



Agilent Technologies

Sinks

August 2005

Notice

The information contained in this document is subject to change without notice.

Agilent Technologies makes no warranty of any kind with regard to this material, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. Agilent Technologies shall not be liable for errors contained herein or for incidental or consequential damages in connection with the furnishing, performance, or use of this material.

Warranty

A copy of the specific warranty terms that apply to this software product is available upon request from your Agilent Technologies representative.

Restricted Rights Legend

Use, duplication or disclosure by the U. S. Government is subject to restrictions as set forth in subparagraph (c) (1) (ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.227-7013 for DoD agencies, and subparagraphs (c) (1) and (c) (2) of the Commercial Computer Software Restricted Rights clause at FAR 52.227-19 for other agencies.

© Agilent Technologies, Inc. 1983-2005
395 Page Mill Road, Palo Alto, CA 94304 U.S.A.

Acknowledgments

Mentor Graphics is a trademark of Mentor Graphics Corporation in the U.S. and other countries.

Microsoft[®], Windows[®], MS Windows[®], Windows NT[®], and MS-DOS[®] are U.S. registered trademarks of Microsoft Corporation.

Pentium[®] is a U.S. registered trademark of Intel Corporation.

PostScript[®] and Acrobat[®] are trademarks of Adobe Systems Incorporated.

UNIX[®] is a registered trademark of the Open Group.

Java[™] is a U.S. trademark of Sun Microsystems, Inc.

SystemC[®] is a registered trademark of Open SystemC Initiative, Inc. in the United States and other countries and is used with permission.

Contents

1 Sinks

Introduction.....	1-1
Connection of a Timed Component Output to a Sink.....	1-2
Sink Data in Binary Dataset.....	1-2
Automatically Opening Data Display after Simulation.....	1-3
Sinks and Optimization.....	1-3
Access to Timed Signal Carrier Frequency in a Dataset.....	1-4
PE Estimator Usage.....	1-4
Typical Baseband and RF System Models.....	1-5
PE Measurement Concepts.....	1-7
Simulation Concepts.....	1-8
Measurement Taps.....	1-8
Noise Sources.....	1-10
Relationship between SNR, Es/No, and Eb/No.....	1-11
Error Detection.....	1-12
Setting Up BER Simulations.....	1-12
References.....	1-16
BER_FER.....	1-18
berIS.....	1-22
berMC.....	1-28
berMC4.....	1-35
EVM.....	1-38
EVM_WithRef.....	1-44
NumericSink.....	1-48
NumericSinkGated.....	1-51
Printer.....	1-53
Sinad.....	1-54
SpectrumAnalyzer.....	1-57
SpectrumAnalyzerResBW.....	1-64
TimedDataWrite.....	1-71
TimedSink.....	1-73
WMAN_EVM.....	1-75

2 Obsolete Sinks

ErrVecMeas.....	2-2
FFTAnalyzer.....	2-7
OutFile.....	2-20
SpecAnalyzer.....	2-22

Index

Chapter 1: Sinks

Introduction

The Sinks library contains sinks that accept input from numeric or timed components. Sinks produce unprocessed data or perform measurement algorithm specific processing and produce processed data.

Data received by a sink can be numeric (scalar or fixed or matrix) or timed (baseband or RF). All data is stored in double-precision floating-point (real) or complex numerical format. Some sinks accept numeric and timed data; other sinks accept specifically either numeric or fixed or timed data; refer to [Table 1-1](#) for details.

Note Information regarding time domain signal differences between ADS Ptolemy simulations and Circuit Envelope and Transient simulations is given in the “[Timed Synchronous Dataflow](#)” section in the *ADS Ptolemy Simulation* manual.

Table 1-1. Sink Summary

Sink Component	Possible Input Data Type	Storage Medium
berMC	timed	binary data set
berMC4	timed	binary data set
berIS	timed	binary data set
EVM	timed	binary data set
NumericSink	numeric or timed (partial data stored)	binary data set
Printer	numeric or timed	ASCII data file
Sinad	timed	binary data set
SpectrumAnalyzer	timed	binary data set
SpectrumAnalyzerResBW	timed	binary data set
TimedSink	timed	binary data set
Obsolete Sink Component	Equivalent New Component	
FFTAAnalyzer	SpectrumAnalyzer	
SpecAnalyzer	SpectrumAnalyzer	
OutFile	TimedDataWrite (for .tim, .bintim, .ascsig, .sig file types); SDFWrite (for .sdf file types) in the Instruments Library	

Table 1-1. Sink Summary (continued)

Sink Component	Possible Input Data Type	Storage Medium
ErrVecMeas	EVM	
WriteVar	<i>not available</i>	

Connection of a Timed Component Output to a Sink

Timed components also have the RLoad and RTemp parameters.

- RLoad by default specifies the resistive load DefaultRLoad, which is a reference to a VAR expression of that name, or a reference to the DF controller parameter of that name that is set to an infinite value by default. If the user expects to collect timed data into a matched resistive load, the user must specify the resistive value to the RLoad parameter. Changes on the DF controller will affect any other timed sink that has RLoad=DefaultRLoad. Changes done on a particular sink will only affect that sink.
- RTemp has a usage similar to the RLoad parameter.

Sink Data in Binary Dataset

The abbreviated dataset name is visible to the user in the data display dialog boxes for the various plot types (Linear, Polar, Smith, Stack, List).

When accessing data stored in a binary dataset within the Data Display window for plotting, the abbreviated name of each set of this data follows one of these naming conventions. The dot separation mark is used to concatenate a partial name into a unique data name.

The abbreviated name of data collected by a sink component that is not part of any hierarchical component in the schematics is:

< sink instance name >

or

< top level schematic design name > . < sink instance name >

The double dot separation mark is used to abbreviate the entire hidden or implicit partial names between the two explicit partial names.

The abbreviated name of data collected by a sink component that is nested one design level below the top level design of a hierarchical schematic design is:

< nested hierarchical design instance name > . < sink instance name >

The naming convention for the full name of the data collected by a sink component that is part of an N-level hierarchical design in the schematics is:

<1st nested design instance name>.<2nd nested design instance name>.....<2nd from last nested instance name>.<last nested design instance name>.<sink instance name>

Automatically Opening Data Display after Simulation

All sinks (except Printer, TimedDataWrite, and OutFile) that generate a binary dataset have a Plot parameter with Rectangular and None options. This variable, in conjunction with the *Open Data Display when simulation completes* setting in the **Simulate > Simulation Setup** dialog box will determine if the data collected by the sink will be automatically plot in a Data Display window.

To automatically plot sink data at the end of simulation, before the start of simulation, from the Schematic window go to **Menu > Simulate > Simulation Setup** and set *Open Data Display when simulation completes* check box. Then, for each sink that you want data to automatically plot, set Plot=Rectangular. At completion of the simulation, a Data Display window will automatically display with rectangular plots of your data. In this window you can set the plotting format, and it will be updated in the same format after the next simulation.

Sinks and Optimization

When optimizing a signal processing schematic design, place a Goal component in conjunction with the Optim controller.

The RangeVar parameter in the Goal component can be used to specify the range variable to be used during optimization; refer to [Table 1-2](#) when it is necessary to specify RangeVar. (Optimization will proceed successfully if RangeVar is not specified.)

Table 1-2. Sinks and Their Independent Variables

Sink	Conditions	Independent Variable
berMC, berMC4	berOutput = ber vs time	Index
	berOutput = ber vs time	time
	berOutput = ber only	Index
	berOutput = ber only	currentTime
berIS		

Table 1-2. Sinks and Their Independent Variables (continued)

Sink	Conditions	Independent Variable
EVM		Index
NumericSink		Index
Sinad		xIndex
SpectrumAnalyzer		freq
SpectrumAnalyzerResBW		freq
TimedSink		time
FFTAAnalyzer, SpecAnalyzer	DisplayFreqUnit=Hz	Hertz
	DisplayFreqUnit=kHz	Kilo_Hertz
	DisplayFreqUnit=MHz	Mega_Hertz
	DisplayFreqUnit=GHz	Giga_Hertz
ErrVecMeas		xIndex

Access to Timed Signal Carrier Frequency in a Dataset

From the Data Display window, the value for carrier frequency f_c for a timed signal is available by use of an expression. To use this dataset value, place an equation (*Insert > Equation*). In the dialog box, write an equation with the name on the left side (*MyFc*), and on the right side using the *get_attr()* function. For example,

$$\text{MyEqn} = \text{get_attr}(\langle \text{sink data name} \rangle, "fc")$$

where $\langle \text{sink data name} \rangle$ is the name of the sink data selected from the list shown in the equation dialog box. (The sink data name is described previously.) This *MyFc* equation can then be used in a plot, table, or other equations.

PE Estimator Usage

This section explains the use of the probability of error (PE) measurement items *berMC*, *berMC4*, and *berIS* in the Signal Processing schematic. With these items, it is possible to measure not only the PE performance of the system but the signal-to-noise ratio of the system as well.

The basic system models and the PE simulation concepts and methodology are explained.

Typical Baseband and RF System Models

Communication systems can be broadly classified as baseband or RF systems. Baseband systems typically use the PAM data format although other formats, such as pulse width modulation and pulse position modulation, can also be used. RF systems use a wide variety of modulation schemes such as QAM, PSK schemes such as QPSK, DQPSK, PI4DQPSK, and others such as MSK and FSK. The primary interest is to measure the probability-of-symbol error (P_s) or bit-error rate (BER) of the system.

In [Figure 1-1](#), a general model of a baseband system link is represented in part (a) and general RF system models are represented in parts (b) and (c).

An RF system consists of the input bit stream, Transmit Data Encoder, Transmit Baseband Network, RF Modulator, Transmit RF Network, RF Channel, Receive RF Network, RF Demodulator, Receive Baseband Network, Receive Data Decoder, and the output bit stream. The RF system may have a single data channel, or may have I and Q data channels. The Baseband system omits the RF sections. However, the following discussion applies to both systems.

Baseband System Model

The bit stream is often in the NRZ (non-return-to-zero) data format. The Transmit Data Encoder is used for different purposes such as to convert the NRZ data format to another format: for example, multi-level PAM (pulse amplitude modulation) symbol format, or for error control coding.

The Baseband Channel has a transmit and receive side and is typically a bandwidth limited environment with intersymbol interference and noise introduced in the channel. The Receive Data Decoder is used to convert the received symbols to the desired output binary data stream.

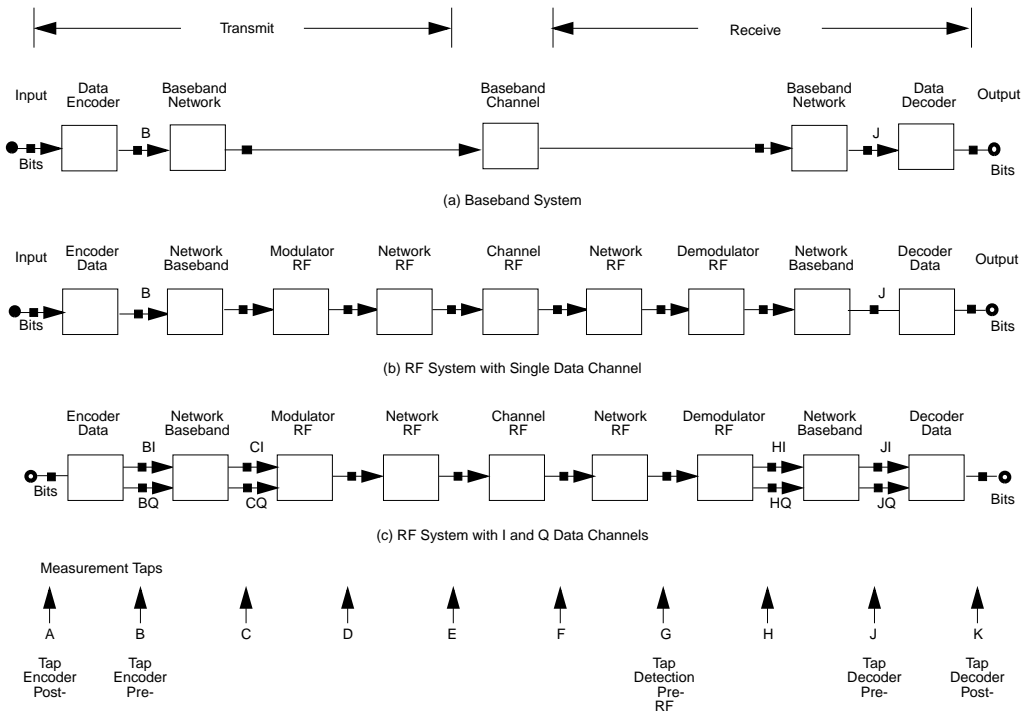


Figure 1-1. Typical Baseband and RF Systems

RF System Model

The bit stream for this system is also often in the NRZ data format. The Transmit Data Encoder again is used to convert the NRZ data format to another format, or for error control coding. The Transmit Baseband Network is typically used to band limit the frequency spectra of the data symbols, and often introduces intersymbol interference to the symbol stream.

The RF modulator can be of many types. Some formats (such as QAM and QPSK) use I and Q data channels while others (such as BPSK) contain a single data channel.

The RF networks and channel include items such as transmit IF and RF filtering, upconverters, high power amplifiers, coaxial cable, antennas, line of sight link, receive RF and IF filters, and receive low-noise amplifier.

The RF Demodulator converts the RF energy back to a baseband symbol stream that includes symbol distortions, interference, and noise. The Receive Baseband Network

is typically used to filter the received symbol stream to reduce symbol distortion and noise. The Receive Data Decoder is used to convert the received symbols to the desired output bit stream.

PE Measurement Concepts

The probability of error of a system is measured by comparing the output data stream to the input data stream. The BER gives the average number of output bit-errors per input bit; for example, a 10^{-6} BER means that, on the average, 1 output bit-error occurs with 10^6 input bits

When the data streams being compared are not simple 2-level (binary) data streams, the measurement is with respect to the data symbols and is called the probability of symbol error (Ps) measurement.

Typical hardware Ps/BER measurements are based on the exact number of symbol/bit-errors and the number of symbols/bits transmitted. This is called a Monte Carlo measurement.

For the PE measurement to be statistically significant, the number of bits transmitted should typically be much greater than $1/PE$ (as a rule of thumb).

The relative variance of the PE for N transmitted bits is:

$$\text{VAR} = (1 - PE) / (PE \times N)$$

This implies that, for a PE of 10^{-6} , with a relative variance of 0.01, a sample size N of approximately 10^8 bits is required.

Figure 1-2 shows the confidence bands on the PE measurement versus total number of bits observed.

For the program PE measurement, the number of samples required is established by setting the relative variance for the PE measurement. As can be seen, for a PE of 1.0×10^{-k} with 100×10^k samples measured, there is a 99% confidence that the actual PE is between 0.77×10^{-k} and 1.3×10^{-k} .

For a low PE uncertainty, a smaller variance is required. However, a smaller variance requires a larger number of transmitted bits.

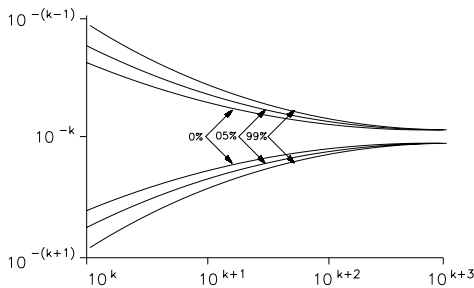


Figure 1-2. PE Confidence Bands

Simulation Concepts

Simulation is performed in discrete time steps. At each time step, the signals generated by all the sources are propagated through the system and the outputs are evaluated. Noise can be introduced in the system by means of the random noise sources available. Another method of introducing noise is to use the electrical models of components and to define their noise properties by means of parameters such as noise figure or noise temperature. This is a useful feature that permits the user to build an accurate physical model of an actual system. The program automatically generates an equivalent noise source that represents the noise properties of the electrical component. No distinction is made between data sources and noise sources during simulation when evaluating the output of the system and the output measured is the net effect due to all sources.

No assumption is made about the nature of the noise at the system output. The statistics of the noise at the output depend on the transformations it undergoes when propagating through the system. The effects of nonlinearities in the system can thus be accurately simulated.

Measurement Taps

The PE performance of a system is calculated by comparing the transmitted data (also referred to as V_{ref}) with the output data of the receiver (also referred to as the V_{test}). V_{ref} can be measured at either the pre-encoder tap or the post encoder tap of the transmitter (see [Figure 1-1](#)); V_{test} should be correspondingly measured at the post-decoder tap or the pre-decoder tap at the receiver. The choice depends on the system under consideration. For example, if the data encoder consists of an NRZ to

4-level PAM data converter, the post-encoder and pre-decoder taps are convenient measurement taps for the reference and test signals, respectively. However the NRZ data bits may first be encoded with an error correction encoder prior to converting them to a 4-level PAM format. The data decoder would then consist of a 4-level PAM to binary converter followed by an error correction decoder. The PE performance of the entire system should then be measured by comparing the pre-encoder signal to the post-decoder signal.

The signal-to-noise ratio of the system cannot be measured by monitoring the receiver output because the output is the combined effect of the data signals and the noise. Therefore, the signal and noise statistics cannot be measured separately. The user must either calculate the SNR by examining the structure of the system and the statistical properties of the noise sources, or make other measurements (depending on the system) to determine the SNR.

However, it is possible to introduce noise in the system by placing an external noise source. Such a noise source would be typically introduced in the channel. If this external noise source is the predominant source of noise in the system, the SNR can be calculated by measuring the power of the noise source and the power of the data signal separately. An additional measurement tap (the tap is labelled L and will be called the Reference noise tap) is then required as shown in Figure 1-3 (also see Figure 1-1).

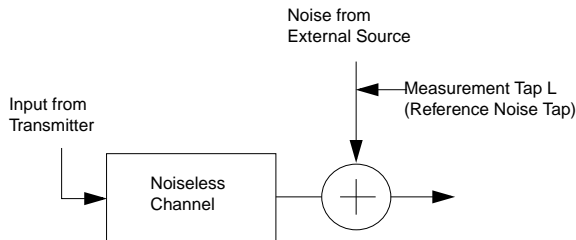


Figure 1-3. Introduction of Noise from an External Source into a System

In Figure 1-3, the *noiseless channel* block represents effects of the channel such as attenuation of the signal, propagation delay, distortion, fading, etc. The noise signal measured at the Reference noise tap is called $E_{N_{ref}}$ and the data signal that is used to measure the signal energy is called $E_{S_{ref}}$. $E_{S_{ref}}$ can be measured at any tap in the system prior to the point where the noise is introduced.

Noise Sources

The nature of the noise introduced in the channel depends on whether the system is a baseband or RF system. External noise is typically generated by using the Noise source that is a baseband or RF noise source. The output of the Noise source can be directly added to the signal in the channel.

Another important aspect is the spectral characteristic of the Noise source (see [Figure 1-4](#)). Output of the Noise source is a bandlimited white noise signal whose bandwidth is dependent on the time step at which the simulation is carried out. The time interval between two consecutive noise samples T_o , is determined by the parameter TStep in the Noise source. The (baseband) bandwidth of the noise is then equal to $1/(2 \times TStep)$ and the power spectral density is constant over this bandwidth.

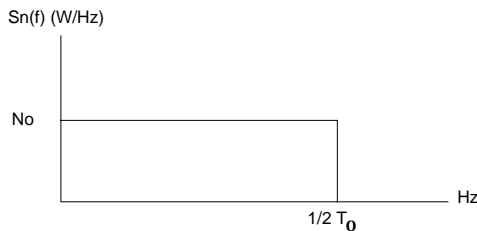


Figure 1-4. Spectral Characteristic of the Noise Source

Let

σ = RMS value of the output of the Noise source in volts

T_o = simulation time step in seconds

R_{ref} = default reference resistance is 50 Ohms

$S_n(f)$ = one-sided power spectral density of the bandlimited white noise in W/Hz

Then

$$S_n(f) = N_o \text{ (W/Hz)} \quad \text{for } 0 \leq f \leq \frac{1}{2T_o} \text{ (Hz)}$$

where

$$N_o = \frac{2\sigma^2 T_o}{R_{ref}}$$

Therefore, the auto-correlation function of the noise is given by

$$R_n(\tau) = \frac{N_o}{2T_o} \text{sinc}\left(\frac{\tau}{T_o}\right)$$

Because the noise is sampled every T_o seconds, the correlation between the noise samples is given by $R_n(k T_o)$, where k is an integer; therefore, the correlation between the noise samples is zero. In the case of a Gaussian noise source, the samples are also independent.

Relationship between SNR, Es/No, and Eb/No

The signal-to-noise ratio of a system can be calculated in different ways. The ratio of the signal power to the noise power is one such measure and is denoted as the SNR of the system. A measure that is used more commonly is the ratio of the energy per symbol to the power spectral density of the noise (E_s/N_o), or the ratio of the energy per bit to the power spectral density of the noise (E_b/N_o).

The berMC4 component measures the power in E_{Sref} and E_{Nref} from which the desired signal-to-noise ratio is calculated as follows.

Define the following:

P_{ESref} = power in the E_{Sref} signal in Watts

P_{ENref} = power in the E_{Nref} signal in Watts

T_o = simulation time step in seconds

T_s = symbol time in seconds

E_s/N_o is calculated according to the equations:

$$E_s = P_{ESref} \times T_s$$

If the noise signal is in a baseband representation, then

$$N_o = P_{ENref} \times (2 \times T_o) \quad \text{and} \quad \frac{E_s}{N_o} = \frac{P_{ESref} \times T_s}{P_{ENref} \times (2 \times T_o)}$$

If the noise signal is in a complex envelope representation, then

$$N_o = P_{ENref} \times T_o \quad \text{and} \quad \frac{E_s}{N_o} = \frac{P_{ESref} \times T_s}{P_{ENref} \times T_o}$$

To convert E_s/N_0 to E_b/N_0 , let each symbol carry L bits of information. Then E_b/N_0 is simply given by $E_b/N_0 = (E_s/N_0)/L$.

SNR is simply given by $SNR = P_{ESref} / P_{ENref}$.

Error Detection

Error detection is performed by sampling V_{ref} and V_{test} every T_s seconds (here T_s is the symbol time) and comparing the samples. If the samples do not lie within the same threshold levels, then an error is declared. The ratio of the number of errors counted to the total number of bits transmitted is the estimated PE.

Setting Up BER Simulations

Three important points must be considered when setting up a BER measurement:

- Synchronizing test (V_{test}) and reference (V_{ref}) signals
- Choosing the optimal sampling instant
- Scaling V_{test} and V_{ref} appropriately

Each point is discussed in the following sections. For these discussions, *BER sink* refers to any berMC, berMC4, or berIS component.

[“BER Simulation Examples” on page 1-16](#) provides information for how to access designs that demonstrate the concepts described in the following sections.

Synchronizing Test and Reference Signals

A successful BER simulation requires that V_{test} and V_{ref} are synchronized.

Otherwise, the BER measurement result will most likely be close to 0.5 (50%). There are two ways V_{test} and V_{ref} can be synchronized.

- **Manual Synchronization**

If the exact delay between V_{test} and V_{ref} is known, synchronization can be easily achieved by introducing the same amount of delay in V_{ref} using the DelayRF component. In this case, set the DelayBound parameter of the BER sink to -1 to turn off the auto synchronization feature.

The exact delay between V_{test} and V_{ref} can be found in several ways:

- Sometimes the delay introduced by each component in the path between V_{test} and V_{ref} is known. For example, the delay introduced by the LPF_RaisedCosineTimed component is known since it is one of the parameters a user needs to set. In this case, adding the delays introduced by each component will give the exact delay between V_{test} and V_{ref} .
- V_{test} and V_{ref} can be recorded (using TimedSink components) and plotted in the data display. By observing the plots, the delay between the two signals can often be determined.
- The CrossCorr component can be used to measure the delay between V_{test} and V_{ref} (see the *MeasuringDelay* design in example project: *File > Example Project > PtolemyDocExamples > BER_Validation_prj*).

When determining the delay by observing the plots of V_{test} and V_{ref} versus time or by using the CrossCorr component, turn off noise and other impairments in the system (set power level of noise sources very low, make amplifiers linear). When these impairments are turned off, the V_{test} waveform will be close to the ideal waveform, which will help determine the delay more easily and accurately.

- **Automatic Synchronization**

If the exact delay between V_{test} and V_{ref} is unknown, the auto synchronization feature of the BER sink can be used. In this case, the DelayBound parameter of the BER sink must be set to a value that is an upper bound of the exact delay between the two signals. The BER sink will then cross correlate the two signals and try to estimate the delay between them. Since the auto synchronization feature relies on the cross correlation to estimate the delay, the BER sink may not be able to always synchronize the two signals, especially at low signal-to-noise ratios.

An upper bound of the delay between V_{test} and V_{ref} can be found using any of the three ways described above used to find the exact delay.

- Knowing the upper bound of the delay introduced by each of the components in the path between V_{test} and V_{ref} and adding these upper bounds.
- Observing the plots of V_{test} and V_{ref} versus time.
- Using the CrossCorr component to get an initial estimate of the delay and adding a few simulation time steps to it.

When the auto synchronization feature is used, no delay needs to be added to the reference signal. However, if the upper bound of the delay is big (greater than 50 symbol periods) and if a lower bound of the delay (D_L) between V_{test} and V_{ref} is known, then it is recommended that V_{ref} is delayed by this amount (D_L) using a DelayRF component and DelayBound is also reduced by the same amount (D_L). This reduces the memory used by the BER sink and will speed up the synchronization process. For example, assume that the delay introduced by the transmitter and receiver filters is 20 μsec and the delay introduced by the rest of the components (e.g. RF channel) has an upper bound of 35 μsec . One way to set up the BER simulation is to not delay the reference signal at all and set the DelayBound parameter of the BER sink to 55 μsec . Another (recommended) way is to delay the reference signal by 20 μsec and set DelayBound to 35 μsec .

When the auto synchronization feature is used, the BER sink saves the value of the estimated delay in the simulation dataset. The name of the dataset variable is *<sink instance name>.Delay*.

Choosing the Optimal Sampling Instant

The BER sink downsamples V_{test} and V_{ref} to one sample per symbol before it compares them in order to detect errors. The first sample of V_{ref} is taken at the time instant specified by the Start parameter. If auto synchronization is turned off (DelayBound=-1), then the first sample of V_{test} is also taken at Start. If auto synchronization is turned on, then the first sample of V_{test} is taken at Start+Delay, where Delay is the delay the BER sink estimated.

The optimal sampling instant is the instant where there is no ISI (intersymbol interference) or where the minimum ISI occurs. For systems, using root raised cosine filters at the transmitter and receiver, the optimal sampling instant is at the center of the symbol period. Therefore, set the Start parameter of the BER sink as follows:

- if no delay is introduced in the reference signal, set Start to

$$N \times \text{SymbolTime} + \text{int}((\text{SampPerSym} - 1) / 2) \times \text{TStep}$$

where N is a positive integer, SymbolTime is the symbol period, SampPerSym is the number of samples per symbol in V_{test} and V_{ref} and TStep is the simulation time step for V_{test} and V_{ref} .

- if a delay of D is introduced in the reference signal, set Start to

$$N \times \text{SymbolTime} + \text{int}((\text{SampPerSym} - 1) / 2) \times \text{TStep} + D$$

If your system has differential encoding/decoding, synchronization loops, carrier recovery sub-systems, or other sub-systems that need time to reach their steady state, set N to a value that is large enough to allow the entire system to reach steady state.

When a square root raised cosine filter with pulse equalization is used in the modulator (this filter is used inside the DBPSK_Mod, QPSK_Mod, DQPSK_Mod, DQOPSK_Pi4Mod components) and the input signal to the modulator is an NRZ waveform (piece-wise constant waveform for multi-level PAM and QAM systems) with an even number of samples per symbol, then the signal at the output of the modulator and eventually at the output of the demodulator will not have a sample at the optimal sampling instant. The optimal sampling instant occurs at the midpoints in between the existing signal samples. This is a limitation that results from the need to represent continuous time signals with samples. If there is no signal sample at the optimal sampling instant, then the BER sink will sample V_{test} at a point where there is ISI and the BER results will be worse than expected. To overcome this limitation, the Interpolator component can be connected to the output of the modulator. When the TimeRef parameter of the Interpolator is set to $\text{TStep} / 2$ and a fourth order Lagrange interpolation is used, the Interpolator will reconstruct the signal samples that occur at the midpoints between the existing samples.

Scaling of Test and Reference Signals

For signals with only two levels, such as 2-level PAM, BPSK, DBPSK, QPSK (2 levels per axis), DQPSK (2 levels per axis), and zero mean value the threshold for deciding whether an error has occurred or not is 0 (if both V_{test} and V_{ref} are positive or both are negative no error is detected; if one is positive and the other one is negative an error is detected). In this case, the exact level of V_{test} and V_{ref} is not important and does not affect the BER measurement result.

For signals with more than two levels, such as multi-level PAM (4-PAM, 8-PAM, 16-PAM, ...) or QAM (16-QAM, 32-QAM, 64-QAM), the exact level of V_{test} and V_{ref} is important because there is more than one error detection threshold.

If the automatic threshold setting is used (BER sink ThresholdSetting parameter set to *automatically*) then the error detection thresholds are set to $(2 \times i - N) / N$, $i = 1, 2, \dots, N$, where N is the number of thresholds (BER sink NumThreshold parameter). NumThreshold must be set to the number of signal levels minus 1 (3 for 4-PAM, 7 for 8-PAM, 3 for 16-QAM (3 levels per axis), 7 for 64-QAM (7 levels per axis)). The

expected signal levels are located at the midpoints between the thresholds, that is at $(2 \times i - 2 - N) / N$, $i = 1, 2, \dots, N+1$. Both V_{test} and V_{ref} must be scaled appropriately so that when they are sampled at the optimal sampling instant and assuming ideal conditions (no noise or other distortions) they generate samples at the expected levels. Turning off the noise and the rest of the impairments in the system and plotting the eye diagrams for V_{test} and V_{ref} can help determine the signal levels at the optimal sampling instant and therefore the appropriate scale factors.

Note When the berIS component is used, the V_{ref} and V_{test} signal levels must be the same (including 2-level PAM, BPSK, DBPSK, QPSK, and DQPSK signals). This is because the amount of noise added to V_{test} inside the berIS component is dependent on signal levels at the berIS inputs.

BER Simulation Examples

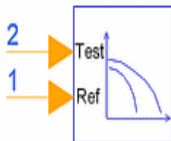
The *BER_Validation_prj* example project (*File > Example Project > PtolemyDocExamples > BER_Validation_prj*) includes several example designs that demonstrate the concepts discussed in the previous sections. These designs show how the theoretical BER performance in an AWGN (additive white gaussian noise) environment can be achieved for various modulation types such as BPSK, DBPSK, QPSK, DQPSK, $\pi/4$ -DQPSK, 16-QAM, 64-QAM. Each example design provides setup information.

References

- [1] J. Baprawski, "Generalized Modulation Mathematics Applied to RF System Simulation," *Proceedings of the RF Expo West Conference*, Santa Clara, CA, February 5-7, 1991, pp. 127-147.
- [2] T. T. Ha, *Digital Satellite Communications*, McGraw-Hill, 1990.
- [3] N. Kanaglekar, et al. "Wave Analysis of Noise in Interconnected Multiport Networks," *IEEE Transactions on Microwave Theory and Techniques*, Vol. MTT-35, No. 2, February 1987, pp. 112-115.
- [4] D. Lu and J. Baprawski, "The Quasi-Analytical Method for Estimating Error Probabilities of M-ary Signaling Systems with Intersymbol Interference," *Proceedings of the 13th Symposium on Information Theory and its Applications*, Tateshina, Japan, January 23-25, 1991, pp. 109-114.

- [5] D. Lu and K. Yao, "Improved Importance Sampling Technique for Efficient Simulation of Digital Communication," *IEEE Journal on Selected Areas in Communication*, J-SAC, Vol. 6, No. 1, January 1988, pp. 67-75.
- [6] R. W. Lucky, J. Salz and E. J. Weldon, Jr., *Principles of Data Communications*, McGraw-Hill, 1968.
- [7] P. Z. Peebles, Jr., *Digital Communication Systems*, Prentice-Hall, 1987.
- [8] G. Proakis, *Digital Communication*, McGraw-Hill, 1989.
- [9] K. S. Shanmugan, *Digital and Analog Communication Systems*, John Wiley & Sons, 1985.
- [10] B. Sklar, *Digital Communications*, Prentice-Hall, 1988.

BER_FER



Description Bit Error Rate and Frame Error Rate estimation

Library Sinks

Class SDFBER_FER

Parameters

Name	Description	Default	Type	Range
Plot	Plot data when set to 'Rectangular' and Simulation Setup set to 'Open Data Display when simulation completes': None, Rectangular	None	enum	
Start	Data collection start index	DefaultNumericStart	int	[0, ∞)
Stop	Data collection stop index when EstRelVariance is not met	DefaultNumericStop	int	(Start, ∞)
ControlSimulation	Let sink control how long the simulation will run? NO, YES	YES	enum	
BitsPerFrame	Bits per frame	100	int	[1, ∞)
EstRelVariance	BER estimation relative variance	0.01	real	[0, 1)
OutputBER	BER output: BER vs index, BER vs index every 10 bits, BER vs index every 100 bits, BER vs index every 1000 bits, BER vs index every BitsPerFrame bits, Final BER	Final BER	enum	
OutputFER	FER output: FER vs frame, FER vs frame every 10 frames, Final FER, No FER	Final FER	enum	

Pin Inputs

Pin	Name	Description	Signal Type
1	ref	reference bit stream	int
2	test	test bit stream	int

Notes/Equations

1. BER_FER can be used to measure the BER (bit error rate) and FER (frame error rate) of a system. In some systems, FER is referred to as PER (packet error rate) or BLER (block error rate). The input signals to the reference (ref) and test (test) inputs must be bit streams. The bit streams must be synchronized, otherwise the BER/FER estimates will be wrong.
2. The Start parameter defines when data processing starts. The end of data processing depends on the settings of the parameters ControlSimulation, Stop, and EstRelVariance:
 - If ControlSimulation is NO, then Stop and EstRelVariance are ignored. Data processing ends when the simulation ends. In this case, the end of the simulation is determined by other sink or source components that control the simulation.
 - If ControlSimulation is YES and EstRelVariance is 0.0, then data processing ends when Stop is reached.
 - If ControlSimulation is YES and EstRelVariance is greater than 0.0, then data processing ends when EstRelVariance is met or when Stop is reached. In this case, Stop acts as an upper bound on how long the simulation will run just in case it takes too long for EstRelVariance to be met. In this mode of operation, messages are printed in the simulation status window showing the value of estimation relative variance as the simulation progresses. The EstRelVariance parameter can be used to control the quality of the BER estimate BER_FER generates. The lower the value of EstRelVariance the more accurate the estimate will be.

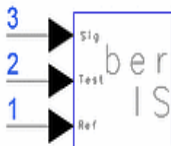
For more details, refer to [“PE Measurement Concepts” on page 1-7](#). Note that the equation for the estimation relative variance described above assumes that the errors happen randomly (as in the case of an AWGN channel) and not in bursts (as in the case of a fading channel).

3. The `BitsPerFrame` parameter sets the number of bits in each frame. A frame is considered to be in error if at least one of the bits in the frame is detected incorrectly. If the bit errors are independent identically distributed events then BER and FER are related through the equation $FER = 1 - (1 - BER)^{BitsPerFrame}$.
To estimate BER/FER over an exact number of frames set `ControlSimulation` to YES, `EstRelVariance` to 0.0, and `Stop` to *Start + N × BitsPerFrame - 1*, where *Start* is the value of the `Start` parameter, *BitsPerFrame* is the value of the `BitsPerFrame` parameter and *N* is the number of frames to be simulated.
4. The `OutputBER` parameter determines how often BER values are written in the dataset.
 - *Final BER* writes the final BER value.
 - *BER vs index* writes BER values as a function of index. The user can see how BER changes throughout the simulation.
 - *BER vs index every 10 bits*, *BER vs index every 100 bits*, *BER vs index every 1000 bits*, and *BER vs index every BitsPerFrame bits* behave similar to *BER vs index* but BER values are written every 10, 100, 1000, and `BitsPerFrame` bits, respectively. The user can see how BER changes throughout the simulation while keeping the dataset at a reasonable size.
5. The `OutputFER` parameter determines how often FER values are written in the dataset.
 - *No FER* does not write any FER values.
 - *Final FER* writes the final FER value.
 - *FER vs frame* writes FER values as a function of frame. The user can see how FER changes throughout the simulation.
 - *FER vs frame every 10 frames* behaves similarly to *FER vs frame* but FER values are written every 10 frames. The user can see how FER changes throughout the simulation while keeping the dataset at a reasonable size.
6. Lengthy BER simulation times can be significantly shortened by using the *Parallel BER* option in the *Simulate > Simulation Setup* window > *Parallel* tab. (To use this option, the user's machine must have LSF client installed and it must be connected to an LSF cluster.) In the *Parallel* tab *Simulation Mode*, choose *Parallel Hosts* to activate the *Parallel BER* field.

For details regarding the Parallel BER option, refer to note 8 of the berMC component documentation.

An example that demonstrates the *Parallel BER* feature using the BER_FER component is provided; to access this example, from the ADS Main window choose *File > Example Project > WLAN > WLAN_80211a_PER_prj > WLAN_80211a_36Mbps_AWGN_System*.

berIS



Description Error Probability measurement using Improved Importance Sampling

Library Sinks

Class TSDF_berIS

Derived From baseAutoDisplaySink

Parameters

Name	Description	Default	Unit	Type	Range
Plot	if simulation is setup to open data display after simulation and if Plot is not set to 'None', then plot the data for this sink: None, Rectangular	None		enum	
RLoad	Load resistance. DefaultRLoad will inherit from DF controller.	DefaultRLoad	Ohm	real	(0, ∞)
RTemp	Resistor physical temperature, in degrees C. DefaultRTemp will inherit from DF controller.	DefaultRTemp	Celsius	real	[-273.15, ∞)
Start	Start time for sampling signals	DefaultTimeStart	sec	real	[0, ∞)
SymbolTime	Symbol time	1.0	sec	real	[TStep, ∞)†
NumThreshold	Number of thresholds for the error detection	1		int	[1, ∞)
ThresholdSetting	Type of threshold settings: automatically, manually	automatically		enum	
Threshold	Threshold values when user selects manually setting	0		real array	
DelayBound	Upper bound of delay for Synchronizing inputs; if DelayBound <= 0, the Synchronizer is turned off	-1.0	sec	real	{-1} or (0, ∞)††

Name	Description	Default	Unit	Type	Range
berOutput	Type of ber output: ber vs time, ber vs time every 10 symbols, ber vs time every 100 symbols, ber vs time every 1000 symbols, ber only	ber only		enum	
NBw	Noise bandwidth	3.0	Hz	real	(0, ∞)
SystemType	System type: PAM, QAM, QPSK, DQPSK, PI4DQPSK	PAM		enum	
EstVar	Estimation relative variance	0.1		real	(0, 1]
EsOverNo	Energy per Symbol over Noise Density in dB	3.0		real	($-\infty$, ∞)
EsOverNoRange	EsOverNo range	0.0		real	[0, ∞)
NumSwpsForEsOverNo	Number of sweep points for EsOverNo	1		int	[1, ∞)
<p>† TStep is the simulation time step for the component input signals.</p> <p>†† if DelayBound is set to -1, berIS will assume test and reference signals are already synchronized.</p>					

Pin Inputs

Pin	Name	Description	Signal Type
1	input1	reference data input	timed
2	input2	test signal input	timed
3	input3	receiving signal input	timed

Notes/Equations

1. Basic Principle

The berIS probability of error estimation is based on the Improved Importance Sampling (IIS) method [1 - 3], which can quickly estimate error probabilities for PAM, QAM, QPSK, DQPSK, and PI/4DQPSK systems. (The IIS method is system dependent—berIS cannot be used to estimate error probability for other systems.) To speed simulation, error events are made to occur more frequently by modifying noise density in the IIS.

Symbol error probability detection is based on comparing modified test data to reference data waveforms, symbol by symbol. When calculating the probability,

a weighting function is used to adjust the error probability to an unbiased estimation.

To measure symbol error probability:

- Synchronize output test data (test data) with reference data.
- Sample reference and test data. Each sampling time must be at the center of each symbol. The sampling time interval is one symbol time.
- Compare the modified test data sample to the reference data sample.
- Detect an error event by using thresholds. An error occurs if the value of the test sample is not the same as reference sample in the threshold detection; otherwise, there is no error.
- When an error event is detected, calculate the weighting function and add the weight in the error probability buffer, then normalize the error probability by using total number of test samples N . Assume the total number of test data samples (or reference samples) is N , the IIS weighting function is w , and the indicating function for error is $I(.)$, where when an error is detected $I(.) = 1$, otherwise $I(.) = 0$. Error probability can be mathematically described by

$$P_s = \frac{1}{N} \sum I(i) w_i$$

2. Data Normalization

To simplify detection, test and reference data need to be normalized from -1 to 1 . All detection thresholds are normalized from -1 to 1 also. The thresholds result in symbol ranges equally spaced between -1 and $+1$ if ThresholdSetting=automatically.

3. Synchronizing Test and Reference Data

Test and reference data can be synchronized automatically or by the user:

Automatic Synchronization

- Set DelayBound > exact delay between test and reference data. Typically, DelayBound = $5\times$ to $10\times$ exact delay. This approach does not require knowing the exact delay between test and reference data. Auto-synchronization estimates the delay based on calculating the auto-correlation function between the two waveforms.

User Synchronization

- Set DelayBound = -1.
- Find the exact delay between test data and reference data.
- Place a Delay component between the reference data source and the reference input pin of the berIS component.

Continue synchronization with these parameters.

- Start is used for specifying a sampling start time for reference signal, and also for positioning test and reference samples. As mentioned in note 1, each sampling time must be at the center of each symbol. For example, for the reference signal from ADS data source, if DelayBound = -1, Start must be set to Delay+Stime/2, where the exact delay between test and reference data is Delay, and symbol time is Stime. Otherwise, Start can be set to Stime/2.
- NumThreshold is determined by signal levels; assume the signal level is M, NumThreshold=M-1.
- ThresholdSetting options:
 - automatically* setting thresholds for uniform threshold interval.
 - manually* setting thresholds for uniform or non-uniform interval
- Threshold: Threshold values when user selects *manually* setting
- NBw is the noise bandwidth specified by the user. In the model the internal noise power will be calculated using the EsOverNo and NBw parameters.
- SystemType can be set according to test system type: PAM, QAM, QPSK, DQPSK, or PI4DQPSK.
- EstVar is for controlling estimation accuracy, the simulation relative variance EstVar must be properly specified. A smaller EstVar takes more time and results in a more accurate simulation; a larger EstVar takes less time but the result is not as accurate. When EstVar=0.01, which is a good estimation, a 10% standard deviation will be received.
- EsOverNo is the minimum Es/No in dB. It can be set by the user. The error probability performance will be performed based on the EsOverNo (Es/No) set by the user. In the model the internal noise power will be calculated by using the EsOverNo and NBw parameters.
- EsOverNoRange is the range from minimum Es/No to maximum Es/No in dB.

- NumSwpsForEsOverNo is the number of plotting points for Ps Vs Es/No curve

4. Symbol Error Probability and Bit Error Probability

berIS measures the symbol error probability P_s for a single channel. For systems with I,Q channels such as QAM, QPSK, DQPSK, and PI/4DQPSK, the user must measure both channels to get P_{si} , P_{sq} . To derive the bit error probability P_{ei} and P_{eq} we can use

$$P_{ei} = P_{si}/L, P_{eq} = P_{sq}/L$$

where L is the number of bits per symbol for I, Q channels.

Then combine the results for the bit error probability of the system by using

$$P_e = P_{ei} + P_{eq} - P_{ei} P_{eq}$$

where P_e is the bit error probability for the system and P_{ei} and P_{eq} are the bit error probabilities for each channel.

5. Noise Source

An external noise source is not required. Noise power is added by berIS in terms of Es/No and the noise bandwidth input from the NBw parameter.

6. Ps Vs Es/No Curve

To generate a P_s curve, P_s Vs Es/No or BER Vs Es/No, previously, the ADS sweep controller must be used for sweeping values of Es/No. In that case, all signals must be re-generated for each Es/No. For the new version of berIS model, signals for testing BER can be re-used for different Es/No, and the simulation is more efficient.

To generate a BER curve properly, the Es/No range can be specified by using EsOverNo and EsOverNoRange, and the number of sweep points can be set by using NumSwpsForEsOverNo. Because the sweep controller is not used any more to re-generate test signals for different Es/No, the BER curve can be generated faster.

7. Delay between reference signal and test signal. If set DelayBound > 0, this component provide an estimation of the delay. The delay value can be found in the data display server.

8. IIS Simulation

For a QAM system at 10^{-6} of BER and Monte Carlo simulation, 10^8 simulation samples are needed for a reasonable accuracy of 0.01 relative variance. However, for the IIS simulation, only 800 samples are needed for accuracy [1].

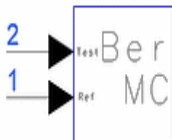
An external noise source is not required. Noise power is added by berIS in terms of E_s/N_0 and noise bandwidth at input 3.

9. For general information regarding sinks, refer to [“Introduction” on page 1-1](#).

References

- [1] D. Lu and K. Yao, “Improved Importance Sampling Technique for Efficient Simulation of Digital Communication Systems,” *IEEE J. Select. Areas Commun.* vol. 6, pp. 67-75, Jan. 1988.
- [2] D. Lu and K. Yao, “Estimation Variance Bounds of Importance Sampling Simulations in Digital Communication Systems,” *IEEE Trans. Commun.* vol 39, pp. 1413-1417, Oct. 1991.
- [3] J. Chen, D. Lu, J. Sadowski, and K. Yao “On Importance Sampling in Digital Communications - Part I: Fundamentals,” *IEEE J. Select. Areas in Commun.* vol 11. No. 3, pp. 289-299, April, 1993.

berMC



Description Error Probability measurement using Monte Carlo Method, 2 inputs

Library Sinks

Class TSDF_berMC

Derived From baseTimeSink

Parameters

Name	Description	Default	Unit	Type	Range
Plot	if simulation is setup to open data display after simulation and if Plot is not set to 'None', then plot the data for this sink: None, Rectangular	None		enum	
RLoad	Load resistance. DefaultRLoad will inherit from the DF controller.	DefaultRLoad	Ohm	real	(0, ∞)
RTemp	Resistor physical temperature, in degrees C. DefaultRTemp will inherit from the DF controller.	DefaultRTemp	Celsius	real	[-273.15, ∞)
Start	Start time for data recording. DefaultTimeStart will inherit from the DF Controller.	DefaultTimeStart	sec	real	[0, ∞)
Stop	Stop time for data recording. DefaultTimeStop will inherit from the DF Controller.	DefaultTimeStop	sec	real	[Start, ∞)
ControlSimulation	if set to YES, 'Stop' time determines how long the simulation will run: NO, YES	YES		enum	
SymbolTime	symbol time	1.0	sec	real	[TStep, ∞)†
EstRelVariance	estimation relative variance (if set to 0.0 simulation will run until Stop is reached)	0.0		real	[0, 1]

Name	Description	Default	Unit	Type	Range
NumThreshold	number of threshold	1		int	[1, ∞)
ThresholdSetting	type of threshold settings: automatically, manually	automatically		enum	
Threshold	threshold values when user selects manually setting	0		real array	
DelayBound	upper bound of delay for Synchronizing inputs; if DelayBound = -1, the Synchronizer is turned off	-1.0	sec	real	{-1} or (0, ∞)††
berOutput	Type of ber output: ber vs time, ber vs time every 10 symbols, ber vs time every 100 symbols, ber vs time every 1000 symbols, ber only	ber only		enum	
† TStep is the simulation time step for the component input signals. †† if DelayBound is set to -1, berMC will assume test and reference signals are already synchronized.					

Pin Inputs

Pin	Name	Description	Signal Type
1	input	timed sink input signal	timed
2	input2	test signal input	timed

Notes/Equations

1. Basic Principle

berMC uses the Monte Carlo method for probability of error measurement of linear as well as nonlinear communication systems.

The measurement is based on comparing test data to reference data waveforms, symbol by symbol.

To measure symbol error probability:

- Synchronize output test data (test data) with reference data.
- Sample reference and test data. Each sampling time must be at the center of each symbol. The sampling time interval is one symbol time.
- Compare the test data sample to the reference data sample.

- Detect an error event by using thresholds. An error occurs if the value of the test sample is not the same as reference sample in the threshold detection, otherwise there is no error.
- Continue counting error events. Assume the number of error events is n , and the total number of test (or reference) data samples is N . Error probability = n/N

2. Data Normalization

To simplify detection, the parameter `ThresholdSetting` can be set to *automatically* and all detection thresholds are normalized from -1 to 1 . The threshold range results in symbol ranges that are equally spaced between -1 and $+1$. This means test and reference data must be normalized from -1 to 1 .

If `ThresholdSetting`=*manually*, test data must be adjusted in the same range of reference data.

3. Synchronizing Test and Reference Data

Test and reference data can be synchronized automatically or by the user:

Automatic Synchronization

- Set `DelayBound` > exact delay between test and reference data. Typically, set `DelayBound` = $5\times$ to $10\times$ exact delay. This approach does not require knowing the exact delay between test and reference data. Auto-synchronization estimates the delay based on calculating the cross-correlation function between the two waveforms.

User Synchronization

- Set `DelayBound` = -1 .
- Find the exact delay between test and reference data.
- Place a `Delay` component between the reference data source and the reference input pin of the `berMC` component.

Continue synchronization with these parameters.

- `Start` is used for specifying a sampling start time for the reference signal, and also for positioning test and reference samples. As mentioned, each sampling time must be at the center of each symbol. For example, for the reference signal from ADS data source if `DelayBound` = -1 , `Start` must be set to `Delay+SymbolTime/2`, where the exact delay between test and reference data

is Delay, and symbol time is SymbolTime. Otherwise, Start must be set to SymbolTime/2.

- Stop is used for setting N total test samples (or reference samples). $N = \text{Stop} / \text{SymbolTime}$. A larger N results in a more accurate error probability.
- NumThreshold is determined by signal levels; assume the signal level is M, $\text{NumThreshold} = M - 1$.
- ThresholdSetting options:
 - automatically* setting thresholds for uniform threshold interval.
 - manually* setting thresholds for uniform or non-uniform interval
- Threshold: Threshold values when user selects *manually* setting
- If berOutput=ber only, berMC provides the BER estimation value; if berOutput=ber vs time, berMC provides the BER estimation value and the BER Vs Time plot.

4. Delay between reference signal and test signal. If set DelayBound > 0, this component provides an estimation of the delay. The delay value can be found in the data display server.
5. The EstRelVariance parameter can be used to control the quality of the estimate that berMC generates (for details, refer to *PE Measurement Concepts* under the *Sinks, PE Estimator Usage* topic). If set to 0, then it is basically ignored and berMC will base its estimate on data collected from Start to Stop.

When EstRelVariance is set to a positive value (≤ 1), then berMC will base its estimate on data collected from Start until the estimate's relative variance drops below EstRelVariance or until Stop is reached (in this case Stop provides an upper bound on how long the simulation should run). In this mode of operation, and since the exact time the simulation will finish is unknown, berMC reports (on the status window) the relative variance of its current estimate every time 1000 symbols are processed.

Note that the equation for the estimation relative variance given (in *PE Measurement Concepts* under the *Sinks, PE Estimator Usage* topic) assumes that the errors happen randomly (as in the case of an AWGN channel) and not in bursts (as in the case of a fading channel).

6. Symbol Error Probability and Bit Error Probability

berMC can measure the symbol error probability or symbol error rate (SER) for a single channel. For a system with only one channel, such as PAM or BPSK, the bit error probability or bit error rate (BER) can be calculated by

$$\text{BER} = \text{SER} / L$$

where L is the number of bits per symbol (for example, for a BPSK system L=1, for an 8-level PAM system L=3). Note that the above formula assumes that Gray coding was used to map the bits to symbols.

For systems with I and Q channels, such as QAM, QPSK, $\pi/4$ -DQPSK, the user must place two berMC components to measure the SER for each channel, SER_i and SER_q. The SER for the whole system can then be calculated by

$$\text{SER} = \text{SER}_i + \text{SER}_q - \text{SER}_i \times \text{SER}_q$$

This formula assumes that both berMC components used to measure SER_i and SER_q processed the same number of symbols in the simulation (see note 7 on how to make sure this is true).

To calculate the BER for a system with two channels, first calculate the BER for each channel BER_i and BER_q using the equations

$$\text{BER}_i = \text{SER}_i / L \text{ and } \text{BER}_q = \text{SER}_q / L$$

where L is now the number of bits per symbol in each channel (for example, for QPSK L=1, for 16-QAM L=2, for 64-QAM L=3). Again Gray coding is assumed in the mapping of bits to symbols. (Note that this does not work for systems with non-rectangular constellations such as 8-PSK, 32-QAM, 128-QAM. berMC cannot be used to measure SER or BER for these systems.) After BER_i and BER_q have been calculated, the BER for the whole system is given by

$$\text{BER} = (\text{BER}_i + \text{BER}_q) / 2$$

7. As mentioned in note 6, for systems with two channels it is important that both berMC components used to measure the SER for each channel process the same number of symbols. If this is not true, the equations given in note 6 to determine the SER and BER of the whole system from the SER_i and SER_q will give biased results.

There are two approaches to make both berMC sinks process the same number of symbols during the simulation.

- if EstRelVariance is not used (it is set to 0) set the Start parameters of both sinks to the same value and the Stop parameters of both sinks to the same value. The ControlSimulation parameter must be set to YES for both sinks.

- if EstRelVariance is used (it is set to a positive value ≤ 1), then set the Start parameters for both sinks to the same value and make one sink to control the simulation and the other sink not to control the simulation.

For example, if you want a relative variance of 0.01, maximum symbols processed 1.0e6, and start processing data after 8.5 msec, then, assuming that symbol time is 1 msec:

Parameters of the first sink must be set as follows:

Start = 8.5 msec
Stop = 8.5 msec + 1.0e6 \times 1 msec
ControlSimulation = YES
SymbolTime = 1 msec
EstRelVariance = 0.01

Parameters of the second sink must be set as follows:

Start = 8.5 msec
Stop = 8.5 msec + 1.0e6 \times 1 msec
(not used when ControlSimulation=NO)
ControlSimulation = NO
SymbolTime = 1 msec
EstRelVariance = 0.01
(not used when ControlSimulation=NO)

Depending on the order the scheduler decides to fire the two sinks the sink that controls the simulation may end up processing one symbol more than the other one. This is insignificant and will not bias the results obtained with the equations given in note 6.

8. Lengthy BER simulation times can be significantly shortened by using the *Parallel BER* option in the *Simulate > Simulation Setup* window > *Parallel* tab. (To use this option, the user's machine must have LSF client installed and it must be connected to an LSF cluster.) In the *Parallel* tab *Simulation Mode*, choose *Parallel Hosts* to activate the *Parallel BER* field.

When the *Parallel BER* option is selected a value should also be entered in the *Number of Partitions* field. This value defines the number of parallel simulations that LSF will spawn. It is also the expected simulation time speedup factor assuming that there are at least this many machines in the LSF cluster and that the machines are equivalent in terms of processing power.

The BER results from a simulation that run using the *Parallel BER* option will NOT be identical to the results if the same simulation was run on a single machine. However, the two results are statistically equivalent.

- For a demonstration of this equivalence and some simulation performance speedup results refer to the design in *Example Project > Com_Sys/ParallelBER_prj > ParallelBER_Example1* and the corresponding data display file.
- For an example of a swept Eb/No BER simulation using the *Parallel BER* option refer to the design in *Example Project > Com_Sys/ParallelBER_prj > ParallelBER_Example2*.

Please note that, although there is no upper limit to the value one can enter in the *Number of Partitions* field, large values for this parameter can lead to wrong results. Consider the following scenario.

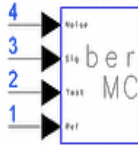
A design is set up for a BER measurement that will simulate 1e6 bits. Let's assume that the time period corresponding to 1e6 bits is 1 sec. If this simulation is split over 100 machines using the *Parallel BER* option then each machine will simulate a time period of approximately 10 msec.

If the goal of the simulation was to observe the effect on BER of fading with fade rate 100 Hz, then the results from the simulation using the *Parallel BER* option will most likely be wrong. The reason is that in a fading environment with a fade rate of 100 Hz, a fade in the signal power occurs every 10 msec on the average. In the single simulation setup, the machine will simulate a time period of 1 sec over which approximately 100 fades will occur. However, in the *Parallel BER* simulation a fade may not occur at all since each machine will simulate a time period of approximately 10 msec. A value of 10 for *Number of Partitions* would be a more appropriate choice for this scenario.

The conclusion is that when selecting a value for *Number of Partitions* the user must ensure that the phenomena he/she is trying to observe will occur in the reduced time period simulated by each of the parallel simulations LSF will spawn.

9. For general information regarding sinks, refer to [“Introduction” on page 1-1](#).

berMC4



Description Error Probability measurement using Monte Carlo Method, 4 inputs

Library Sinks

Class TSDF_berMC4

Derived From _berMC

Parameters

Name	Description	Default	Unit	Type	Range
Plot	if simulation is setup to open data display after simulation and if Plot is not set to 'None', then plot the data for this sink: None, Rectangular	None		enum	
RLoad	Load resistance. DefaultRLoad will inherit from the DF controller.	DefaultRLoad	Ohm	real	(0, ∞)
RTemp	Resistor physical temperature, in degrees C. DefaultRTemp will inherit from the DF controller.	DefaultRTemp	Celsius	real	[-273.15, ∞)
Start	Start time for data recording. DefaultTimeStart will inherit from the DF Controller.	DefaultTimeStart	sec	real	[0, ∞)
Stop	Stop time for data recording. DefaultTimeStop will inherit from the DF Controller.	DefaultTimeStop	sec	real	[Start, ∞)
ControlSimulation	if set to YES, 'Stop' time determines how long the simulation will run: NO, YES	YES		enum	
SymbolTime	symbol time	1.0	sec	real	[TStep, ∞)†
EstRelVariance	estimation relative variance (if set to 0.0 simulation will run until Stop is reached)	0.0		real	[0, 1]

Sinks

Name	Description	Default	Unit	Type	Range
NumThreshold	number of threshold	1		int	[1, ∞)
ThresholdSetting	type of threshold settings: automatically, manually	automatically		enum	
Threshold	threshold values when user selects manually setting	0		real array	
DelayBound	upper bound of delay for Synchronizing inputs; if DelayBound = -1, the Synchronizer is turned off	-1.0	sec	real	{-1} or (0, ∞) ^{††}
berOutput	Type of ber output: ber vs time, ber vs time every 10 symbols, ber vs time every 100 symbols, ber vs time every 1000 symbols, ber only	ber only		enum	

[†] TStep is the simulation time step for the component input signals.
^{††} if DelayBound is set to -1, berMC will assume test and reference signals are already synchronized.

Pin Inputs

Pin	Name	Description	Signal Type
1	input	timed sink input signal	timed
2	input2	test signal input	timed
3	input3	receiving signal input	timed
4	input4	receiving noise input	timed

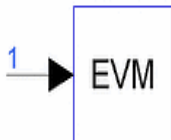
Notes/Equations

1. berMC4 works the same way as berMC and provides all features that berMC provides. It also provides the feature of measuring the Es/No (energy per symbol to noise power spectral density ratio).

To measure Es/No, connect the pure signal (before noise is added) to berMC4 input pin 3 and the noise signal to input pin 4. This is only possible if the noise is generated by a single Noise source on the schematic and SummerRF is used to combine the signal and the noise. If the noise is generated in more than one place, for example, by setting resistor temperatures to values > -273.15, by setting NoiseFigure parameters of GainRF and/or MixerRF components, by setting the Loss parameter of filters to non-zero values, then it is more complicated if not impossible to separate the signal and the noise.

2. The E_s term is calculated by measuring the power of the signal at input 3 and multiplying it by `SymbolTime`. The N_0 term is calculated by measuring the power of the signal at input 4 and dividing it by the simulation bandwidth ($1/TStep$ for RF signals, $1/(2 \times TStep)$ for baseband signal; `TStep` is the simulation time step).
3. For general information regarding sinks, refer to [“Introduction” on page 1-1](#).

EVM



Description Error Vector Magnitude Measurement

Library Sinks

Class TSDF_EVM

Derived From baseAnalysis

Parameters

Name	Description	Default	Unit	Type	Range
Plot	if simulation is setup to open data display after simulation and if Plot is not set to 'None', then plot the data for this sink: None, Rectangular	None		enum	
RLoad	Load resistance. DefaultRLoad will inherit from the DF controller.	DefaultRLoad	Ohm	real	(0, ∞)
RTemp	Resistor physical temperature, in degrees C. DefaultRTemp will inherit from the DF controller.	DefaultRTemp	Celsius	real	[-273.15, ∞)
Start	Start time for data recording. DefaultTimeStart will inherit from the DF Controller.	DefaultTimeStart	sec	real	[0, ∞)
SymTime	symbol duration time	1e-3	sec	real	[TStep, ∞)†
SymBurstLen	burst length in number of symbols	100		int	[1, ∞)
MeasType	options for measurement type: EVM RMS, EVM Peak, C0, Frequency error, Droop, Error sequence, Magnitude error sequence, Phase error sequence, Sampled data	EVM RMS		enum	

Name	Description	Default	Unit	Type	Range
ModType	options for modulation type of input signal: BPSK, QPSK, HPSK, PI/4 DQPSK, PSK8, PSK16, QAM4, QAM16, QAM32, QAM64, QAM128, QAM256, PAM4, PAM8, User defined	QPSK		enum	
Constellation	used if ModType="User defined" to define constellation as an array of complex values (re, im)	(1,1) (1,-1) (-1,-1) (-1,1)		complex array	
OptimizeSamplingInstant	if YES, optimal sampling instant will be automatically found: NO, YES	YES		enum	
† TStep is the simulation time step for the component input signal.					

Pin Inputs

Pin	Name	Description	Signal Type
1	input	timed sink input signal	timed

Notes/Equations

1. Error vector magnitude measurements are used to evaluate the modulation accuracy of modulators. It is used, for example, in the IS-54 TDMA digital cellular to set the minimum specifications for modulation accuracy of $\pi/4$ -DQPSK modulators.

The defining equations for the EVM measurement follow the definition in the *EIA/TIA IS-54-B TDMA Cellular System Dual-Mode Mobile Station-Base Station Compatibility Standard, section 2.1.3.3.1.3.3 (Error Vector Magnitude Requirement)*. Let $Z(k)$ denote the actual complex vectors (I and Q) produced by observing the real transmitter through an ideal receiver filter at instants k , one symbol period apart. $S(k)$ is defined as the ideal reference symbol (normalized such that its maximum energy symbol falls on the unit circle). Then, $Z(k)$ is modeled as:

$$Z(k) = [C_0 + C_1(S(k) + E(k))]W^k$$

where

$W = e^{Dr+jDa}$, accounts for both a frequency offset (Da radians/symbol phase rotation) and an amplitude change rate (of Dr nepers/symbol)

C_0 is a complex constant origin offset

C_1 is a complex constant representing the arbitrary phase and output power of the transmitter, and

$E(k)$ is the residual vector error on sample $S(k)$

The sum square error vector is

$$\sum_{k=0}^{N-1} |E(k)|^2 = \sum_{k=0}^{N-1} \left| \frac{[Z(k)W^{-k} - C_0]}{C_1} - S(k) \right|^2$$

where N is equal to `SymBurstLen` and C_0 , C_1 , W are chosen such as to minimize the above expression.

EVM (rms) is defined to be the rms value of $|E(k)|$ normalized by the rms value of $|S(k)|$. Therefore,

$$EVM(rms) = \frac{\sqrt{\frac{1}{N} \cdot \sum_{k=0}^{N-1} |E(k)|^2}}{\sqrt{\frac{1}{N} \cdot \sum_{k=0}^{N-1} |S(k)|^2}} = \frac{\sqrt{\sum_{k=0}^{N-1} |E(k)|^2}}{\sqrt{\sum_{k=0}^{N-1} |S(k)|^2}}$$

The symbol EVM at symbol k is defined as

$$EVM(k) = \frac{|E(k)|}{\sqrt{\frac{1}{N} \cdot \sum_{k=0}^{N-1} |S(k)|^2}}$$

which is the vector error magnitude at symbol k normalized by the rms value of $|S(k)|$.

2. The `MeasType` parameter determines the output of the component.

- If `MeasType = EVM RMS`, the output is the $EVM(rms)$ value as defined in the equations of note 1.
- If `MeasType = EVM Peak`, the output is $\max\{EVM(k)\}$, where the maximum is evaluated over $k = 0, 1, \dots, N-1$.
- If `MeasType = C0`, the output is

$$20 \times \log_{10} \left(\frac{|C_0|}{RMS(|S(k)|)} \right)$$

- If MeasType = Frequency error, the output is

$$(Da)/(2 \times \pi \times SymTime)$$

- If MeasType = Droop, the output is

$$(-20 \times \log_{10}(e^{Dr}))$$

- If MeasType = Error sequence, the output is the sequence

$$\frac{Re\{E(k)\} + j \times Im\{E(k)\}}{RMS(|S(k)|)}$$

- If MeasType = Magnitude Error sequence, the output is the magnitude error sequence. For the definition of magnitude error, see [Figure 1-5](#).
- If MeasType = Phase Error sequence, the output is the phase error sequence. For the definition of phase error, see [Figure 1-5](#).
- If MeasType = Sampled data, the output is the downsampled (1 sample per symbol period) input signal.

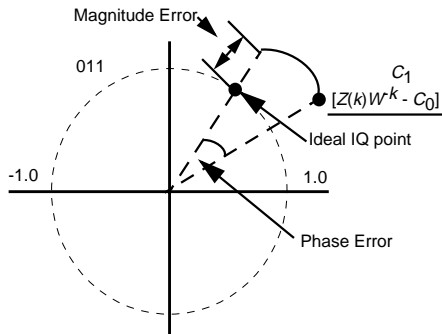


Figure 1-5. Magnitude and Phase Error

3. The ModType parameter is used to select one of the pre-defined or a user-defined signal constellation. All signal constellations (including user-defined) are normalized so that the maximum magnitude constellation point lies on the unit circle. For example, for a QPSK constellation, all four IQ points lie on the unit circle.

For a PAM-type signal (includes BPSK, 4-PAM, 8-PAM) the Q-channel of the signal is set to 0.0.

Figure 1-6 shows a few examples of the IQ constellation sets used in the EVM measurement.

The HPSK option is for an HPSK signal that has equal gains applied to the I and Q channels resulting in a 4-point constellation. In effect, this is the same as a QPSK constellation. If the gains applied to the I and Q channels are different, the resulting constellation has 8 points and using the HPSK option will give the wrong results; in this case, the *User defined* option should be used, which allows the user to specify an arbitrary constellation in the Constellation parameter.

4. The Start and OptimizeSamplingInstant parameters define how the input signal is downsampled. Note that the EVM algorithm operates on the downsampled input signal.

If OptimizeSamplingInstant is set to NO, the input signal is downsampled at one sample per symbol period starting at Start. If Start is not an exact multiple of the simulation time step, interpolation is used to find the input signal values in between the available samples.

If OptimizeSamplingInstant is set to YES, the input signal is downsampled at one sample per symbol period starting at StartDownSampling, where StartDownSampling is swept from Start to (Start + SymTime) with a step of (simulation time step) / 10. This way the optimal sampling instant can be found. The optimal value for StartDownSampling is also displayed on the status window and can be used in subsequent simulations as the value of Start with OptimizeSamplingInstant set to NO. This will result in faster simulations and it will give the correct results as long as no parameter that may affect the optimal sampling instant (for example the delay of a filter) is changed. If such a parameter changes, then OptimizeSamplingInstant should be set to YES.

Another way to find when the optimal sampling instant is without setting the OptimizeSamplingInstant parameter to YES is to observe the eye diagram and find where its maximum opening occurs.

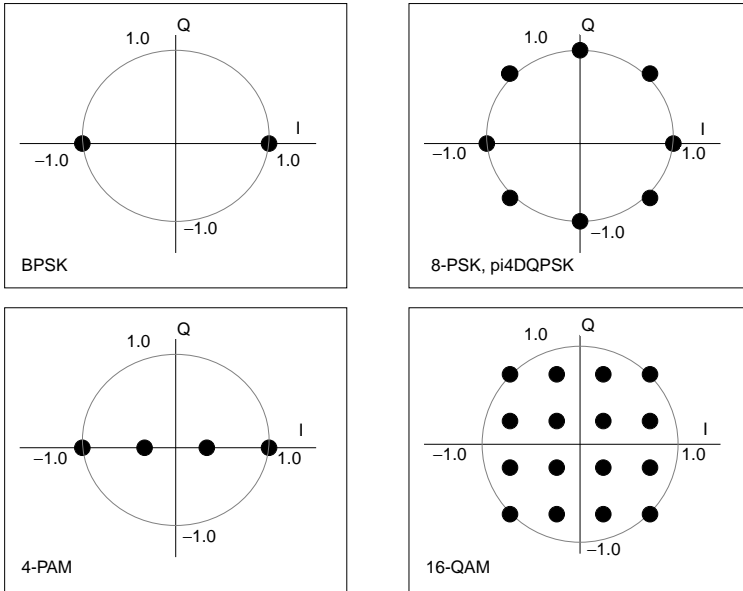
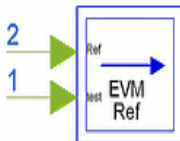


Figure 1-6. Examples of IQ-Constellations as Used in EVM Measurement

5. For general information regarding sinks, refer to [“Introduction” on page 1-1](#).

EVM_WithRef



Description EVM measurement with reference signal input
Library Sinks

Parameters

Name	Description	Default	Sym	Unit	Type	Range
StartSym	start symbol	20			int	[0, ∞)
SymBurstLen	number of symbols within burst to be measured	100	N		int	[1, 10000]
SampPerSym	number of samples per symbol	16			int	[1, ∞)
SymDelayBound	upper bound of delay detection, in symbols, -1 for no detection	3			int	[-1, ∞)
NumBursts	number of bursts to be measured	5			int	[1, ∞)
MeasType	type of measurement: EVM_rms, EVM_peak, EVM_95th_percentile	EVM_rms			enum	
SymbolRate	symbol rate	1e3		Hz	real	(0, ∞)

Pin Inputs

Pin	Name	Description	Signal Type
1	testDataInput	test signal for EVM measurement	complex
2	RefDataInput	reference signal for EVM measurement	complex

Notes/Equations

1. EVM_WithRef performs the error vector magnitude measurement on the signal applied at its testDataInput pin using the signal applied at the RefDataInput pin as a reference. The fact that a signal is used as a reference makes this

component modulation independent, that is it can be used to measure EVM for any modulation format.

2. EVM measurements are used to evaluate the modulation accuracy of modulators. For example, in the IS-54 TDMA digital cellular, they are used to set the minimum specifications for the accuracy of $\pi/4$ -DQPSK modulators.

Let $Z(k)$ denote the actual complex vectors (I and Q) produced by observing the real transmitter through an ideal receiver filter at instants k , one symbol period apart. Let $S(k)$ denote the ideal reference symbol. Then, $Z(k)$ is modeled as:

$$Z(k) = \{ C_0 + C_1 \times [S(k) + E(k)] \} \times W^k, 0 \leq k \leq N-1$$

where

$W = e^{Dr + jDa}$ accounts for both a frequency offset (Da radians per symbol phase rotation) and an amplitude change rate (Dr nepers per symbol)

C_0 is a complex constant origin offset

C_1 is a complex constant representing the arbitrary phase and output power of the transmitter

$E(k)$ is the residual vector error on sample $S(k)$

The sum square error vector is

$$\sum_{k=0}^{N-1} |E(k)|^2 = \sum_{k=0}^{N-1} \left| \frac{[Z(k)W^{-k} - C_0]}{C_1} - S(k) \right|^2$$

where C_0 , C_1 , W are chosen such as to minimize the above expression.

EVM (rms) is defined to be the rms value of $|E(k)|$ normalized by the rms value of $|S(k)|$. Therefore,

$$EVM(rms) = \frac{\sqrt{\frac{1}{N} \times \sum_{k=0}^{N-1} |E(k)|^2}}{\sqrt{\frac{1}{N} \times \sum_{k=0}^{N-1} |S(k)|^2}} = \frac{\sqrt{\sum_{k=0}^{N-1} |E(k)|^2}}{\sqrt{\sum_{k=0}^{N-1} |S(k)|^2}}$$

The symbol EVM at symbol k is defined as

$$EVM(k) = \frac{|E(k)|}{\sqrt{\frac{1}{N} \times \sum_{k=0}^{N-1} |S(k)|^2}}$$

which is the vector error magnitude at symbol k normalized by the rms value of $|S(k)|$.

3. The MeasType parameter determines the output of the component.

- If MeasType = EVM_rms, the output is the EVM(rms) value as defined in the equations of note 2.
- If MeasType = EVM_peak, the output is $\max\{EVM(k)\}$, where the maximum is evaluated over $k = 0, 1, \dots, L-1$.
- If MeasType = EVM_95th_percentile, the output is the 95th percentile of the EVM(k) values. The 95th percentile of the EVM(k) values is the value that is greater than 95% of the EVM(k) values and less than 5% of the EVM(k) values.

In addition the following quantities are displayed on the status window

- $20 \times \log_{10}\left(\frac{|C0|}{RMS(|S(k)|)}\right)$, origin offset expressed in dB with respect to the rms value of $|S(k)|$.
 - $(Da \times SymbolRate)/(2 \times \pi)$, frequency error.
 - $-20 \times \log_{10}(e^{Dr})$, droop expressed in dB with respect to 1.
4. The StartSym parameter can be used to set the symbol at which data collection for the EVM measurement starts. The primary use of this parameter is to exclude antisocial symbols (during filter transients) so that they do not affect the measurement results.
5. The SymBurstLen parameter defines the burst size on which the measurement is performed.
6. The SymDelayBound parameter is used for synchronizing the test and reference signals. Since the two signals go through different paths the test signal will most likely be delayed with respect to the reference signal. If SymDelayBound is set to a value greater than 0, then the component will cross-correlate the two signals trying to detect the delay between them. However, only delays between 0 and SymDelayBound are considered. Note that SymDelayBound is in number of symbols, that is if an upper bound of the delay

is 3 symbols, SymDelayBound should be set to 3 and not to $3 \times \text{SampPerSym}$. If SymDelayBound is set to -1 then no synchronization is performed.

7. Most standards specify that the EVM measurement should be performed over multiple bursts and the results should be averaged. This functionality is provided by the NumBursts parameter. If NumBursts is set to a value greater than 1 then the EVM measurement is performed over NumBursts consecutive bursts and the results of the individual measurements are averaged. Averaging is applied to both the values sent to the dataset and the ones displayed on the status window.
8. The SymbolRate parameter is only used to calculate the frequency error based on the formula given in note 3. It does not affect the EVM measurement in any other way.
9. For general information regarding sinks, refer to [“Introduction” on page 1-1](#).

NumericSink



Description Numeric Data Sink

Library Sinks

Class SDFNumericSink

Derived From NumericSinkGated

Parameters

Name	Description	Default	Type	Range
Plot	If simulation is setup to open data display after simulation and if Plot is not set to 'None', then plot the data for this sink: None, Rectangular	None	enum	
Start	Sample number to start collecting data. DefaultNumericStart will inherit from the DF controller.	DefaultNumericStart	int	[0, ∞)
Stop	Sample number to stop collecting data. DefaultNumericStop will inherit from the DF controller.	DefaultNumericStop	int	[Start, ∞)
ControlSimulation	If set to YES, 'Stop' sample number determines how long the simulation will run: NO, YES	YES	enum	

Pin Inputs

Pin	Name	Description	Signal Type
1	input	input signal	multiple anytype

Notes/Equations

1. NumericSink collects data from the output of the connected component and saves it to the simulation dataset.

When the connected component is a numeric component, the collected data is in the form of integer, real, or complex data values versus an integer index value.

When the connected component is a timed component, the collected data is in the form of complex data values (for baseband timed signals the imaginary part is set to 0) versus an integer index value. The collected data will not have information on the characterization frequency of the timed signal and it will not preserve the timestamps.

2. The NumericSink component can record data of all the following data types: integer scalar, floating-point (real) scalar, fix scalar, complex scalar, timed scalar, integer matrices, floating-point (real) matrices, fix matrices, complex matrices, integer busses, floating-point (real) busses, fix busses, and complex busses. All types of fix data are converted to the corresponding floating-point (real) type before being recorded.

Busses of timed data, busses of different data types, and busses of matrices (integer, floating-point (real), fix, or complex) are not supported. In addition, swept matrix dimensions and/or swept bus sizes are not supported.

3. The name of the variable (holding the simulation data) that is created in the dataset by each NumericSink is the same as the sink's instance name. For bus or matrix data the variable with the sink's instance name represents the whole bus or matrix structure. The individual bus or matrix elements are also available for plotting or use in AEL expressions and meas equations. For example, a matrix of size 2×3 connected to a NumericSink named MyMatrix will result in the following variables being listed in the set of variables available for plotting in the data display window: MyMatrix, MyMatrix(1,1), MyMatrix(1,2), MyMatrix(1,3), MyMatrix(2,1), MyMatrix(2,2), MyMatrix(2,3).

When iterated NumericSink instances are used (for example, N<1:3>) the names of the instantiated sinks will contain the special characters < and > (for example, N<1>, N<2>, and N<3>). These characters have special meaning and they must be quoted when used in AEL expressions, meas equations, or as the goal expression in an optimization simulation. For AEL expressions and meas equations the correct usage is var("N<1>"); for optimization goal expressions the correct usage is var(""N<1>"") (the Expression parameter of the Goal component is a string; two double quotes are needed to represent one double quote inside a string).

4. When a bus is connected to an iterated NumericSink instance then the bus size S must be an integer multiple of the number of the NumericSink instances N , otherwise an error is reported.

- If $S = N$, then each NumericSink instance is connected to a single bus element so the data saved by each sink is scalar.
- If $S = k \times N$, where k is an integer greater than 1, then the first k bus elements are connected to the first NumericSink instance, the second k bus elements are connected to the second NumericSink instance, and so on.

In this case, each NumericSink instance is connected to a bus of size k so the data saved by each sink will have the bus format. A known problem in this case is that the individual bus elements are not listed in the set of variables available for plotting in the data display window. For example, assuming that $k = 2$ and the iterated NumericSink instance is $A\langle 1:3 \rangle$, one would expect to see $A\langle 1 \rangle$, $A\langle 1 \rangle(1)$, $A\langle 1 \rangle(2)$, $A\langle 2 \rangle$, $A\langle 2 \rangle(1)$, $A\langle 2 \rangle(2)$, $A\langle 3 \rangle$, $A\langle 3 \rangle(1)$, $A\langle 3 \rangle(2)$ listed in the set of variables available for plotting in the data display window but in fact only $A\langle 1 \rangle$, $A\langle 2 \rangle$, and $A\langle 3 \rangle$ are listed. However, one can still plot the individual bus elements by first selecting to plot the whole bus (for example, $A\langle 2 \rangle$) then manually editing the expression that is being plotted, for example, $\text{var}("A\langle 2 \rangle")(2)$ to plot the second element of the bus $A\langle 2 \rangle$.

5. The amount of data collected by a NumericSink is controlled by the Start, Stop, and ControlSimulation parameters. Data collection always begins at the input sample with an index value equal to the value of the Start parameter (index values start at 0 and increment by 1 for every sample).

- If ControlSimulation is set to YES, then data collection ends at the input sample with an index value equal to the value of the Stop parameter.
- If ControlSimulation is set to NO, then data collection ends when the simulation ends; in this case, there must be at least one other source or sink component that controls how long the simulation will run.

6. For general information regarding sinks, refer to [“Introduction” on page 1-1](#).

NumericSinkGated



Description Numeric Data Sink with Control Input

Library Sinks

Class SDFNumericSinkGated

Parameters

Name	Description	Default	Type	Range
Plot	If simulation is setup to open data display after simulation and if Plot is not set to 'None', then plot the data for this sink: None, Rectangular	None	enum	
Start	Sample number to start collecting data. DefaultNumericStart will inherit from the DF controller.	DefaultNumericStart	int	[0, ∞)
Stop	Sample number to stop collecting data. DefaultNumericStop will inherit from the DF controller.	DefaultNumericStop	int	[Start, ∞)
ControlSimulation	If set to YES, 'Stop' sample number determines how long the simulation will run: NO, YES	YES	enum	

Pin Inputs

Pin	Name	Description	Signal Type
1	control	control signal	int
2	input	input signal	multiple anytype

1. The NumericSinkGated component is similar to the NumericSink component; it gives the user more flexibility in controlling when data is collected. Refer to

documentation for “[NumericSink](#)” on page 1-48 for information about the basic functionality.

2. When the control input is unconnected then this component behaves exactly the same as NumericSink.

When the control input is connected then data is collected as described in note 5 of NumericSink documentation with the additional requirement that the control signal is not 0. To control the data collection only through the control pin set Start to 0 and ControlSimulation to NO. In this case, there must be at least one other source or sink component that controls how long the simulation runs.

The use of a control signal allows the user to capture one or more signal segments whose start and end is not known before the simulation starts but for which a binary signal can be generated that goes high when the segment starts and low when the segment ends. It can also significantly reduce the amount of collected data if the desired segments are spaced far apart from each other in a long simulation.

Printer



Description Plain output data file writer

Library Sinks

Class SDFPrinter

C++ Code

Parameters

Name	Description	Default	Type	Range
Start	sample Index to start recording data	DefaultNumericStart	int	[0, ∞)
Stop	sample Index to stop recording data	DefaultNumericStop	int	[Start, ∞)
ControlSimulation	control simulation: NO, YES	YES	enum	
FileName	output file name	print.txt	filename	

Pin Inputs

Pin	Name	Description	Signal Type
1	input		multiple anytype

Notes/Equations

1. Printer prints out one sample from each input port per line.
2. The output file will be a text file that contains data in ADS Ptolemy format specific to the input data type: real array (for floating-point (real), fixed, and integer scalar input data), complex array, string array, real matrix, integer matrix, fixed-point matrix, or complex matrix. For format information, refer to [“Understanding File Formats”](#) in the *ADS Ptolemy Simulation* manual.
3. For general information regarding sinks, refer to [“Introduction” on page 1-1](#).

Sinad



Description SINAD measurement

Library Sinks

Class TSDF_Sinad

Derived From baseAnalysis

Parameters

Name	Description	Default	Unit	Type	Range
Plot	if simulation is setup to open data display after simulation and if Plot is not set to 'None', then plot the data for this sink: None, Rectangular	None		enum	
RLoad	Load resistance. DefaultRLoad will inherit from the DF controller.	DefaultRLoad	Ohm	real	(0, ∞)
RTemp	Resistor physical temperature, in degrees C. DefaultRTemp will inherit from the DF controller.	DefaultRTemp	Celsius	real	[-273.15, ∞)
Start	Start time for data recording. DefaultTimeStart will inherit from the DF Controller.	DefaultTimeStart	sec	real	[0, ∞)
Stop	Stop time for data recording. DefaultTimeStop will inherit from the DF Controller.	DefaultTimeStop	sec	real	[Start, ∞)
SignalFrequency	signal Frequency of interest.	1.0e6	Hz	real	(0, $1/(2 * TStep)$)†
BandRejectSpan	band reject filter bandwidth.	0.0	Hz	real	[0, $1/(2 * TStep)$)††

Name	Description	Default	Unit	Type	Range
HanningWindow	windowing of signal with a Hanning window.: OFF, ON	ON		enum	
† TStep is the simulation time step for the component input signal. †† Refer to Note 3 for details regarding use of this parameter.					

Pin Inputs

Pin	Name	Description	Signal Type
1	input	timed sink input signal	timed

Notes/Equations

1. The SINAD measurement calculates the ratio $(S+N+D)/(N+D)$ in dB, where S denotes signal power (that is, the signal power centered at SignalFrequency), N denotes the noise power and D denotes distortion power (that is, the residual power not accounted for in S or N).

The $(S+N+D)$ power term is estimated from the signal over the time interval from Start to Stop.

The $(N+D)$ power term is estimated by first taking an FFT (fast Fourier transform) of the signal over the time interval from Start to Stop, applying an ideal band-reject filter (with a reject span of BandRejectSpan) centered at SignalFrequency on the FFT'd signal (in the frequency domain) and finally summing up the power of the filtered signal.

In the SINAD measurement, it is assumed that the signal is a real signal (the Q-channel information is not used).

2. SignalFrequency is the center frequency of the desired signal. For example, in a 1-tone SINAD measurement of an FM demodulator, SignalFrequency should be set equal to the frequency of the sinusoidal tone.
3. BandRejectSpan is the bandwidth of the band reject filter centered at SignalFrequency which is to be applied on the signal in order to estimate the $(N+D)$ noise plus distortion power term.

If BandRejectSpan is set equal to zero or less than $1/(\text{Stop}-\text{Start})$, (that is, $\text{BandRejectSpan} < 1/(\text{Stop}-\text{Start})$), the actual band reject span used internally in the SINAD measurement is set equal to $5/(\text{Stop}-\text{Start})$.

4. HanningWindow is used to allow the user the ability to window the signal before the estimation of the SINAD ($(S+N+D)/(N+D)$) is performed. If HanningWindow=ON, then a Hanning window is applied to the signal before the SINAD measurement is performed.

Generally, set HanningWindow=ON to avoid excessive spectral leakage problems in the computation of the FFT of the signal.

Note Due to a migration problem, when an ADS 1.5 (or earlier) design using this component is opened in a later ADS release, the HanningWindow parameter will be set to OFF no matter what the value is in the original design.

5. For general information regarding sinks, refer to [“Introduction” on page 1-1](#).

SpectrumAnalyzer



Description Spectrum analyzer
Library Sinks
Class TSDF_SpectrumAnalyzer
Derived From baseAnalysis

Parameters

Name	Description	Default	Unit	Type	Range
Plot	if simulation is setup to open data display after simulation and if Plot is not set to 'None', then plot the data for this sink: None, Rectangular	None		enum	
RLoad	Load resistance. DefaultRLoad will inherit from the DF controller.	DefaultRLoad	Ohm	real	(0, ∞)
RTemp	Resistor physical temperature, in degrees C. DefaultRTemp will inherit from the DF controller.	DefaultRTemp	Celsius	real	[-273.15, ∞)
Start	Start time for data recording. DefaultTimeStart will inherit from the DF Controller.	DefaultTimeStart	sec	real	[0, ∞)
Stop	Stop time for data recording. DefaultTimeStop will inherit from the DF Controller.	DefaultTimeStop	sec	real	[Start+16*TStep, ∞)†
Window	Window with default constant applied to collected data (default constant is used when WindowConstant is 0.0): none, Hamming 0.54, Hanning 0.50, Gaussian 0.75, Kaiser 7.865, _8510 6.0, Blackman, Blackman-Harris	none		enum	

Sinks

Name	Description	Default	Unit	Type	Range
WindowConstant	Window constant used for windows of type Hamming, Hanning, Gaussian, Kaiser, 8510	0.0		real	[0, ∞)
Bias	Spectrum normalization bias type to correct for effects of windowing and/or averaging: no bias, power	power		enum	
FStart	Start frequency for spectrum calculation	0.0	Hz	real	[0, ∞)
FStop	Stop frequency for spectrum calculation	100e9	Hz	real	(FStop, ∞)
NumFreqs	Number of frequencies uniformly spaced from FStart to FStop; NumFreqs=0 results in default frequency resolution	0		int	[0, ∞)
NPoints	Number of points to be used in a segment; NPoints=0 results in one segment of (Stop-Start)/TStep+1 points and Overlap is ignored	0		int	[0, (Stop-Start)/TStep+1)†
Overlap	Number of overlapping points between two adjacent segments	0		int	[0, NPoints - 1]

† TStep is the simulation time step for the component input signal.

Pin Inputs

Pin	Name	Description	Signal Type
1	input	timed sink input signal	timed

Notes/Equations

1. The SpectrumAnalyzer component can be used to measure the spectrum of a baseband or an RF signal. In the following notes $TStep$ will be used to denote the simulation time step and fc will be used to denote the signal characterization frequency ($fc=0$ for a baseband signal and $fc>0$ for an RF signal).

Note Information regarding time domain signal differences between ADS Ptolemy simulations and Circuit Envelope and Transient simulations is given in the “[Timed Synchronous Dataflow](#)” section in the *ADS Ptolemy Simulation* manual.

2. The component outputs the complex amplitude voltage values at the frequencies of the spectral tones. The component does not output the power at the frequencies of the spectral tones. However, one can calculate and display the power spectrum as well as the magnitude and phase spectrum by using the dBm, mag, and phase functions of the data display window. Note that the dBm function assumes a 50-ohm resistive load. If a different load was used in the simulation, its value can be specified as a second argument to the dBm function, for example, dBm(VRF, 75) where VRF is the instance name of the SpectrumAnalyzer component and 75 ohms is the resistive load used. To integrate the power spectrum over a frequency range see note 7.

Note that, for baseband signals and for the frequency of 0 Hz the dBm function returns a power value that is 3 dB less than the actual power. This is because the primary use of the dBm function is with RF signals from an Analog/RF schematic simulation, where the 0 Hz frequency corresponds to the carrier frequency and not really 0 Hz signal frequency.

If the baseband signal has no significant power at dc, this 3 dB error is insignificant and can be ignored—otherwise, it must be considered. For a baseband signal, the energy at 0 Hz is more typically measured as the average voltage value of the signal and not by power.

3. By default, the displayed spectrum extends from 0 Hz to $1/(2 \times TStep)$ Hz for a baseband signal and from $fc - 1/(2 \times TStep)$ Hz to $fc + 1/(2 \times TStep)$ Hz for an RF signal. When $fc < 1/(2 \times TStep)$, the default spectrum extends to negative frequencies. The spectral content at these negative frequencies is conjugated, mirrored, and added to the spectral content of the closest positive frequency. This way, the negative frequency tones are displayed on the positive frequency axis as would happen in an RF spectrum analyzer measurement instrument. This process may introduce an error in the displayed frequency for the mirrored tones. The absolute error introduced is less than $\delta f / 2$ (see note 5 for the definition of δf).
4. The basis of the algorithm used by SpectrumAnalyzer is the $fs()$ function (the chirp-Z transform option of $fs()$ is used). The algorithm allows $fs()$ to be called on

multiple signal segments and averages the results to achieve video averaging (see note 5). The algorithm can also scale the output spectrum to correct errors in power introduced by windowing.

For details regarding the $fs()$ function, refer to the *Measurement Expressions* manual.

5. The *Start* and *Stop* parameters define the time interval over which the measurement is performed. The number of data points collected is

$$\frac{Stop - Start}{TStep} + 1$$

If $NPoints = 0$, all collected data is processed as one segment. The assumptions made by $fs()$ lead to better results if the segment size is odd. Therefore, if the total number of data points collected is even, *Stop* is increased by *TStep*. The frequency resolution achieved in this case is $deltaf = 1/(Stop - Start)$.

If $NPoints > 0$, the collected data is broken down in segments of size *NPoints*, the spectrum of each segment is calculated and at the end all the spectra are averaged (video averaging). For the same reason explained above, if *NPoints* is even it is increased by 1 to make it odd. The frequency resolution achieved in this case is $deltaf = 1/(TStep \times NPoints)$. If *Overlap* = 0, the *Npoints*-long segments are non-overlapping, otherwise they overlap by *Overlap* points.

Since it is very unlikely that an integer number of *Npoints*-long segments that overlap by *Overlap* points will fit exactly in the interval from *Start* to *Stop*, *Stop* may be increased/decreased so that the resulting number of segments is an integer. Let *NExtra* be the number of extra points that need to be collected in order to have an integer number of segments. If $NExtra < Npoints / 2$, *Stop* is increased by $NExtra \times TStep$, otherwise it is decreased by $(NPoints - NExtra - Overlap) \times TStep$.

Figure 1-7 shows the multiple segments and video averaging concepts discussed above.

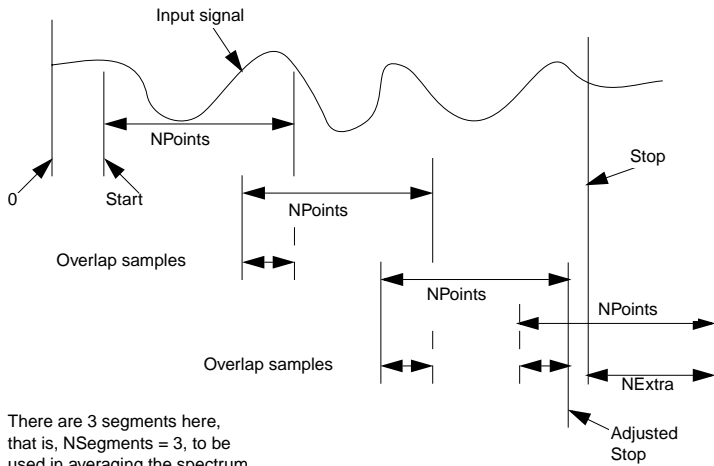


Figure 1-7. Input Signal and SpectrumAnalyzer Parameters Start, NPoints, and Overlap

6. The *Window* and *WindowConstant* parameters are used to define the window that will be applied to each segment before its spectrum is calculated. Windowing is often necessary in transform-based (chirp-Z, FFT) spectrum estimation. Without windowing, the estimated spectrum may suffer from spectral leakage that can cause misleading measurements or masking of weak signal spectral detail by spurious artifacts.

Window Type Definitions

Hanning Window:

$$w(kT_s) = \begin{cases} W - (1 - W) \cos\left(\frac{2\pi k}{NPoints}\right) & 0 \leq k \leq NPoints \\ 0.0 & otherwise \end{cases}$$

where $W = WindowConstant$

Hamming Window:

$$w(kT_s) = \begin{cases} W - (1 - W) \cos\left(\frac{2\pi k}{NPoints}\right) & 0 \leq k \leq NPoints \\ 0.0 & otherwise \end{cases}$$

where $W = WindowConstant$

Blackman Window:

$$w(kT_s) = \begin{cases} 0.42 - 0.5 \cos\left(\frac{2\pi k}{NPoints}\right) + 0.08 \cos\left(\frac{4\pi k}{NPoints}\right) & 0 \leq k \leq NPoints \\ 0.0 & otherwise \end{cases}$$

Blackman-Harris Window

$$w(kT_s) = \begin{cases} 0.36 - 0.49 \cos\left(\frac{2\pi k}{NPoints}\right) + 0.14 \cos\left(\frac{4\pi k}{NPoints}\right) - 0.0117 \cos\left(\frac{6\pi k}{NPoints}\right) & 0 \leq k \leq NPoints \\ 0.0 & otherwise \end{cases}$$

Kaiser (Kaiser-Bessel) Window:

$$w(kT_s) = \begin{cases} \frac{I_0\left(\beta \left[1 - \left(\frac{k-\alpha}{\alpha}\right)^2\right]^{1/2}\right)}{I_0(\beta)} & 0 \leq k \leq NPoints \\ 0.0 & otherwise \end{cases}$$

Here $\alpha = NPoints / 2$, β is the shape parameter (set by *WindowConstant*), and $I_0(.)$ is the 0th order modified Bessel function of the first kind.

8510 6.0 Window

This is a Kaiser window with $\beta = 6.0$

Gaussian (Weierstrass) Window:

$$w(kT_s) = \begin{cases} \exp\left(-\frac{1}{2}\left(\alpha \frac{(2k - NPoints)}{NPoints}\right)^2\right) & 0 \leq \left|k - \frac{NPoints}{2}\right| \leq \frac{NPoints}{2} \\ 0.0 & otherwise \end{cases}$$

where $\alpha = WindowConstant$.

Window Options

Available window options (some with default constants) are:

Hamming 0.54

Hanning 0.50

Gaussian 0.75

Kaiser 7.865

8510 6.0
Blackman
Blackman-Harris

The default window constants for Hamming, Hanning, Gaussian, Kaiser, and 8510 can be changed by using the *WindowConstant* parameter.

7. The windowing of a signal in time has the effect of changing its power. The *Bias* parameter can be used to compensate for this. If *Bias* is set to *no bias*, then no normalization of the spectrum is done. If *Bias* is set to *power*, then the spectrum is normalized so that the power contained in it is the same as the power of the input signal.

To calculate the power contained in a spectrum, the *spec_power()* function can be used in the data display window. The *spec_power()* function expects its input to be in dBm and returns a value in dBm. Therefore, if spectral data generated using the SpectrumAnalyzer is passed to the *spec_power()* function, the dBm function must be applied to the data before *spec_power()* is called. If frequency limits needs to be passed to *spec_power()*, specify them in Hz.

For details regarding the *spec_power()* function, refer to refer to the *Measurement Expressions* manual.

8. The *FStart* and *FStop* parameters control the frequency range over which the spectrum is calculated overriding the default values (see note 3). If the values of these parameters are outside the valid range
 - $[0, 1/(2 \times TStep)]$ for a baseband signal
 - $[fc - 1/(2 \times TStep), fc + 1/(2 \times TStep)]$ for an RF signal

FStart is forced to the lower limit of the valid range and *FStop* is forced to the upper limit of the valid range.

9. The *NumFreqs* parameter controls how many spectral tones will be calculated between *FStart* and *FStop*. The default value of 0 results in $(FStop - FStart)/(\delta f) + 1$ spectral tones being calculated (see note 5 for the definition of δf). If $(FStop - FStart)/(NumFreqs) < \delta f$ then *NumFreqs* is forced to $(FStop - FStart)/(\delta f) + 1$.
10. For general information regarding sinks, refer to [“Introduction” on page 1-1](#).

References

- [1] A. V. Oppenheim, R. W. Schaffer, *Discrete-Time Signal Processing*, Prentice Hall, 1989, Chapter 9, section 9.7.

SpectrumAnalyzerResBW



Description Spectrum analyzer with resolution bandwidth setting

Library Sinks

Class TSDF_SpectrumAnalyzerResBW

Derived From _SpectrumAnalyzer

Parameters

Name	Description	Default	Unit	Type	Range
Plot	if simulation is setup to open data display after simulation and if Plot is not set to 'None', then plot the data for this sink: None, Rectangular	None		enum	
RLoad	Load resistance. DefaultRLoad will inherit from the DF controller.	DefaultRLoad	Ohm	real	(0, ∞)
RTemp	Resistor physical temperature, in degrees C. DefaultRTemp will inherit from the DF controller.	DefaultRTemp	Celsius	real	[-273.15, ∞)
Start	Start time for data recording. DefaultTimeStart will inherit from the DF Controller.	DefaultTimeStart	sec	real	[0, ∞)
Stop	Stop time for data recording. DefaultTimeStop will inherit from the DF Controller.	DefaultTimeStop	sec	real	[Start, ∞)
Window	Window with default constant applied to collected data (default constant is 0.0): none, Hamming 0.54, Hanning 0.50, Gaussian 0.75, Kaiser 7.865, _8510 6.0, Blackman, Blackman-Harris	none		enum	
ResBW	Resolution bandwidth	30 kHz	Hz	real	[0, ∞)

Name	Description	Default	Unit	Type	Range
NumSegments	Number of segments	0		int	[0, ∞)
SegmentTime	Segment time	1.0 msec	sec	real	(0, ∞)

Pin Inputs

Pin	Name	Description	Signal Type
1	input	timed sink input signal	timed

Notes/Equations

1. The SpectrumAnalyzerResBW component can be used to measure the spectrum of a baseband or an RF signal. In the following notes, $TStep$ will be used to denote the simulation time step and fc will be used to denote the signal characterization frequency ($fc=0$ for a baseband signal and $fc>0$ for an RF signal).

Note Information regarding time domain signal differences between ADS Ptolemy simulations and Circuit Envelope and Transient simulations is given in the [“Timed Synchronous Dataflow”](#) section in the *ADS Ptolemy Simulation* manual.

2. The component outputs the complex amplitude voltage values at the frequencies of the spectral tones. The component does not output the power at the frequencies of the spectral tones. However, one can calculate and display the power spectrum as well as the magnitude and phase spectrum by using the dBm, mag, and phase functions of the data display window. Note that the dBm function assumes a 50-ohm resistive load. If a different load was used in the simulation, its value can be specified as a second argument to the dBm function, for example, dBm(VRF, 75) where VRF is the instance name of the SpectrumAnalyzerResBW component and 75 ohms is the resistive load used. To integrate the power spectrum over a frequency range see note 7.

Note that, for baseband signals and for the frequency of 0 Hz the dBm function returns a power value that is 3 dB less than the actual power. This is because the primary use of the dBm function is with RF signals from an Analog/RF schematic simulation, where the 0 Hz frequency corresponds to the carrier frequency and not to the 0 Hz signal frequency.

If the baseband signal has no significant power at dc, this 3 dB error is insignificant and can be ignored—otherwise, it must be considered. For a baseband signal, the energy at 0 Hz is more typically measured as the average voltage value of the signal and not by power.

3. The displayed spectrum extends from 0 Hz to $1/(2 \times TStep)$ Hz for a baseband signal and from $fc - 1/(2 \times TStep)$ Hz to $fc + 1/(2 \times TStep)$ Hz for an RF signal. When $fc < 1/(2 \times TStep)$, the default spectrum extends to negative frequencies. The spectral content at these negative frequencies is conjugated, mirrored, and added to the spectral content of the closest positive frequency. This way, the negative frequency tones are displayed on the positive frequency axis as would happen in an RF spectrum analyzer measurement instrument. This process may introduce an error in the displayed frequency for the mirrored tones. The absolute error introduced is less than the frequency step in the displayed spectrum (see note 5 for the more details about frequency step in the displayed spectrum).
4. The basis of the algorithm used by SpectrumAnalyzerResBW is the $fs()$ function (the chirp-Z transform option of $fs()$ is used). Depending on the values of the parameters Start, ResBW, and NumSegments, the algorithm may call $fs()$ on multiple signal segments and average the results to achieve video averaging (see note 5 for more details).

For details regarding the $fs()$ function, refer to the *Measurement Expressions* manual.

5. Depending on the values of the parameters ResBW and NumSegments the spectrum measurement will be performed in a different way. There are four modes of operation that are described:
 - ResBW > 0, NumSegments = 0

This is the default mode of operation. In this mode, the input signal (in the time interval [Start, Stop]) is broken down in multiple segments of length $NENBW / ResBW$, where NENBW is the normalized equivalent noise bandwidth for the selected window (see note 8 for the definition of NENBW). The spectra of the individual segments are calculated and averaged (video averaging). If the interval [Start, Stop] is not long enough, that is $Stop - Start < NENBW / ResBW$, then Stop is increased so that one segment of length $NENBW / ResBW$ is processed. If $Stop - Start > NENBW / ResBW$ but the interval [Start, Stop] does not contain an integer multiple of $NENBW / ResBW$ long segments, Stop is again adjusted (increased) so that an integer number of $NENBW / ResBW$ long segments is processed.

The resolution bandwidth achieved is ResBW. The frequency step in the displayed spectrum is also ResBW.

- ResBW > 0, NumSegments > 0

This mode of operation is very similar to the first one. The only difference is that the value of the Stop parameter is ignored and Stop is set to Start + NumSegments x SegmentTime. Stop may then be further adjusted as explained in the first mode of operation.

- ResBW = 0, NumSegments = 0

In this mode of operation, data is collected from Start to Stop and the spectrum analysis is performed on the whole data treated as one segment. This achieves the highest possible resolution bandwidth for the collected data given by $NENBW / (Stop - Start)$, where NENBW is the normalized equivalent noise bandwidth for the selected window (see note 8 for the definition of NENBW). The frequency step in the displayed spectrum is $1 / (Stop - Start)$. No video averaging occurs in this mode of operation.

- ResBW = 0, NumSegments > 0

In this mode of operation, data is collected from Start to Start + NumSegments x SegmentTime. The data is broken down in NumSegments segments of length SegmentTime. The spectra of the individual segments are calculated and averaged (video averaging). The resolution bandwidth achieved is $NENBW / SegmentTime$, where NENBW is the normalized equivalent noise bandwidth for the selected window (see note 8 for the definition of NENBW). The frequency step in the displayed spectrum is $1 / SegmentTime$. Video averaging occurs if NumSegments > 1.

6. The *Window* parameter is used to define the window that will be applied to each segment before its spectrum is calculated. Windowing is often necessary in transform-based (chirp-Z, FFT) spectrum estimation. Without windowing, the estimated spectrum may suffer from spectral leakage that can cause misleading measurements or masking of weak signal spectral detail by spurious artifacts.

Window Definitions

- none

$$w(kT_s) = \begin{cases} 1.0 & 0 \leq k \leq N \\ 0.0 & \textit{otherwise} \end{cases}$$

where N is the window size

- **Hamming 0.54**

$$w(kT_s) = \begin{cases} 0.54 - 0.46 \cos\left(\frac{2\pi k}{N}\right) & 0 \leq k \leq N \\ 0.0 & \textit{otherwise} \end{cases}$$

where N is the window size

- **Hanning 0.50**

$$w(kT_s) = \begin{cases} 0.5 - 0.5 \cos\left(\frac{2\pi k}{N}\right) & 0 \leq k \leq N \\ 0.0 & \textit{otherwise} \end{cases}$$

where N is the window size

- **Gaussian 0.75**

$$w(kT_s) = \begin{cases} \exp\left(-\frac{1}{2}\left(0.75\frac{(2k-N)}{N}\right)^2\right) & 0 \leq \left|k - \frac{N}{2}\right| \leq \frac{N}{2} \\ 0.0 & \textit{otherwise} \end{cases}$$

where N is the window size

- **Kaiser 7.865**

$$w(kT_s) = \begin{cases} \frac{I_0\left(7.865\left[1 - \left(\frac{k-\alpha}{\alpha}\right)^2\right]^{1/2}\right)}{I_0(7.865)} & 0 \leq k \leq N \\ 0.0 & \textit{otherwise} \end{cases}$$

where N is the window size, $\alpha = N/2$, and $I_0(\cdot)$ is the 0th order modified Bessel function of the first kind

- **8510 6.0 (Kaiser 6.0)**

$$w(kT_s) = \begin{cases} \frac{I_0\left(6.0\left[1 - \left(\frac{k-\alpha}{\alpha}\right)^2\right]^{1/2}\right)}{I_0(6.0)} & 0 \leq k \leq N \\ 0.0 & \textit{otherwise} \end{cases}$$

where N is the window size, $\alpha = N/2$, and $I_0(\cdot)$ is the 0th order modified Bessel function of the first kind

- Blackman

$$w(kT_s) = \begin{cases} 0.42 - 0.5 \cos\left(\frac{2\pi k}{N}\right) + 0.08 \cos\left(\frac{4\pi k}{N}\right) & 0 \leq k \leq N \\ 0.0 & \textit{otherwise} \end{cases}$$

where N is the window size

- Blackman-Harris

$$w(kT_s) = \begin{cases} 0.35875 - 0.48829 \cos\left(\frac{2\pi k}{N}\right) + 0.14128 \cos\left(\frac{4\pi k}{N}\right) - 0.01168 \cos\left(\frac{6\pi k}{N}\right) & 0 \leq k \leq N \\ 0.0 & \textit{otherwise} \end{cases}$$

where N is the window size

7. The windowing of a signal in time has the effect of changing its power. SpectrumAnalyzerResBW normalizes the measured spectrum so that the power contained in it is the same as the power of the input signal.

To calculate the power contained in a spectrum, the `spec_power()` function can be used in the data display window. The `spec_power()` function expects its input to be in dBm and returns a value in dBm. Therefore, if spectral data generated using SpectrumAnalyzerResBW is passed to the `spec_power()` function, the dBm function must be applied to the data before `spec_power()` is called. If frequency limits need to be passed to `spec_power()`, they must be specified in Hz.

For details regarding the `spec_power()` function, refer to the *Measurement Expressions* manual.

8. The windowing of a signal in time also affects the resolution bandwidth that can be achieved. When calculating the spectrum of a signal segment the resolution bandwidth achieved with a window is always lower (worse) than the resolution bandwidth achieved without a window. The normalized equivalent noise bandwidth (NENBW) of a window is one measure of how much it reduces the resolution bandwidth that can be achieved. To compensate for this a longer signal segment should be processed.

Equivalent noise bandwidth (ENBW) compares the window to an ideal, rectangular filter. It is the equivalent width of a rectangular filter that passes

the same amount of white noise as the window. The normalized ENBW (NENBW) is the ENBW multiplied by the time duration of the of the signal being windowed. NENBW values for windows available in SpectrumAnalyzerResBW are listed in [Table 1-3](#).

Table 1-3. Normalized Equivalent Noise Bandwidth of Available Windows

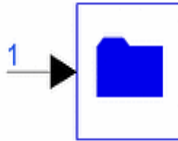
Window	NENBW
none	1
Hamming 0.54	1.363
Hanning 0.50	1.5
Gaussian 0.75	1.883
Kaiser 7.865	1.653
8510 6.0	1.467
Blackman	1.727
Blackman-Harris	2.021

9. For general information regarding sinks, refer to [“Introduction” on page 1-1](#).

References

- [1] A. V. Oppenheim, R. W. Schaffer, *Discrete-Time Signal Processing*, Prentice Hall, 1989, Chapter 9, section 9.7.

TimedDataWrite



Description **Format Specific Output Data File** Library **Sinks**

Parameters

Name	Description	Default	Unit	Type	Range
Start	start time for data recording. DefaultTimeStart will inherit from the DF Controller.	DefaultTimeStart	sec	real	[0, ∞)
Stop	stop time for data recording. DefaultTimeStop will inherit from the DF Controller.	DefaultTimeStop	sec	real	[Start, ∞)
RLoad	load resistance. DefaultRLoad will inherit from the DF controller.	DefaultRLoad	Ohm	real	(0, ∞)
RTemp	physical temperature, in degrees C. DefaultRTemp will inherit from the DF controller.	DefaultRTemp	Celsius	real	[-273.15, ∞)
ControlSimulation	if set to YES, 'Stop' time determines how long the simulation will run: NO, YES	YES		enum	
FileName	filename that you assign before beginning signal analysis to create a data file. filename must begin with a letter and cannot exceed 32 characters; choose format using <i>.tim</i> , <i>.bintim</i> , <i>.ascig</i> , <i>sig</i> extension. During analysis, the file is generated and saved to the <i>data</i> subdirectory under your current project (<i>._prj</i>) directory.			filename	

Pin Inputs

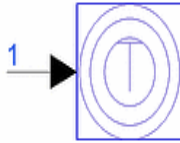
Pin	Name	Description	Signal Type
1	input	timed sink input signal	timed

Notes

1. TimedDataWrite enables the generation of one of the following output files.
 - *.tim* and *.bintim* files use MDIF format (*.tim* in ASCII form, *.bintim* in binary form).
 - *.ascsig* and *.sig* files use a signal file format used in the Cadence Signal Processing Workstation product (*.ascsig* in ASCII form, *.sig* in binary form).For format information, refer to [“Understanding File Formats”](#) in the *ADS Ptolemy Simulation* manual.

The file generated by TimedDataWrite can be used in combination with the source TimedDataRead (Timed Sources library) to provide source input data.
2. For general information regarding sinks, refer to [“Introduction” on page 1-1](#).

TimedSink



Description Timed Data Collector

Library Sinks

Class TSDF_TimedSink

Derived From baseSink

Parameters

Name	Description	Default	Unit	Type	Range
Plot	If simulation is setup to open data display after simulation and if Plot is not set to 'None', then plot the data for this sink: None, Rectangular	None		enum	
RLoad	Load resistance. DefaultRLoad will inherit from the DF controller.	DefaultRLoad	Ohm	real	(0, ∞)
RTemp	Resistor physical temperature, in degrees C. DefaultRTemp will inherit from the DF controller.	DefaultRTemp	Celsius	real	[-273.15, ∞)
Start	Start time for data recording. DefaultTimeStart will inherit from the DF Controller.	DefaultTimeStart	sec	real	[0, ∞)
Stop	Stop time for data recording. DefaultTimeStop will inherit from the DF Controller.	DefaultTimeStop	sec	real	[Start, ∞)
ControlSimulation	if set to YES, 'Stop' time determines how long the simulation will run: NO, YES	YES		enum	

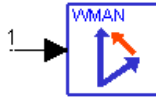
Pin Inputs

Pin	Name	Description	Signal Type
1	input	input signal	timed

Notes

1. TimedSink collects timed (baseband or complex envelope) data from the output of the connected component and saves it to the simulation dataset. Timed baseband data is in the form of real voltage values versus time. Timed complex envelope data is in the form of complex voltage values versus time.
2. The name of the variable (holding the simulation data) that is created in the dataset by each TimedSink is the same as the sink's instance name.
3. In addition to the voltage data, the TimedSink component will also save the input signal characterization frequency as an attribute named *fc*. The ael expression *get_attr()* can be used to retrieve the value of the characterization frequency. For example, if the sink instance name is T1, then *fc1=get_attr(T1, "fc")* will return the characterization frequency of the signal T1.
4. The amount of data collected by a TimedSink is controlled by the Start, Stop, and ControlSimulation parameters. Data collection always begins at the time instant specified by Start.
 - If ControlSimulation is set to Yes, then data collection ends at the time instant specified by Stop.
 - If ControlSimulation is set to No, then data collection ends when the simulation ends; in this case, there must be at least one other source or sink component that controls how long the simulation will run.
5. For general information regarding sinks, refer to [“Introduction” on page 1-1](#).

WMAN_EVM



Description WMAN EVM measurement

Library Sinks

Class TSDF_WMAN_EVM

Parameters

Name	Description	Default	Unit	Type	Range
RLoad	load resistance. DefaultRLoad will inherit from the DF controller.	DefaultRLoad	Ohm	real	(0, ∞)
RTemp	physical temperature, in degrees C, of load resistance. DefaultRTemp will inherit from the DF controller.	DefaultRTemp	Celsius	real	[-273.15, ∞)
FCarrier	carrier frequency	1.9 GHz	Hz	real	(0, ∞)
Start	start time for data recording. DefaultTimeStart will inherit from the DF Controller.	DefaultTimeStart	sec	real	[0, ∞)
AverageType	average type: Off, RMS (Video)	RMS (Video)		enum	
FramesToAverage	number of frames that will be averaged if AverageType is RMS (Video)	20		int	[1, ∞)
DataSubcarrierModulation	modulation format of the data subcarriers: Auto Detect, FCH, BPSK, QPSK, QAM 16, QAM 64	Auto Detect		enum	
GuardInterval	guard interval time, expressed as a fraction of the FFT time length	0.25		real	[0, 1]
NominalBandwidth	nominal channel bandwidth of the input signal	7.0 MHz	Hz	real	[2, 370e6]
FsBW_Ratio	ratio between the FFT sampling frequency and the nominal bandwidth: Auto, $_{57/50}$, $_{8/7}$, $_{86/75}$, $_{316/275}$, $_{144/125}$, Other	Auto		enum	

Sinks

Name	Description	Default	Unit	Type	Range
OtherFsBW_Ratio	sampling frequency to nominal bandwidth ratio when FsBW_Ratio is set to Other	8/7		real	[0.5, 2]
SearchLength	search length	4.8 msec	sec	real	(0, ∞)
PulseSearch	search for burst in the input signal: NO, YES	YES		enum	
ResultLengthType	used together with ResultLength parameter, see help for details: Auto Select, Try FCH, Manual Override	Auto Select		enum	
ResultLength	in terms of symbol-times, see help for details	60		int	[1, 1367]
MeasurementOffset	measurement offset (symbol-times)	0		int	[0, ∞)
MeasurementInterval	measurement interval (symbol-times)	60		int	[1, ∞)
SubchannelIndex	specify the subchannel to check the possible uplink subchannel preamble	16		int	[1, 31]
SymbolTimingAdjust	amount of time (expressed as a percent of the FFT time length) to back away from the end of the symbol time when deciding the part of the symbol that the FFT will be performed on	-3.125		real	[-100*GuardInterval, 0]
TrackAmplitude	pilot amplitude tracking: NO, YES	NO		enum	
TrackPhase	pilot phase tracking: NO, YES	YES		enum	
TrackTiming	pilot timing tracking: NO, YES	NO		enum	
EqualizerTraining	specify how the equalizer is initialized, or trained: CH Estimation Sequence Only, CH Estimation Sequence & Data	CH Estimation Sequence Only		enum	
SubcarrierSelect	carriers that will be analyzed: All, Single Carrier, Pilots Only	All		enum	

Name	Description	Default	Unit	Type	Range
CarrierIndex	index of carrier to be analyzed when SubcarrierSelect = Single carrier	1		int	
Debug	display instant RCE and other information when set to YES: NO, YES	NO		enum	

Pin Inputs

Pin	Name	Description	Signal Type
1	input	input signal	timed

Notes

1. This component performs an EVM measurement for an 802.16 OFDM signal.

Note The input signal must be a timed RF (complex envelope) signal or the component will error out.

This measurement provides results for RCErms_percent, RCE_dB, RCEpk_percent, PilotRCE_dB, CPERms_percent, FrequencyError_Hz, IQ_Offset_dB, and SyncCorrelation. To use these results in an AEL expression or in the *Goal* expression in an optimization setup, you must prefix them with the instance name of the component followed by a period (for example *W1.RCE_dB*).

Note The 802.16 OFDM standard uses *RCE* (*relative constellation error*) instead of *EVM* (*error vector magnitude*); while the names are different, the calculated values are exactly the same. *CPE* means *common pilot error*.

The following notes provide a brief description of the algorithm (which is the same as that used in Agilent 89600 VSA) and parameters used in this component.

The WMAN burst structure is shown in [Figure 1-8](#). Many of the terms used in the following notes such as the preamble, channel estimation sequence, data

symbol, guard-band (G, also called guard interval or cyclic prefix) are shown in this figure.

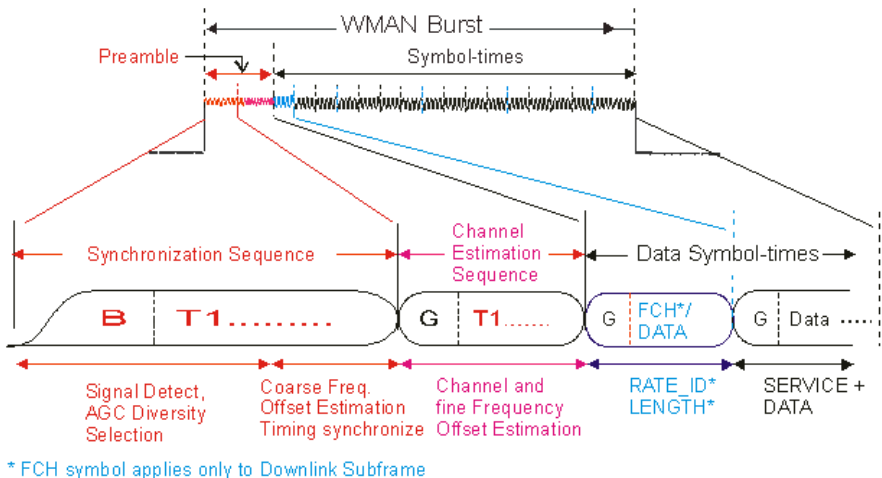


Figure 1-8. WMAN Burst Structure

- Starting at the time instant specified by the Start parameter, a signal segment of SearchLength is acquired. This signal segment is searched in order for a complete burst to be detected. The burst search algorithm looks for both a *burst on* and a *burst off* transition. In order for the burst search algorithm to detect a burst, an idle part must exist between consecutive bursts and the bursts must be at least 15 dB above the noise floor.

If the acquired signal segment does not contain a complete burst, the algorithm will not detect any burst and the analysis that follows will most likely produce incorrect results. Therefore, SearchLength must be long enough to acquire at least one complete burst. Because the time instant specified by the Start parameter can be soon after the beginning of a burst, it is recommended that SearchLength be set to a value approximately equal to $2 \times \text{burstLength} + 3 \times \text{idle}$, where burstLength is the duration of a burst in seconds and idle is the duration of the idle part in seconds. If it is known that Start is close to the beginning of a burst then SearchLength can be set to $\text{burstLength} + 2 \times \text{idle}$. If the duration of the burst or the idle part is unknown, then a TimedSink component can be used to record the signal and the signal can be plotted in the data display. By observing the magnitude of the signal's envelope versus time one can determine the duration of the burst and the idle interval.

3. If `AverageType=Off`, only one burst is detected, demodulated, and analyzed.

If `AverageType=RMS (Video)`, after the first burst is analyzed the signal segment corresponding to it is discarded and new signal samples are collected from the input to fill in the signal buffer of `SearchLength`. When the buffer is full again a new burst search is performed; when a burst is detected it is demodulated and analyzed. These steps repeat until `FramesToAverage` bursts are processed.

If a burst is misdetectd for any reason, results from its analysis are discarded. The EVM results obtained from all successfully detected, demodulated, and analyzed bursts are averaged to give the final result.

4. `DataSubcarrierModulation` specifies the data subcarrier modulation format. This parameter does not affect the demodulation of the pilot subcarriers, which are always BPSK modulation.

- When `DataSubcarrierModulation=Auto Detect` The algorithm will use the information detected within the 802.16 OFDM burst to automatically determine the data subcarrier modulation format.
- When `DataSubcarrierModulation=FCH` The algorithm will use the information within the frame control header (FCH) to determine the data subcarrier modulation format. If the FCH symbol is not present, the data subcarrier modulation format is auto-detected instead.
- When `DataSubcarrierModulation=BPSK, QPSK, QAM 16, or QAM 64` The algorithm will ignore the format determined from the OFDM burst and set the demodulator to the modulation format specified.

5. `GuardInterval` specifies the guard interval (also called cyclic extension) length for each symbol-time, as a fraction of the FFT time period. The value must match the guard interval length actually used in the input signal in order for the demodulation to work properly.

6. `NominalBandwidth` specifies the nominal channel bandwidth defined in the 802.16 standard. The actual signal bandwidth will be slightly less than this and the OFDM FFT sample rate will be slightly more. `NominalBandwidth`, combined with `FsBW_Ratio` (and the knowledge of the FFT length) determines the OFDM subcarrier spacing.

7. `FsBW_Ratio` and `OtherFsBW_Ratio` specify the ratio (n in the standard) between OFDM FFT sample rate (F_s in the standard) and the nominal channel

bandwidth (BW in the standard). The *ratio* is generally a number slightly larger than one.

- When $FsBW_Ratio=Auto$, the *ratio* is automatically detected based on the bandwidth. The value of $OtherFsBW_Ratio$ is ignored.
- When $FsBW_Ratio=_57/50$, $_8/7$, $_86/75$, $_316/275$, or $_144/125$, the value of $OtherFsBW_Ratio$ is ignored.
- When $FsBW_Ratio=Other$, the value of $OtherFsBW_Ratio$ will be used. An arbitrary value between 0.5 and 2.0 is allowed.

8. $PulseSearch$ controls how the algorithm will search for the preambles.

- When $PulseSearch=YES$, the algorithm will search for the preambles at the beginning of the burst. This is the default and recommended setting.
- When $PulseSearch=NO$, the algorithm will try to search for the preambles over the entire signal. This is only needed when some 802.16 OFDM signals do not appear to be made up of RF bursts; these signals appear to be continually on, with preambles embedded within the signal.

9. $ResultLengthType$ and $ResultLength$ control how much data is demodulated.

- When $ResultLengthType=Auto Select$, the measurement result length is automatically determined. In this case, the $ResultLength$ defines a Max Result Length for the burst in symbol-times; that is, the measurement will compare the number of symbol-times detected within the burst to the Max Result Length and uses the smaller value as the measurement result length.
- When $ResultLengthType=Try FCH$, the algorithm will first try to use the Frame Control Header to determine the measurement result length. If the FCH symbol is not present, the measurement result length will be auto-detected based on the RF pulse length.
- When $ResultLengthType=Manual Override$, the measurement result length is set to $ResultLength$ and the number of symbol-times detected within the burst is ignored.

Table 1-4 summarizes how Auto Select, Try FCH, and Manual Override modes determine the measurement result length. The table lists the measurement result lengths actually used for three different values of $ResultLength$ (30, 26 and 20 symbol-times). It is assumed that the input burst is 26 symbol-times long.

Table 1-4. ResultLength Parameter Settings

ResultLengthType	ResultLength	Measurement Result Length Actually Used
Auto Select / Try FCH	20	20
Auto Select / Try FCH	26	26
Auto Select / Try FCH	30	26
Manual Override	20	20
Manual Override	26	26
Manual Override	30	30

Note that when ResultLengthType=Manual Override and ResultLength=30 (larger than the actual burst size) the algorithm will demodulate the full 30 symbol-times even though this is 4 symbol-times beyond the burst width.

- With the MeasurementInterval and MeasurementOffset parameters the user can isolate a specific segment of the ResultLength for analysis. Only the segment specified by these two parameters will be analyzed in order to get the EVM results. Figure 1-9 shows the interrelationship between the SearchLength, ResultLength, MeasurementInterval, and MeasurementOffset.

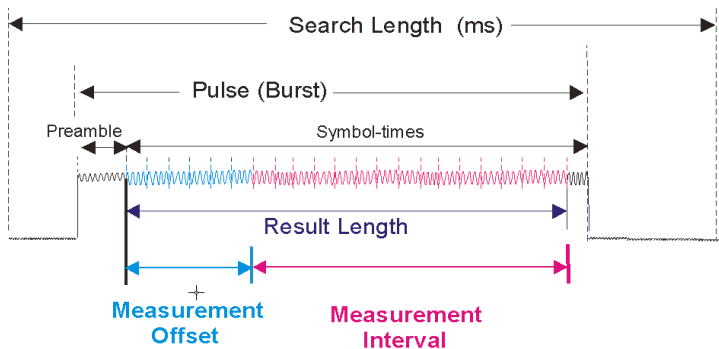


Figure 1-9. Interrelationship between SearchLength, ResultLength, MeasurementInterval, and MeasurementOffset.

- SubchannelIndex controls analysis of 802.16-2004 uplink subchannels. When demodulating an 802.16 OFDM signal, the algorithm checks for several different kinds of preambles, including Long preamble (found on Downlink subframes), Short preamble (found on most Uplink bursts), and a Subchannel preamble (found on Uplink subchannel bursts). The type of preamble found

determines the type of demodulation that is done. However, there are 31 different possible uplink subchannels specified in the 802.16-2004 standard, so the algorithm checks for only one of the possible subchannels. This parameter controls which of the subchannels the analyzer checks for. The default value is 16, which is a subchannel that uses all 200 subcarriers.

12. **SymbolTimingAdjust** adjusts the symbol timing used for demodulation. Normally, when demodulating an OFDM symbol, the guard interval is skipped and an FFT is performed on the last portion of the symbol-time. However, this means that the FFT will include the transition region between this symbol and the following symbol. To avoid this, it is generally beneficial to back away from the end of the symbol-time and use part of the guard interval.

SymbolTimingAdjust controls how far the FFT part of the symbol is adjusted away from the end of the symbol-time. The value is in terms of percent of the used (FFT) part of the symbol-time. Note that the **SymbolTimingAdjust** parameter value is negative, because the FFT start time is moved back by this parameter. **Figure 1-10** demonstrates this concept. When setting **SymbolTimingAdjust**, do not back away from the end of the symbol-time too much because this may make the FFT include corrupt data from the transition region at the beginning of the symbol-time.

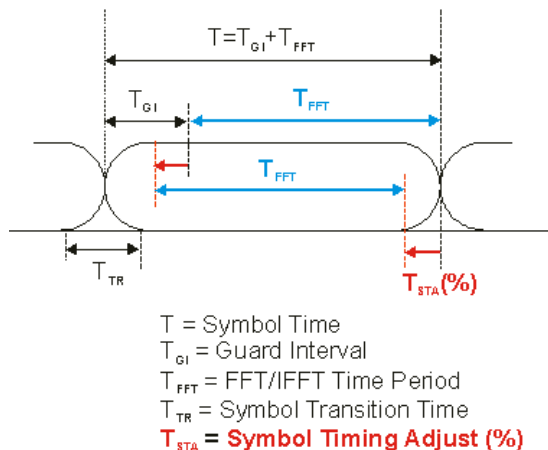


Figure 1-10. **SymbolTimingAdjust** Definition

13. **TrackAmplitude**, **TrackPhase**, and **TrackTiming** specify how pilot tracking is used to correct for imperfections in the equalizer response (calculated from the burst preamble) and for imperfections that change over the length of the burst.

- When TrackAmplitude=YES, the algorithm applies pilot subcarrier amplitude error correction to the pilot and data subcarriers.
- When TrackPhase=YES, the algorithm applies pilot subcarrier phase error correction to the pilot and data subcarriers.
- When TrackTiming=YES, the algorithm applies pilot subcarrier timing error correction to the pilot and data subcarriers.

14. EqualizerTraining specifies how the equalizer is trained. When demodulating the 802.16 OFDM signal, the algorithm uses an equalizer to correct for linear impairments in the signal path, such as multi-path.

- When EqualizerTraining=CH Estimation Sequence Only The equalizer is trained by looking at the channel estimation sequence in the preamble of the OFDM burst. After this initialization, the equalizer coefficients are held constant while demodulating the rest of the burst.

Advantages of this method: it models what a typical OFDM receiver would do, so the measured RCE (EVM) more accurately reflects the signal quality seen by a typical OFDM receiver; and it complies with the description in the *Transmit constellation error and test method* section of the 802.16 standard.

Disadvantage of this method: the measured RCE (EVM) value may be higher for signals whose impairments change during the burst than it would be if the equalizer were trained over the entire burst.

- When EqualizerTraining=CH Estimation Sequence & Data The equalizer is trained by analyzing the entire OFDM burst, including the channel estimation sequence (in the preamble) and the data symbols. This type of training generally gives a more accurate estimate of the true response of the transmission channel.

Advantages of this method: the equalizer coefficients typically reflect the linear channel impairment with greater accuracy, as the dataset used to train the equalizer is larger and is less affected by turn-on transient effects in the burst; and the RCE (EVM) is typically lower because the equalizer is less impacted by noise and some other forms of distortion.

Disadvantage of this method: it is less likely to accurately reflect the performance of a typical OFDM receiver. Because this type of equalizer calculation is more complicated and therefore more expensive to implement, it is less likely to be used in practical receivers.

15. SubcarrierSelect and CarrierIndex specify what subcarrier data will be analyzed.

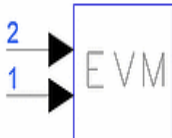
- When SubcarrierSelect=All, all subcarriers (-100 through 100, skipping 0) will be analyzed.
- When SubcarrierSelect=Single Carrier, the subcarrier number specified by CarrierIndex will be analyzed.
- When SubcarrierSelect=Pilots Only, all pilot subcarriers will be analyzed.

References

- [1] IEEE Std 802.16-2004, “*Part 16: Air Interface for Fixed Broadband Wireless Access Systems*” 2004.
- [2] Agilent 89600 VSA software, Option B7S 802.16 OFDM.
<http://www.agilent.com/find/89600>.

Chapter 2: Obsolete Sinks

ErrVecMeas



Description Error Vector Magnitude Measurement

Library Obsolete (not to be discontinued), Sinks

Class TSDFerrVecMeas

Derived From baseAnalySink

Parameters

Name	Description	Default	Unit	Type	Range
Plot	if simulation is setup to open data display after simulation and if Plot is not set to 'None', then plot the data for this sink: None, Rectangular	None		enum	
Start	Start time for data recording. DefaultTimeStart will inherit from the DF Controller.	DefaultTimeStart	sec	real	[0, ∞)
Stop	Stop time for data recording. DefaultTimeStop will inherit from the DF Controller.	DefaultTimeStop	sec	real	[Start, ∞)
SymbolTime	Symbol duration time	1.0	sec	real	(0, ∞)
Measurement_Type	Options for type of measurement in EVM: ERROR_VECTOR_MAGNITUDE, C0_DC_OFFSET, C1_PHASE_AND_POWER, W_FREQ_OFFSET_AND_GAIN_RATE, MAGNITUDE_BURST_ERROR, PHASE_BURST_ERROR, SAMPLED_TEST_DATA_SIGNAL	ERROR_VECTOR_MAGNITUDE		enum	

Name	Description	Default	Unit	Type	Range
Symbol_Burst_Length	Number of symbols within burst to be measured for the EVM	16		int	[1, ∞)
Modulation_Type	Options for modulation type of input signal: BPSK_MOD, PSK8_MOD, PI4_DQPSK_MOD, QPSK_MOD, QAM16_MOD, QAM64_MOD, PAM4_MOD, PAM8_MOD	PI4_DQPSK_MOD		enum	

Pin Inputs

Pin	Name	Description	Signal Type
1	input	timed sink input signal	timed
2	inputQ	Q phase input	timed

Notes/Equations

Note This component is obsolete for new designs. (It is available only for compatibility with designs created with ADS 1.3 or earlier.) There are no plans to remove this component from future ADS releases; however, enhancements or fixes of any existing defects will not be made.

Use the improved EVM for new design work.

1. Error vector magnitude measurements are used to evaluate the modulation accuracy of modulators. It is used, for example, in the IS-54 TDMA digital cellular to set the minimum specifications for modulation accuracy of $\pi/4$ -DQPSK modulators.

The defining equations for the EVM measurement follows the definition in the *EIA/TIA IS-54-B TDMA Cellular System Dual-Mode Mobile Station-Base Station Compatibility Standard, section 2.1.3.3.1.3.3 (Error Vector Magnitude Requirement)*. Let $Z(k)$ denote the actual complex vectors (I and Q) produced by observing the real transmitter through an ideal receiver filter at instants k , one symbol period apart. $S(k)$ is defined as the ideal referenced symbol (normalized such that its maximum energy symbol falls on the unit circle). Then, $Z(k)$ is modeled as:

$$Z(k) = [C_0 + C_1(S(k) + E(k))] W^k$$

where

$W = e^{Dr+jDa}$, accounts for both a frequency offset (Δa radians/symbol phase rotation) and an amplitude change rate (of Δr nepers/symbol)

C_0 is a complex constant origin representing quadrature modulator imbalance

C_1 is a complex constant representing the arbitrary phase and output power of the transmitter, and

$E(k)$ is the residual vector error on sample $S(k)$

The sum square error vector is

$$\sum_{k=MIN}^{MAX} |E(k)|^2 = \sum_{k=MIN}^{MAX} \left| \frac{[Z(k) W^{-k} - C_0]}{C_1} - S(k) \right|^2$$

where C_0 , C_1 , W are chosen such as to minimize the above expression, and are then used to calculate the individual vector errors $E(k)$ on each symbol.

EVM (rms) is defined to be the root-mean-square (rms) of the sum square error vector expression above. Therefore,

$$EVM(rms) = \sqrt{\frac{1}{MAX - MIN + 1} \sum_{k=MIN}^{MAX} |E(k)|^2}$$

MIN and MAX are the integer indices of the first and last symbols within the symbol burst for which the EVM is to be measured. By setting the parameter *Start* in the EVM measurement we can select which symbol we start with (thus determining MIN). By setting *Symbol_Burst_Length* in the EVM measurement we can select how long the symbol burst is (note that *Symbol_Burst_Length*=MAX-MIN+1).

The user must set the Start parameter for the EVM measurement to sample the signal at the optimal symbol timing phase. A suitable value for Start can be determined by viewing an eye diagram of the signal.

2. The Measurement_Type options for *Error Vector Magnitude (rms)*, *C0 (DC offset)*, *C1 (Output power and Phase (deg))*, *W (amplitude change (nepers/symbol))*, and *phase rotation (radians/symbol)* are self-explanatory

from note 1. The selected value option will then be displayed by the EVM measurement in the Graphics window.

The Measurement_Type options for magnitude error burst, or phase error burst (deg) refer to the magnitude or phase error of the individual received symbol. The selected value option for magnitude burst error or phase burst error will result in the EVM measurement displaying the magnitude or phase error for all the symbols within the burst. Figure 2-1 shows how the magnitude and phase error are defined.

3. The IQ signal constellations as specified by Measurement_Type are all scaled such that the maximum energy IQ point lie on the unit circle. For example, for a QPSK constellation, all four IQ points lie on the unit circle.

For a PAM-type signal (includes BPSK, 4-PAM, 8-PAM) the Q-channel of the signal is set to 0.0 and the maximum energy IQ point has a 1.0 energy value.

Figure 2-2 shows a few examples of the IQ constellation sets used in the EVM measurement.

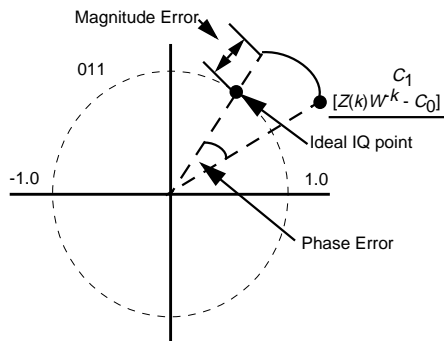


Figure 2-1. Magnitude and Phase Error

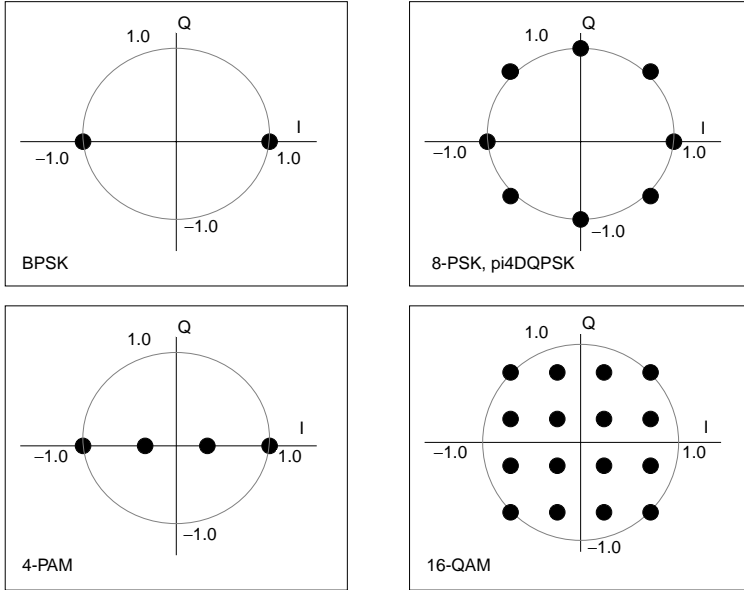
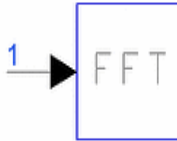


Figure 2-2. Examples of IQ-Constellations as Used in EVM Measurement

FFTAnalyzer



Description FFT Analyzer

Library **Obsolete (not to be discontinued), Sinks**

Class **TSDFFFTAnalyzer**

Derived From **baseAnalySink**

Parameters

Name	Description	Default	Unit	Type	Range
Plot	if simulation is setup to open data display after simulation and if Plot is not set to 'None', then plot the data for this sink: None, Rectangular	None		enum	
Start	Start time for data recording. DefaultTimeStart will inherit from the DF Controller.	DefaultTimeStart	sec	real	[0, ∞)
Stop	Stop time for data recording. DefaultTimeStop will inherit from the DF Controller.	DefaultTimeStop	sec	real	[Start, ∞)
DisplayFreqUnit	Options for display frequency unit of output.: GHz, MHz, KHz, Hz	MHz		enum	
NPoints	number of points per segment -- must be a power of 2	64		int	[2, ∞)†
NAverage	number of points used in smoothing window	1		int	[1, NPoints]
Param	window parameter used as needed by Window	0.5		real	(0, ∞)

Obsolete Sinks

Name	Description	Default	Unit	Type	Range
Window	windowing Type, Hamming, etc.: RECTANGULAR, BARTLETT, HANNING, HAMMING, BLACKMAN, KAISER, RIESZ, RIEMANN, DE_LA_VALLE, TUKEY, BOHMAN, POISSON, HANNING_POISSON, CAUCHY, GAUSSIAN, FLAT_TOP	HANNING		enum	
NSegments	number of segments to be used	1		int	[1, ∞)
Overlap	overlapping factor (number of overlapping points between two adjacent segments)	0		int	[0, NPoints-1]
Bias	bias type for normalizing spectrum: BIAS_NONE, BIAS_MEAN, BIAS_POWER	BIAS_POWER		enum	
Display	display spectrum in dBV, or magnitude, or dBm: dBV, MAGNITUDE, dBm	dBm		enum	
Ref_Resistance	Load reference for dBm calculation, in Ohms	50.0		real	(0, ∞)
† NPoints must be a power of 2. If not a power of 2, then the nearest (lower) power of 2 number is used. For example, given NPoints = 130, then the nearest (lower) power of 2 number is 128.					

Pin Inputs

Pin	Name	Description	Signal Type
1	input	timed sink input signal	timed

Notes/Equations

Note This component is obsolete for new designs. (It is available only for compatibility with designs created with ADS 1.3 or earlier.) There are no plans to remove this component from future ADS releases; however, enhancements or fixes of any existing defects will not be made.

Use the improved SpectrumAnalyzer for new design work; refer to note 8 for details regarding migrating FFTAnalyzer to SpectrumAnalyzer in your existing design.

1. The FFTAnalyzer measures the power spectrum of the input signal by breaking up the input signal samples into segments, windowing the segments (with a rectangular, Hamming, etc. window), taking an FFT (fast Fourier transform) of the spectrum of the segments, and averaging the spectrum over several segments.

For typical use, when only one signal segment is used with the FFTAnalyzer, the following settings may be used:

Start = 0

Stop = 1 signal time period as desired by the user

TimeUnit = sec

DisplayFreqUnit = Hz

NPoints = 2^N , where N is selected such that Stop/TStep = $2^N + 1$, and TStep is the simulation time step set by the user

NAverage = 1

Window = Rectangular

NSegments = 1

Overlap = 0

Bias = BIAS_NONE

2. Given an RF signal

$$V(t) = \text{Re}\{v(t)e^{j2\pi f_c t}\}$$

where

$$v(t) = v_I(t) + j v_Q(t)$$

is the complex envelope and f_c is the carrier frequency, the spectrum of RF signal $V(f)$ is obtained by:

$$V(f) = \int_{-\infty}^{+\infty} V(t)e^{-j2\pi ft} dt = \frac{1}{2}[V(f-f_c) + V^*(-f-f_c)]$$

Depending on the value of f_c and signal bandwidth BW (which is inversely proportional to sampling interval TStep), three cases can be identified:

- $f_c \geq \text{BW}$: The signal is bandpass and there is no overlap between $V(f-f_c)$ and $V^*(-f-f_c)$.
- $f_c < \text{BW}$: The signal is lowpass and $V(f-f_c)$ and $V^*(-f-f_c)$ overlap partially. In the overlapped region the two spectrums are added together. Since $V(f-f_c)$ and $V^*(-f-f_c)$ are numerically truncated, when added, the resulting spectrum may show small glitches. These can be removed by applying a window on the spectrum.
- $f_c = 0$: This is a special case of $f_c < \text{BW}$ where there is a complete overlap between $V(f-f_c)$ and $V^*(-f-f_c)$.

3. Start, NPoints, NSegments, Overlap, and NAverage Parameters

Figure 2-3 shows how the parameters Start, NPoints, NSegments, and Overlap are used within the FFTAnalyzer measurement.

From Figure 2-3, NPoint FFTs are applied to the NSegments segments shown and their respective power spectra calculated. The final calculated power spectrum is the average power spectrum of the NSegments segments.

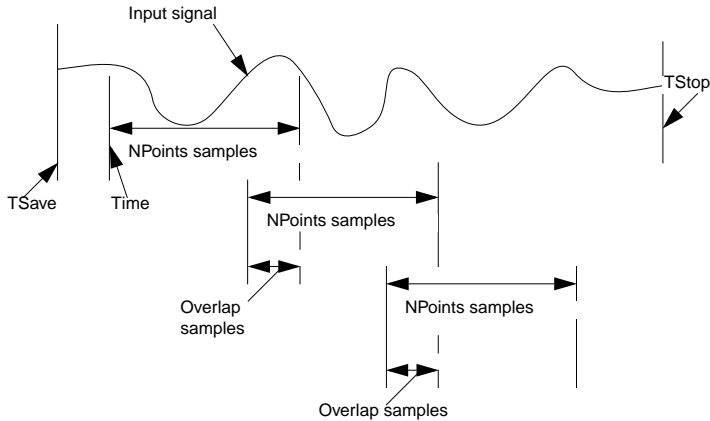
Smoothing of the average power spectrum (this is also referred to as video filtering) is performed by sliding a window of NAverage points across the average power spectrum and averaging over the points that fall inside the window. Consider the example in Figure 2-4, where NAverage = 5.

The smoothed power spectrum at a frequency point is found by averaging the (NAverage-1)/2 points below it, itself and the (NAverage-1)/2 points above it.

4. Window and Param Parameters

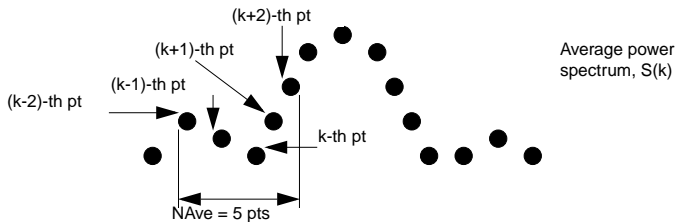
Before taking the FFT of the segments as shown in Figure 2-3, the samples within the segment are windowed by the window type as specified in the Window parameter. Windowing is often necessary in FFT-based power spectrum estimation because, without windowing, the calculated spectrum may suffer excessively from spectral leakage, which may lead to misleading measurements or the masking of weak signal spectral detail. Computing the

FFT of a segment without windowing is equivalent to applying a rectangular window on the data.



There are 3 segments here, i.e., $NS = 3$, to be used in averaging the spectrum.

Figure 2-3. Input Signal and FFT Analyzer Parameters Start, NPoints, NSegments, and Overlap



Smoothed power spectrum at k-th point:
 $\text{Smoothed}(S(k)) = (S(k-2) + S(k-1) + S(k) + S(k+1) + S(k+2)) / N_{\text{Ave}}$,
 where $N_{\text{Ave}} = 5$.

Figure 2-4. Smoothing of Average Power Spectrum by NAverage Point Moving Average Window

Suppose that the original input signal is denoted as $x(kT_s)$, where T_s is time step size of the input signal. The resulting windowed signal is represented as

$$y(kT_s) = w(kT_s)x(kT_s)$$

where $w(kT_s)$ denotes the windowing function specified by the window type parameter.

The windowing functions used follow standard signal processing conventions. Rectangular, Bartlett, Hanning, Hamming, Blackman, and Kaiser windows are described in [1]; refer to section 7.4 for window definitions and properties and chapter 11 for signal processing applications.

Window Type Definitions

Rectangular (Dirichlet) Window:

$$w(kT_s) = \begin{cases} 1.0 & (0 \leq k \leq NPoints) \\ 0.0 & otherwise \end{cases}$$

Bartlett (triangular, Fejer) Window:

$$w(kT_s) = \begin{cases} \frac{2k}{NPoints} & 0 \leq k \leq \frac{NPoints}{2} \\ 2 - \frac{2k}{NPoints} & \frac{NPoints}{2} \leq k \leq NPoints \\ 0.0 & otherwise \end{cases}$$

Hanning Window:

$$w(kT_s) = \begin{cases} 0.5 - 0.5 \cos\left(\frac{2\pi k}{NPoints}\right) & 0 \leq k \leq NPoints \\ 0.0 & otherwise \end{cases}$$

Hamming Window:

$$w(kT_s) = \begin{cases} 0.54 - 0.46 \cos\left(\frac{2\pi k}{NPoints}\right) & 0 \leq k \leq NPoints \\ 0.0 & otherwise \end{cases}$$

Blackman Window:

$$w(kT_s) = \begin{cases} 0.42 - 0.5 \cos\left(\frac{2\pi k}{NPoints}\right) + 0.08 \cos\left(\frac{4\pi k}{NPoints}\right) & 0 \leq k \leq NPoints \\ 0.0 & otherwise \end{cases}$$

Kaiser (Kaiser-Bessel) Window:

$$w(kT_s) = \begin{cases} \frac{I_0\left(\beta\left[1 - \left(\frac{k-\alpha}{\alpha}\right)^2\right]^{1/2}\right)}{I_0(\beta)} & 0 \leq k \leq NPoints \\ 0.0 & otherwise \end{cases}$$

Here $\alpha = NPoints/2$, β is the shape parameter (set by Param), and $I_0(\cdot)$ is the 0th order modified Bessel function of the first kind.

Riesz (Bochner) Window:

$$w(kT_s) = \begin{cases} 1.0 - \left(\frac{2k - NPoints}{NPoints}\right)^2 & 0 \leq k \leq NPoints \\ 0.0 & otherwise \end{cases}$$

Riemann Window:

$$w(kT_s) = \begin{cases} \frac{\sin\left(\frac{2\pi\left(k - \frac{NPoints}{2}\right)}{NPoints}\right)}{\left(\frac{2\pi\left(k - \frac{NPoints}{2}\right)}{NPoints}\right)} & 0 \leq k \leq NPoints \\ 0.0 & otherwise \end{cases}$$

de la Valle'-Poussin Window:

$$w(kT_s) = \begin{cases} 1 - 6\left(\frac{2k - NPoints}{NPoints}\right)^2 \left(1 - \frac{|2k - NPoints|}{NPoints}\right) & 0 \leq \left|k - \frac{NPoints}{2}\right| \leq \frac{NPoints}{4} \\ 2\left(1 - \frac{|2k - NPoints|}{NPoints}\right)^3 & \frac{NPoints}{4} \leq \left|k - \frac{NPoints}{2}\right| \leq \frac{NPoints}{2} \\ 0.0 & otherwise \end{cases}$$

Tukey (cosine-tapered) Window:

$$w(kT_s) = \begin{cases} 0.5 \left[1 + \cos \left(\frac{2k - (1 - \alpha)NPoints}{(1 - \alpha)NPoints} \pi \right) \right] & \frac{(NPoints)\alpha}{2} \leq \left| k - \frac{NPoints}{2} \right| \leq \frac{NPoints}{2} \\ 1.0 & 0 \leq \left| k - \frac{NPoints}{2} \right| \leq \frac{(NPoints)\alpha}{2} \\ 0.0 & \textit{otherwise} \end{cases}$$

where α is set by Param, such that $0 \leq \alpha < 1.0$.

Bohman Window:

$$w(kT_s) = \begin{cases} \left(1 - \frac{|2k - NPoints|}{NPoints} \right) \cos \left(\frac{|2k - NPoints|}{NPoints} \pi \right) + \frac{1}{\pi} \sin \left(\frac{|2k - NPoints|}{NPoints} \pi \right) & 0 \leq \left| k - \frac{NPoints}{2} \right| \leq \frac{NPoints}{2} \\ 0.0 & \textit{otherwise} \end{cases}$$

Poisson Window:

$$w(kT_s) = \begin{cases} \exp \left(-\alpha \frac{|2k - NPoints|}{NPoints} \right) & 0 \leq \left| k - \frac{NPoints}{2} \right| \leq \frac{NPoints}{2} \\ 0.0 & \textit{otherwise} \end{cases}$$

where α is set by Param, such that $0 \leq \alpha$.

Hanning-Poisson Window:

$$w(kT_s) = \begin{cases} 0.5 \left(1 + \cos \left(\frac{\pi(2k - NPoints)}{NPoints} \right) \right) \exp \left(-\alpha \frac{|2k - NPoints|}{NPoints} \right) & 0 \leq \left| k - \frac{NPoints}{2} \right| \leq \frac{NPoints}{2} \\ 0.0 & \textit{otherwise} \end{cases}$$

where α is set by Param, such that $0 \leq \alpha$.

Cauchy (Abel, Poisson) Window:

$$w(kT_s) = \begin{cases} \frac{1}{1 + \left(\alpha \frac{(2k - NPoints)}{NPoints} \right)^2} & 0 \leq \left| k - \frac{NPoints}{2} \right| \leq \frac{NPoints}{2} \\ 0.0 & \textit{otherwise} \end{cases}$$

where α is set by the parameter Param.

Gaussian (Weierstrass) Window:

$$w(kT_s) = \begin{cases} \exp\left(-\frac{1}{2}\left(\alpha\frac{(2k - NPoints)}{NPoints}\right)^2\right) & 0 \leq \left|k - \frac{NPoints}{2}\right| \leq \frac{NPoints}{2} \\ 0.0 & otherwise \end{cases}$$

where α is set by the parameter Param.

Flat Top Window:

$$wkT_s = \begin{cases} \frac{1}{4.64}\left(1 - 1.93\cos\left(\frac{2\pi k}{NPoints}\right) + 1.29\cos\left(\frac{4\pi k}{NPoints}\right) - 0.388\cos\left(\frac{6\pi k}{NPoints}\right) + 0.0322\cos\left(\frac{8\pi k}{NPoints}\right)\right) & 0 \leq k \leq NPoints \\ 0.0 & otherwise \end{cases}$$

5. Bias Parameter

The windowing of a signal in time has the effect of changing the power of the signal. Therefore, the Bias parameter is used to normalize the windowed signal by dividing the windowed signal by a factor that we denote here by ξ . This normalization changes the power of the windowed signal in the following manner.

When Bias=BIAS_NONE

$$\xi = 1.0$$

Thus, when Bias=BIAS_NONE, no normalization is done on the windowed signal.

When Bias = BIAS_MEAN, then

$$\xi = \frac{1}{NPoints} \sum_{k=0}^{NPoints-1} w(kT_s)$$

Here $w(kT_s)$ denotes the windowing function specified by the window type parameter (refer to note 4 for more details on window types). The result of this normalization is that the mean (or DC component) of the signal is kept the same in the windowed signal as in the un-windowed signal case.

When Bias=BIAS_POWER, then

$$\xi = \sqrt{\frac{1}{NPoints} \sum_{k=0}^{NPoints-1} w^2(kT_s)}$$

Here $w(kT_s)$ denotes the windowing function specified by the window type parameter (refer to note 4 for more details on the window types). The result of this normalization is that the power of the signal remains the same in the windowed signal as in the un-windowed signal case.

6. Display Parameter

The Display parameter allows the power spectrum to be displayed in dBV, or Magnitude, or dBm units.

7. Example Application

Here we show an application using the FFTAnalyzer measurement to measure the power spectrum of the output of a 0.3GMSK modulator. The test bench shown in [Figure 2-5](#) contains a random NRZ data source that feeds the 0.3GMSK modulator at whose output an FFTAnalyzer has been placed to estimate the power spectrum of the modulated signal.

The FFTAnalyzer measurement parameters are set such that the measurement will begin from Time=0; each segment will consist of 1024 points (NPoints = 2¹⁰ points); a smoothing window of size 3 (NAve=3); each segment will be windowed by a Hanning window, (note that the Param=0.5 value will be ignored by the measurement since the Hanning window does not require a parameter to specify it); 100 segments will be used to estimate the power spectrum (NSegments=100) with no overlapping between segments (Overlap=0 points); the mean value of the windowed signal will be kept the same as the unwindowed signal (Bias=Mean); the units used to display the spectrum will be in dBV.

[Figure 2-6](#) shows the power spectrum estimate as found by the FFTAnalyzer measurement.

8. The SpectrumAnalyzer component obsoletes the FFTAnalyzer. There is no one-to-one relationship between the SpectrumAnalyzer and FFTAnalyzer parameters. Except for very simple cases, such as when the number of data points in the timed interval [Start, Stop] is an exact power of 2 and only one segment is used, the output of the SpectrumAnalyzer and the FFTAnalyzer will differ.

The recommended way to set parameters common to FFTAnalyzer and SpectrumAnalyzer follows:

- Plot: set to the same value as the Plot parameter of FFTAnalyzer.
- Start: set to the same value as the Start parameter of FFTAnalyzer.
- Stop: set to the same value as the Stop parameter of FFTAnalyzer.
- Window: set to the same value as the Window parameter of FFTAnalyzer. Not all options available in the FFTAnalyzer are supported by SpectrumAnalyzer. SpectrumAnalyzer provides Blackman-Harris and HP8510 6.0 options that are not available in FFTAnalyzer. The *RECTANGULAR* option of FFTAnalyzer corresponds to the *none* option of SpectrumAnalyzer.
- WindowConstant: set to the same value as Param parameter of FFTAnalyzer
- Bias: BIAS_NONE option of FFTAnalyzer corresponds to "no bias" option of SpectrumAnalyzer. BIAS_POWER option of FFTAnalyzer corresponds to "power" option of SpectrumAnalyzer. BIAS_MEAN option of FFTAnalyzer is not available in SpectrumAnalyzer.
- NPoints: set to the same value as NPoints parameter of FFTAnalyzer. Note that if NPoints is not an exact power of 2, FFT_Analyzer resets NPoints to the biggest power of 2 that is smaller than the NPoints value given. FFT_Analyzer outputs (NPoints+1) points. SpectrumAnalyzer requires NPoints to be an odd number (if not it increases NPoints by 1 to make it odd).
- Overlap: set to the same value as Overlap parameter of FFTAnalyzer

The following parameters are unique to SpectrumAnalyzer.

- RLoad: FFTAnalyzer does not provide something equivalent to this parameter but its behavior is equivalent to having an infinite RLoad.
- RTemp: FFTAnalyzer does not provide something equivalent to this parameter.

- **FStart**: FFTAnalyzer does not provide something equivalent to this parameter (it uses $FStart = 0$ for baseband signals and $FStart = fc - 0.5/TStep$ for RF signals, where fc is the signal characterization frequency and $TStep$ is the simulation time step).
- **FStop**: FFTAnalyzer does not provide something equivalent to this parameter (it uses $FStop = 0.5/TStep$ for baseband signals and $FStop = fc + 0.5/TStep$ for RF signals, where fc is the signal characterization frequency and $TStep$ is the simulation time step).
- **NumFreqs**: FFTAnalyzer does not provide something equivalent to this parameter (it uses $NumFreqs = NPoints + 1$; see $NPoints$).

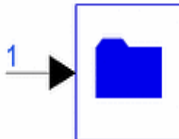
The following parameters are unique to FFTAnalyzer.

- **DisplayFreqUnit**: SpectrumAnalyzer does not provide something equivalent to this parameter.
- **NAverage**: SpectrumAnalyzer does not provide something equivalent to this parameter but its behavior is equivalent to setting $NAverage$ to 1.
- **NSegments**: SpectrumAnalyzer does not provide something equivalent to this parameter. It will try to fit as many $NPoints$ -long segments as possible in the time interval $[Start, Stop]$ by properly adjusting (increasing/decreasing) $Stop$.
- **Display**: SpectrumAnalyzer does not provide something equivalent to this parameter (it always outputs the complex amplitude voltage values at the frequencies of the spectral tones).
- **Ref_Resistance**: SpectrumAnalyzer does not provide something equivalent to this parameter.

References

- [1] A. V. Oppenheim, R. W. Schaffer, *Discrete-Time Signal Processing*, Prentice Hall, 1989, Chapter 11, section 7.4.

OutFile



Description Format Specific Output Data File
Library Obsolete (not to be discontinued), Sinks
Class TSDFOutFile
Derived From baseTimedSink

Parameters

Name	Description	Default	Unit	Type	Range
Start	Start time for data recording. DefaultTimeStart will inherit from the DF Controller.	DefaultTimeStart	sec	real	[0, ∞)
Stop	Stop time for data recording. DefaultTimeStop will inherit from the DF Controller.	DefaultTimeStop	sec	real	[Start, ∞)
FileName	filename that you assign before beginning signal analysis to create a data file. filename must begin with a letter and cannot exceed 32 characters; choose format using <i>.tim</i> , <i>.bintim</i> , <i>ascig</i> , <i>sig</i> , or <i>.dat</i> extension. During analysis, the file is generated and saved to the <i>data</i> subdirectory under your current project (<i>_prj</i>) directory.	temp		filename	

Pin Inputs

Pin	Name	Description	Signal Type
1	input	timed sink input signal	timed

Notes/Equations

Note This component is obsolete for new designs. (It is available only for compatibility with designs created with ADS 1.3 or earlier.) There are no plans to remove this component from future ADS releases; however, enhancements or fixes of any existing defects will not be made.

Use the improved `TimedDataWrite` or `SDFWrite` for new design work.

1. OutFile enables the generation of one of the following output files.

.tim and *.bintim* files use MDIF format (*.tim* in ASCII form, *.bintim* in binary form). *.ascsig* and *.sig* files use a signal file format used in the Cadence Signal Processing Workstation product (*.ascsig* in ASCII form, *.sig* in binary form). For format information, refer to “*Understanding File Formats*” in the *ADS Ptolemy Simulation* manual.

.dat signal file format used with the Agilent 89440 series of test instruments for vector modulation generators/analyzers. Refer to the *Standard Data Format Utilities User's Guide*, Agilent Part No. 5960-5759.

The file generated by OutFile can be used in combination with the source TimeFile (Timed Sources library) to provide source input data.

SpecAnalyzer



Description Spectrum Analyzer
Library Obsolete (not to be discontinued), Sinks
Class TSDFSpecAnalyzer
Derived From baseAnalySink

Parameters

Name	Description	Default	Unit	Type	Range
Plot	if simulation is setup to open data display after simulation and if Plot is not set to 'None', then plot the data for this sink: None, Rectangular	None		enum	
Start	Start time for data recording. DefaultTimeStart will inherit from the DF Controller.	DefaultTimeStart	sec	real	[0, ∞)
Stop	Stop time for data recording. DefaultTimeStop will inherit from the DF Controller.	DefaultTimeStop	sec	real	[Start, ∞)
DisplayFreqUnit	Options for display frequency unit of output.: GHz, MHz, KHz, Hz	MHz		enum	
MeasMode	measurement Mode of Spectrum Analyzer -- determines if Resolution Bandwidth value is specified or if Time Gating value is specified.: RESOLUTION_BANDWIDTH_MODE, TIME_GATING_MODE	RESOLUTION_BANDWIDTH_MODE		enum	

Name	Description	Default	Unit	Type	Range
ResBW_or_TimeGate	this parameter specifies the Resolution Bandwidth when MeasMode = RESOLUTION_BANDWIDTH_MODE. Otherwise, when MeasMode = TIME_GATING_MODE, then this parameter specifies the Time Gating value.	0.1		real	(0, ∞)
Window	windowing Type, Hamming, etc.: RECTANGULAR, BARTLETT, HANNING, HAMMING, BLACKMAN, KAISER, RIESZ, RIEMANN, DE_LA_VALLE, TUKEY, BOHMAN, POISSON, HANNING_POISSON, CAUCHY, GAUSSIAN, FLAT_TOP	HANNING		enum	
Display	display spectrum in dBV, or magnitude, or dBm: dBV, MAGNITUDE, dBm, PHASE	dBm		enum	
VideoAveraging	this parameter turns video averaging of the spectrum on, and off.: OFF, ON	OFF		enum	
Ref_Resistance	load reference for dBm calculation, in Ohms	50.0		real	(0, ∞)

Pin Inputs

Pin	Name	Description	Signal Type
1	input	timed sink input signal	timed

Notes/Equations

Note This component is obsolete for new designs. (It is available only for compatibility with designs created with ADS 1.3 or earlier.) There are no plans to remove this component from future ADS releases; however, enhancements or fixes of any existing defects will not be made.

Use the improved SpectrumAnalyzer for new design work; refer to note 8 for details regarding migrating SpecAnalyzer to SpectrumAnalyzer in your existing design.

1. SpecAnalyzer measures the spectrum (power and phase) of the input signal. The user can set the resolution bandwidth of the spectrum analyzer or set the gating time over which the spectrum analyzer measures the spectrum. Smoothing of the power spectrum can be done by video averaging.

Estimation of the signal spectrum is performed by computing an FFT (fast Fourier transform) on the signal. The power spectrum is displayed for positive frequencies only.

2. For typical use when the input signal I and Q envelopes have time periodic data, the following settings can be used:

Start = start time for periodic I and Q envelopes

Stop = stop time for the end of one time period of the I and Q data where the simulation time step TStep is selected so that $(\text{Stop} - \text{Start}) / \text{TStep} = 2^N$

TimeUnit = sec

DisplayFreqUnit = Hz

MeasMode = RESOLUTION_BANDWIDTH_MODE

ResBW = $1 / (\text{Stop} - \text{Start})$ (user must enter actual value)

Window = Rectangular

VideoAveraging = OFF

3. The FFT is a radix 2 FFT and requires 2^N data points. The collected number of data points NumPoints is equal to $(\text{Stop} - \text{Start}) / \text{TStep}$, where TStep is the simulation time step; the first data point occurs at Start and the last data point occurs at $\text{Stop} - \text{TStep}$.

Let $N = \text{ceil}(\ln(\text{NumPoints}) / \ln(2))$.

If $\text{NumPoints} < 2^N$, the dataset will be padded with zeros before NumPoints of data will be scaled by a factor of $\text{NumPoints}/(2^N)$ before the radix 2 FFT is calculated. Scaling of data is done to correct for zero padding because the desired signal length is less than the FFT data array length.

If $\text{NumPoints} < 2^N$, the FFT will contain spectral splatter caused by zero padding. To reduce this FFT effect, the user should consider using a windowing option and, perhaps, video averaging.

If $\text{NumPoints} = 2^N$, the user can insert an appropriate multi-rate design (with up/down samplers and multi-rate filters) in front of the SpecAnalyzer component to achieve the desired simulation time step that satisfies requirements for $\text{NumPoints} = 2^N$.

4. Given an RF signal

$$V(t) = \text{Re}\{v(t)e^{j2\pi f_c t}\}$$

where

$$v(t) = v_I(t) + j v_Q(t)$$

is the complex envelope and f_c is the carrier frequency, the spectrum of RF signal $V(f)$ is obtained by:

$$V(f) = \int_{-\infty}^{+\infty} V(t)e^{-j2\pi ft} dt = \frac{1}{2}[V(f-f_c) + V^*(-f-f_c)]$$

Depending on the value of f_c and signal bandwidth BW (which is inversely proportional to input signal time step TStep), three cases can be identified:

- $f_c \geq \text{BW}$: The signal is bandpass and there is no overlap between $V(f-f_c)$ and $V^*(-f-f_c)$.
- $f_c < \text{BW}$: The signal is lowpass and $V(f-f_c)$ and $V^*(-f-f_c)$ overlap partially. In the overlapped region the two spectrums are added together. Since $V(f-f_c)$ and $V^*(-f-f_c)$ are numerically truncated, when added, the resulting spectrum may show small glitches. These can be removed by applying a window on the spectrum.
- $f_c = 0$: This is a special case of $f_c < \text{BW}$ where there is a complete overlap between $V(f-f_c)$ and $V^*(-f-f_c)$.

MeasMode and ResBW_or_TimeGate parameters

When MeasMode=RESOLUTION_BANDWIDTH_MODE, the spectrum analyzer will set its resolution bandwidth (in frequency units) as specified by the value in ResBW_or_TimeGate. Note that the minimum resolution bandwidth available from the signal is inversely proportional to the total time interval of the signal (Stop – Start). If the resolution bandwidth desired is smaller than this or is set to zero (for example, ResBW_or_TimeGate=0.0 and MeasMode=RESOLUTION_BANDWIDTH_MODE), the resolution bandwidth is set equal to the minimum resolution bandwidth (that is, resolution bandwidth = 1/(Stop – Start)).

When MeasMode=TIME_GATING_MODE, the spectrum analyzer samples the signal over the time interval from Start to (Start+ResBW_or_TimeGate) and calculate the spectrum of the signal over this interval.

5. **Window parameter.** Before taking the FFT of the signal, windowing of the signal by the Window specified is performed. Windowing is often necessary in FFT-based power spectrum estimation because, without windowing, the spectrum that is calculated may suffer excessively from spectral leakage that can lead to misleading measurements or the masking of weak signal spectral detail by spurious artifacts.

Certain window options require setting a parameter α value.

- Kaiser window: $\alpha = 7.865$
- Tukey window: $\alpha = 0.75$
- Poisson window: $\alpha = 2.0$
- Hanning-Poisson window: $\alpha = 2.0$
- Cauchy window: $\alpha = 3.0$
- Gaussian window: $\alpha = 0.75$

The windowing functions used follow standard signal processing conventions. Rectangular, Bartlett, Hanning, Hamming, Blackman, and Kaiser windows are described in [1]; refer to section 7.4 for window definitions and properties and chapter 11 for signal processing applications.

6. **Display parameter.** Allows the user to view the spectrum power in dBV, magnitude, or dBm units, or it can be set to Display=Phase (deg) in which case the phase (in degrees) of the signal (versus frequency) is displayed.
7. **VideoAveraging parameter.** Smoothing of the power spectrum output can be done by setting the VideoAveraging parameter to *on*. This smoothing is done by performing FFTs on several signal segments (where the time interval within

each segment is commensurate with the desired ResBW or Time Gating) and averaging the power spectrum. If VideoAveraging is set to *off*, no averaging is done and only the power spectrum for the first signal segment (from Start and for a time interval that is commensurate with the desired ResBW or Time Gating) will be shown.

8. The SpectrumAnalyzer component obsoletes the SpecAnalyzer component. There is no one-to-one relationship between the SpectrumAnalyzer and SpecAnalyzer parameters. Except for very simple cases, such as when the number of data points in the timed interval [Start, Stop] is an exact power of 2 and only one segment is used, the output of SpectrumAnalyzer and SpecAnalyzer will differ.

The recommended way to set parameters common to SpecAnalyzer and SpectrumAnalyzer follows:

- Plot: set to the same value as Plot parameter of SpecAnalyzer.
- Start: set to the same value as Start parameter of SpecAnalyzer.
- Stop: set to the same value as Stop parameter of SpecAnalyzer.
- Window: set to the same value as Window parameter of SpecAnalyzer. Not all options available in the SpecAnalyzer are supported by SpectrumAnalyzer. SpectrumAnalyzer provides Blackman-Harris and HP8510 6.0 options, which were not available in SpecAnalyzer. RECTANGULAR option of SpecAnalyzer corresponds to none option of SpectrumAnalyzer.

The following parameters are unique to SpectrumAnalyzer

- RLoad: SpecAnalyzer does not provide something equivalent to this parameter but its behavior is equivalent to having an infinite RLoad.
- RTemp: SpecAnalyzer does not provide something equivalent to this parameter.
- WindowConstant: SpecAnalyzer does not provide something equivalent to this parameter (the constants used for Windows that require a parameter are given in its documentation).
- Bias: SpecAnalyzer does not provide something equivalent to this parameter but its behavior is equivalent to setting Bias to "power".
- FStart: SpecAnalyzer does not provide something equivalent to this parameter (it uses $FStart = 0$ for baseband signals and $FStart = fc - 0.5/TStep$

for RF signals, where f_c is the signal characterization frequency and T_{Step} is the simulation time step).

- **FStop:** SpecAnalyzer does not provide something equivalent to this parameter (it uses $F_{Stop} = 0.5/T_{Step}$ for baseband signals and $F_{Stop} = f_c + 0.5/T_{Step}$ for RF signals, where f_c is the signal characterization frequency and T_{Step} is the simulation time step).
- **NumFreqs:** SpecAnalyzer does not provide something equivalent to this parameter (it uses $NumFreqs = NPoints + 1$; see $NPoints$).
- **NPoints:** $NPoints$ used in SpecAnalyzer is calculated with a complicated expression based on $DisplayFreqUnit$, $MeasMode$, $ResBW_or_TimeGate$, and $Window$ parameters. To find out what value of $NPoints$ was used see how many points exist in the output spectrum and subtract 1.
- **Overlap:** if the $VideoAveraging$ parameter of SpecAnalyzer = OFF, $Overlap$ should be set to 0, otherwise it should be set to $NPoints / 2$.

Table 2-1. Example 1: Equivalent Settings for SpecAnalyzer and SpectrumAnalyzer

SpecAnalyzer		SpectrumAnalyzer	
Parameter	Value	Parameter	Value
Plot	Rectangular	Plot	Rectangular
Start	0	Start	0
Stop	256 usec	Stop	256 usec
DisplayFreqUnit	Hz	Rload	50
MeasMode	RESOLUTION_BANDWIDTH_MODE	Rtemp	-273.15
ResBW_or_TimeGate	0	Window	none
Window	RECTANGULAR	WindowConstant	0
Display	dBm	Bias	power
VideoAveraging	OFF	Fstart	0
Ref_Resistance	50	Fstop	1.00e+11
		NumFreqs	0
		Npoints	0
		Overlap	0

The following assumptions are made for this example:

1. SpecAnalyzer has only one associated external shunt resistor to ground providing the 50-ohm terminal and validates setting $Ref_Resistance$ to 50 ohms; the R_{Temp} value of that resistance is set to -273.15.
2. The $(Start-Stop)/T_{Step}$ value is a power of 2, where T_{Step} is the simulation time step (1 usec in this example).
3. The SpectrumAnalyzer display was set up in the ADS Data Display by selection of the $dBm(.)$ for plotting data.

Table 2-2. Example 2: Equivalent Settings for SpecAnalyzer and SpectrumAnalyzer

SpecAnalyzer		SpectrumAnalyzer	
Parameter	Value	Parameter	Value
Plot	Rectangular	Plot	Rectangular
Start	0	Start	0
Stop	256 usec	Stop	256 usec
DisplayFreqUnit	Hz	Rload	50
MeasMode	TIME_GATING_MODE	Rtemp	-273.15
ResBW_or_TimeGate	128 usec	Window	none
Window	RECTANGULAR	WindowConstant	0
Display	dBm	Bias	power
VideoAveraging	ON	Fstart	0
Ref_Resistance	50	Fstop	1.00e+11
		NumFreqs	0
		Npoints	(128 usec)/(1 usec) + 1
		Overlap	(128 usec)/(1 usec)/2 + 1
<p>The following assumptions are made for this example:</p> <ol style="list-style-type: none"> 1. SpecAnalyzer has only one associated external shunt resistor to ground providing the 50-ohm terminal and validates setting Ref_Resistance to 50 ohms; the RTemp value of that resistance is set to -273.15. 2. The (Start-Stop)/TStep value is a power of 2, where TStep is the simulation time step (1 usec in this example). 3. The SpectrumAnalyzer display was set up in the ADS Data Display by selection of the dBm(.) for plotting data. 			

References

- [1] A. V. Oppenheim, R. W. Schaffer, *Discrete-Time Signal Processing*, Prentice Hall, 1989, Chapter 11, section 7.4.

Index

B

BER_FER, 1-18
berIS, 1-22
berMC, 1-28
berMC4, 1-35

E

ErrVecMeas, 2-2
EVM, 1-38
EVM_WithRef, 1-44

F

FFTAnalyzer, 2-7

N

NumericSink, 1-48
NumericSinkGated, 1-51

O

OutFile, 2-20

P

Printer, 1-53

S

Sinad, 1-54
sinks, 1-1
SpecAnalyzer, 2-22
SpectrumAnalyzer, 1-57
SpectrumAnalyzerResBW, 1-64

T

TimedDataWrite, 1-71
TimedSink, 1-73

W

WMAN_EVM, 1-75

