Agilent Technologies

Advanced Design System 2011.01

Feburary 2011
Instruments

Acknowledgments

GNU Lesser General Public License along with this library; if not, write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA. unRAR copyright: The decompression engine for RAR archives was developed using source code of unRAR program.All copyrights to original unRAR code are owned by Alexander Roshal. unRAR License: The unRAR sources cannot be used to re-create the RAR compression algorithm, which is proprietary. Distribution of modified unRAR sources in separate form or as a part of other software is permitted, provided that it is clearly stated in the documentation and source comments that the code may not be used to develop a RAR (WinRAR) compatible archiver. 7-zip Availability: http://www.7-zip.org/

AMD Version 2.2 - AMD Notice: The AMD code was modified. Used by permission. AMD copyright: AMD Version 2.2, Copyright © 2007 by Timothy A. Davis, Patrick R. Amestoy, and Iain S. Duff. All Rights Reserved. AMD License: Your use or distribution of AMD or any modified version of AMD implies that you agree to this License. This library is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version. This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details. You should have received a copy of the GNU Lesser General Public License along with this library; if not, write to the Free Software Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA Permission is hereby granted to use or copy this program under the terms of the GNU LGPL, provided that the Copyright, this License, and the Availability of the original version is retained on all copies.User documentation of any code that uses this code or any modified version of this code must cite the Copyright, this License, the Availability note, and "Used by permission." Permission to modify the code and to distribute modified code is granted, provided the Copyright, this License, and the Availability note are retained, and a notice that the code was modified is included. AMD Availability: http://www.cise.ufl.edu/research/sparse/amd

UMFPACK 5.0.2 - UMFPACK Notice: The UMFPACK code was modified. Used by permission. UMFPACK Copyright: UMFPACK Copyright © 1995-2006 by Timothy A. Davis. All Rights Reserved. UMFPACK License: Your use or distribution of UMFPACK or any modified version of UMFPACK implies that you agree to this License. This library is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version. This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details. You should have received a copy of the GNU Lesser General Public License along with this library; if not, write to the Free Software Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA Permission is hereby granted to use or copy this program under the terms of the GNU LGPL, provided that the Copyright, this License, and the Availability of the original version is retained on all copies. User documentation of any code that uses this code or any modified version of this code must cite the Copyright, this License, the Availability note, and "Used by permission." Permission to modify the code and to distribute modified code is granted, provided the Copyright, this License, and the Availability note are retained, and a notice that the code was modified is included. UMFPACK Availability: http://www.cise.ufl.edu/research/sparse/umfpack UMFPACK (including versions 2.2.1 and earlier, in FORTRAN) is available at http://www.cise.ufl.edu/research/sparse . MA38 is available in the Harwell Subroutine Library. This version of UMFPACK includes a modified form of COLAMD Version 2.0, originally released on Jan. 31, 2000, also available at http://www.cise.ufl.edu/research/sparse . COLAMD V2.0 is also incorporated as a built-in function in MATLAB version 6.1, by The MathWorks, Inc. http://www.mathworks.com . COLAMD V1.0 appears as a column-preordering in SuperLU (SuperLU is available at http://www.netlib.org). UMFPACK v4.0 is a built-in routine in MATLAB 6.5. UMFPACK v4.3 is a built-in routine in MATLAB 7.1.

Qt Version 4.6.3 - Qt Notice: The Qt code was modified. Used by permission. Qt copyright: Qt Version 4.6.3, Copyright (c) 2010 by Nokia Corporation. All Rights Reserved. Qt License: Your use or distribution of Qt or any modified version of Qt implies that you agree to this License. This library is free software; you can redistribute it and/or modify it under

the
terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version. This library is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details. You should have received a copy of the GNU Lesser General Public License along with this library; if not, write to the Free Software Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA Permission is hereby granted to use or copy this program under the terms of the GNU LGPL, provided that the Copyright, this License, and the Availability of the original version is retained on all copies.User
documentation of any code that uses this code or any modified version of this code must cite the Copyright, this License, the Availability note, and "Used by permission." Permission to modify the code and to distribute modified code is granted, provided the Copyright, this License, and the Availability note are retained, and a notice that the code was modified is included. Qt Availability: http://www.qtsoftware.com/downloads Patches Applied to Qt can be found in the installation at:
$HPEESOF_DIR/prod/licenses/thirdparty/qt/patches. You may also contact Brian Buchanan at Agilent Inc. at brian_buchanan@agilent.com for more information.

The HiSIM_HV source code, and all copyrights, trade secrets or other intellectual property rights in and to the source code, is owned by Hiroshima University and/or STARC.

Errata The ADS product may contain references to "HP" or "HPEESOF" such as in file names and directory names. The business entity formerly known as "HP EEsof" is now part of Agilent Technologies and is known as "Agilent EEsof". To avoid broken functionality and to maintain backward compatibility for our customers, we did not change all the names and labels that contain "HP" or "HPEESOF" references.

Warranty The material contained in this document is provided "as is", and is subject to being changed, without notice, in future editions. Further, to the maximum extent permitted by applicable law, Agilent disclaims all warranties, either express or implied, with regard to this documentation and any information contained herein, including but not limited to the implied warranties of merchantability and fitness for a particular purpose. Agilent shall not be liable for errors or for incidental or consequential damages in connection with the furnishing, use, or performance of this document or of any information contained herein. Should Agilent and the user have a separate written agreement with warranty terms covering the material in this document that conflict with these terms, the warranty terms in the separate agreement shall control.

Technology Licenses The hardware and/or software described in this document are furnished under a license and may be used or copied only in accordance with the terms of such license. Portions of this product include the SystemC software licensed under Open Source terms, which are available for download at http://systemc.org/ . This software is redistributed by Agilent. The Contributors of the SystemC software provide this software "as is" and offer no warranty of any kind, express or implied, including without limitation warranties or conditions or title and non-infringement, and implied warranties or conditions merchantability and fitness for a particular purpose. Contributors shall not be liable for any damages of any kind including without limitation direct, indirect, special, incidental and consequential damages, such as lost profits. Any provisions that differ from this disclaimer are offered by Agilent only.

Restricted Rights Legend U.S. Government Restricted Rights. Software and technical data rights granted to the federal government include only those rights customarily provided to end user customers. Agilent provides this customary commercial license in Software and technical data pursuant to FAR 12.211 (Technical Data) and 12.212 (Computer Software) and, for the Department of Defense, DFARS 252.227-7015 (Technical Data - Commercial Items) and DFARS 227.7202-3 (Rights in Commercial Computer Software or Computer Software Documentation).

About Instruments Components

ADS Ptolemy supports a suite of components that provide interfaces from the schematic to:

- Agilent instruments to transfer data and commands
- Agilent VEE and Agilent VSA 89600 series software
- Read/write files compatible with other Agilent products such as Signal Studio, VSA 89600 software, and Baseband Studio

New Connection-Manager-based components are available with the 2003C release; these benefits are provided:

- Supported on all ADS platforms
- Interactive instrument selection
- Industry standard instrument drivers
- Support newer IO interfaces

> **Note**
> The Connection Manager instrument links pull the ADS Connection Manager license.

Connection Manager

The Connection-Manager-based instrument link components (named CM_ and located in the Instruments library) perform measurements based on the Connection Manager (CM) architecture. The term *measurement* is a general term used here to describe functions in a server library; generally, a measurement includes gathering physical data from one or more instruments, but a measurement is not limited to instrument control.

The Connection-Manager-based simulation components are part of the *client* half of the CM *client/server* system and operate through an instance of the CM server. You can think of the CM-based simulation components as interfaces into the measurements available on the server. For more information, refer to *Client-Server Architecture* (connectmui) under the section on *Operational and Functional Concepts* (connectmui) in the *Connection Manager* (connectmui) documentation.

The client also provides instrument *discovery* that makes it easy to identify the instruments that are connected to your server workstation, enabling the client to easily associate a particular instrument with a particular measurement. For more information, refer to *Discovering Connected Hardware* (connectmui) under the section on *Getting Started with Connection Manager* (connectmui) in the *Connection Manager* (connectmui) documentation.

Detailed information regarding Connection Manager basics, server IO setup and functional concepts is provided in the *Connection Manager* (connectmui) documentation.

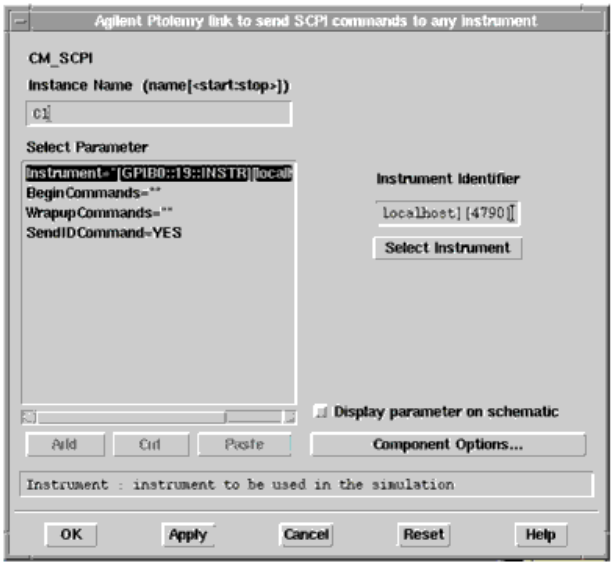
Instrument Selection

The CM components enable interactive selection of the instrument before simulation. These components include an Instrument parameter that is set to a string following a specific syntax:

```
[GPIB::20::INSTR][localhost][4790]
```

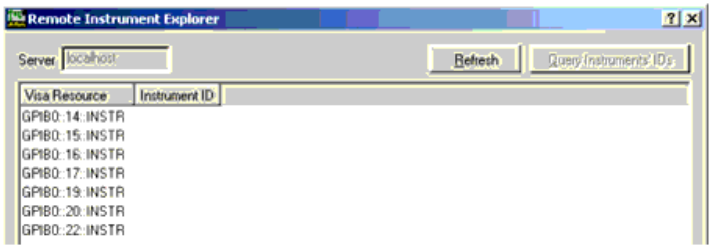
To interactively select an instrument:

1. From the ADS Schematic window, double-click a CM component symbol to access the component dialog box.

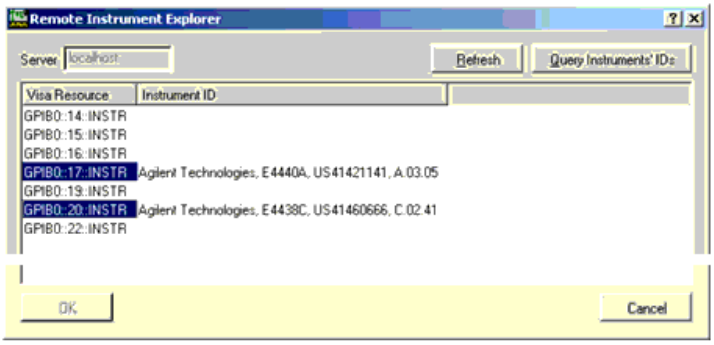


2. Select Instrument from the parameter list.
3. Click **Select Instrument** to access the *Set Server* dialog box.

4. Enter the DNS host name or IP address of the workstation on which an instance of the Connection Manager server is running.
5. Enter the port number on which the server is waiting for incoming connection requests. Unless the server settings have been manually changed, use the default port number 4790.
6. Click **OK** to access the *Remote Instrument Explorer* dialog box.



This dialog box shows the VISA Resource identifiers of all instruments that are currently connected through interfaces configured on the workstation running the Connection Manager server. A VISA Resource identifier is similar to an IP address and can uniquely identify an instrument among all instruments connected to the workstation through configured interfaces.
To map the VISA Resource identifiers to the associated instrument model number, select one or more of the VISA Resource entries and click Query Instruments' IDs to tell the Connection Manager server to send the standard IEEE 488.2 identification command **IDN?* to the instrument(s). Most modern instruments that understand this command respond with an identification string. After all selected instruments are queried, the Instrument IDs are displayed:



7. Select a specific instrument based on the Instrument ID string displayed and click **OK**.
8. In the component dialog box, click **OK** to return to the Schematic window. Click **Cancel** in any of the dialogs to retain your old settings.

For information on renaming the instrument IDs, refer to *Customizing Instrument Identification Strings* (connectmui) under the section on *Getting Started with Connection Manager* (connectmui) in the *Connection Manager* (connectmui) documentation.

VeeLink Setup

ADS Ptolemy supports the VeeLink component which enables you to use VEE UserFunctions as if these are integral parts of your Ptolemy simulation. Typical uses for the VeeLink component include:

- Controlling instruments, wafer probers, and IC handlers.
- Adding measurements or other functions not provided in ADS.

Requirements for using VeeLink are:

- Windows 2000 and XP Professional
- ADS Ptolemy and Matrix licenses

> **Note**
> While the VeeLink component is not a separately-licensed component, the use of Ptolemy matrices requires that simulations using VeeLink have a Ptolemy matrix license.

- Agilent IO libraries version (optional). The IO libraries enable calling a VEE user function that communicates with instruments.

For details on installing and configuring Agilent IO libraries, refer to "Configuring IO Interfaces" in the *Agilent IO Libraries Installation and Configuration Guide*; you can download the latest version of this document at http://www.agilent.com/find/iolib .

> **Note**
> Agilent VEE runtime environments are not supported by VeeLink.

Required Support Files

Before you can use the VeeLink component, you must install certain DLL and VEE Service Manager files.

VeeLink component implementation depends on the presence of two DLLs in the directory paths listed in your environment PATH variable.

- *VeeLink.dll* - A library that provides generic capability to call VEE functions. This library is installed as part of Ptolemy in ADS.
- *libvapi.dll* - A library that is distributed with Agilent VEE. *VeeLink.dll* needs this library in order to load. The library is installed in the *lib* subdirectory of the VEE installation directory during Ptolemy installation.

Using VeeLink requires a program called VEE Service Manager.

- The VEE Service Manager is spawned from a Windows Service called *VeeService.exe*, and enables a client to use VEE's remote service capability. The *veesm.exe* file is installed in the VEE installation directory.
- Beginning with the 2004A release, the Vee Service installation is moved out from under the main ADS installation so you can choose where to install Vee Service. The default is *c:\ProgramFiles\Agilent\ConnectionManager*.

For more information about the VEE Service Manager, refer to VEE Service Manager.

Using VeeLink

The VeeLink component is listed in the *Instruments* palette. Static configuration information, such as the name of the VEE library containing the VEE UserFunction you want to call, is specified using the component's configuration parameters. Arguments to the function and the function return values are supplied through data flow (described in the next section on Data Flow Semantics). For information about developing VEE UserFunctions, refer to the Agilent VEE documentation.

The VeeLink component provides an extension mechanism which you can use to implement additional functionality in an ADS Ptolemy schematic.

To use a VeeLink component:

1. In the Agilent VEE development environment, develop a VEE UserFunction that implements the additional functionality you need.
2. In an ADS Ptolemy schematic, include the VeeLink component to gain schematic access to that UserFunction.

Data Flow Semantics

Ptolemy calls your VEE UserFunction when data is delivered to the VeeLink component's input pin. You should understand the relationship of data inputs and outputs, and data types between a UserFunction and the VeeLink component; these are discussed in the following sections.

VEE UserFunction Arguments - Data Types

The number of input data connections to the VeeLink component must match exactly the number of data input terminals contained in your VEE UserFunction. The data types you can specify currently include:

- int
- float (real)
- complex
- int_matrix
- float_matrix
- complex_matrix

Note that the matrix types must have one data row. These can have as many columns as you want in that row.

VEE UserFunction input terminals can be constrained to expect data of specific types and shapes. A data shape describes the dimensionality of the data. For example, a data shape could be constrained to be a scalar (a single value) or an array with one dimension. The

defaults, when you add a terminal to a UserFunction, are data type *Any* and data shape *Any*. In this case, *Any* means that the terminal accepts any data type or shape that the VEE system understands. When you constrain a terminal, you restrict the data type or shape the terminal accepts. In the event that you supply data to a UserFunction that does not specifically match the type or shape a terminal specifies, VEE tries to coerce the supplied data to match the type and shape the function expects. There are some conversions VEE cannot make, since doing so causes a loss of data precision. If it cannot perform the data type or shape conversion, VEE emits an error, which is propagated back to the VeeLink component.

VEE UserFunction Arguments - Evaluation Order

A VEE UserFunction shares another characteristic with functions typically written in a textual language: the arguments are evaluated based on position, not based on the terminal name. For instance, if a UserFunction has three arguments, the first argument you supply is sent to the top-most pin, the second argument is sent to the middle pin, and so on.

There is no requirement to connect anything to the VeeLink component's input pin, unless the VEE UserFunction has an input terminal. When connections are required, if you simply connect Ptolemy components to the VeeLink component, Ptolemy calls the function supplying arguments whose order is based on the order in which you connected the data source to the VEE link component input. Sometimes this is acceptable. However, if it is important that a particular data source maps to a particular UserFunction input terminal, you can use a *Bus Merge* component (located in *Numeric Control*) to specify the ordering of arguments. Note that the *Bus Merge* component sends out data from bottom to top, so the bottom-most pin on the *Bus Merge* component directs data to the top-most UserFunction data terminal and so on.

VEE UserFunction Results

A VEE UserFunction can return any number of values, all of which are carried on the VeeLink component's single data output pin. The number of UserFunction results actually propagated into the rest of the Ptolemy simulation depends on the number of connections you make to the VeeLink component's output pin. You do not need to match the number of VeeLink component output pin connections to the number of output terminals the VEE UserFunction has. The VeeLink component propagates UserFunction output terminal results sequentially from top to bottom.

For example, suppose your VEE UserFunction has three output terminals. If you have two connections to the VeeLink component output pin, results from the two top-most UserFunction output terminals propagate into the Ptolemy schematic. The results from the remaining (bottom) UserFunction output terminal are discarded. If, for example, you want results from the first and third UserFunction outputs only, you must make three connections to the VeeLink component output pin.

A VEE UserFunction can return different types on the same output terminal on different invocations. Because of this, the VeeLink component output terminal is of data type *Anytype*. The VeeLink component constructs a Ptolemy particle whose type corresponds to the data type and shape that the VEE UserFunction returns. Output data types currently supported are:

- int
- float (real)
- complex
- int_matrix
- float_matrix
- complex_matrix
- anytype

Note that the matrix types currently support only one data row, having as many columns as there are elements in the VEE UserFunction result array.

Results propagate from the VeeLink according to the order in which you make output pin connections. VEE UserFunction results are position dependent, so if you need to specify that a specific UserFunction terminal result propagates to a given VeeLink component output pin connection, you must use a *BusSplit* component (located in the *Numeric Control* library). The *BusSplit* component propagates the VEE UserFunction results from bottom to top: the bottom-most *BusSplit* component output pin propagates the top-most VEE UserFunction result terminal; the next *BusSplit* output pin up propagates the next UserFunction terminal down; and so on.

Data Type Resolution

Both the VeeLink input and output pins are of type *Anytype*. In order for the Ptolemy simulator to resolve the data types flowing through data connections, the connections you make to the VeeLink component must have concrete data types. Connecting the VeeLink component to a component having a data pin with a concrete type, enables the data type resolution to determine the type of data to flow from the VeeLink component.

VEE Service Manager

The VEE Service Manager is a Windows Service that wraps itself around the *veesm.exe* executable. *Veesm.exe* is a library that enables other programs to make RPC calls to VEE functions across a network. *Veesm* is just a Win32 executable; it has no UI. In the past, to make VEE RPC calls into a server workstation, there needed to be an interactive logon session running, and that session had to start *veesm.exe*. Most people would put *veesm.exe* in the system startup program group. Then, when someone logged into the system, *veesm* would start up in the security context of the interactively logged-on user. When the user logged out, the system would shut *veesm.exe* down, breaking whatever connection a client had to an instance of a VEE server.

The necessity to have an interactively logged-on user causes a subtle problem. Because *veesm* would run in the environment provided by the user's logon session, the VEE server programs would look for the IO configuration files in the home directory of whoever logged in. If *Lucy* wrote a VEE server program you were interested in using, she would probably reference instruments in the IO configuration in her home directory. But if *Ricky* logs into the server workstation, *veesm* runs using his account credential, and his IO configuration files may not have the instruments *Lucy's* program needs.

Wrapping a Windows Service around *veesm* solves the problems created by requiring an interactive logon session. You can configure a service to start automatically at system startup, and that service remains available through all interactive logons and logoffs, until the system shuts down or someone whose account has appropriate privileges stops it. Also, you can configure a service to start only when specifically directed to do so and specify which user account environment you want a Service to run under. Because the service has control of the environment under which the spawned *veesm* runs, the service can direct *veesm* to the correct IO configuration file.

Service Manager Composition

The VEE Service Manager consists of these files:

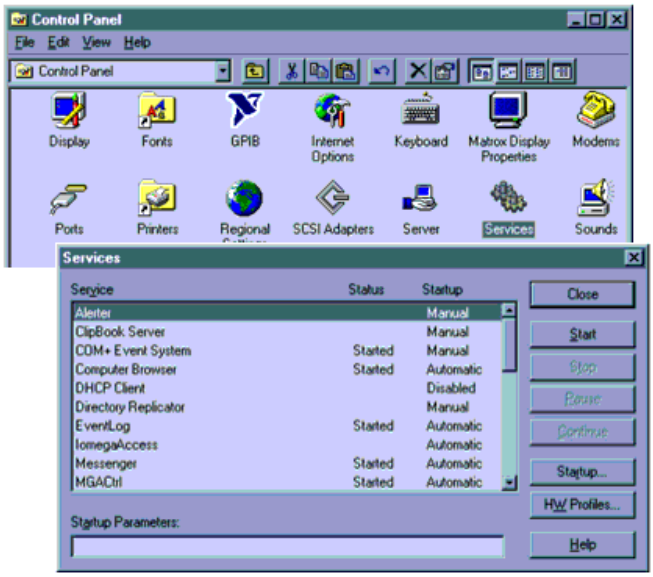
- *Veesm.exe* is the VEE RPC provider shipped as part of the VEE product.
- *VeeService.exe* is the implementation file for the Windows Service. This file is shipped as part of ADS and is located in the *<adsInstalldir>/bin* directory.
- *VeeEvents.dll* is a resource-only dll that is part of the Windows Event Viewer. This is

the standard mechanism by which Windows system components log informational and error information. This is shipped as part of ADS and must be in the same directory as *VeeService.exe*.

- *VeeSrv.cpl* is Windows Control Panel applet that provides a convenient interface to determine where the system looks to find your VEE IO configuration file. This file is copied into the *%SystemRoot%\system32* directory during the VeeLink component installation.

Putting the Vee Service Manager Under SCM Control

All Services, including device drivers (Windows Services are degenerate forms of device drivers) run under the auspices of the system Service Control Manger (SCM). The Windows Control Panel contains a *Services* applet.



The Services applet is a system component that enables you to control a Service. To put the Vee Service Manager under SCM control, run *VeeService.exe -install*. After installation, you see that there is an entry in the Services applet. By default, this entry has been marked to start manually. To run the program, push the Service Start button. In a release build, we set the Service to start automatically when the system starts.

Unwiring from the System

To remove the Service from the SCM, effectively making it inaccessible, run *VeeService.exe -remove*. This stops the service if the service is running, then removes it from the SCM list of configured Services.

Starting and Stopping

You can start or stop the Service from the command line, by running *VeeService -start* or *VeeService -stop*. Or, you can use the start and stop buttons on the Services Control Panel applet.

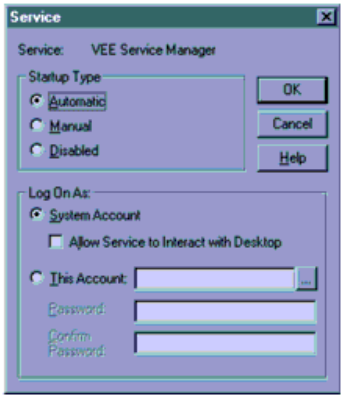
Configuring

When using a Service, you should consider the environment in which you expect it to run. One implication here is that any processes the service spawns runs under the same environment as the Service itself, unless the Service makes special provisions to do

otherwise. The VEE Service Manager has made no such provisions.

When Does It Start?

The Service installs itself to start automatically at system startup. You can always use the Services Control Panel applet to change its startup time, pressing *Startup*, which opens this dialog.

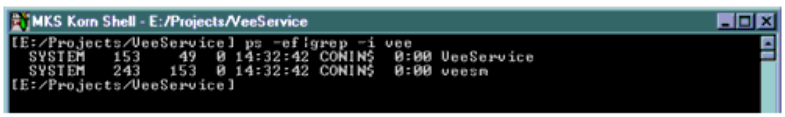


Who Does It Run As?

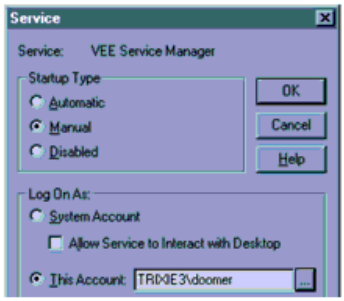
The Service window *Log On As* part of the Services Control Panel applet defines the environment under which a Service runs. If you do not do anything else, the Service runs under the *System Account*. This account exists on every Windows 2000 and XP machine. It is a privileged account, in that the account has most of the rights normally granted to the Administrator group. On Windows 2000 and later OS versions, the System account does have network share privileges.

Notice the check box labeled *Allow Service to Interact with Desktop*; and, note that this check box is not available if you choose to run the Service under an account other than the System account. This check box enables the Service to create an interactive window on the active desktop. When ADS installs the Vee Service, ADS sets the account to be *Local System* and enables the *Allow Service to Interact with Desktop* capability. Most services do not have a UI; instead, these provide a special-purpose Control Panel applet to configure them. This has some important implications for Callable VEE clients. If your VEE server program needs to have UI interaction, you must check this box. This condition can show up if the called VEE function has a button (or whatever) on a panel. Also, this condition can happen if a Ptolemy simulation has a VeeLink component with the *Debug* property set.

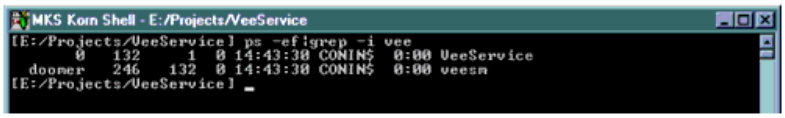
You can run the Service under the environment of a particular account. In this display, you can see that the Service, and the *veesm* process which it spawned, are running under the System account.



If you change the account the Service runs under (as displayed),



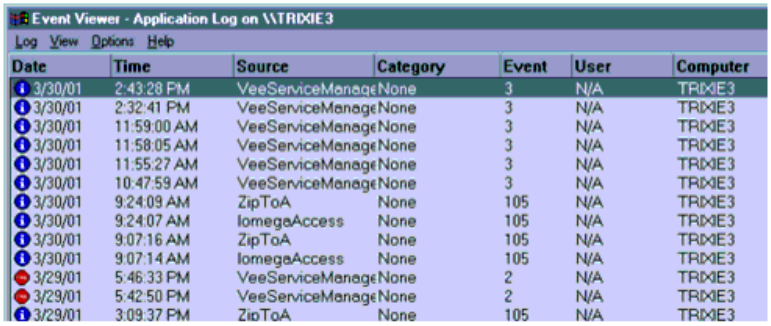
this change is reflected the next time the Service starts.



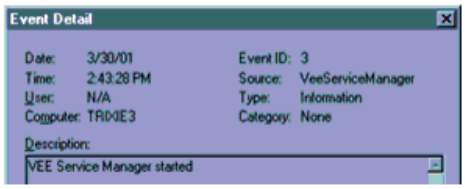
If you run under an account other than the System account, you must supply a password. Also, the Service is not able to interact with the desktop when running under an account other than the System account.

Checking Its Health

The Service logs informational and error messages to the system Event Viewer.



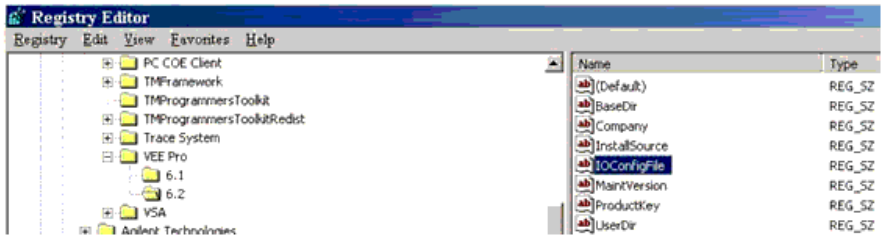
When the Service starts, the program inserts an informational message that contains text similar to this:



If the Service encounters an error, the Service logs the error with a message that is flagged with a stop sign.

Where the Service Finds Its vee.io File

When the Service starts up, the program looks for a registry entry like this:

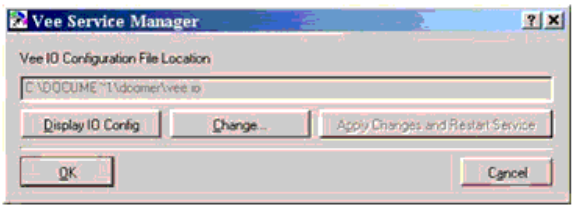


The service expects the registry entry to be a single directory path.

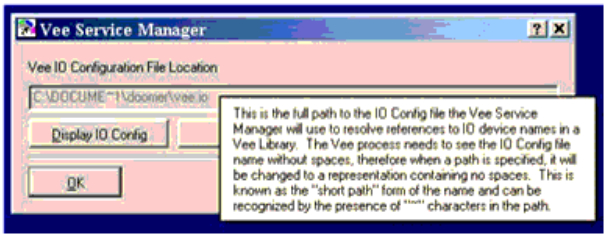
If the service finds this entry, it spins off the *veesm.exe* program with the *-veeio* option, which instructs any instances of Vee resolving remote User Function calls to reference the instrument names contained in the file the registry entry points to. Otherwise, the service starts *veesm* without any options. This is an important point, because the content of the IO file resolves references to instruments in your Vee User Functions. If your User Function contains a reference to an instrument named *Lucy*, you are referring to the *Lucy* instrument name in the IO file the registry entry identifies. Another IO file might contain an instrument definition also called *Lucy*, which may or may not point to the same instrument as the IO file specified in the registry. Another important point is that, for your Vee User Function to load, all instruments your function references must be defined in the IO file. Otherwise, the simulation generates an error.

A Windows Control Panel applet provides a convenient mechanism to specify where you want the VEE Service to look for the VEE IO configuration file. When launching the Control Panel, you should see an entry titled *VEE Service Manager*.

When you launch this configuration utility, you see a wizard dialog.



You can get help on any component by clicking **?** in the title bar; then, when the cursor changes to **?**, select the component.



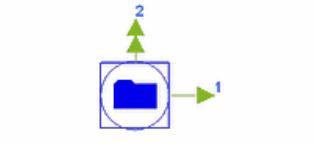
The *File Location* field contains the path to the IO file used to resolve references to instrument names. The Vee RPC server must see this path without spaces; so the server uses what is known as the *short path* form, which is recognized by **~** characters in the path.

The *Display IO Config* button opens an instance of Vee Pro, displaying the IO Manager dialog with the contents of the IO file listed.

To select a different IO file for future use in the Vee Service, press the *Change* button; select the file from the dialog box that opens.

To apply your changes immediately, press *Apply Changes and Restart Service*. This stops Vee Service (disconnecting any clients), then starts the service, picking up the value of the new IO file. To apply your changes to future sessions without having an immediate effect on Vee Service, click **OK**. To discard any changes (preventing those changes from effecting future sessions) click **Cancel**.

CM_BB_StudioStreamRead



Description: Baseband Studio for Streaming file reader
Library: Instruments
Class: SDFCM_BB_StudioStreamRead

Parameters

Name	Description	Default	Type
FileName	input baseband studio file name	file.bin	filename
ControlSimulation	control simulation: NO, YES	NO	enum
Periodic	periodic output: NO, YES	YES	enum
MarkerFormat	format of marker bits embedded in IQ data: zero Markers (16 Bit IQ), two Markers (15 Bit IQ), four Markers (14 Bit IQ)	zero Markers (16 Bit IQ)	enum

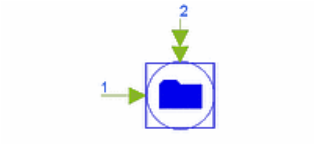
Pin Outputs

Pin	Name	Description	Signal Type
1	output		complex
2	outputmarkers		multiple complex

Notes/Equations

1. Baseband Studio for waveform streaming is an Agilent product that contains elements of PC hardware and software that, when coupled with an Agilent PSG or ESG vector signal generator and a PC, provides a deep memory signal source needed for thorough testing early in the development process.
   The I/Q waveform data from a Baseband Studio file can be played using the Baseband studio for streaming UI directly on the ESG/PSG internal baseband generator DACs in a continuous fashion. This results in long, unique test signals, up to hours in length directly from the ESG/PSG analog I/Q and RF output ports.
   The CM_BB_StudioStreamRead component can be used to read data from a file format compatible with the Baseband Studio product in an ADS simulation. In addition, the CM_BB_StudioStreamRead and CM_BB_StudioStreamWrite components can be used to save and playback long I/Q sequences within ADS.
2. The component has two outputs: a data output pin for complex I/Q data and a control output pin for marker values. The number of outputs in the marker output pin is driven by the MarkerFormat parameter.
3. The MarkerFormat parameter specifies if the waveform data in the file is interpreted as 16-, 15-, or 14-bit data with 0, 2 or 4 markers are embedded in each data value respectively.
   zero Markers(16 Bit I,Q) - no output from the marker pin
   two Markers(15 Bit I,Q) - one complex output from marker pin in the form (Event1 − bit + j Event2 − bit)
   four Markers(14 Bit I,Q) - two complex outputs from marker pin in the form (Event1 − bit + jEvent2 − bit) and (Event3 − bit + jEvent4 − bit)
4. If the length of simulation is larger than the available data in the file, use the Periodic parameter to repeat the old data. If Periodic = YES, all data from the file is read and repeated. If Periodic = NO, the output is zero after all data is read.
5. If ControlSimulation = YES, the simulation runs until the last data in the waveform file is read.

CM_BB_StudioStreamWrite



Description: Baseband Studio for Streaming file writer
Library: Instruments
Class: SDFCM_BB_StudioStreamWrite
Derived From: CM_BinaryArbWrite

Parameters

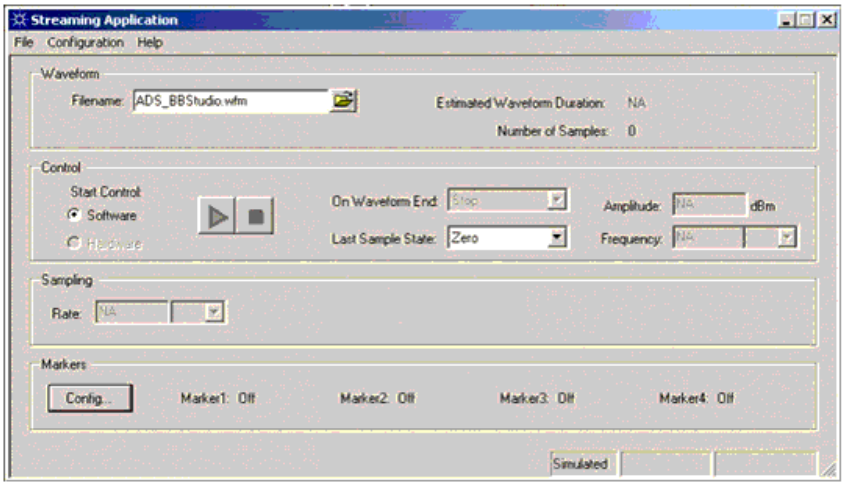
Name	Description	Default	Type
Start	sample number to start waveform recording	DefaultNumericStart	int
Stop	sample number to stop waveform recording	DefaultNumericStop	int
FileName	Output baseband studio file name	file.bin	filename
ControlSimulation	control simulation: NO, YES	YES	enum
AutoScaling	scale input data to fit dynamic range of ESG DAC: NO, YES	YES	enum

Pin Inputs

Pin	Name	Description	Signal Type
1	input		complex
2	inputmarkers		multiple complex

Notes/Equations

1. Baseband Studio for waveform streaming is an Agilent product that contains elements of PC hardware and software that, when coupled with an Agilent PSG or ESG vector signal generator and a PC, provides a deep memory signal source needed for thorough testing early in the development process.
   The CM_BB_StudioStreamWrite component can be used to create a signal compatible with the Baseband Studio product. In addition, the CM_BB_StudioStreamRead and CM_BB_StudioStreamWrite components can be used to save and playback long I/Q sequences within ADS.
   The file generated by this sink can be transferred to a PC configured for use with Baseband Studio; and, the I/Q waveform data from the file can be played using the Baseband studio for streaming UI directly on the ESG/PSG internal baseband generator DACs in a continuous fashion. This results in long, unique test signals, up to hours in length directly from the ESG/PSG analog I/Q and RF output ports.



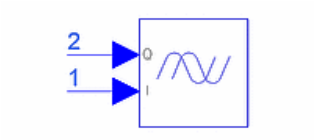
2. The component has two inputs: an input pin for complex I/Q data and an optional complex bus input pin to enable marker bits to be embedded in the file. To avoid creating marker bits, leave the marker input pin disconnected. Up to two complex inputs can be connected to the marker input pin. For every point in the waveform, up to four marker bits can be generated based on whether the corresponding input is >1

or 0. The marker bits are embedded into each I/Q data point in the file like this:
Zero connections to marker input - Data format: 0 Markers; 16 Bit I,Q
One connection to marker input - Data format: 2 Markers; 15 Bit I,Q
Two connections to marker input - Data Format: 4 Markers; 14 Bit I,Q
For example, to enable event markers 1 and 4 for the first 10 points of the waveform, generate two complex signals A and B that take the values (1 + j0) and (0 + j1) respectively for the first 10 points in the simulation and revert to (0 + j0) for the rest of the simulation. Use the BusMerge2 component to merge signals A and B and connect the output of BusMerge2 to the marker bus input pin.

Note
The Baseband Studio file format contains only data values as 16-bit integers with optional markers embedded in each data value. It does not contain any header information such as Amplitude, Frequency, and SampleClk. If required, these instrument settings must be controlled from the instrument's front panel or by using appropriate SCPI commands.

3. The AutoScaling parameter specifies whether input scaling is to be performed. When AutoScaling = YES, the input I/Q data is scaled to the range {−1, 1}, and the scaling factor is displayed towards the end of the simulation. To perform scaling, data is cached in a temporary file, which can require significantly more disk space. When AutoScaling = NO, no scaling is performed on the input data; however, data outside the range {−1, 1} is clipped to −1 or 1.
4. If ControlSimulation = YES, the simulation runs from Start to Stop − Start + 1.

CM_ESG_E443xB_Sink



Description: Agilent ESG E443xB RF Signal Generator
Library: Instruments
Class: SDFCM_ESG_E443xB_Sink
Derived From: CM_ESG_E4438C_Sink

Parameters

Name	Description	Default	Unit	Type
Instrument	instrument to be used in the simulation	[GPIB0::19::INSTR][localhost][4790]		instrument
Enabled	When YES communicates with instrument: NO, YES	YES		enum
Start	sample number to start waveform recording	DefaultNumericStart		int
Stop	sample number to stop waveform recording	DefaultNumericStop		int
Frequency	RF output frequency	3e9	Hz	real
Amplitude	RF output power level (dBm)	-135		real
ARBRef	reference for the waveform clock: Internal, External	Internal		enum
ARBRefFreq	reference frequency of the external clock generator	10e6	Hz	real
SampleClk	sample clock rate for sequencer and DAC	4.9152e6	Hz	real
FileName	waveform file name	esg.wfm		filename
ArbOn	select waveform and turn ARB ON after download: NO, YES	NO		enum
RFPowOn	turn RF power ON after download: NO, YES	NO		enum
EventMarkers	enable ESG markers: Neither, Event1, Event2, Both	Neither		enum
MarkerLength	select number of points for ESG markers	10		int
RecFilter	baseband filter applied to the IQ signals: through, filter_250kHz, filter_2500kHz, filter_8MHz	through		enum
ScalingFactor	scaling factor applied to the IQ signals	1.0		real

Pin Inputs

Pin	Name	Description	Signal Type
1	I	I input waveform data	real
2	Q	Q input waveform data	real

Notes/Equations

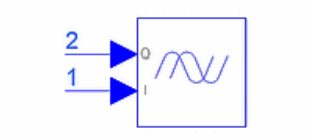
Note
On 64-bit Linux this component must run in 32-bit mode to avoid "socket error."

1. The CM_ESG_E443xB_Sink model collects data from a simulation and downloads the data to an ESG-D/DP Series Signal Generator. This model uses the connection manager architecture to communicate with the instrument. For more information, refer to *Client-Server Architecture* (connectmui) under the section on *Operational and Functional Concepts* (connectmui) in the *Connection Manager* (connectmui) documentation .
2. Prerequisites for using CM_ESG_E443xB_Sink are:
   - ESG-D/DP Series Signal Generator: E4430B, E4431B, E4432B, or E4433B; for information, visit the website http://www.agilent.com/find/esg .
   - ESG, Dual Arbitrary Waveform Generator module option UND.
   - Supported method of connecting the instrument to your computer through the Connection Manager architecture. For detailed setup information, refer to the *Connection Manager* (connectmui) documentation.
3. The Instrument parameter specifies a triplet that identifies the specific instrument to be used in the simulation, along with the Connection Manager server information to be used to connect to the instrument.
   To interactively select an instrument, refer to *Instrument Selection* (instruments) under the section *About Instruments Components* (instruments).
   For more information regarding instrument selection, refer to *Discovering Connected Hardware* (connectmui) under the section on *Getting Started with Connection Manager* (connectmui) in the *Connection Manager* (connectmui) documentation.
4. The Start and Stop parameters specify when to start and stop data collection. The

number of samples collected, Stop − Start + 1, must be in the range 16 samples to 8 Msamples, where 1 Msample = 1,048,576 samples. The ESG requires an even number of samples; the last sample is discarded if Stop − Start + 1 is odd.

5. The ARBRef parameter specifies an internal or external reference for the ESG clock generator. If set to *External*, the ARBRefFreq parameter sets the frequency of this clock.
6. The SampleClk parameter sets the sample clock rate for the DAC output.
7. The FileName parameter sets the name of the waveform inside the ESG that holds the downloaded data. If FileName is blank, this model does not attempt any communication with the instrument.
8. The I/Q data values passed to the sink is offset and scaled to fit the instrument waveform data format. To scale the data before downloading to the ESG, set the ScalingFactor parameter. For full waveform signal range, set ScalingFactor = 1.0.
9. If the ArbOn parameter = YES, the ESG starts generating the signal immediately after simulation data is downloaded. If ArbOn = NO (default), waveform generation must be turned on at the ESG front panel.
10. If the RFPowOn parameter = YES, the ESG turns on RF power immediately after simulation data is downloaded. If RFPowOn = NO (default), RF power must be turned on at the ESG front panel.
11. The EventMarkers parameter specifies which ESG Event markers are enabled: *Event1* , *Event2*, *Both*, or *Neither*. Event markers are used for synchronizing other instruments to the ESG. When one or both EventMarkers are enabled, *Event1* and/or *Event2* is set to the first sample of the downloaded Arb waveform. This is equivalent to setting the events on the front panel of the ESG.
    For more information, refer to the *Agilent ESG-D/DP Series Options UND and UN5 Signal Generators* manual, Chapter 2.
12. The RecFilter parameter specifies the cutoff frequency for the reconstruction filter that lies between the DAC output and the Dual Arbitrary Waveform Generator output inside the ESG.
13. To access usage examples, from the ADS Main window, choose **File > Open > Example > Instruments > CM_ESG_wrk**.

CM_ESG_E4438C_Sink



Description: Agilent ESG Vector Signal Generators
Library: Instruments

Parameters

Name	Description	Default	Unit	Type	Range
Instrument	instrument to be used in the simulation	[GPIB0::20::INSTR][localhost][4790]		instrument	
Enabled	When YES communicates with instrument: NO, YES	YES		enum	
Start	sample number to start waveform recording	DefaultNumericStart		int	
Stop	sample number to stop waveform recording	DefaultNumericStop		int	
Frequency	RF output frequency	3e9	Hz	real	
Amplitude	RF output power level (dBm)	-135		real	
ARBRef	reference for the waveform clock: Internal, External	Internal		enum	
ARBRefFreq	reference frequency of the external clock generator	10e6	Hz	real	
IQModFilter	baseband filter applied to the IQ signals: through, filter_2100kHz, filter_40MHz	through		enum	
SampleClk	sample clock rate for sequencer and DAC	4.9152e6	Hz	real	
FileName	waveform file name	esg.wfm		filename	
DownloadMode	select mode of operation of the sink: download_to_VARB, write_to_datafile, both	download_to_VARB		enum	
AutoScaling	scale inputs to range { -1, +1 } before download: NO, YES	YES		enum	
ArbOn	select waveform and turn ARB ON after download: NO, YES	NO		enum	
RFPowOn	turn RF power ON after download: NO, YES	NO		enum	
EventMarkers	enable ESG markers: Neither, Event1, Event2, Both	Neither		enum	
MarkerLength	select number of points for ESG markers	10		int	
InstructionTimeout	Instrument instruction timeout	300 sec	sec	int	[1,∞)

Pin Inputs

Pin	Name	Description	Signal Type
1	I	I input waveform data	real
2	Q	Q input waveform data	real

Notes/Equations

Note
On 64-bit Linux this component must run in 32-bit mode to avoid "socket error."

1. The CM_ESG_E4438C_Sink model collects data from a simulation and downloads the data to an E4438C Vector Signal Generator. Parameters specify how data is interpreted by the ESG. This model uses the connection manager architecture to communicate with the instrument. In addition to the E4438C mentioned above, the E8267C (PSG-C), E8267D (PSG-D), and N5182A (MXG) are also supported.
2. Prerequisites for using CM_ESG_E4438C_Sink are:
   - ESG Vector Signal Generator E4438C; for information, visit the website http://www.agilent.com/find/esg .
   - Supported method of connecting the instrument to your computer through the Connection Manager architecture. For detailed setup information, refer to

Connection Manager (connectmui) documentation.

3. The Instrument parameter specifies a triplet that identifies the specific instrument to be used in the simulation, along with the Connection Manager server information to be used to connect to the instrument.
   To interactively select an instrument, refer to *Instrument Selection* (instruments) under *About Instruments Components* (instruments).
   For more information regarding instrument selection, refer to *Discovering Connected Hardware* (connectmui) under the section on *Getting Started with Connection Manager* (connectmui) in the *Connection Manager* (connectmui) documentation.
4. The Enabled parameter specifies whether communication with the instrument needs to be established during the simulation. For example, if the sink needs to send a command to the instrument, Enabled should be set to "YES".
5. The Start and Stop parameters specify when to start and stop data collection. The number of samples collected, Stop − Start + 1, must be in the range 60 samples to 64 Msamples, where 1 Msample = 1,048,576 samples. The ESG requires an even number of samples; the last sample is discarded if Stop − Start + 1 is odd.
6. The ARBRef parameter specifies an internal or external reference for the ESG clock generator. If set to External, the ARBRefFreq parameter sets the frequency of this clock.
7. The IQModFilter parameter specifies the cutoff frequency for the reconstruction filter that lies between the DAC output and the Dual Arbitrary Waveform Generator output inside the ESG.
8. The SampleClk parameter sets the sample clock rate for the DAC output.
9. The FileName parameter sets the name of the file into which data is written. Depending on the DownloadMode parameter, data can be written to the ESG, to a local data file, or both. The same filename is used for both options.
10. The DownloadMode parameter specifies the mode of operation of this sink component.
    download_to_VARB - The waveform is downloaded to the ESG volatile ARB (VARB) memory.
    Use this mode of operation to avoid writing data to the local file system.
    write_to_datafile - The waveform is written to a proprietary encrypted binary file format in the data directory of the workspace. Then the file can be transferred to the SECUREWAVE directory in the ESG file system using an FTP client (select the binary data transfer option during the FTP process). The ESG automatically decrypts the file and makes the file available for selection from the list of waveforms in its NVARB memory.
    Use this mode of operation to avoid communication with the ESG instrument during simulation.
    Note that, besides data, only the Frequency and SampleClk values are embedded in the file as header information. Other ESG settings (such as Amplitude) must be set manually from the ESG front panel or by using appropriate SCPI commands.
    both - *download_to_VARB* and *write_to_datafile* options are enabled.

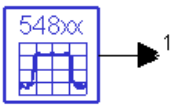
    Note
    When the *write_to_datafile* option is used to create a datafile in the SignalStudio Wavform Package Library Format, the signal is normalized to the {−1, 1} range. In order to enable the recovery of the signal voltage level, the scaling factor used for normalization `(abs(max(I,Q)))` is embedded in the header of the datafile. When this file is used with the CM_SStudioFileRead component, the embedded scaling factor value is used to restore the signal to its original level.

11. The ESG driver expects data in the range {−1, 1}. The AutoScaling parameter specifies whether to scale inputs to fit this range.
    If AutoScaling = YES, inputs are scaled to the range {−1, 1} and scaling is applied to data written to the local file (refer to *note 10*).
    If AutoScaling = NO, raw simulation data is downloaded to the ESG without any scaling, but data outside the range {−1, 1} is clipped to −1 or 1.
12. If ArbOn = YES, the ESG starts generating the signal immediately after simulation data is downloaded; if ArbOn = NO, waveform generation must be turned on at the ESG front panel.
13. If RFPowOn = YES, the ESG turns on RF power immediately after simulation data is downloaded; if RFPowOn = NO (default), RF power must be turned on at the ESG front panel.
14. The EventMarkers parameter specifies which ESG Event markers are enabled: *Event1* , *Event2*, *Both*, or *Neither*. Event markers are used for synchronizing other

instruments to the ESG. When one or both EventMarkers are enabled, *Event1* and/or *Event2* is set to the first sample of the downloaded Arb waveform. This is equivalent to setting the corresponding event from the front panel of the ESG.

15. The MarkerLength parameter specifies the range of points over which the markers must be set starting from the first point of the waveform. Depending on the setting of the EventMarkers parameter, the length of trigger *Event1* and/or *Event2* is set to a multiple of the pulse-width which, in turn, is determined by the sample clock rate of the DAC output.
16. The InstructionTimeout parameter (added for ADS 2008 Update 1 Release) specifies how long the application will wait for an instrument instruction execution to complete before a timeout error is reported This parameter must be at least 1 second. The default value is 300 seconds. The purpose for this parameter is to accomodate different instrument operation complete times, especially the downloading time of lengthy waveforms (tens of Mega-samples) for N5182 (MXG). This parameter will have no impact on performance. The only drawback for specifying a long waiting time, is that if no instrument is present or there is a communication problem between the host computer and the instrument, the SW will appear to hang before reporting a timeout error.
17. To access usage examples, from the ADS Main window, choose **File > Open > Example > Instruments > CM_ESG_wrk**.
18. For ESG, PSG, MXG families of Agilent ARB sources, you can do an ARB download at one specific carrier amplitude/power level at a time using a single CM_ESG_E4438C_Sink simulation. However, if you want to achieve a swept power measurement (Ex: BER Bathtub curve measurement verification), you can use the ADS CM_SCPI components in combination with the CM_ESG_E4438C_Sink/Sequencer controller to send SCPI control commands to the MXG/ESG/PSG source which allow you to change carrier amplitude dynamically. In effect, CM_SCPI allows you to sweep ARB Source power values without doing multiple signal downloads from ADS. Certainly, you could do multiple CM_ESG_E4438C_Sink download simulations in sequence to test at different power levels, but depending upon your signal length, this may take a long time in series... Instead, we recommend using the more efficient CM_SCPI use model. For more details/examples, see the following which demonstrate how the CM_SCPI component can be used in ADS simulation sequence to more efficiently sweep the power level of MXG/ESG/PSG via parameter WrapupCommands=":POW {Next_Power} DBM":
    - **$HPEESOF_DIR/examples/Connected_Solutions/3GPP_BER_wrk(3GPP_Uplink_VSA_BER)**
    - **$HPEESOF_DIR/examples/Connected_Solutions/WLAN_BER_wrk(WLAN_802_11ag_VSA_BER)**
    - **Sending SCPI Commands to my Signal Generator to change the Power**

CM_Infiniium_548xx_Source1



Description: Capture one channel from an Infiniium oscilloscope
Library: Instruments
Class: TSDF_CM_Infiniium_548xx_Source

Parameters

Name	Description	Default	Unit	Type	Range
Instrument	instrument to be used in the simulation	[GPIB0::7::INSTR] [localhost][4790]		instrument	
OverrideInstrumentModelCheck	if no, error if instrument model is unsupported: NO, YES	NO		enum	
Waveform1	source data to write to output#1: Channel 1, Channel 2, Channel 3, Channel 4, Memory 1, Memory 2, Memory 3, Memory 4	Channel 1		enum	
SetupFile	name of setup file to recall state into Infiniium			filename	
UseInstrumentSettings	if yes, ignore Autoscale, TimebaseDelay and TriggerHoldoff component parameters: NO, YES	YES		enum	
InstructionTimeout	Time to wait for an instrument instruction to complete before giving up	10	sec	int	(0, 600)
Autoscale	perform autoscale before acquiring source data: NO, YES	YES		enum	
TimebaseDelay	time interval between the trigger event and the delay reference point	0	sec	real	(-∞, ∞)
TriggerHoldoff	time the oscilloscope should wait after receiving a trigger before enabling the trigger again	80 nsec	sec	real	[50 nsec, 10 sec]
ROut	resistance	DefaultROut	Ohm	real	[0, ∞)
RTemp	physical temperature, in degrees C	DefaultRTemp	Celsius	real	[-273.15, ∞)
TStep	simulation time step: Derived from source data, Derived from output pin	Derived from source data	sec	real enum	{-1} or [0, ∞)
ControlSimulation	control simulation: NO, YES	NO		enum	
SegmentLength	if non-zero, truncate or zero pad source data to this length	0	sec	real	[0, ∞)
RepeatData	control operation at the end of source data: Single pass, Repeat, Reacquire	Single pass		enum	
InterpolationType	interpolation technique to use: Lagrange, Sample and hold, Linear	Lagrange		enum	
InterpolationOrder	number of points to use if Lagrange interpolation is set	4		int	[2, ∞)

Pin Outputs

Pin	Name	Description	Signal Type
1	output#1		timed

Notes/Equations

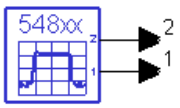
Note
On 64-bit Linux this component must run in 32-bit mode to avoid "socket error."

1. The Instrument parameter specifies a triplet that identifies the specific instrument to

be used in the simulation, along with the Connection Manager server information to be used to connect to the instrument.
To interactively select an instrument, refer to *Instrument Selection* (instruments).
For more information regarding instrument selection, refer to *Discovering Connected Hardware* (connectmui) under the section on *Getting Started with Connection Manager* (connectmui) in the *Connection Manager* (connectmui) documentation.

2. The OverrideInstrumentModelCheck parameter enables overriding the check of the instrument's firmware version. For more details regarding overriding this check, refer to the *Override instrument model check* (connectmui) under the section on *Operational and Functional Concepts* (connectmui) in the *Connection Manager* (connectmui) documentation.
3. The Connection Manager can be used to create the file for the SetupFile parameter. This file can be generated using the *Save Instrument State* button on the *Advanced* tab of the *Voltage Waveform Measurements - Agilent 5483xB and 5485xB Oscilloscope Families* window. The *Advanced Tab* (connectmui) details are described under the section on *Operational and Functional Concepts* (connectmui) in the *Connection Manager* (connectmui) documentation.
4. The InstructionTimeout parameter enables you to take measurement sweeps that can take some time to finish, as described in the following examples.
   If the instrument is set to single sweep mode and the default number of points is selected, the instrument attempts to transfer a large amount of data. Set InstructionTimeout to a number of seconds slightly greater than the time needed to read the instrument data; this prevents the simulation from aborting before data transfer is complete. Or, set the number of points to transfer explicitly from the instrument.
   If the instrument is configured to trigger off of an external event that takes a while, set InstructionTimeout to a value slightly longer than the time it takes to receive a trigger; this enables the instrument enough time to read its data and prevents the simulation from aborting before data transfer is complete.
5. For more information on setting ROut, RTemp and TStep parameters, refer to the *Timed Sources* (timed) introduction. Additionally, this component supports a TStep value of −1 that automatically sets TStep to the source data sampling period.
6. If ControlSimulation = YES and RepeatData = Single pass, the simulation runs until the last data of the source data is read.
7. The InterpolationType and InterpolationOrder parameters are used only when TStep and the source data sampling period are not equal.
8. To access example designs that use this model, from the ADS Main window, choose **File > Open > Example > Instruments > CMInfiniium_wrk**.
9. Note: The state of the instrument is set by this model during its use and is not reset to its initial state after use. Customer needs to rely on manual reset of the instrument to its former state as desired, such as instrument internal interpolation state, after its use by this model.

CM_Infiniium_548xx_Source2



Description: Capture two channels from an Infiniium oscilloscope
Library: Instruments
Class: TSDF_CM_Infiniium_548xx_Source

Parameters

Name	Description	Default	Unit	Type	Range
Instrument	instrument to be used in the simulation	[GPIB0::7::INSTR] [localhost][4790]		instrument	
OverrideInstrumentModelCheck	if no, error if instrument model is unsupported: NO, YES	NO		enum	
Waveform1	source data to write to output#1: Channel 1, Channel 2, Channel 3, Channel 4, Memory 1, Memory 2, Memory 3, Memory 4	Channel 1		enum	
Waveform2	source data to write to output#2: Channel 1, Channel 2, Channel 3, Channel 4, Memory 1, Memory 2, Memory 3, Memory 4	Channel 3		enum	
SetupFile	name of setup file to recall state into Infiniium			filename	
UseInstrumentSettings	if yes, ignore Autoscale, TimebaseDelay and TriggerHoldoff component parameters: NO, YES	YES		enum	
InstructionTimeout	Time to wait for an instrument instruction to complete before giving up	10	sec	int	(0, 600)
Autoscale	perform autoscale before acquiring source data: NO, YES	YES		enum	
TimebaseDelay	time interval between the trigger event and the delay reference point	0	sec	real	(-∞, ∞)
TriggerHoldoff	time the oscilloscope should wait after receiving a trigger before enabling the trigger again	80 nsec	sec	real	[50 nsec, 10 sec]
ROut	resistance	DefaultROut	Ohm	real	[0, ∞)
RTemp	physical temperature, in degrees C	DefaultRTemp	Celsius	real	[-273.15, ∞)
TStep	simulation time step: Derived from source data, Derived from output pin	Derived from source data	sec	real enum	{-1} or [0, ∞)
ControlSimulation	control simulation: NO, YES	NO		enum	
SegmentLength	if non-zero, truncate or zero pad source data to this length	0	sec	real	[0, ∞)
RepeatData	control operation at the end of source data: Single pass, Repeat, Reacquire	Single pass		enum	
InterpolationType	interpolation technique to use: Lagrange, Sample and hold, Linear	Lagrange		enum	
InterpolationOrder	number of points to use if Lagrange interpolation is set	4		int	[2, ∞)

Pin Outputs

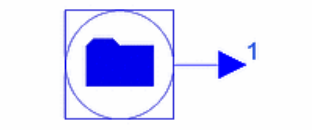
Pin	Name	Description	Signal Type
1	output#1		timed
2	output#2		timed

Notes/Equations

Note
On 64-bit Linux this component must run in 32-bit mode to avoid "socket error."

1. The Instrument parameter specifies a triplet that identifies the specific instrument to be used in the simulation, along with the Connection Manager server information to be used to connect to the instrument.
   To interactively select an instrument, refer to *Instrument Selection* (instruments).
   For more information regarding instrument selection, refer to Discovering Connected Hardware under the section on Gettting Started with Connection Manager in the *Connection Manager* (connectmui) documentation.
2. The OverrideInstrumentModelCheck parameter enables overriding the check of the instrument's firmware version. For more details regarding overriding this check, refer to the *Override instrument model check* (connectmui) under the section on *Operational and Functional Concepts* (connectmui) in the *Connection Manager* (connectmui) documentation.
3. The Connection Manager can be used to create the file for the SetupFile parameter. This file can be generated using the *Save Instrument State* button on the *Advanced* tab of the *Voltage Waveform Measurements - Agilent 5483xB and 5485xB Oscilloscope Families* window. The *Advanced Tab* (connectmui) details are described under the section on *Operational and Functional Concepts* (connectmui) in the *Connection Manager* (connectmui) documentation.
4. The InstructionTimeout parameter enables you to take measurement sweeps that can take some time to finish, as described in the following examples.
   If the instrument is set to single sweep mode and the default number of points is selected, the instrument attempts to transfer a large amount of data. Set InstructionTimeout to a number of seconds slightly greater than the time needed to read the instrument data; this prevents the simulation from aborting before data transfer is complete. Or, set the number of points to transfer explicitly from the instrument.
   If the instrument is configured to trigger off of an external event that takes a while, set InstructionTimeout to a value slightly longer than the time it takes to receive a trigger; this enables the instrument enough time to read its data and prevents the simulation from aborting before data transfer is complete.
5. For more information on setting ROut, RTemp and TStep parameters, refer to the *Timed Sources* (timed) introduction. Additionally, this component supports a TStep value of −1 that automatically sets TStep to the source data sampling period.
6. If ControlSimulation = YES and RepeatData = Single pass, the simulation runs until the last data of the source data is read.
7. The InterpolationType and InterpolationOrder parameters are used only when the TStep and the source data sampling period are not equal.
8. To access example designs that use this model, from the ADS Main window, choose **File > Open > Example > Instruments > CMInfiniium_wrk**.
9. Note: The state of the instrument is set by this model during its use and is not reset to its initial state after use. Customer needs to rely on manual reset of the instrument to its former state as desired, such as instrument internal interpolation state, after its use by this model.

CM_LA_167xx_Source



Description: Waveform output using data downloaded from LA 167xx instrument
Library: Instruments
Class: SDFCM_LA_167xx_Source

Parameters

Name	Description	Default	Type
Instrument	instrument hostname or IP address	141.121.237.74	string
AnalyzerName	name of the analyzer module to be used	Analyzer< C>	string
LabelName	label name of data in analyzer memory	Label1	string
StartState	starting sample in analyzer	0	int
RunAnalyzer	run analysis before retrieving data: NO, YES	NO	enum
ScalingFactor	scale data from instrument before output	1.0	real
ControlSimulation	control simulation: NO, YES	NO	enum
Periodic	periodic output: NO, YES	YES	enum

Pin Outputs

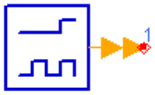
Pin	Name	Description	Signal Type
1	output		real

Notes/Equations

Note
On 64-bit Linux this component must run in 32-bit mode to avoid "socket error."

1. The CM_LA_167xx_Source model reads data attached to a specific label in a 167xx Logic Analyzer module plugged into a 16700 series Logic Analysis System mainframe.
2. Prerequisites for using the CM_LA_167xx_Source are
   - 167xx A/B Logic Analyzer module.
   - 16700 series Logic Analysis System mainframe; for information, visit the website http://www.agilent.com/find/logic and select *Modular System 16700 Series*.
   - LAN connectivity from your workstation to the Logic Analysis System mainframe.
3. The Instrument parameter identifies the Logic Analysis System mainframe that contains the Logic Analyzer as one of its modules. The mainframe is identified by its DNS hostname or IP address.
4. The AnalyzerName parameter identifies the Logic Analyzer module that contains the data. Multiple analyzer modules can be plugged into the same LA system.
5. The LabelName parameter specifies the label name in the Logic Analyzer that contains the data of interest.
6. The StartState parameter specifies which *state number* to use to start the waveform. States are basically samples in the Logic Analyzer; so if there are 128K samples (states) and the trigger is in the middle, the state numbers would go from −64K to +64K. Usually the trigger position is at the start of data, so state number 0 is the trigger point as well as the beginning of data. In general, the trigger point is when the analyzer receives a stimulus, say from the Event1 marker output of an ESG.
7. The RunAnalyzer parameter specifies whether the Logic Analyzer uses the current measurement or performs a new measurement. When RunAnalyzer = YES, the analyzer is forced to run a measurement and collect data before simulation begins. When RunAnalyzer = NO, data from the existing measurement in the analyzer is imported into ADS.
8. The ScalingFactor parameter specifies the scaling applied to the raw data imported from the analyzer before the data is used in the simulation.
9. If the length of simulation is larger than the available data in the analyzer, use the Periodic parameter to repeat the old data. If Periodic = YES, all data from the analyzer is read and repeated. If Periodic = NO, the output is zero after all data is read.
10. If ControlSimulation = YES, the simulation runs until the last data in the logic analyzer is read.

CM_LA_169xx_Source



Description: Waveform output using data downloaded from an analyzer that resides in a 169xx mainframe
Library: Instruments
Class: SDFCM_LA_169xx_Source

Parameters

Name	Description	Default	Type
InstrumentHostname	Logic Analysis System hostname or IP address	la16900	string
InstrumentSetupFile	name of instrument state file		string
ModuleName	name of the Analyzer module to be used	My 168x/9x-1	string
BusSignalNames	names of bus signals in analyzer memory	MyBus1	string array
StartingSampleNumber	index of starting sample in analyzer	DefaultNumericStart	int
EndingSampleNumber	index of ending sample in analyzer	DefaultNumericStop	int
RunBeforeCapture	run new measurement before retrieving data: NO, YES	NO	enum
RepeatData	zero-pad, repeat data: SinglePass, Repeat	Repeat	enum
ControlSimulation	control simulation: NO, YES	NO	enum
TimeoutInSeconds	The length of time to wait, in seconds, before giving up on data transfer	120	int

Pin Outputs

Pin	Name	Description	Signal Type
1	output		multiple int

Notes/Equations

Note
On 64-bit Linux this component must run in 32-bit mode to avoid "socket error."

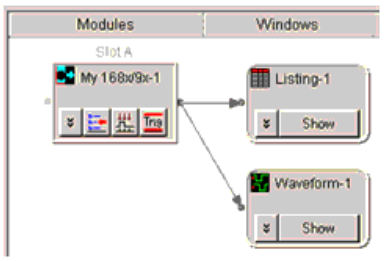
1. The CM_LA_169xx_Source model retrieves data from a Logic Analyzer module residing in a 169xx Logic Analysis (LA 169xx) mainframe and propagates the retrieved data into an ADS Ptolemy simulation.

   Note
   The notation *LA 169xx mainframe* is used here to represent all Agilent 16900-series logic analysis systems and 1680/1690-series logic analyzer mainframes. A 169xx Logic Analysis mainframe is essentially a Windows XP computer housed in a modular frame with slots that accept measurement (Analyzer) and stimulus (Pattern Generator) plug-in cards.
   For more information regarding logic analysis systems, options, and modules, visit the website http://www.agilent.com/find/logic .

2. CM_LA_169xx_Source uses the Agilent Connection Manager (CM) architecture to communicate with the instrument. Compared to other CM-based ADS Ptolemy instrument link models, this component has notable differences that are described here.
   CM_LA_169xx_Source communicates with the instrument by using the CM server to load a COM Automation Server that is pre-installed on LA 169xx mainframes (as part of the Agilent Logic Analyzer application.) The COM object provides a mechanism for controlling the Agilent Logic Analyzer application from remote computers on the LAN. To enable the CM_LA_169xx_Source to communicate with an LA 169xx mainframe, install the CM server on the LA 169xx mainframe itself.
   For details regarding installing the CM server on a Windows XP PC, refer to *Installing Connection Manager Server on Windows* (instalpc) in the *Windows Installation* (instalpc) documentation.

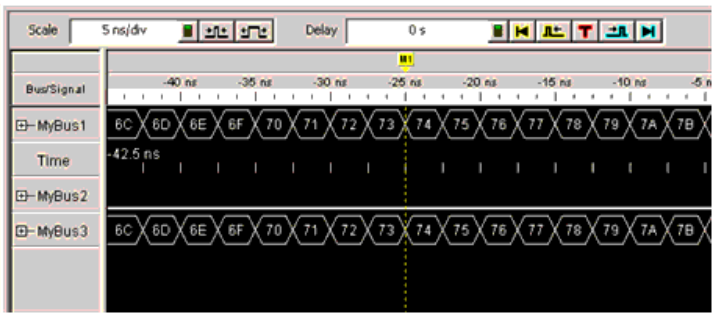
Note that also the Agilent Logic Analyzer application can be installed and used on a Windows XP/2000 computer for remote access of 16900- or 1680/1690-series logic analyzers on the network, or for offline analysis of data captured on 16900-, 1680/1690-, or 16700-series logic analyzers. In this setup, install the CM server on the same PC on which the Agilent Logic Analyzer application resides. This documentation refers to such a Windows XP/2000 computer as the LA 169xx mainframe and its file-system as the LA 169xx mainframe file-system, although it could be a standalone Windows XP/2000 computer running the Agilent Logic Analyzer application that emulates an LA 169xx mainframe.
CM_LA_169xx_Source does not enable interactive instrument selection using the Remote Instrument Explorer (a feature common to many CM-based instrument link models). This restriction is due to the nature of IO connectivity options available in the LA mainframes that the model interacts with.

3. Prerequisites for using CM_LA_169xx_Source are:
   - An LA 169xx mainframe with the Agilent Logic Analyzer application installed (Refer to *note 1* for the definition of an LA 169xx mainframe.)
   - An active Connection Manager server instance installed and running on the LA 169xx mainframe.
   - Network connectivity over the LAN from the workstation running ADS to the LA 169xx mainframe.
4. InstrumentHostname specifies the DNS hostname or IP address of the LA 169xx mainframe to be used.
5. Optionally, InstrumentSetupFile specifies an LA 169xx mainframe state file. The LA 169xx mainframe enables you to save the current instrument state and/or data into a file on its file-system. Any file path specified in InstrumentSetupFile must be relative to the instrument's file system. If a full path is not provided, the program assumes that the file exists in the default config directory (usually *C:\Documents and Settings\<username>\MyDocuments\Agilent Technologies\Logic Analyzer\config*).
6. ModuleName specifies the name of the analyzer module that you want to retrieve data from into the simulation. In Module Name Example, the module name would be *My 168x/9x-1*.



Module Name Example

7. BusSignalNames specifies a string array containing the names of the bus signals whose values you want to propagate into a simulation. The data output pin carrys one connection per bus signal name defined by this parameter. For example, to capture the bus signals in BusSignalNames Example, set BusSignalNames = MyBus1 MyBus2 MyBus3.



BusSignalNames Example

8. Signals are ordered from bottom to top on the schematic. To retrieve more than one

signal using CM_LA_169xx_Source, you can use a BusSplit component to control the order.
For example, when using BusSplit2 and BusSignalNames = MyBus1 MyBus2, the bottom output is MyBus1.

9. StartingSampleNumber specifies the index value that represents the first instrument data point you want to propagate into the simulation. This value can be any valid index as shown in Start and End Sample Number Example.
10. EndingSampleNumber specifies the index value that represents the last instrument data point you want to propagate into the simulation. This can be any valid index value (as shown in Start and End Sample Number Example) that is greater than the index value specified by StartingSampleNumber.

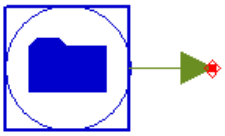
Start and End Sample Number Example

11. RunBeforeCapture specifies whether a new measurement run is initiated before data capture.
    - RunBeforeCapture = NO The simulation collects data from the instrument without a new measurement run.
    - RunBeforeCapture = YES The simulation collects data from the instrument after a new measurement run.
12. RepeatData instructs the simulator on what to do when all the data retrieved from the LA 169xx mainframe has been propagated into the simulation.
    - RepeatData = SinglePass Pad zeros until simulation end.
    - RepeatData = Repeat Repeat the same data until simulation end. CM_LA_169xx_Source retrieves data again from the LA 169xx mainframe and propagates it into the simulation. To run a measurement before retrieving the data each time it is repeated, set RunBeforeCapture = YES.
13. If ControlSimulation = YES, the number of data points specified (EndingSampleNumber - StartingSampleNumber + 1) determines how long the simulation runs.
14. CM_LA_169xx_Source has one output pin of type int. However, this is a multi-output pin and can be used to propagate as many integer outputs in parallel into the

simulation as needed. To see an example on how this can be set up using BusSignalNames, refer to *note 7*.
The Analyzer module in an LA 196xx mainframe stores data as bit vectors. For each simulation step, the bit vector associated with each signal is retrieved and interpreted as an integer. Further, no attempt is made to interpret the most significant bit as a sign bit if the vector is less than 32 bits wide. Any such interpretation can be done on the schematic using appropriate ADS Ptolemy models. Ptolemy interprets a vector with the most significant bit set to 1 as a negative integer only when the vector is exactly 32 bits wide.
To bring in data that is wider than 32 bits, split the data into multiple bus signals on the logic analyzer and propagate the data concurrently using multiple connections on the CM_LA_169xx_Source output.

15. To access example designs that use this model, from the ADS Main window, choose **File > Open > Example > Instruments > CM_LA_169xx_wrk**.

CM_N6030_FileRead



Description: N6030 Arbitrary Waveform file read
Library: Instruments
Class: SDFCM_N6030_FileRead

Parameters

Name	Description	Default	Type
FileNameI	Output baseband studio file name for In-channel data	file_i.bin	filename
FileNameQ	Output baseband studio file name for Quadrature-channel data	file_q.bin	filename
ControlSimulation	Control simulation: NO, YES	YES	enum
Periodic	Periodic output: NO, YES	YES	enum

Pin Outputs

Pin	Name	Description	Signal Type
1	output		complex

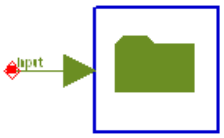
Notes/Equations

1. CM_N6030_FileRead can read Arbitrary Waveform Generator file for N6030 instrument. Agilent's N6030 series of arbitrary waveform generators deliver unprecedented performance for creation of complex wideband waveforms. High sampling rate and high bit resolution provided in a single instrument enable designers to create ideal waveforms for accurate test of radar, satellite and frequency agile systems. For example, each channel of the N6030A provides 500 MHz of modulation bandwidths and over 65 dBc of spurious free dynamic range. When the N6030A is combined with a wideband I/Q upconverter, modulation bandwidth of 1 GHz can be realized at microwave frequencies for authentic signal simulations for IF and RF subsystem test.
2. This component as an ADS Source can read in recorded baseband IQ data from two files (16 bit I, Q data) and create a complex signal with IQ components compatible with the Baseband Studio product. The data is in long, unique test signals, up to hours in length directly.
3. The CM_N6030_FileWrite and CM_N6030_FileRead components can be used to save and playback long I/Q sequences within ADS.
4. If the length of simulation is larger than the available data in the file, use the Periodic parameter to repeat the old data. If Periodic=YES, all data from the file will be read

and repeated. If Periodic=NO, the output will be zero after all data is read.

5. If ControlSimulation=YES, the simulation will run from Start to Stop-Start+1.

CM_N6030_FileWrite



Description: N6030 Arbitrary Waveform Generator file writer
Library: Instruments
Class: SDFCM_N6030_FileWrite
Derived From: CM_BinaryArbWrite

Parameters

Name	Description	Default	Type	Range
Start	sample number to start waveform recording	DefaultNumericStart	int	
Stop	sample number to stop waveform recording	DefaultNumericStop	int	
FileNameI	Output baseband studio file name for In-channel data	file_i.bin	filename	
FileNameQ	Output baseband studio file name for Quadrature-channel data	file_q.bin	filename	
ControlSimulation	control simulation: NO, YES	YES	enum	
AutoScaling	scale input data to fit dynamic range of ESG DAC: NO, YES	YES	enum	
FullScaleFactor	Full scale factor for peak value, when autoscale is used	1.0	real	(0, 1]

Pin Inputs

Pin	Name	Description	Signal Type
1	input		complex

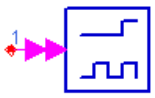
Notes/Equations

1. CM_N6030FileWrite can write Arbitrary Waveform Generator file for N6030 instrument. Agilent's N6030 series of arbitrary waveform generators deliver unprecedented performance for creation of complex wideband waveforms. High sampling rate and high bit resolution provided in a single instrument enable designers to create ideal waveforms for accurate test of radar, satellite and frequency agile systems. For example, each channel of the N6030A provides 500 MHz of modulation bandwidths and over 65 dBc of spurious free dynamic range. When the N6030A is combined with a wideband I/Q upconverter, modulation bandwidth of 1 GHz can be realized at microwave frequencies for authentic signal simulations for IF and RF subsystem test.
2. This component as an ADS Sink can be used to create a signal with IQ components compatible with the Baseband Studio product. The file generated by this sink can be transferred to a PC configured for use baseband signal with N6030. The data is in long, unique test signals, up to hours in length directly.
3. The component has one complex input: an input pin for complex I/Q data. The outputs can be written in two files: one for I data and another for Q data that are formatted to: 0 Markers; 16 Bit I, Q.
4. If ControlSimulation=YES, the simulation will run from Start to Stop-Start+1.
5. The AutoScaling parameter specifies if input scaling is to be performed. When AutoScaling=YES, the input I/Q data is scaled to fit dynamic range of N6030 DAC range, and the full scaling factor is displayed towards the end of the simulation. When AutoScaling=NO, no scaling is performed on the input data; however, data outside the range {-1, 1} will be clipped to -1 or 1.

6. Full scale factor is for peak value, when autoscale is used.

> **Note**
> The output data file with Baseband Studio file format contains only data values as 16-bit integers with no marker embedded in each data value. It does not contain any header information such as Amplitude, Frequency, and SampleClk. If required, these instrument settings must be controlled from the instrument's front panel or by using appropriate SCPI commands.

CM_PatGen_169xx_Sink



Description: Downloads data to a pattern generator module housed in a 169xx mainframe
Library: Instruments
Class: SDFCM_PatGen_169xx_Sink

Parameters

Name	Description	Default	Unit	Type	Range
InstrumentHostname	Logic Analysis System hostname or IP address	la16900.wlv.agilent.com		string	
InstrumentSetupFile	name of instrument state file			filename	
ModuleName	name of the PattGen module to be used	My 16720A-1		string	
BusSignalNames	names of bus signals in analyzer memory	MyBus1		string array	
Start	sample number to start data recording	DefaultNumericStart		int	
Stop	sample number to stop data recording	DefaultNumericStop		int	
PGBFileName	name of pattern generator binary file	PattGen.pgb		filename	
OverwriteExistingFile	overwrite if file exists: NO, YES	YES		enum	
OutputMode	output channel mode: HalfChannel, FullChannel	HalfChannel		enum	
ClkSource	clock source: Internal, External	Internal		enum	
IntClkFrequency	internal clock frequency	180e6	Hz	real	(1, 300e6)
ClkOutDelay	delay the clock by multiple of 500 ps	0		int	(0, 14)
RunAfterLoading	clock data out of pattgen after download: NO, YES	YES		enum	
RunMode	cycle through data once or repeat data: SinglePass, Repeat	Repeat		enum	
InitSequenceFile	text file containing initial sequence data			filename	
ControlSimulation	control simulation: NO, YES	YES		enum	

Pin Inputs

Pin	Name	Description	Signal Type
1	input		multiple fix

Notes/Equations

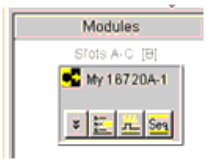
> **Note**
> On 64-bit Linux this component must run in 32-bit mode to avoid "socket error."

1. The CM_PatGen_169xx_Sink model collects data from an ADS Ptolemy simulation

and downloads the data into a Pattern Generator module residing in a 169xx Logic Analysis (LA 169xx) mainframe.

> **Note**
> The notation *LA 169xx mainframe* is used here to represent all Agilent 16900-series logic analysis systems and 1680/1690-series logic analyzer mainframes. A 169xx Logic Analysis mainframe is essentially a Windows XP computer housed in a modular frame with slots that accept measurement (Analyzer) and stimulus (Pattern Generator) plug-in cards.
> For more information regarding logic analysis systems, options, and modules, visit the website http://www.agilent.com/find/logic .

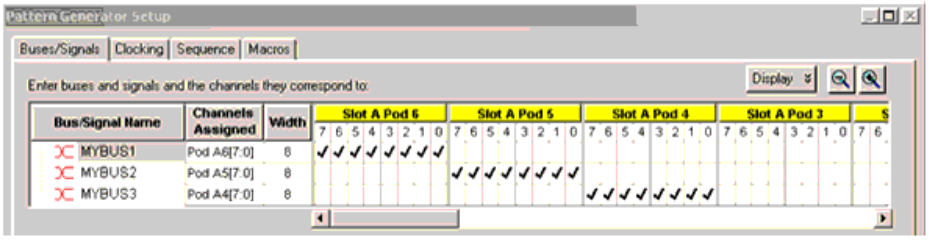
2. CM_PatGen_169xx_Sink uses the Agilent Connection Manager (CM) architecture to communicate with the instrument. Compared to other CM-based ADS Ptolemy instrument link models, this component has notable differences that are described here.
   CM_PatGen_169xx_Sink communicates with the instrument by using the CM server to load a COM Automation Server that is pre-installed on LA 169xx mainframes (as part of the Agilent Logic Analyzer application.) The COM object provides a mechanism for controlling the Agilent Logic Analyzer application from remote computers on the LAN. To enable CM_PatGen_169xx_Sink to communicate with an LA 169xx mainframe, install the CM server on the LA 169xx mainframe itself.
   For details regarding installing the CM server on a Windows XP PC, refer to *Installing Connection Manager Server on Windows* (instalpc) in the *Windows Installation* (instalpc) documentation.
   Note that also the Agilent Logic Analyzer application can be installed and used on a Windows XP/2000 computer for remote access of 16900- or 1680/1690-series logic analyzers on the network, or for offline analysis of data captured on 16900-, 1680/1690-, or 16700-series logic analyzers. In this setup, you must install the CM server on the same PC on which the Agilent Logic Analyzer application resides. This documentation refers to such a Windows XP/2000 computer as the LA 169xx mainframe and its file-system as the LA 169xx mainframe file-system, although it could be a standalone Windows XP/2000 computer running the Agilent Logic Analyzer application that emulates a LA 169xx mainframe.
   CM_PatGen_169xx_Sink does not enable interactive instrument selection using the Remote Instrument Explorer (a feature common to many CM-based instrument link models). This restriction is due to the nature of IO connectivity options available in the LA mainframes that the model interacts with.
3. Prerequisites for using CM_PatGen_169xx_Sink are:
   - An LA 169xx mainframe with the Agilent Logic Analyzer application installed (Refer to *note 1* (instruments) for the definition of an LA 169xx mainframe.)
   - An active Connection Manager server instance installed and running on the LA 169xx mainframe.
   - Network connectivity over the LAN from the workstation running ADS to the LA 169xx mainframe.
4. InstrumentHostname specifies the DNS hostname or IP address of the LA 169xx mainframe to be used.
5. InstrumentSetupFile optionally specifies an LA 169xx mainframe state file that should be loaded before simulation begins. The LA 169xx mainframe enables you to save the current instrument state and/or data into a file on its file-system. Any file path specified in InstrumentSetupFile must be relative to the instrument's file system. If a full path is not provided, the program assumes that the file exists in the default config directory (usually *C:\Documents and Settings\<username>\MyDocuments\Agilent Technologies\Logic Analyzer\config*).
6. ModuleName specifies the name of the Pattern Generator module that you want to send data to from the simulation. In Module Name Example, the module name would be *My 16720A-1*.



Module Name Example

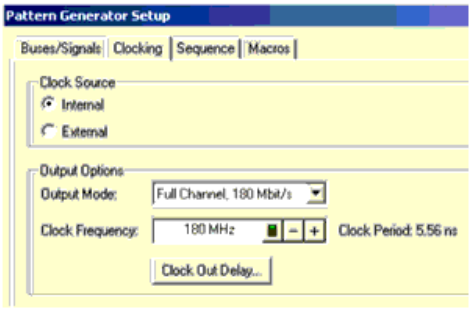
7. BusSignalNames specifies a string array containing the names you want to assign to

the individual bus signals. Each signal is defined to be the width of the corresponding fixed-point data value on the input pin. For example, to assign incoming data from 3 connections at the input pin to 3 corresponding bus signals as in BusSignalNames Example, set BusSignalNames = MyBus1 MyBus2 MyBus3.



BusSignalNames Example

8. Signals are ordered from bottom to top on the schematic. To download more than one signal using the CM_PatGen_169xx_Sink, use a BusMerge component to control the order.
   For example, when using a BusMerge and BusSignalNames = MyBus1 MyBus2, the bottom input is assigned the bus signal name MyBus1.
9. Start specifies the sample number of the first simulation data point you want to propagate into the instrument.
10. Stop specifies the sample number of the first simulation data point you want to propagate into the instrument. Stop can be any valid integer greater than Start.
11. PGBFileName specifies the name of the file on the instrument file system that holds the sequence data. If a full path is provided, the path must be relative to the LA 169xx mainframe file-system.
12. OverwriteExistingFile = YES instructs the pattern generator module to overwrite an existing file on the LA 169xx mainframe file-system whose name matches PGBFilename. OverwriteExistingFile = NO forces a simulation error if a file name matching InstrumentSetupFile is already on the LA 169xx mainframe file system.
13. OutputMode specifies the mode to put the pattern generator in to clock the data out. OutputMode = FullChannel limits the data rate to a maximum of 180 MHz, but enables 48 channels per card. OutputMode = HalfChannel enables an output rate of up to 300 MHz, but limits the number of channels to 24 per card.



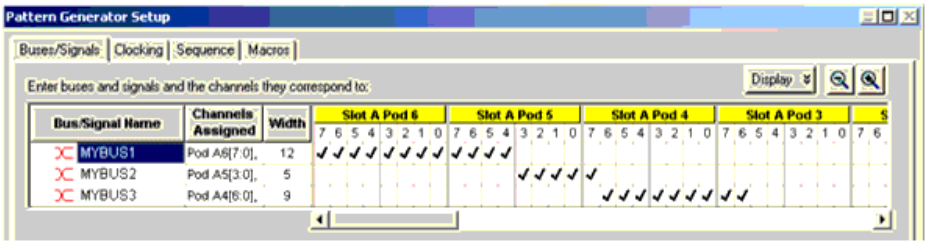
Output Mode Selection

14. ClkSource specifies the clock source to be used (Internal or External as shown in Output Mode Selection) to pace the pattern generator when moving data from its internal memory to its data pins.
15. IntClkFreq, used when ClkSource = Internal, specifies the rate at which data is clocked out of the pattern generator data pins.
16. ClkOutDelay specifies the positioning of the output clock relative to data signals. Setting this to a value greater than 0 increments the clock delay by the value multiplied by 500 ps.
17. RunAfterLoading specifies (NO, YES) whether you want to clock data out of the pattern generator after downloading data.
18. RunMode, used when RunAfterLoading = YES, specifies whether the pattern generator should clock one cycle of data from its data pins (RunMode = SinglePass) or cycle the main sequence repetitively (RunMode = Repeat).
19. InitSequenceFile specifies a filename that holds the data defining the initial sequence

to be played prior to clocking the main sequence. The text file holds one integer value per line you want in the initial sequence; such a file can be generated using the Printer component in ADS Ptolemy. Note that ideally, the width of the integer data should be the same as the total width of the incoming data for all bus signals defined. However, because the incoming data could be wider than the maximum number of bits that can be represented by an integer, the vectors in the init sequence is zero-padded up to the required width.

> **Important**
> The number of data points specified in the init sequence file must be a multiple of 4 in half-channel mode and a multiple of 2 in full-channel mode. If this condition is not met, the additional data points in the file are discarded and a warning stating the same is generated during simulation.

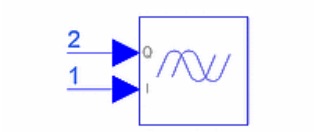
20. ControlSimulation is used to specify (NO, YES) whether this component is to control the simulation lifetime.
21. CM_PatGen_169xx_Sink has a fixed-point multi-input pin. Each incoming fixed-point data can be of arbitrary width, but the width must be constant over a simulation run. If the total bit-widths of incoming data corresponding to all specified bus signals is greater than the vector bit-width that can be supported by the Pattern Generator module, an error is reported.
    When multiple bus signals are defined, incoming data is packed together without leaving any gaps between the bus signals. The bits are always packed starting from the most significant bit on the Pattern Generator.
    For example, in Bus Signals Are Packed Consecutively Across Byte/Slot/Pod Boundaries the incoming bus signal data widths for bus signals MyBus1, MyBus2, and MyBus3 are 12, 5, and 9, respectively. Bus Signals Are Packed Consecutively Across Byte/Slot/Pod Boundaries shows that bus signal assignments begin from the highest available slot (slot A in this example) on its highest available pod number (pod 6 in this example).



Bus Signals Are Packed Consecutively Across Byte/Slot/Pod Boundaries

22. To access example designs that use this model, from the ADS Main window, choose **File > Open > Example > Instruments > CM_LA_169xx_wrk**.

CM_PSG_E8267C_Sink



Description: Agilent PSG E8267C Vector Signal Generator
Library: Instruments
Class: SDFCM_PSG_E8267C_Sink
Derived From: CM_ESG_E4438C_Sink

Parameters

Name	Description	Default	Unit	Type
Instrument	instrument to be used in the simulation	[GPIB0::21::INSTR][localhost][4790]		instrument
Enabled	When YES communicates with instrument: NO, YES	YES		enum
Start	sample number to start waveform recording	DefaultNumericStart		int
Stop	sample number to stop waveform recording	DefaultNumericStop		int
Frequency	RF output frequency	3e9	Hz	real
Amplitude	RF output power level (dBm)	-135		real
ARBRef	reference for the waveform clock: Internal, External	Internal		enum
ARBRefFreq	reference frequency of the external clock generator	10e6	Hz	real
SampleClk	sample clock rate for sequencer and DAC	4.9152e6	Hz	real
FileName	waveform file name to be stored in PSG	ads_psg_download		filename
DownloadMode	select mode of operation of the sink: download_to_VARB, write_to_datafile, both	download_to_VARB		enum
AutoScaling	scale inputs to range { -1, +1 } before download: NO, YES	YES		enum
ArbOn	select waveform and turn ARB ON after download: NO, YES	NO		enum
RFPowOn	turn RF power ON after download: NO, YES	NO		enum
EventMarkers	Enable PSG markers: Neither, Event1, Event2, Both	Neither		enum
MarkerLength	select number of points for ESG markers	10		int
IQ_ModFilter	baseband filter applied to the IQ signals: through, filter_40MHz	through		enum

Pin Inputs

Pin	Name	Description	Signal Type
1	I	I input waveform data	real
2	Q	Q input waveform data	real

Notes/Equations

Note
On 64-bit Linux this component must run in 32-bit mode to avoid "socket error."

1. The CM_PSG_E8267C_Sink model collects data from a simulation and downloads the data to an E8267C Vector Signal Generator. Parameters specify how data is interpreted by the PSG. This model uses the connection manager architecture to communicate with the instrument. See *Client-Server Architecture* (connectmui) under the section on *Operational and Functional Concepts* (connectmui) in the *Connection Manager* (connectmui) documentation for more information.
2. Prerequisites for using CM_PSG_E8267C_Sink are:
   - PSG Vector Signal Generator E8267C; for information, visit the website http://www.agilent.com/find/psg .
   - Supported method of connecting the instrument to your computer through the Connection Manager architecture. For detailed setup information, refer to *Configuring the Server IO on the PC* (connectmui) under the section on *Getting Started with Connection Manager* (connectmui) in the *Connection Manager* (connectmui) documentation.
3. The Instrument parameter specifies a triplet that identifies the specific instrument to be used in the simulation, along with the Connection Manager server information to be used to connect to the instrument.
   To interactively select an instrument, refer to *Instrument Selection* (instruments)

under *About Instruments Components* (instruments).
For more information regarding instrument selection, see *Discovering Connected Hardware* (connectmui) under the section on *Getting Started with Connection Manager* (connectmui) in the *Connection Manager* (connectmui) documentation.

4. The Start and Stop parameters specify when to start and stop data collection. The number of samples collected, Stop − Start + 1, must be in the range 60 samples to 64 Msamples where 1 Msample = 1048576 samples. The PSG requires an even number of samples; the last sample is discarded if Stop − Start + 1 is odd.
5. The ARBRef parameter specifies an internal or external reference for the PSG clock generator. If set to *External*, the ARBRefFreq parameter sets the frequency of this clock.
6. The SampleClk parameter sets the sample clock rate for the DAC output.
7. The FileName parameter sets the name of the file into which data is written. Data can be written to the ESG, to a local data file, or to both, based on the DownloadMode parameter. The same filename is used for both options.
8. The DownloadMode parameter specifies the mode of operation of this sink component.
   download_to_VARB - The waveform is downloaded to the ESG volatile ARB (VARB) memory.
   To avoid writing data to the local file system, use this mode of operation.
   write_to_datafile - The waveform is written to a proprietary encrypted binary file format in the data directory of the workspace. Then the file can be transferred to the WAVEFORM directory in the ESG file system using an FTP client; select the binary data transfer option during the FTP process. The ESG automatically decrypts the file and makes the file available for selection from the list of waveforms in its VARB memory.
   To avoid communication with the ESG instrument during simulation, use this mode of operation.
   Note that, besides data, only the Frequency and SampleClk values are embedded in the file as header information. Other ESG settings (such as Amplitude) must be set manually from the ESG front panel or by using appropriate SCPI commands.
   both - *download_to_VARB* and *write_to_datafile* options are enabled.
9. The ESG driver expects data in the range {−1, 1}. The AutoScaling parameter specifies whether to scale inputs to fit this range.
   If AutoScaling = YES, inputs are scaled to the range {−1, 1}. And, scaling is applied to data written to the local file (refer to *note 8*).
   If AutoScaling = NO, raw simulation data is downloaded to the ESG without any scaling, but data outside the range {−1, 1} is clipped to −1 or 1.
10. If ArbOn = YES, the PSG starts generating the signal immediately after simulation data is downloaded; if ArbOn = NO (default), waveform generation must be turned on at the PSG front panel.
11. If RFPowOn = YES, the PSG turns on RF power immediately after simulation data is downloaded; if RFPowOn = NO (default), RF power must be turned on at the PSG front panel.
12. The EventMarkers parameter specifies which PSG Event markers are enabled: *Event1* , *Event2*, *Both*, or *Neither*. Event markers are used for synchronizing other instruments to the PSG. When one or both EventMarkers are enabled, *Event1* and/or *Event2* is set to the first sample of the downloaded Arb waveform. This is equivalent to setting the corresponding event from the front panel of the PSG.
13. The MarkerLength parameter specifies the range of points over which the markers must be set starting from the first point of the waveform. Depending on the setting of the EventMarkers parameter, the length of trigger *Event1* and/or *Event2* is set to a multiple of the pulsewidth which, in turn, is determined by the sample clock rate of the DAC output.
14. The RecFilter parameter specifies the cutoff frequency for the reconstruction filter that lies between the DAC output and the Dual Arbitrary Waveform Generator output inside the ESG.
15. To access example designs that use this model, from the ADS Main window, choose **File > Open > Example > Instruments > CM_PSG_wrk**.

CM_SCPI

Description: ADS Ptolemy link to send SCPI commands to any instrument
Library: Instruments
Class: SDFCM_SCPI

Parameters

Name	Description	Default	Type
Instrument	instrument to be used in the simulation	[GPIB0::19::INSTR][localhost][4790]	instrument
SetupCommands	commands to send at simulation setup time		string
BeginCommands	commands to send at simulation begin time		string
WrapupCommands	commands to send at simulation wrapup		string
SendIDCommand	send SCPI "*IDN" command to instrument (select NO for non-SCPI instrument): NO, YES	YES	enum
Enabled	When YES communicates with instruments: NO, YES	YES	enum

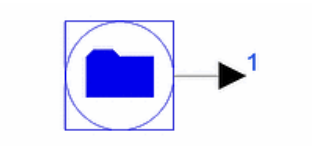
Notes/Equations

Note
On 64-bit Linux this component must run in 32-bit mode to avoid "socket error."

1. CM_SCPI sends arbitrary commands to an instrument at the beginning and end of simulation. This model can be used for configuration of an instrument beyond what ADS Ptolemy models provide. This model uses the connection manager architecture to communicate with the instrument. See *Client-Server Architecture* (connectmui) in the *Connection Manager* (connectmui) documentation for more information.
2. Prerequisites for using CM_SCPI are:
   - Any instrument that supports commands in ASCII text format; for example, instruments that comply with the IEEE 488.2 standard.
   - Supported method of connecting the instrument to your computer through the Connection Manager architecture. For detailed setup information, refer to *Configuring the Server IO on the PC* (connectmui) in the *Connection Manager* (connectmui) documentation.
3. The Instrument parameter specifies a triplet that identifies the specific instrument to be used in the simulation, along with the Connection Manager server information to be used to connect to the instrument.
   To interactively select an instrument, refer to *Instrument Selection* (instruments). For more information regarding instrument selection, see *Discovering Connected Hardware* (connectmui) in the *Connection Manager* (connectmui) documentation.
4. Commands sent to the instrument are echoed to the status server for confirmation. Enclose variable names inside your commands with curly braces. For example, if you have a variable F signifying the frequency, setting BeginCommands to *:FREQ {F} Hz* sets the instrument frequency.
5. The SendIDCommand parameter provides the flexibility to use this SCPI model to send commands to instruments that are not SCPI compliant. SendIDCommand = YES by default, which causes the SCPI identification command *IDN? to be transmitted during initialization. Set SendIDCommand to NO when used with non-SCPI instruments.
6. The Enabled parameter, when set to YES, enables CM_SCPI to log information into the simulation status window and communicate with instruments; when set to NO, logging and communication activities are disabled.
7. Refer to the appropriate instrument documentation for commands that correspond to

the settings you want. Use semicolons to separate multiple commands. If you send a query command (one containing a question mark), CM_SCPI reads the result and display the result in the status window.

CM_SStudioFileRead



Description: Time domain signal generator with signalstudio encrypted file based data
Library: Instruments

Parameters

Name	Description	Default	Unit	Type	Range
FileName	Input file name	esg.wfm		filename	
TStep	simulation time step	Derived from source data	sec	real enum	{-1} or [0,∞)
ControlSimulation	control simulation: NO, YES	NO		enum	
SegmentLength	if non-zero, truncate or zero pad source data to this length	0	sec	real	[0,∞)
RepeatData	control operation at the end of source data: Single pass, Repeat	Single pass		enum	
InterpolationType	interpolation technique to use: Lagrange, Sample and hold, Linear	Lagrange		enum	
InterpolationOrder	number of points to use if Lagrange interpolation is set	4		int	[2,∞)
UseFileFCarrier	Determines whether to use the carrier frequency in the WFM file or FCarrier parameter: NO, YES	YES		enum	
FCarrier	Carrier frequency used when UseFileFCarrier is set to NO	1000000.0	Hz	real	[0,∞)

Pin Outputs

Pin	Name	Description	Signal Type
1	output	output signal	timed

Notes/Equations

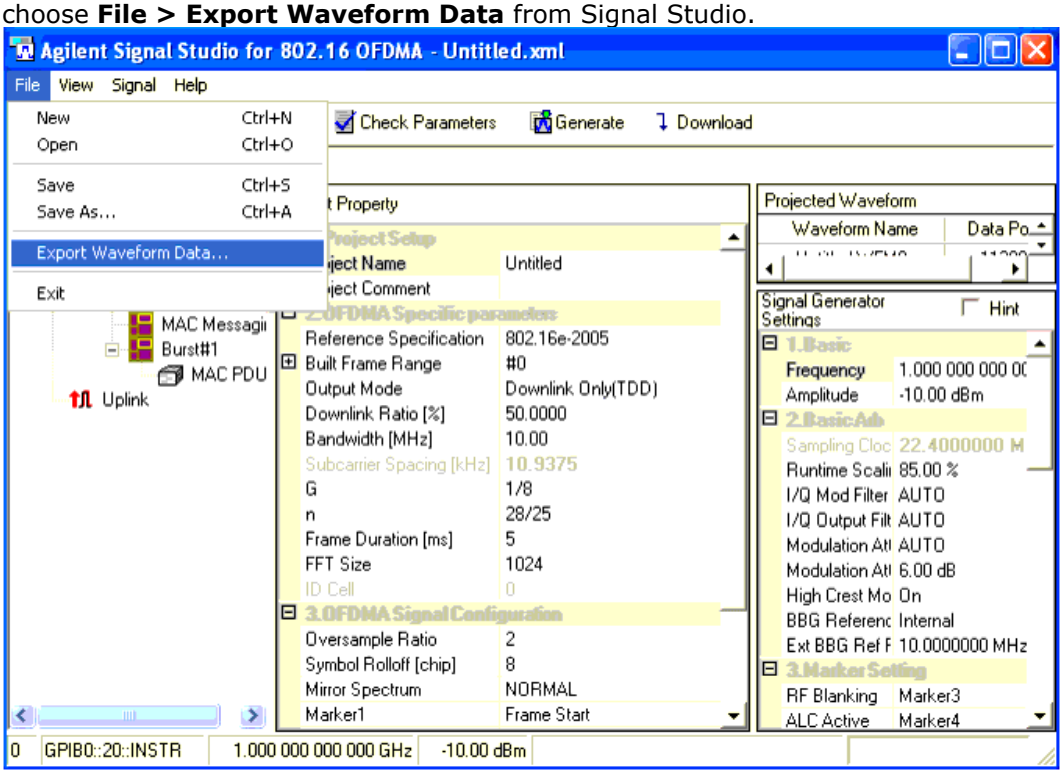
1. The CM_SStudioFileRead component is used to generate timed data output evaluated using IQ data from a Signal Studio waveform (.wfm) file. This type of file can be created by:
   - Using the CM_ESG_E4438C_Sink or CM_PSG_E8267C_Sink components (see DownloadMode parameter) or
   - Using the next generation Signal Studio personalities.
     Signal Studio is an Agilent product that consists of a suite of PC-based software tools used to create waveforms for popular communication formats. Signal Studio has an intuitive, easy-to-use Graphical User Interface to set various signal parameters for flexible waveform generation.
     Agilent's Signal Studio software can be downloaded from the Signal Studio website at http://www.agilent.com/find/signalstudio . The Signal Studio models with different Personalities and Product Model Numbers listed in *Signal Studio Personalities Compatible with this Component* support exporting the generated waveform signal into an encrypted-binary file, a wfm file. This component reads a wfm file and outputs the simulation signal to test your designs. In the simulation, this model also checks the waveform personality and pulls the corresponding ADS license as shown in *Signal Studio Personalities Compatible with this Component*.

Signal Studio Personality	E4438C ESG Compatible Signal Studio Model Number	E8267D PSG Compatible Signal Studio Model Number	N5182A MXG Compatible Signal Studio Model Number	ADS Wireless Library License Pulled
3GPP W-CDMA with HSDPA/HSUPA	N7600B	N7600B	N7600B	WCDMA3G
IS-95, cdma2000, 1xEV-DO	N7601B	N7601B	N7601B	CDMA1xEV
GSM and EDGE	N7602B	N7602B	N7602B	EDGE
TD-SCDMA	N7612B	N/A	N7612B	TDSCDMA
802.16-2004	N7613A	N7613A	N7613A	WLAN_Fixed
802.16 OFDMA	N7615B	N7615B	N7615B	WLAN_Mobile
802.11 WLAN	N7617B	N7617B	N7617B	WLAN
DVB	N7623B	N7623B	N7623B	DTV
Bluetooth	E4438C-406	N/A	N/A	WLAN
3GPP LTE	N7624B	N/A	N7624B	LTE

Some of the Signal Studio models are not shown in *Signal Studio Personalities Compatible with this Component* above. These models typically apply to one of the following cases:

- Some Signal Studio models; such as T-DMB and Pulse Building, do not support exporting wfm files to ADS.
- Embedded software models that only work with ESG hardware, and there is no capability to export wfm files to ADS.

To export the data to Advanced Design System from one of these applications, choose **File > Export Waveform Data** from Signal Studio.

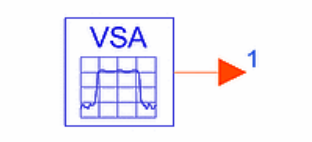


2. FileName specifies the encrypted binary wfm file from which the signal is to be read. The default location for the wfm file is the data directory of the current ADS workspace.
3. For more information on setting TStep parameter, refer to the *Timed Sources* (timed) Introduction. Additionally, this component supports a TStep value of −1 that automatically sets TStep to the sampling period specified in the wfm file.
4. If ControlSimulation = YES and RepeatData = Single pass, the simulation runs until the last data from the datafile is read.
5. The InterpolationType and InterpolationOrder parameters are used only when TStep and the datafile sampling period are not equal.

Note
The following note is applicable only when the datafile used was created by using the *write_to_datafile* option of the *CM_ESG_E4438C_Sink* component. This sink normalizes the signal to the {−1, 1} range. In order to enable the recovery of the signal voltage level, the scaling factor used for normalization `(abs(max(I,Q)))` is embedded in the header of the datafile. When this file is used with the CM_SStudioFileRead component, the embedded scaling factor value is used to restore the signal to its original level.

6. UseFileFCarrier determines whether to use carrier frequency specified in the input file or in the parameter FCarrier. If UseFileFCarrier is set to YES the file must have the carrier frequency.
7. FCarrier is used when UseFileFCarrier is set to NO. In that case FCarrier overrides carrier frequency in the input file.

CM_VSA_E444xA_Source



Description: Agilent E444xA Vector Signal Analyzer
Library: Instruments
Class: TSDFCM_VSA_E4406A_Source

Parameters

Name	Description	Default	Unit	Type
Instrument	instrument to be used in the simulation	[GPIB0::17::INSTR][localhost][4790]		instrument
Measurement	measurement type: Measured time (timed), Measured spectrum (freq complex)	Measured time (timed)		enum
ControlSimulation	control simulation: NO, YES	NO		enum
RepeatData	repeat data or resample instrument: Repeat, Resample	Repeat		enum
UseCurrentSettings	use current instrument settings: NO, YES	YES		enum
CenterFrequency	set center frequency	100 MHz	Hz	real
Span	set frequency span	1 MHz	Hz	real
ResBW	set resolution bandwidth	100 kHz	Hz	real
SweepTime	set sweep time	2 msec	sec	real
VSA_Trigger	set VSA trigger type: Immediate, Front, Rear	Immediate		enum

Pin Outputs

Pin	Name	Description	Signal Type
1	out	data	anytype

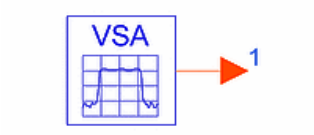
Notes/Equations

Note
On 64-bit Linux this component must run in 32-bit mode to avoid "socket error."

1. The CM_VSA_E444xA_Source model downloads data from any E444xA VSA series instrument when used in vector signal analyzer mode. It outputs different amounts and types of data depending on which measurement you are performing. This model uses the connection manager architecture to communicate with the instrument. See *Client-Server Architecture* (connectmui) in the *Connection Manager* (connectmui) documentation for more information.

2. Prerequisites for using CM_VSA_E444xA_Source are:
   - E444x6A Performance Spectrum Analyzer; for information, visit the website http://www.agilent.com/find/psa .
   - Supported method of connecting the instrument to your computer through the Connection Manager architecture. For detailed setup information, refer to *Configuring the Server IO on the PC* (connectmui) in the *Connection Manager* (connectmui) documentation.
3. The Instrument parameter specifies a triplet that identifies the specific instrument to be used in the simulation, along with the Connection Manager server information to be used to connect to the instrument.
   To interactively select an instrument, refer to *Instrument Selection* (instruments).
   For more information regarding instrument selection, see *Discovering Connected Hardware* (connectmui) in the *Connection Manager* (connectmui) documentation.
4. The Measurement parameter controls what data this model downloads. This model configures its output port based on data.
   Measured time (timed) - IQ measured time data is downloaded and output as timed baseband data in volts.
   Measured spectrum (freq complex) - IQ measured spectrum data is downloaded and output as pairs of complex numbers. The first number is the real frequency (the complex part is zero), and the second number is the complex voltage at that frequency. To separate the signal, place a Distributor2 component at the VSA E444xA output.
5. The ControlSimulation parameter determines how this model acts inside the simulation. When ControlSimulation = YES, this model keeps the simulation running during the first pass through the data. If your sinks are set to not control the simulation, these automatically collect all data from this model. When ControlSimulation = NO, this model produces data *ad infinitum*, and you must set your sink to control the simulation, then set the Start and Stop parameters to appropriate values.
6. When RepeatData = Repeat, this model downloads data from the instrument once and repeats it; when RepeatData = Resample, this model continues to download data from the instrument. Note that resampled time-based data is not continuous.
7. To access example designs that use this model, from the ADS Main window, choose **File > Open > Example > Instruments > CM_VSA_wrk**.

CM_VSA_E4406A_Source



Description: Agilent E4406A Vector Signal Anayzer
Library: Instruments
Class: TSDFCM_VSA_E4406A_Source
Derived From: WVS

Parameters

Name	Description	Default	Unit	Type
Instrument	instrument to be used in the simulation	[GPIB0::17::INSTR][localhost][4790]		instrument
Measurement	measurement type: Measured time (timed), Measured spectrum (freq complex)	Measured time (timed)		enum
ControlSimulation	control simulation: NO, YES	NO		enum
RepeatData	repeat data or resample instrument: Repeat, Resample	Repeat		enum
UseCurrentSettings	use current instrument settings: NO, YES	YES		enum
CenterFrequency	set center frequency	100 MHz	Hz	real
Span	set frequency span	1 MHz	Hz	real
ResBW	set resolution bandwidth	100 kHz	Hz	real
SweepTime	set sweep time	2 msec	sec	real
VSA_Trigger	set VSA trigger type: Immediate, Front, Rear	Immediate		enum

Pin Outputs

Pin	Name	Description	Signal Type
1	out	data	anytype

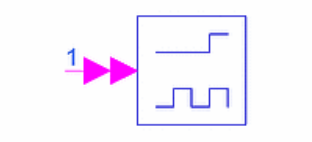
Notes/Equations

Note
On 64-bit Linux this component must run in 32-bit mode to avoid "socket error."

1. The CM_VSA_E4406A_Source model downloads data from any E4406A vector signal analyzer. It outputs different amounts and types of data depending on which measurement you are performing. This model uses the connection manager architecture to communicate with the instrument. See *Client-Server Architecture* (connectmui) in the *Connection Manager* (connectmui) documentation for more information.
2. Prerequisites for using CM_VSA_E4406A_Source are:
   - E4406A Vector Signal Analyzer; for information, visit the website http://www.agilent.com/find/vsa .
   - Supported method of connecting the instrument to your computer through the Connection Manager architecture. For detailed setup information, refer to *Configuring the Server IO on the PC* (connectmui) in the *Connection Manager* (connectmui) documentation.
3. The Instrument parameter specifies a triplet that identifies the specific VSA instrument to be used in the simulation, along with the Connection Manager server information to be used to connect to the instrument.
   To interactively select an instrument, refer to *Instrument Selection* (instruments).
   For more information regarding instrument selection, see *Discovering Connected Hardware* (connectmui) in the *Connection Manager* (connectmui) documentation.
4. The Measurement parameter controls what data this model downloads. The model configures its output port based on data.
   Measured time (timed) - IQ measured time data is downloaded and output as timed baseband data in volts.
   Measured spectrum (freq complex) - IQ measured spectrum data is downloaded and output as pairs of complex numbers. The first number is the real frequency (the complex part is zero), and the second number is the complex voltage at that frequency. To separate the signal, place a Distributor2 component at the VSA E4406A output.
5. The ControlSimulation parameter determines how this model acts inside the simulation. When ControlSimulation = YES, this model keeps the simulation running during the first pass through the data. If your sinks are set to not control the simulation, these automatically collects all data from this model. When ControlSimulation = NO, this model produces data ad infinitum, and you must set your sink to control the simulation, then set the Start and Stop parameters to appropriate values.

6. When RepeatData = Repeat, this model downloads data from the instrument once and repeats it; when RepeatData = Resample, this model continues to download data from the instrument. Note that resampled time-based data is not continuous.
7. To access example designs that use this model, from the ADS Main window, choose **File > Open > Example > Instruments > CM_VSA_wrk**.

PatGen_16522A_Sink



Description: Agilent 16522A Pattern Generator
Library: Instruments
Class: SDFPatGen_16522A_Sink

Parameters

Name	Description	Default	Unit	Type	Range
Interface	IP address	192.0.0.1		string	
Login	Login name and password separated by colon	anonymous:ads		string	
Slot	Slot of pattern generator card in mainframe	A		string	
Start	Sample number to start pattern recording	DefaultNumericStart		int	
Stop	Sample number to stop pattern recording	DefaultNumericStop		int	
Labels	Labels for data	A		string array	
ChannelMode	Full or Half channel mode: Half, Full	Full		enum	
Clock	Clock source for pattern generator: Internal (see InternalClock), External (period >= 20ns), External (10ns <= p < 20ns), External (period < 10ns)	Internal (see InternalClock)		enum	
InternalClock	Internal clock period	10e-9	sec	real	

Pin Inputs

Pin	Name	Description	Signal Type
1	data	input data	multiple fix

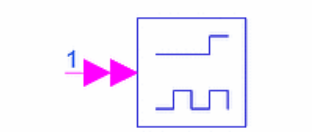
Notes/Equations

1. The PatGen_16522A_Sink model collects bit sequences from a simulation and downloads them to a 16522A pattern generator module that is plugged into a 16700 series logic analyzer.
2. Prerequisites for using the PatGen_16522A_Sink are
   - 16522A Pattern Generator; for information, visit the website http://www.agilent.com/find/xbv_pro_16522a .
   - 16700B Logic Analysis System; for information, visit the website http://www.agilent.com/find/xbv_pro_16700b .
   - Supported method of connecting the instrument to your computer. For detailed setup information, refer to *Configuring the Server IO on the PC* (connectmui) in the *Connection Manager* (connectmui) documentation.
3. This model programs the 16522A to output each of the model inputs as a separate labelled sequence. The 16522A sequences have the same bit width as the model's inputs, and these are named according to the Labels parameter.
4. The Interface parameter identifies the Logic Analysis System mainframe that contains the Pattern Generator as one of its modules. Set the Instrument parameter to the DNS hostname or IP address of the Logic Analysis system.
5. If you are using secure mode on your logic analyzer, set the Login parameter to a

valid user and password that are separated by a colon. If you are using insecure mode, the default Login setting of *anonymous:ads* is fine.

6. The Slot parameter specifies the slot (A to J) in which the 16522A resides. The Pattern Generator in this slot must be activated before this instrument link can be used.
7. The Start and Stop parameters (as with all sinks) specify how much data to collect.
8. The Labels parameter names the signals input to PatGen_16522A_Sink. The same labels are assigned to the signals inside the 16522A. Use a minus sign as the first character of a label to set negative polarity for that signal.
   Labels are assigned in the same order as signals are connected. If you connect more than one signal to PatGen_16522A_Sink, use a BusMerge component to control the order. Remember that signals are ordered from bottom to top on the schematic. For example, when using a BusMerge4, the Labels format is Labels = {"A", "B", "C", "D"}, where the bottom input is A.
9. The ChannelMode parameter programs the 16522A to half- or full-channel output mode. *Half* channel mode enables a higher data rate of up to 200 MHz, but with only 20 channels; *Full* channel mode limits the maximum data rate to 100 MHz but enables use of all 40 channels.
10. The Clock and InternalClock parameters specify how to clock the data. For internal clocking, set the Clock parameter to *Internal* , and set the InternalClock parameter to the desired period. For external clocking, set the Clock parameter to the external setting corresponding to the external clock period.
11. Multiple pattern generator models can be plugged into the same mainframe.
12. To access example designs that use this model, from the ADS Main window, choose **File > Open > Example > Instruments > PatGen_wrk**.

PatGen_16720A_Sink



Description: Agilent 16720A Pattern Generator
Library: Instruments
Class: SDFPatGen_16720A_Sink

Parameters

Name	Description	Default	Unit	Type	Range
Interface	IP address	192.0.0.1		string	
Login	Login name and password separated by colon	anonymous:ads		string	
Slot	Slot of pattern generator card in mainframe	A		string	
Start	Sample number to start pattern recording	DefaultNumericStart		int	
Stop	Sample number to stop pattern recording	DefaultNumericStop		int	
Labels	Labels for data	A		string array	
ChannelMode	Full or Half channel mode: Half, Full	Full		enum	
ClkSource	clock source for pattern generator: Internal (see IntClkFreq), External (frequency <= 50MHz), External (50MHz < f <= 180MHz), External (frequency > 180MHz)	Internal (see IntClkFreq)		enum	
IntClkFreq	internal clock frequency	10e6	Hz	real	

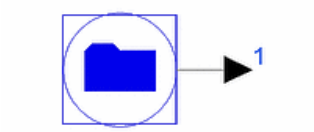
Pin Inputs

Pin	Name	Description	Signal Type
1	data	input data	multiple fix

Notes/Equations

1. The PatGen_16720A_Sink model collects bit sequences from a simulation and downloads them to a 16720A pattern generator module that is plugged into any of the 16700 series of logic analyzers.
2. Prerequisites for using the PatGen_16720A_Sink are
   - 16720A Pattern Generator; for information, visit the website http://www.agilent.com/find/xbv_pro_16720a .
   - 16700B Logic Analysis System; for information, visit the website http://www.agilent.com/find/xbv_pro_16700b .
   - LAN connectivity from your workstation to the Logic Analysis System mainframe. For detailed setup information, refer to *Configuring the Server IO on the PC* (connectmui) in the *Connection Manager* (connectmui) documentation.
3. This model programs the 16720A to output each of the model inputs as a separate labelled sequence. The 16720A sequences have the same bit width as the model's inputs, and these are named according to the Labels parameter.
4. The Interface parameter identifies the Logic Analysis System mainframe that contains the Pattern Generator as one of its modules. Set the Instrument parameter to the DNS hostname or IP address of the Logic Analysis System.
5. If you are using secure mode on your logic analyzer, set the Login parameter to a valid user and password that are separated by a colon. If you are using insecure mode, the default Login setting of *anonymous:ads* is fine.
6. The Slot parameter specifies the slot (A to J) in which the 16720A resides. The pattern generator in this slot must be activated before this instrument link can be used.
7. The Start and Stop parameters (as with all sinks) specify how much data to collect.
8. The Labels parameter names the signals input to PatGen_16720A_Sink. The same labels are assigned to the signals inside the 16720A. Use a minus sign as the first character of a label to set negative polarity for that signal.
   Labels are assigned in the same order as signals are connected. If you connect more than one signal to PatGen_16720A_Sink, use a BusMerge component to control the order. Remember that signals are ordered from bottom to top on the schematic. For example, when using a BusMerge4, the Labels format is Labels = {"A", "B", "C", "D"}, where the bottom input is A.
9. The ChannelMode parameter programs the 16720A to half- or full-channel output mode. *Half* channel mode enables a higher data rate of up to 300 MHz, but with only 24 channels; *Full* channel mode limits the maximum data rate to 180 MHz but enables use of all 48 channels.
10. The ClkSource and IntClkFreq parameters specify how to clock the data. For internal clocking, set ClkSource to Internal, and set IntClkFreq to the desired period. For external clocking, set ClkSource to the external setting corresponding to the external clock period.
11. Multiple pattern generator models can be plugged into the same mainframe.
12. To access example designs that use this model, from the ADS Main window, choose **File > Open > Example > Instruments > PatGen_wrk**.

SDFRead



Description: Read SDF Files
Library: Instruments
Class: TSDFSDFRead

Parameters

Name	Description	Default	Unit	Type	Range
FileName	input file	trace1.dat		filename	
ControlSimulation	control simulation: NO, YES	NO		enum	
Periodic	Periodic output: NO, YES	YES		enum	
AllPoints	Include unalised region, for frequency data: NO, YES	NO		enum	
DataRecord	which data record to output	1		int	

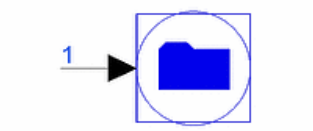
Pin Outputs

Pin	Name	Description	Signal Type
1	out	data	timed

Notes/Equations

1. The SDFRead model reads an SDF file and outputs the contents as timed data. The carrier frequency and timestep are set from the file.
2. The ControlSimulation parameter determines how the model acts inside the simulation. When ControlSimulation = YES, the model keeps the simulation running during the first pass through the data. If your sinks are set to not control the simulation, these automatically collect all data from the model. When ControlSimulation = NO, the model produces data *ad infinitum*, and you must set your sinks to control the simulation, then set the Start and Stop parameters to appropriate values.
3. By default, the model repeats data from the file. If Periodic = *NO*, the model outputs 0 until the end of the simulation.
4. The AllPoints parameter is only relevant if the SDF file contained frequency data. Setting this parameter to *YES* causes data in the unaliased region to be output as well.
5. The DataRecord parameter controls which data record inside the SDF file to output. Most SDF files only contain one data record, so you can usually leave this parameter set to its default of 1.
6. See also, *SDFWrite* (instruments).

SDFWrite



Description: Write SDF Files
Library: Instruments
Class: TSDFSDFWrite

Parameters

Name	Description	Default	Unit	Type	Range
ControlSimulation	"control simulation": NO, YES	YES		enum	
Start	number to start recording data	DefaultTimeStart	sec	real	
Stop	number to stop recording data	DefaultTimeStop	sec	real	
FileName	input files	trace.dat		filename	

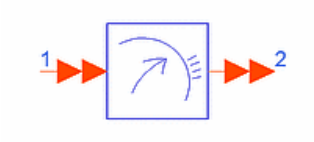
Pin Inputs

Pin	Name	Description	Signal Type
1	in	data	timed

Notes/Equations

1. The SDFWrite model writes an SDF file of the timed data it receives as input.
2. The ControlSimulation parameter determines how the model acts inside the simulation. When ControlSimulation = YES, the model collects data according to the Start and Stop parameters. When ControlSimulation = NO, the model collects as much data as it receives, starting at Start. Some other source or sink in the simulation must be set to control the simulation.
3. See also, *SDFRead* (instruments).

VeeLink



Description: A Link to Agilent VEE
Library: Instruments
Class: SDFVeeLink

Parameters

Name	Description	Default	Type
HostName	The IP address or host name of the workstation that should host the remote VEE session		string
VeeLibraryName	An Agilent VEE saved program file containing User Functions		filename
BeginUserFunctionName	A User Function called once at the beginning of a simulation		string
UserFunctionName	A User Function called at each iteration of a simulation		string
WrapupUserFunctionName	A User Function called once the end of a simulation		string
TimeOut	Seconds to wait before giving up	10.0	real
DebugMode	Run Agilent VEE function in debug mode? NO, YES	NO	enum

Pin Inputs

Pin	Name	Description	Signal Type
1	VeeFunctionArguments		multiple anytype

Pin Outputs

Pin	Name	Description	Signal Type
2	VeeFunctionResults		multiple anytype

Notes/Equations

Note
This item is for use on Win32 platforms only.

1. The VeeLink component enables you to use VEE UserFunctions as if these were integral parts of your Ptolemy simulation.
   Agilent VEE for Windows is a powerful graphical programming environment for fast measurement analysis results; for information, visit the website http://www.agilent.com/find/vee .
2. Invoke an Agilent VEE function by using the *Callable VEE* C- language library. Invoking an Agilent VEE function in this manner requires that the Agilent VEE development environment has been installed. Additionally, you must start the Agilent VEE Service Manager.
3. An Agilent Vee Library is a saved program file containing User Functions.
   The HostName parameter specifies which workstation (the *remote host*) actually is running the Vee User Function. If HostName is blank (default) the function runs on the same workstation as the one that initiated the simulation. The *remote host* must have Agilent Vee Pro installed.
   For a User Function to run on a workstation other than the one on which you initiated

the simulation, the remote host must have the Vee Service Manager running. Unlike the Agilent Vee Pro software (a licensed product), you can install the Agilent Vee Service on any workstation. The Vee Service is installed as an optional part of the Connection Manager Server installation. Installing the Vee Service does not require that you install the Connection Manager Server.
The Vee Service installation places a Windows Control Panel applet in the target PC Windows installation directory system32 sub-directory.
A Vee Library is a saved Vee program file that contains UserFunctions. Vee programs typically have *.vee* or *.vxe* extensions. You can set the VeeLibraryName parameter to specify the path to the Vee Library using relative or absolute paths.

- If you specify an absolute path (a path that contains a file separator character \ or /) Vee loads a library from the specified location only.
- If you specify a relative path (a path that does not contain a file path separator), VeeLink looks first in the workspace data directory, then in the directory specified in the registry key.
  _ HKEY_LOCAL_MACHINE\Software\Agilent\<latest>\UserDir_
  where <*latest*> is the highest-numbered version of Vee Pro installed on the server PC.

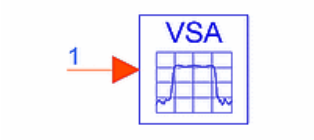
> **Note**
> The VeeLibraryName parameter must contain a file name complete with the extension, for example *rz.vee*.

4. The BeginUserFunctionName parameter identifies a UserFunction to be called once per VeeLink component at the start of a simulation run. You might use this to set instruments to a reset state before doing any other setup. If you leave this parameter empty, the VeeLink start just continues on without trying to call a VEE function at simulation start time.
5. The UserFunctionName parameter specifies the name of a User Function within the library; this function is called at each iteration of the simulation.
   It is legal to have User Functions of the same name in different libraries.
   Also, the UserFunctionName parameter can remain blank. For example, leave this parameter blank if you want something to happen at the beginning and/or end of a simulation but not at each iteration of the simulation.
6. The WrapupUserFunctionName parameter identifies a UserFunction to be called once at the end of a simulation (if the simulation has not stopped with an exception). For example, use this function to set instruments to a known state before initiating another simulation run.
7. The TimeOut parameter specifies the length of time (in seconds) the VeeLink component waits before determining that the VEE UserFunction can not complete. A TimeOut value of zero means the VeeLink component waits indefinitely for the UserFunction to complete.
   Common reasons a UserFunction can not complete in the expected time include:
   - An instrument is set to trigger on a hardware signal that never arrives.
   - A UserFunction contains an interactive control that you never interact with, possibly because the control is not contained in a Vee panel (a presentation view of the function) and therefore, is not visible.
   - A UserFunction contains an interactive control that you never interact with, possibly because the Vee Service Manager is not set up to enable interactive VEE Server sessions.
8. DebugMode = YES enables you to run the Agilent VEE process hosting your function in a mode that enables you to use Agilent VEE development environment debugging tools.

   > **Note**
   > For interactive debugging to work, the Agilent VEE Service Manager must be configured so that the service process can interact with the desktop. See *Who Does It Run As?* (instruments).

9. This component has been tested with VEE Pro 6.0, 6.1, 6.2, 7.0, and 7.5x.

VSA_89600_1_Sink



Description: Agilent 89600 Vector Signal Analyzer
Library: Instruments
Class: TSDFVSA_89600_Sink

Parameters

Name	Description	Default	Unit	Type
VSATitle	Text for VSA title bar	Simulation output		string
TStep	Simulation time step; use a value of 0 for time step synchronization with other network timed signals	0	sec	real
SamplesPerSymbol	Digital demodulation samples per symbol; NOT to be confused with VSA points per symbol	0.0		real
SetupFile	Name of measurement setup file to recall			filename
RestoreHW	YES to restore VSA hardware selection at end of simulation; NO to not: NO, YES	NO		enum
SetFreqProp	Set VSA 89600 frequency properties (center, span, zoom): NO, YES	YES		enum

Pin Inputs

Pin	Name	Description	Signal Type
1	input#1		anytype

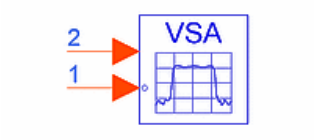
Notes/Equations

Note
This item is for use on Win32 platforms only.

1. The VSA_89600 models provide a dynamic link to integrate the 89600 series VSA software with ADS.
   Before using this model, the VSA_89600 software must be installed. The VSA 89600 software can be downloaded from http://www.agilent.com/find/89600 .
   For 89600 analyzer reference information, example measurements, or a getting started video, access *Help* or choose **Start > Programs > Agilent 89600 VSA > Help**.
2. The VSA_89600_Sink models provide a stream interface where you can input digitized waveforms directly from ADS to 89600-series VSA software without using the 89600-series hardware. The full functionality of the 89600 analyzer is available to analyze and display the ADS signal.
3. VSA_89600_1_Sink is a 1-channel, continuous measurement subset of the VSA_89600_Sink base component.
4. An input can be float (real), complex, or timed; timed can be BaseBand or ComplexEnv.
5. The default TimeStep is 1 second unless the TStep parameter is set >0. If TimeStep is available from the input, TimeStep is used instead. Then, this is used to set the sample rate of the VSA analyzer to 1/TimeStep. This places an upper limit on the analyzer frequency span.
6. The analyzer re-zooms data when its measurement center frequency is set different from that of the simulation. Also, the frequency span can be reduced in the same way. Refer to *note 16*.
7. The SamplesPerSymbol parameter provides a convenient way to set the analyzer's Symbol Rate. This is different from the analyzer's Points/Symbol parameter, which adjusts the analyzer's interpolation of demodulated data. SamplesPerSymbol takes precedence over the Symbol Rate in the setup file, if a SetupFile is specified.
8. The analyzer's digital demodulation algorithm limits the analyzer's span to a maximum of 15.625 times the Symbol Rate. This, plus the analyzer's maximum decimation rate, places a lower limit on the Symbol Rate.

9. The SetupFile parameter can be used to recall automatically a VSA setup file during simulation start-up. The VSA measurement setup file, SetupFile, is saved from the VSA application file menu, **File > Save > Save Setup**. To avoid the need to supply an absolute file name on the schematic, save the setup file in the workspace data directory. Note that the VSA does not default to saving in this directory the first time. To refine an existing SetupFile, you can modify the VSA application while simulating and then save the setup file before restarting the simulation.
10. Triggering is not available in the 89600 when the input is from a simulation.
11. Unless a setup file is specified, the VSA application begins the simulation with its previous setup, with a few modifications dictated by the simulation. This setting enables for continuity of the VSA setup between simulations. RestoreHW = YES can interfere with this (see *note 12*). This continuity means that the initial VSA application setup can be important when a setup file is not specified. This includes the Hardware Setup. Starting a simulation in a known state, such as a preset state, can resolve some VSA initialization problems. The VSA can be preset in various ways under *File > Preset* on the VSA application. The *Simulate Hardware* selection under *Hardware > Utilities* can be a useful starting point.
12. If RestoreHW = NO (default), the VSA application remains in its Stream hardware setup at the end of a simulation. It can be returned to using other VSA hardware setups through the *Hardware > Utilities* menu on the VSA application.
    If RestoreHW = YES, the hardware setup is saved at the beginning of the simulation and then restored at the end of the simulation. Since some measurement setups change when the hardware setup changes, this setting can interfere with the continuity between simulations as described in Note 9. In general, setting RestoreHW to NO or specifying SetupFile can have the least problems.
    One need for RestoreHW = YES is when changing between simulations where a VSA with the same Instance Name is used as a source in one simulation and a sink in another and the VSA instance is not closed between the simulations. In this case, the sink VSA restores the source hardware setting at the end of the simulation. Otherwise, the VSA is left configured with Stream hardware and does not function correctly as a source. The same problem arises if you would like to alternate between using the same running VSA instance as a sink in a simulation and then would like to make hardware based measurements.
13. Multiple VSA9600-type components can be active in a simulation and you can configure each one independently (For ADS versions earlier than 1.5, only one VSA_89600_Sink component can be active in a simulation.)
14. In some simulation contexts, you can add a shunt resistor to the simulation inputs of a VSA_89600_Sink-type component.
    VSA_89600_Sink-type components do not have an R_{IN} parameter, so the simulator can assume that R_{IN} is infinite without a shunt resistor in the simulation.
15. When a simulation that contains an active VSA_89600_Sink component starts, it attaches to a running 89600 whose title begins with the component's instance name. If no such 89600 instance is found, a new 89600 is created, and its title is set to the associated component's instance name and VSATitle text.
16. If SetFreqProp = NO, the VSA's measurement center frequency, span, and zoom mode are left unchanged (except that these are loaded from a setup file, if SetupFile is specified). If these are different from the simulation center (carrier) frequency, span (sample rate), or zoom (complex) mode, then the display on the VSA can be quite different from expected. If SetFreqProp = YES, the VSA's measurement center frequency, span, and zoom parameters is set to those of the simulation, overriding those in the setup file, if specified.
17. To access example designs that use this model, from the ADS Main window, choose **File > Open > Example > Instruments > VSA89600_demos_wrk**.

VSA_89600_2_Sink



Description: Agilent 89600 Vector Signal Analyzer
Library: Instruments
Class: TSDFVSA_89600_Sink

Parameters

Name	Description	Default	Unit	Type
VSATitle	Text for VSA title bar	Simulation output		string
TStep	Simulation time step; use a value of 0 for time step synchronization with other network timed signals	0	sec	real
SamplesPerSymbol	Digital demodulation samples per symbol; NOT to be confused with VSA points per symbol	0.0		real
SetupFile	Name of measurement setup file to recall			filename
RestoreHW	YES to restore VSA hardware selection at end of simulation; NO to not: NO, YES	NO		enum
SetFreqProp	Set VSA 89600 frequency properties (center, span, zoom): NO, YES	YES		enum

Pin Inputs

Pin	Name	Description	Signal Type
1	input#1		anytype
2	input#2		anytype

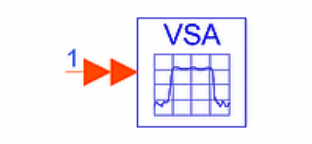
Notes/Equations

Note
This item is for use on Win32 platforms only.

1. The VSA_89600 models provide a dynamic link to integrate the 89600 series VSA software with ADS.
   Before using this model, the VSA_89600 software must be installed. The VSA 89600 software can be downloaded from http://www.agilent.com/find/89600 .
   For 89600 analyzer reference information, example measurements, or a getting started video, access *Help* or choose **Start > Programs > Agilent 89600 VSA > Help**.
2. The VSA_89600_Sink models provide a stream interface where you can input digitized waveforms directly from ADS to 89600-series VSA software without using the 89600-series hardware. The full functionality of the 89600 analyzer is available to analyze and display the ADS signal.
3. VSA_89600_2_Sink is a 2-channel, continuous measurement subset of the VSA_89600_Sink base component.
4. An input can be float (real), complex, or timed; timed can be BaseBand or ComplexEnv.
5. The default TimeStep is 1 second unless the TStep parameter is set > *0* . If TimeStep is available from the input, TimeStep is used instead. Then, this is used to set the sample rate of the VSA analyzer to 1/TimeStep. This places an upper limit on the analyzer frequency span.
6. The analyzer can be configured to create complex data from two float (real) inputs with a VSA_89600_2_Sink component by bringing the analyzer up first and configuring it in the I+jQ mode (*Input* > *Channels* > *I+jQ*). The simulation then leaves this mode undisturbed.
7. The analyzer re-zooms data when its measurement center frequency is set different from that of the simulation. Also, the frequency span can be reduced in the same way. Refer to *note 17*.
8. When the analyzer is in a demodulation mode at the beginning of a simulation and has two float (real) inputs, the analyzer is set to I+jQ mode.

9. The SamplesPerSymbol parameter provides a convenient way to set the analyzer's Symbol Rate. This is different from the analyzer's Points/Symbol parameter, which adjusts the analyzer's interpolation of demodulated data. SamplesPerSymbol takes precedence over the Symbol Rate in the setup file, if a SetupFile is specified.
10. The analyzer's digital demodulation algorithm limits the analyzer's span to a maximum of 15.625 times the Symbol Rate. This, plus the analyzer's maximum decimation rate, places a lower limit on the Symbol Rate.
11. The SetupFile parameter can be used to recall automatically a VSA setup file during simulation start-up. The VSA measurement setup file, SetupFile, is saved from the VSA application file menu, *File* > *Save* > *Save Setup*. To avoid the need to supply an absolute file name on the schematic, save the setup file in the workspace data directory. Note that the VSA does not default to saving in this directory the first time. To refine an existing SetupFile, you can modify the VSA application while simulating and then save the setup file before restarting the simulation.
12. Triggering is not available in the 89600 when the input is from a simulation.
13. Unless a setup file is specified, the VSA application begins the simulation with its previous setup, with a few modifications dictated by the simulation. This setting enables for continuity of the VSA setup between simulations. RestoreHW = YES can interfere with this (refer to *note 14*). This continuity means that the initial VSA application setup can be important when a setup file is not specified. This includes the Hardware Setup. Starting a simulation in a known state, such as a preset state, can resolve some VSA initialization problems. The VSA can be preset in various ways under *File* > *Preset* on the VSA application. The *Simulate Hardware* selection under *Hardware* > *Utilities* can be a useful starting point.
14. If RestoreHW = NO (default), the VSA application remains in its Stream hardware setup at the end of a simulation. It can be returned to using other VSA hardware setups through the *Hardware* > *Utilities* menu on the VSA application.
    If RestoreHW = YES, the hardware setup is saved at the beginning of the simulation and then restored at the end of the simulation. Since some measurement setups change when the hardware setup changes, this setting can interfere with the continuity between simulations as described in *note 11*. In general, setting RestoreHW to NO or specifying SetupFile can have the least problems.
    One need for RestoreHW = YES is when changing between simulations where a VSA with the same Instance Name is used as a source in one simulation and a sink in another and the VSA instance is not closed between the simulations. In this case, the sink VSA restores the source hardware setting at the end of the simulation. Otherwise, the VSA is left configured with Stream hardware and does not function correctly as a source. The same problem arises if you would like to alternate between using the same running VSA instance as a sink in a simulation and then would like to make hardware based measurements.
15. Multiple VSA_89600_Sink-type components can be active in a simulation and you can configure each one independently (For ADS versions earlier than 1.5, only one VSA_89600_Sink component can be active in a simulation.)
    In some simulation contexts you can add a shunt resistor to the simulation inputs of a VSA_89600_Sink-type component. VSA_89600_Sink-type components do not have an R_{IN} parameter, so the simulator can assume that R_{IN} is infinite without a shunt resistor in the simulation.
16. When a simulation that contains an active VSA_89600_Sink component starts, it attaches to a running 89600 whose title begins with the component's instance name. If no such 89600 instance is found, a new 89600 is created, and its title is set to the associated component's instance name and VSATitle text.
17. If SetFreqProp = NO, the VSA's measurement center frequency, span, and zoom mode are left unchanged (except that these are loaded from a setup file, if SetupFile is specified). If these are different from the simulation center (carrier) frequency, span (sample rate), or zoom (complex) mode, then the display on the VSA can be quite different from expected. If SetFreqProp = YES, the VSA's measurement center frequency, span, and zoom parameters is set to those of the simulation, overriding those in the setup file, if specified.
18. To access example designs that use this model, from the ADS Main window, choose **File > Open > Example > Instruments > VSA89600_demos_wrk**.

VSA_89600_Sink



Description: Agilent 89600 Vector Signal Analyzer
Library: Instruments
Class: TSDFVSA_89600_Sink

Parameters

Name	Description	Default	Unit	Type
VSATitle	Text for VSA title bar	Simulation output		string
TStep	Simulation time step; use a value of 0 for time step synchronization with other network timed signals	0	sec	real
SamplesPerSymbol	Digital demodulation samples per symbol; NOT to be confused with VSA points per symbol	0.0		real
SetupFile	Name of measurement setup file to recall			filename
RestoreHW	YES to restore VSA hardware selection at end of simulation; NO to not: NO, YES	NO		enum
Start	Sample number to start measuring	DefaultNumericStart		int
Stop	Sample number to stop measuring	DefaultNumericStop		int
TclTkMode	YES enables Tcl/Tk mode (continuous simulation); NO disables: NO, YES	NO		enum
RecordMode	YES enables VSA 89600 Recording mode; NO disables; Stop must be > Start, TclTkMode = NO: NO, YES	NO		enum
SetFreqProp	Set VSA 89600 frequency properties (center, span, zoom): NO, YES	YES		enum

Pin Inputs

Pin	Name	Description	Signal Type
1	input		multiple anytype

Notes/Equations

Note
This item is for use on Win32 platforms only.

1. The VSA_89600 models provide a dynamic link to integrate the 89600 series VSA software with ADS.
   Before using this model, the VSA_89600 software must be installed. The VSA 89600 software can be downloaded from http://www.agilent.com/find/89600 .
   For 89600 analyzer reference information, example measurements, or a getting started video, access *Help* or choose *Start > Programs > Agilent 89600 VSA > Help.
2. The VSA_89600_Sink models provide a stream interface where you can input digitized waveforms directly from ADS to 89600-series VSA software without using the 89600-series hardware. The full functionality of the 89600 analyzer is available to analyze and display the ADS signal.
3. For continuous measurements, set Stop to −1, TclTKMode to YES, and RecordMode to NO. (For continuous measurements, the components VSA_89600_1_Sink and VSA_89600_2_Sink are configured correctly and can be easier to use.)
4. An input can be float (real), complex, or timed; timed can be BaseBand or ComplexEnv.
5. The default TimeStep is 1 second unless the TStep parameter is set >0. If TimeStep is available from the input, TimeStep is used instead. Then, this is used to set the sample rate of the VSA analyzer to 1/TimeStep. This places an upper limit on the analyzer frequency span.
6. The analyzer can be configured to create complex data from two float (real) inputs with a VSA_89600_2_Sink component by bringing the analyzer up first and

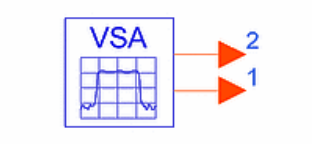
configuring it in the I+jQ mode (*Input* > *Channels* > *I+jQ*). The simulation then leaves this mode undisturbed.

7. The analyzer re-zooms data when its measurement center frequency is set different from that of the simulation. Also, the frequency span can be reduced in the same way. Refer to *note 19*.
8. When the analyzer is in a demodulation mode at the beginning of a simulation and has two float (real) inputs, the analyzer is set to I+jQ mode.
9. The SamplesPerSymbol parameter provides a convenient way to set the analyzer's Symbol Rate. This is different from the analyzer's Points/Symbol parameter, which adjusts the analyzer's interpolation of demodulated data. SamplesPerSymbol takes precedence over the Symbol Rate in the setup file, if a SetupFile is specified.
10. The analyzer's digital demodulation algorithm limits the analyzer's span to a maximum of 15.625 times the Symbol Rate. This, plus the analyzer's maximum decimation rate, places a lower limit on the Symbol Rate.
11. The SetupFile parameter can be used to recall automatically a VSA setup file during simulation start-up. The VSA measurement setup file, SetupFile, is saved from the VSA application file menu, *File* > *Save* > *Save Setup*. To avoid the need to supply an absolute file name on the schematic, save the setup file in the workspace data directory. Note that the VSA does not default to saving in this directory the first time. To refine an existing SetupFile, you can modify the VSA application while simulating and then save the setup file before restarting the simulation.
12. Triggering is not available in the 89600 when the input is from a simulation.
13. Except when the VSA_89600_Sink RecordMode = YES, TclTkMode = NO, and $0 \le \text{Start} < \text{Stop}$, the you must ensure the simulation generates enough points to fill the 89600 Record buffer when operating the 89600 in the Record mode.
14. When in Record (Time Capture) mode, the 89600 Record buffer needs additional settling points under some conditions, such as when the span is reduced.
15. In Record mode, if necessary, you can stop or pause the simulation in order to give the 89600 priority to run in playback mode after the capture.
16. Unless a setup file is specified, the VSA application begins the simulation with its previous setup, with a few modifications dictated by the simulation. This setting enables for continuity of the VSA setup between simulations. RestoreHW = *YES* can interfere with this (refer to *note 18*). This continuity means that the initial VSA application setup can be important when a setup file is not specified. This includes the Hardware Setup. Starting a simulation in a known state, such as a preset state, can resolve some VSA initialization problems. The VSA can be preset in various ways under *File* > *Preset* on the VSA application. The *Simulate Hardware* selection under *Hardware* > *Utilities* can be a useful starting point.
17. If RestoreHW = *NO* (default), the VSA application remains in its Stream hardware setup at the end of a simulation. It can be returned to using other VSA hardware setups through the *Hardware* > *Utilities* menu on the VSA application.
18. If RestoreHW = *YES*, the hardware setup is saved at the beginning of the simulation and then restored at the end of the simulation. Since some measurement setups change when the hardware setup changes, this setting can interfere with the continuity between simulations as described in Note 17. In general, setting RestoreHW to *NO* or specifying SetupFile can have the least problems.
    One need for RestoreHW = *YES* is when changing between simulations where a VSA with the same Instance Name is used as a source in one simulation and a sink in another and the VSA instance is not closed between the simulations. In this case, the sink VSA restores the source hardware setting at the end of the simulation. Otherwise, the VSA is left configured with Stream hardware and does not function correctly as a source. The same problem arises if you would like to alternate between using the same running VSA instance as a sink in a simulation and then would like to make hardware based measurements.
19. Multiple VSA_89600_Sink-type components can be active in a simulation and you can configure each one independently (For ADS versions earlier than 1.5, only one VSA_89600_Sink component can be active in a simulation.)
20. In some simulation contexts you can add a shunt resistor to the simulation inputs of a VSA_89600_Sink-type component. VSA_89600_Sink-type components do not have an R_{IN} parameter, so the simulator can assume that R_{IN} is infinite without a shunt resistor in the simulation.
21. When a simulation that contains an active VSA_89600_Sink component starts, it attaches to a running 89600 whose title begins with the component's instance name.

If no such 89600 instance is found, a new 89600 is created, and its title is set to the associated component's instance name and VSATitle text.

22. If SetFreqProp = NO, the VSA's measurement center frequency, span, and zoom mode are left unchanged (except that these are loaded from a setup file, if SetupFile is specified). If these are different from the simulation center (carrier) frequency, span (sample rate), or zoom (complex) mode, then the display on the VSA can be quite different from expected. If SetFreqProp = YES, the VSA's measurement center frequency, span, and zoom parameters is set to those of the simulation, overriding those in the setup file, if specified.
23. To access example designs that use this model, from the ADS Main window, choose **File > Open > Example > Instruments > VSA89600_demos_wrk**.

VSA_89600_Source



Description: Agilent 89600 Vector Signal Analyzer
Library: Instruments
Class: TSDFVSA_89600_Source
Derived From: VSA_89600_1_Source

Parameters

Name	Description	Default	Unit	Type	Range
VSATitle	Text for VSA title bar	Simulation source		string	
ControlSimulation	Control simulation: NO, YES	NO		enum	
OutputType	Output port type: Timed, Frequency, Demod Errors, Complex Scalar, Float Scalar, Integer Scalar	Timed		enum	
Pause	Pause for VSA setup: NO, YES	YES		enum	
VSATrace	VSA trace that will supply data: A, B, C, D	A		enum	
RepeatData	VSA hardware measurement control: Repeat, Reacquire, Single pass	Repeat		enum	
TStep	Simulation time step	0	sec	real	
SetupFile	Name of setup file to recall into VSA			filename	
RecordingFile	Name of recording file to recall into VSA			filename	
SetupUse	VSA setup file recall control: Always, Once, No	Always		enum	
AutoCapture	Capture VSA input data at start-up: NO, YES	NO		enum	
DefaultHardware	Use VSA default hardware configuration: NO, YES	NO		enum	
FrequencySpan	If non-zero, set the frequency span.: Unchanged	Unchanged	Hz	real enum	[0, ∞)
SetCenterFrequency	Controls the CenterFrequency parameter.: NO, YES	NO		enum	
CenterFrequency	If the parameter SetCenterFrequency is set to YES, set the center frequency.	0	Hz	real	(-∞, ∞)
Range	If non-zero and RecordingFile not set, set the range.: Unchanged	Unchanged	V	real enum	[0, ∞)
RecordingLength	If non-zero and RecordingFile not set, this parameter set the recording length.: Unchanged	Unchanged	sec	real enum	[0, ∞)

Pin Outputs

Pin	Name	Description	Signal Type
1	out	Measurement data	anytype
2	gap	VSA input data gap signal	int

Notes/Equations

Note
This component is for use on Win32 platforms only.

1. The VSA_89600 models provide a dynamic link to integrate the 89600 series VSA software with ADS.
   Before using this model, the VSA_89600 software must be installed. The VSA 89600 software can be downloaded from http://www.agilent.com/find/89600 .
   For 89600 analyzer reference information, example measurements, or a getting started video, access *Help* or choose **Start > Programs > Agilent 89600 VSA > Help**.
2. The VSA_89600_Source model transfers measurement data from the 89600 Vector Signal Analyzer. The model outputs different amounts and types of data depending on OutputType, as described in *note 5*. Measurements can be acquired from any 89600 trace and are transferred in raw (unformatted) state.
3. The VSATitle parameter sets the title in the 89600 window title bar.
4. The ControlSimulation parameter determines how the model acts inside the simulation. When ControlSimulation = YES, the model keeps the simulation running during the first pass through the data. If your sinks are set to not control the simulation, these automatically collect all data from the model. When ControlSimulation = NO, the model produces data ad infinitum. You must set your sink to control the simulation, then set the Start and Stop parameters to appropriate values.
5. The OutputType parameter establishes the configuration of the model's output data port.

   Important
   The data type of the 89600 trace selected by VSATrace (*note 7*) must be compatible with the output type.

   Timed - Timed data is output at the 89600 center frequency and timestep (refer to TStep in *note 9*). The data can be complex or baseband, depending on the 89600 zoom state. Requires time domain measurements from the 89600. Example 89600 trace data type: Main Time.
   Frequency - Spectrum data is output as pairs of complex numbers. The first number is the real frequency (the imaginary part is zero), and the second number is the complex voltage at that frequency. You can connect a Distributor2 to the VSA_89600_Source data port to separate the signal. Requires frequency domain measurements from the 89600. Example 89600 trace data type: Spectrum.
   Demod Errors - Demodulation error data is output as sets of floating-point (real) values. The number of values and required 89600 configuration vary with demod type. See *note 21*.
   Complex Scalar - Complex numbers are output. If the 89600 measurement data is real-valued, the imaginary part of the output values is zero. Example 89600 trace data type: "Error Vector Time" in Digital Demod mode.
   Float Scalar - Floating point numbers are output. Requires real-valued 89600 measurement data. Example 89600 trace data type: "IQ Mag Error" in Digital Demod mode.
   Integer Scalar - Integer numbers are output. Requires real-valued 89600 measurement data. Useful for sourcing demodulation symbols when the 89600 trace data type is Syms/Errs.
6. The Pause parameter controls the 89600 start-up sequence. When Pause is set to YES, the simulation displays a message dialog and pause before acquiring measurement data from the 89600. Then you can configure the 89600 before pressing the OK button to proceed.
7. The VSATrace parameter specifies which 89600 trace provides measurement data.
8. The RepeatData parameter controls the transfer of data from the 89600 during hardware-based measurement. When RepeatData = Single pass, the model supplies data from a single measurement. The *Repeat* option acquires a single measurement and repetitively sources it into the simulation. *Reacquire* repeatedly acquires and sources new measurements. Note that in the *Repeat* and *Reacquire* modes time-based data is not continuous across measurements.
   When 89600 input is from a recording the RepeatData setting is ignored. Refer to

note 20 for information on operation with recordings.

9. The TStep parameter can be used to specify a target timestep when sourcing timed data. The model trys to adjust the 89600's span to achieve the requested step size. If the 89600 cannot be set to the required span, a warning message is output to the status window.
10. The SetupFile parameter can be used to recall automatically a VSA setup file during simulation start-up. The VSA measurement setup file, SetupFile, is saved from the VSA application file menu, *File > Save > Save Setup*. To avoid the need to supply an absolute file name on the schematic, save the setup file in the workspace data directory. Note that the VSA does not default to saving in this directory the first time. To refine an existing SetupFile, you can modify the VSA application while simulating and then save the setup file before restarting the simulation. If Pause is set to *YES*, the setup file is loaded before the Pause dialog is displayed.
11. The RecordingFile parameter can be used to recall automatically a recording into the 89600 during simulation start-up. If Pause is set to *YES,* the recording file is loaded before the Pause dialog is displayed.
12. The SetupUse parameter specifies when an 89600 setup file (specified in the SetupFile parameter) is recalled. Options are:
    Always - Recall setup file on every simulation run.
    Once - Recall setup file only when 89600 is started.
    No - Do not recall setup file.
13. The AutoCapture parameter can be set to *YES* to have the 89600 automatically initiate a capture recording of hardware input data at the beginning of a simulation. When the capture is complete, the model begins sourcing the recorded data. Time-based measurements is continuous (*note 20*). Recording length can be controlled via an 89600 setup file.
14. The DefaultHardware parameter can be set to *YES* to have the 89600 automatically select an appropriate hardware configuration. Otherwise, the 89600 uses previous hardware settings. Note that when a source 89600 reuses the same running 89600 instance that was used by a sink 89600 (VSA_89600_1_Sink, VSA_89600_2_Sink, or VSA_89600_Sink), it can be left set up with Stream input hardware. This problem can be solved via the RestoreHW parameter of the sink component.
15. Each 89600 measurement produces a block of data points, which are pipelined to the simulation. As a result, simulation plots typically lag 89600 traces. The number of points in a block varies with 89600 measurement type and configuration. For reference, the block size is output to the status window during startup. This value can be useful when configuring other components (for example, plot persistence and update size). Also, any change in block size during simulation is logged in the status window.
16. The units associated with transferred 89600 data are output to the status window during startup. Unless otherwise indicated, values are peak units.
17. To avoid transfer of overlapped measurement data, the 89600's maximum overlap for averaging off is set to zero when a simulation starts.
18. If the model detects a change to one of the following 89600 settings while sourcing data, the model disconnects from the 89600.
    - Input data source
    - Demodulator configuration
    - Data type of the trace supplying measurements (VSATrace)
    - Sweep mode
    - Center frequency (for timed output)
    - Timestep size (for timed output)

    The recommended practice is to use the *Pause* and/or *SetupFile* options to preconfigure the 89600.
19. When a simulation that contains an active VSA_89600_Source component starts, it attaches to a running 89600 whose title begins with the component's instance name. If no such 89600 instance is found, a new 89600 is created, and its title is set to the associated component's instance name and VSATitle text.
20. When 89600 input is from a recording the model steps through the recording, transferring all measurements. 89600 recording playback properties can be used to control start/stop points and looping. In this mode time-based data are continuous, with the exception of wrapped values when a playback loop occurs.
21. Demodulation Error/Summary information is available by configuring the trace data. For details on results available, refer to the 89600 help index topic *symbol table* and look for the demodulation format of interest. The demod formats available depend on

89600 licensing (For example, Digital Demod requires option AYA, and cdma2000 requires option B7.)
The error summary data and the numbering for each demodulation format are documented in tables in the 89600 help documentation (look in the index under *demod errors and ADS*). Note that some demodulation formats include several of these tables. The table values are output in a fixed sequence, which can be different from the order seen on the 89600 display. The number of output error values depends on the configuration.
The easiest way to get an individual error value is to place a DownSample component in series with the VSA89600Source output. For example, consider the list of Digital Demod error items. The DownSample Factor parameter would be set to the number of error values (21). To select EVM, the Phase parameter would be set to 20 (EVM position in the list subtracted from 21).

Note
When OutputType is Integer Scalar and trace data is configured to one of the above types (Syms/Errs, for example), the module sources demodulation symbols.

22. The gap output port emits a 1 when there is a discontinuity in 89600 input data. Otherwise it emits 0. A TkBreak component with Condition parameter set to *"$input(1) > 0"* can be used to trigger control actions based on this signal. The data rate of the gap port is the same as that of the out port.
23. Some uncommon simulation errors can orphan the *hpeesofsim.exe* process when an 89600 is active. If this occurs, additional simulation runs start a new *hpeesofsim* process and other errors can result. If you encounter such a problem, select the ADS *Simulate > Stop and Release Simulator* command, then use the Windows Task Manager's *End Process* command to end any remaining *hpeesofsim* processes.
24. Multiple VSA_89600_Source components can be active in a simulation.
25. When both VSA_89600_Source and VSA_89600_Sink components are present in a design, configuring the VSA_89600_Sink for 0% time record overlap minimizes the effect of gaps in source data.
26. The VSA_89600_Source component can be used with a PSA E444xA/89601A combination. The VSA89600 software runs on a PC connected to the E444xA, via LAN or GPIB, and provides hardware control, modulation analysis, and complete results displays. The controls and display of the E4406A are disabled while operating with the 89601A software.
    For more information, refer to the Agilent PSA website http://www.agilent.com/find/psa .
    Special options required for use of the Performance Spectrum Analyzers E444xA and the VSA89600 software are described in these product notes:
    - Product Note 5988-5015EN; available at the website http://cp.literature.agilent.com/litweb/pdf/5988-5015EN.pdf .
    - Product Note 5988-4094EN; available at the website http://cp.literature.agilent.com/litweb/pdf/5988-4094EN.pdf .
27. The VSA_89600_Source component can be used with a VSA E4406A/89601A combination. The VSA89600 software runs on a PC connected to the E4406A, via LAN or GPIB, and provides hardware control, modulation analysis, and complete results displays. The controls and display of the E4406A are disabled while operating with the 89601A software.
    For more information, refer to the Agilent VSA E4406A website http://www.agilent.com/find/vsa .
    Special options required for use of the Vector Signal Analyzer E4406A and the VSA89600 software are described in Product Note 5988-2906EN, which is available at the website http://cp.literature.agilent.com/litweb/pdf/5988-2906EN.pdf .
28. The VSA_89600_Source component can be used with an ESA/89601A combination. The VSA89600 software runs on a PC connected to the ESA, via GPIB, and provides hardware control, modulation analysis, and complete results displays. The controls and display of the ESA are disabled while operating with the 89601A software.
    For more information, refer to the Agilent ESA website http://www.agilent.com/find/esa .
    Special options required for use of the ESA and the VSA89600 software are described in Product Note 5988-4097EN, which is available at the website http://cp.literature.agilent.com/litweb/pdf/5988-4097EN.pdf .
29. The VSA_89600_Source component can be used with an Infiniium oscilloscope/89601A combination. The VSA89600 software runs on a PC connected to

the Infiniium scope, via LAN or GPIB, and provides hardware control, modulation analysis, and complete results displays. The controls and display of the scope are disabled while operating with the 89601A software.
Special options required for use of the Infiniium scope and the VSA89600 software are described in this product note:
http://cp.literature.agilent.com/litweb/pdf/5988-4096EN.pdf .

30. The VSA_89600_Source component does not handle averaged measurements as a special case; each 89600 analyzer trace update is output to the simulation. The default 89600 analyzer average setup has Fast Average disabled, so the traces are updated each time new measurement results are added to the average, starting with the first measurement.
    To force only the final, fully averaged result to be output to the simulation, select MeasSetup > Average on the 89600 analyzer and check both the Fast Average and the Same as Count check boxes. This prevents the analyzer from updating the screen until the selected number of measurements have been averaged together.
31. To access example designs that use this model, from the ADS Main window, choose **File > Open > Example > Instruments > VSA89600_demos_wrk**.