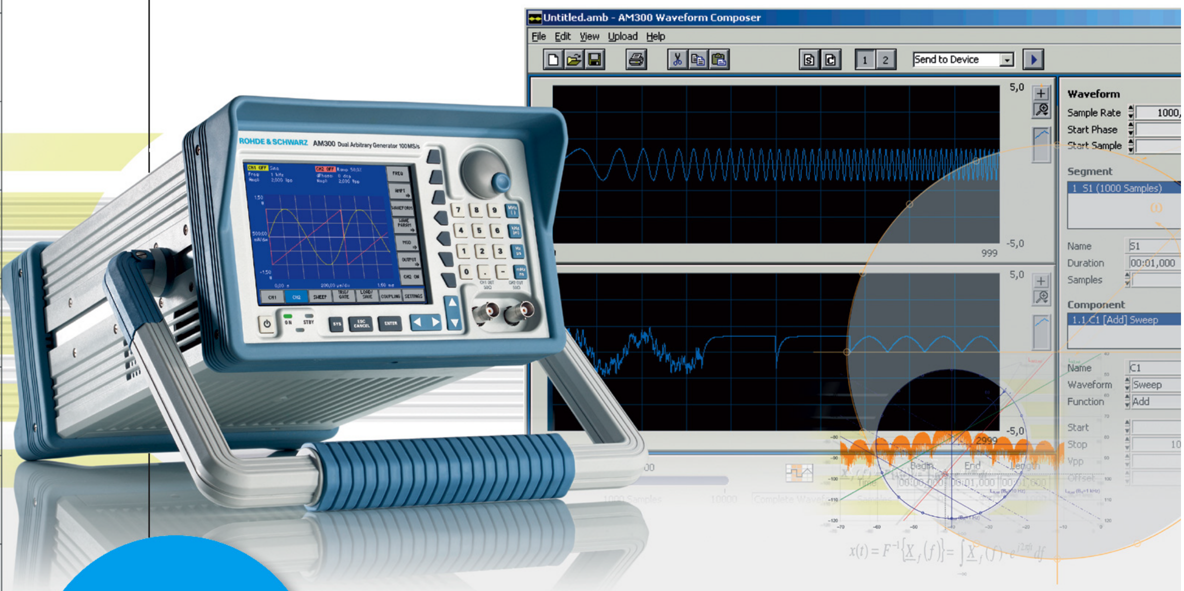


R&S Smart instruments™

Die neue Produktfamilie von Rohde & Schwarz

CE

Remote Control Manual AM300 Arbitrary Generator

VXI Plug & Play Style Instrument Driver

Version 03

ROHDE & SCHWARZ GmbH & Co. KG
Test and Measurement Division
Mühldorfstraße 15
81671 München, Germany

3rd edition 04/2006
Instrument Driver Revision 1.8, 04/2006

This document is subject of change. Errors expected.

Table of Contents

About the Manual 8

Why Instrument Drivers? 9

1 Manual Concept 10

1.1 Introduction 10

1.2 Instrument-Specific Information 11

1.2.1 Instrument Addresses (Resource Strings) 12
1.2.2 Using Callbacks 13
1.2.3 Threat Safety 13
1.2.4 Device Identification and Logical Names 14
1.2.5 Hot Plug & Unplug Support 15
1.2.6 SiTools 15

1.3 Using An Instrument Driver in Application Development Environments 17

1.3.1 Microsoft Visual C++ 4.0 (or higher) and Borland C++ 4.5 (or higher) 17
1.3.2 Microsoft Visual Basic 5.0 (or higher) 17
1.3.3 HP VEE Version 3.2 (or higher) 17
1.3.4 National Instruments LabWindows/CVI(R) 4.0.1 (or higher) 17
1.3.5 National Instruments LabVIEW(R) 6.1 (or higher) 17

1.4 VXIPNP Directory Location 18

1.5 Files Installed 19

2 Programmer’s Reference Manual 21

2.1 Instrument Driver Tree Structure 21

2.2 Function Tree Layout of the AM300 Arbitrary Generator 27

2.2.1 Initialization 28
2.2.2 Application Functions 30
2.2.2.1 Apply Instrument Setup 30
2.2.2.2 Apply Instrument Query 32
2.2.3 Configuration Functions 33
2.2.3.1 Channel Settings 33
2.2.3.1.1 Output Channel Setup 33
2.2.3.1.2 Output Channel Query 34
2.2.3.1.3 Output Channel Coupling Setup 34
2.2.3.1.4 Output Channel Coupling Query 35
2.2.3.2 Output Configuration 36
2.2.3.2.1 Output Function Shape Setup 36
2.2.3.2.2 Output Function Shape Query 37
2.2.3.2.3 Output Frequency Setup 37
2.2.3.2.4 Output Frequency Query 38
2.2.3.2.5 Output Voltage Setup 39
2.2.3.2.6 Output Voltage Query 39
2.2.3.2.7 Output Voltage Offset Setup 40
2.2.3.2.8 Output Voltage Offset Query 42
2.2.3.2.9 Output Voltage High Setup 43
2.2.3.2.10 Output Voltage High Query 43

2.2.3.2.11 Output Voltage Low Setup 44
2.2.3.2.12 Output Voltage Low Query 45
2.2.3.2.13 Output Voltage Unit Setup 45
2.2.3.2.14 Output Voltage Unit Query 46
2.2.3.2.15 Output Square Duty Cycle Setup 47
2.2.3.2.16 Output Square Duty Cycle Query 47
2.2.3.2.17 Output Ramp Symmetry Setup 48
2.2.3.2.18 Output Ramp Symmetry Query 48
2.2.3.2.19 Output Load Setup 49
2.2.3.2.20 Output Load User Setup 50
2.2.3.2.21 Output Load Query 50
2.2.3.2.22 Output Sync Setup 52
2.2.3.2.23 Output Sync Query 53
2.2.3.2.24 Output Setup 53
2.2.3.2.25 Output Query 54
2.2.3.2.26 Output Start Phase Setup 54
2.2.3.2.27 Output Start Phase Query 55
2.2.3.2.28 Output Filter Setup 55
2.2.3.2.29 Output Filter Query 56
2.2.3.2.30 Output Filter Type Setup 56
2.2.3.2.31 Output Filter Type Query 57
2.2.3.2.32 Output External Filter Setup 57
2.2.3.2.33 Output External Filter Query 58
2.2.3.2.34 Output Aditional Setup 58
2.2.3.2.35 Output Aditional Query 59
2.2.3.3 Pulse Generator Settings 59
2.2.3.3.1 Pulse Period Setup 59
2.2.3.3.2 Pulse Period Query 61
2.2.3.3.3 Pulse Width Setup 61
2.2.3.3.4 Pulse Width Query 62
2.2.3.3.5 Pulse Polarity Setup 63
2.2.3.3.6 Pulse Polarity Query 63
2.2.3.4 Modulation Settings 64
2.2.3.4.1 Modulation Points Setup 64
2.2.3.4.2 Modulation Points Query 64
2.2.3.4.3 AM Modulation 65
2.2.3.4.3.1 Amplitude Mod Depth Setup 65
2.2.3.4.3.2 Amplitude Mod Depth Query 66
2.2.3.4.3.3 Amplitude Mod Int Func Setup 66
2.2.3.4.3.4 Amplitude Mod Int Func Query 67
2.2.3.4.3.5 Amplitude Mod Int Freq Setup 68
2.2.3.4.3.6 Amplitude Mod Int Freq Query 68
2.2.3.4.3.7 Amplitude Mod Stat Setup 69
2.2.3.4.3.8 Amplitude Mod Stat Query 69
2.2.3.4.4 FM Modulation 70
2.2.3.4.4.1 Frequency Mod Dev Setup 70
2.2.3.4.4.2 Frequency Mod Dev Query 71
2.2.3.4.4.3 Frequency Mod Int Func Setup 71
2.2.3.4.4.4 Frequency Mod Int Func Query 72
2.2.3.4.4.5 Frequency Mod Int Freq Setup 73
2.2.3.4.4.6 Frequency Mod Int Freq Query 73
2.2.3.4.4.7 Frequency Mod Stat Setup 74
2.2.3.4.4.8 Frequency Mod Stat Query 74
2.2.3.4.5 PM Modulation 75
2.2.3.4.5.1 Phase Mod Dev Setup 75
2.2.3.4.5.2 Phase Mod Dev Query 75
2.2.3.4.5.3 Phase Mod Int Func Setup 76
2.2.3.4.5.4 Phase Mod Int Func Query 77
2.2.3.4.5.5 Phase Mod Int Freq Setup 77
2.2.3.4.5.6 Phase Mod Int Freq Query 78
2.2.3.4.5.7 Phase Mod Stat Setup 79

2.2.3.4.5.8 Phase Mod Stat Query 80
2.2.3.4.6 FSK Modulation 81
2.2.3.4.6.1 FSK Frequency Setup 81
2.2.3.4.6.2 FSK Frequency Query 82
2.2.3.4.6.3 FSK Internal Rate Setup 83
2.2.3.4.6.4 FSK Internal Rate Query 84
2.2.3.4.6.5 FSK Source Setup 84
2.2.3.4.6.6 FSK Source Query 85
2.2.3.4.6.7 FSK Stat Setup 86
2.2.3.4.6.8 FSK Stat Query 86
2.2.3.4.7 PSK Modulation 87
2.2.3.4.7.1 PSK Phase Setup 87
2.2.3.4.7.2 PSK Phase Query 88
2.2.3.4.7.3 PSK Internal Rate Setup 89
2.2.3.4.7.4 PSK Internal Rate Query 89
2.2.3.4.7.5 PSK Source Setup 90
2.2.3.4.7.6 PSK Source Query 91
2.2.3.4.7.7 PSK Stat Setup 91
2.2.3.4.7.8 PSK Stat Query 92
2.2.3.5 Arbitrary Waveform Settings 93
2.2.3.5.1 Data Arb Points Setup 93
2.2.3.5.2 Data Arb Points Query 94
2.2.3.5.3 Data Arb Start Setup 94
2.2.3.5.4 Data Arb Start Query 95
2.2.3.5.5 Data Arb Stop Setup 95
2.2.3.5.6 Data Arb Stop Query 96
2.2.3.5.7 Data Arb Rate Setup 96
2.2.3.5.8 Data Arb Rate Query 97
2.2.3.5.9 Data Arb Rate Mode Setup 97
2.2.3.5.10 Data Arb Rate Mode Query 98
2.2.3.5.11 Data Arb Volatile Setup 98
2.2.3.5.12 Data Arb Dac Volatile Setup 99
2.2.3.6 Sweep Settings 100
2.2.3.6.1 Sweep Freq Start Setup 100
2.2.3.6.2 Sweep Freq Start Query 100
2.2.3.6.3 Sweep Freq Stop Setup 101
2.2.3.6.4 Sweep Freq Stop Query 102
2.2.3.6.5 Sweep Freq Center Setup 102
2.2.3.6.6 Sweep Freq Center Query 103
2.2.3.6.7 Sweep Freq Span Setup 104
2.2.3.6.8 Sweep Freq Span Query 104
2.2.3.6.9 Sweep Direction Setup 105
2.2.3.6.10 Sweep Direction Query 105
2.2.3.6.11 Sweep Points Setup 106
2.2.3.6.12 Sweep Points Query 106
2.2.3.6.13 Sweep Spacing Setup 107
2.2.3.6.14 Sweep Spacing Query 107
2.2.3.6.15 Sweep Time Setup 108
2.2.3.6.16 Sweep Time Query 108
2.2.3.6.17 Sweep Stat Setup 109
2.2.3.6.18 Sweep Stat Query 110
2.2.3.7 Frequency Marker Settings 111
2.2.3.7.1 Marker Frequency Setup 111
2.2.3.7.2 Marker Frequency Query 112
2.2.3.7.3 Marker Stat Setup 112
2.2.3.7.4 Marker Stat Query 113
2.2.3.8 Trigger Subsystem Settings 114
2.2.3.8.1 Trigger Settings 114
2.2.3.8.1.1 Trigger Source Setup 114
2.2.3.8.1.2 Trigger Source Query 114
2.2.3.8.1.3 Trigger Polarity Setup 115

2.2.3.8.1.4 Trigger Polarity Query 115
2.2.3.8.1.5 Trigger Delay Setup 116
2.2.3.8.1.6 Trigger Delay Query 116
2.2.3.8.1.7 Trigger Frequency Setup 117
2.2.3.8.1.8 Trigger Frequency Query 117
2.2.3.8.1.9 Trigger Period Setup 118
2.2.3.8.1.10 Trigger Period Query 118
2.2.3.8.2 Burst Mode 119
2.2.3.8.2.1 Burst Mod Ncycles Setup 119
2.2.3.8.2.2 Burst Mod Ncycles Query 119
2.2.3.8.2.3 Burst Mod Int Rate Setup 120
2.2.3.8.2.4 Burst Mod Int Rate Query 121
2.2.3.8.2.5 Burst Mod Sour Setup 121
2.2.3.8.2.6 Burst Mod Sour Query 122
2.2.3.8.2.7 Burst Mod Stat Setup 123
2.2.3.8.2.8 Burst Mod Stat Query 123
2.2.3.8.3 Sync Settings 124
2.2.3.8.3.1 Sync Type Setup 124
2.2.3.8.3.2 Sync Type Query 124
2.2.3.8.3.3 Sync Polarity Setup 125
2.2.3.8.3.4 Sync Polarity Query 126
2.2.3.8.4 Gate Settings 127
2.2.3.8.4.1 Gate Function Setup 127
2.2.3.8.4.2 Gate Function Query 127
2.2.3.8.4.3 Gate Length Setup 128
2.2.3.8.4.4 Gate Length Query 128
2.2.3.8.4.5 Gate Source Setup 129
2.2.3.8.4.6 Gate Source Query 129
2.2.3.8.4.7 Gate Polarity Setup 130
2.2.3.8.4.8 Gate Polarity Query 130
2.2.4 Action/Status Functions 132
2.2.4.1 Send Trigger 132
2.2.4.2 Abort Trigger 133
2.2.5 Utility Functions 134
2.2.5.1 Time Out 134
2.2.5.1.1 Time Out Setup 134
2.2.5.1.2 Time Out Query 135
2.2.5.2 Reference Oscillator 136
2.2.5.2.1 Reference Oscillator Source Setup 136
2.2.5.2.2 Reference Oscillator Source Query 136
2.2.5.2.3 Reference Oscillator Stat Setup 137
2.2.5.2.4 Reference Oscillator Stat Query 137
2.2.5.3 Flush Error Queue 138
2.2.5.4 State Checking 138
2.2.5.5 Reset 139
2.2.5.6 Self-Test 139
2.2.5.7 Error-Query 140
2.2.5.8 Error Message 141
2.2.5.9 Revision Query 142
2.2.6 Obsolete 143
2.2.6.1.1 Trigger Slope Setup 143
2.2.6.1.2 Trigger Slope Query 143
2.2.6.1.3 Output Summary Setup 144
2.2.6.1.4 Output Summary Query 144
2.2.6.1.5 Gate Time Setup 145
2.2.6.1.6 Gate Time Query 145
2.2.7 Close 146

2.3 Error (Status) Codes 147

2.4 Execution Timeout...149

2.5 Alphabetical List of Functions 150

2.6 Contacts 153

2.7 Remote Control Programming Examples 154

2.7.1.1.1 Error Handling & Time Profiling...154
2.7.1.1.2 Loading Arbitrary data to Channel 1 DAC data and Channel 2 volatile data (sample accurate) 158
2.7.1.1.3 Loading Arbitrary data to Channel 1 DAC data and Channel 2 volatile data (normal) 161
2.7.1.1.4 Setting Sweep Mode 164
2.7.1.1.5 Setting two different waveforms with different frequencies on both channels 166
2.7.1.1.6 Setting the burst mode with external source 169
2.7.1.1.7 Setting of FM modulation 171
2.7.1.1.8 Setting of FSK with internal source...173
2.7.1.1.9 Setting of PSK with external source..175
2.7.1.1.10 Setting of phase differences between Ch1 and Ch2.................................177

About the Manual

Information

This manual is intended to provide you with all the information that is necessary for remote control of the Rohde&Schwarz AM300 Arbitrary Generator via VXIplug&play style Instrument Driver.

Why Instrument Drivers?

Information

Many Rohde&Schwarz customers prefer the graphical programming languages LabVIEW from National Instruments or VEE from Agilent when writing applications for T&M equipment. Quite often, C-based LabWindows/CVI from National Instruments and Visual Basic or Visual C++ from Microsoft/ Borland is also used.

As a service, Rohde&Schwarz provides software device drivers free of charge for all these programming languages. All recent T&M equipment is supported, and demo programs are often available also.

Writing an application can take a lot of time. But writing the application is not the end of the story, since the T&M device must also be driven. That is not so simple with complex equipment, since just the description of the command set may comprise several hundred pages. Applying yourself to this task can be quite time-consuming. To relieve designers of these efforts to a great extend, Rohde&Schwarz provides ready-to-use software device drivers for all major interfaces. You no longer have to search through manuals looking for commands; this has already been done when developing the driver.

1 Manual Concept

About this chapter

Chapter 1 contains information for the user/installer of the Rohde&Schwarz AM300 Arbitrary Generator VXI Plug & Play style Instrument Driver.

More Information

Chapter 2 describes the function tree layout of the AM300 Arbitrary Generator.

1.1 Introduction

Introduction

The Rohde&Schwarz AM300 Arbitrary Generator drivers are single 32-bit drivers.

This Rohde&Schwarz AM300 Arbitrary Generator driver conforms to some parts of the VXI Plug& Play driver standard, which are applicable to conventional GPIB and other non-VXI instruments (that is, rack and stack instruments). The formal VXI Plug & Play standard only covers VXI Instruments, and some elements of the standard do not apply to the Rohde&Schwarz AM300 Arbitrary Generator since it is not a VXI instrument. One of the differences is, that there is no soft front panel, as the Rohde&Schwarz AM300 Arbitrary Generator can be controlled from its hardware front panel.

Options of the driver:

1. Conformance with the VXI Plug & Play standard. The only exception is that it does not have a soft front panel.
2. It is not built on top of, and does not uses the services provided by VISA. VISA is used for its prototype definitions and future compatibility with the other VXI Plug & Play drivers. If VISA library is not available, then rssitype.h provides data type definitions.
3. It includes a "Function Panel" (.fp) file, which allows it to be used with visual programming environments such as HP-VEE, LabWindows/CVI, and LabVIEW.
4. It includes a comprehensive on-line help file, which complements the instrument manual. The help file presents detailed documentation of each function.
5. The programming sources are included so that the driver can be modified if needed. The source conforms to VXI Plug & Play standards. Modifications should only be carried out by people who are familiar with the VXI Plug & Play standard.
6. It includes a Visual Basic include file (.bas) which contains the function calls in Visual Basic syntax, so that driver functions can be called from Visual Basic. If you use Visual Basic with this driver, you should be familiar with C/C++ function declarations. In particular, care must be taken when working with C/C++ pointers.

1.2 Instrument-Specific Information

Specific Information

The AM300 Arbitrary Generator instrument driver is used for remote control of device(s) connected via USB bus. The distribution package (installer) provides the host computer with all the support files necessary to be able establish a communication session between the host computer and device. The installation process is self-guided.

Instrument Description

The AM300 Arbitrary Generator instrument consists of two USB instruments, i.e. measurement module, and the system controller associated with the instrument platform in the power supply.

1.2.1 Instrument Addresses (Resource Strings)

Uppercase letters

When using VXI Plug & Play instrument drivers, instrument addresses should be written completely in uppercase letters. Implementation of the addressing scheme is vendor specific and some vendors support mixed cases. However, for maximum portability, the instrument address should use uppercase characters only.

Instrument Descriptor

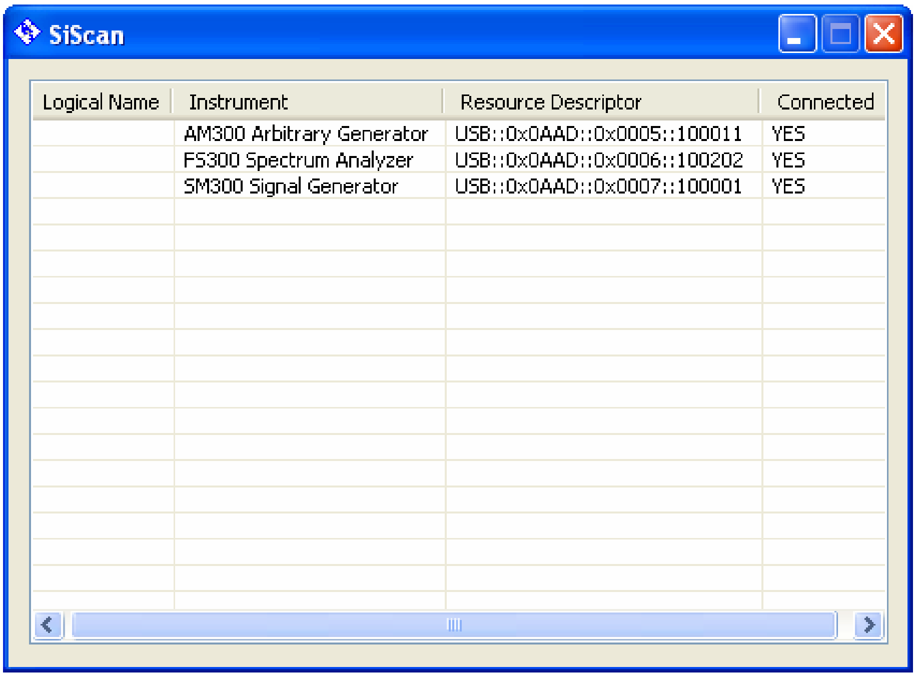
Based on the Instrument Descriptor (instrument address, resource string), the driver establishes a communication session with a device. The syntax of the Instrument Descriptor is shown below. Optional parameters are shown in square brackets ([]).

USB::manufacturer_ID::model_code::*serial_number*

Where:

- **USB** denotes used interface
- **Manufacturer ID** is 0xAAD for Rohde&Schwarz
- **Model Code** is the product identification (0x5 for AM300 Arbitrary Generator)
- **Serial Number** is the serial number printed on the instrument box

Instrument descriptors of the connected devices can be taken using build in utility ***SiScan***.



Logical Name	Instrument	Resource Descriptor	Connected
	AM300 Arbitrary Generator	USB::0x0AAD::0x0005::100011	YES
	FS300 Spectrum Analyzer	USB::0x0AAD::0x0006::100202	YES
	SM300 Signal Generator	USB::0x0AAD::0x0007::100001	YES

1.2.2 Using Callbacks

⚠ **Caution**

Callbacks are not supported with this driver.

1.2.3 Threat Safety

⚠ **Caution**

We recommend that no multiple threats are used to communicate with the instrument in parallel.

1.2.4 Device Identification and Logical Names

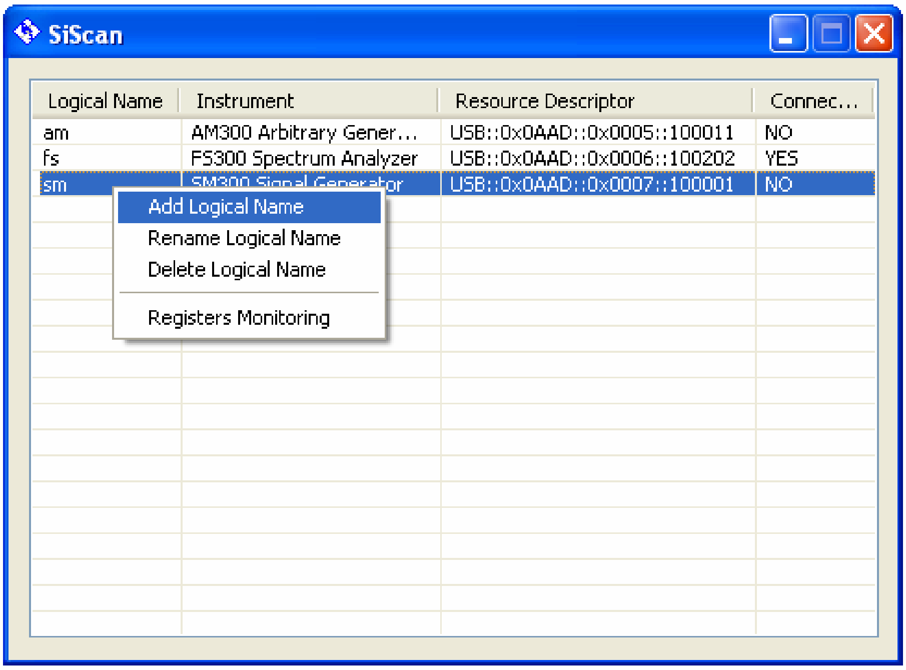
SiScan

For easy identification of devices on the USB bus use the ***SiScan*** application. ***SiScan*** is distributed with the driver, stored in the system's program folder (typically C:\Program Files\Series300\SiTools).

Instrument driver supports logical names as aliases of resource strings. You can pass logical name instead of instrument descriptor (resource string).

For example: **device_1** instead of **USB::0x0aad::0x5::123456**.

Logical names can be configured with the ***SiScan*** application.



Logical Name	Instrument	Resource Descriptor	Connec...
am	AM300 Arbitrary Gener...	USB::0x0AAD::0x0005::100011	NO
fs	FS300 Spectrum Analyzer	USB::0x0AAD::0x0006::100202	YES
sm	SM300 Signal Generator	USB::0x0AAD::0x0007::100001	NO

Windows registry

All the logical names are accessible under the Windows system registry key:

HKEY_LOCAL_MACHINE\SOFTWARE\Rohde&Schwarz\SiControl

Logical name record comprises the following items:

- **BoxResource,** which is the resource string used with device specific drivers (VXIpnp style instrument drivers)
- **ModuleResource** is the resource string used to open session with specified module (low-level function SiOpenDevice)
- **ModelName** is the string description of the device

Logical names are passed as alphanumeric strings. It is allowed to use more than one logical name for one device.

1.2.5 Hot Plug & Unplug Support

Description

Device can be set to local mode (unplugged) and then back to the remote mode (plugged) without loosing the initialized communication session. Once the session is opened (rssiam_init), session based data are safe until the session is closed (rssiam_close).

Communication session

A session is a communication path between a software element on host computer and a resource (instrument). Every communication session is unique. It is allowed to open up to 256 sessions per device.

1.2.6 SiTools

Description

A set of the utilities called ***SiTools*** is used to manage (***SiScan***) or monitor (***SiMonitor***) connected devices. All the respective utilities are stored under path defined in the windows registry under

HKEY_LOCAL_MACHINE\SOFTWARE\Rohde&Schwarz\SiTools

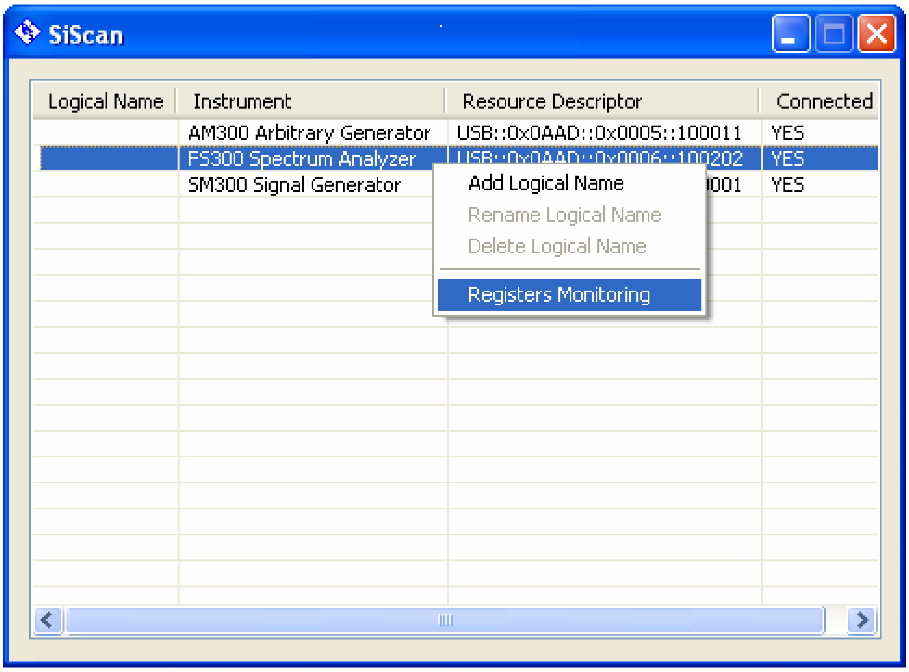
key (***SiToolsDir***). Each of ***SiTools*** provides comprehensive help how to use it. ***SiTools*** are installed with the driver and stored in the system's program folder (typically C:\Program Files\Series300\SiTools).

Note

Please note that the ***SiTools*** are not necessary for the system.

SiScan

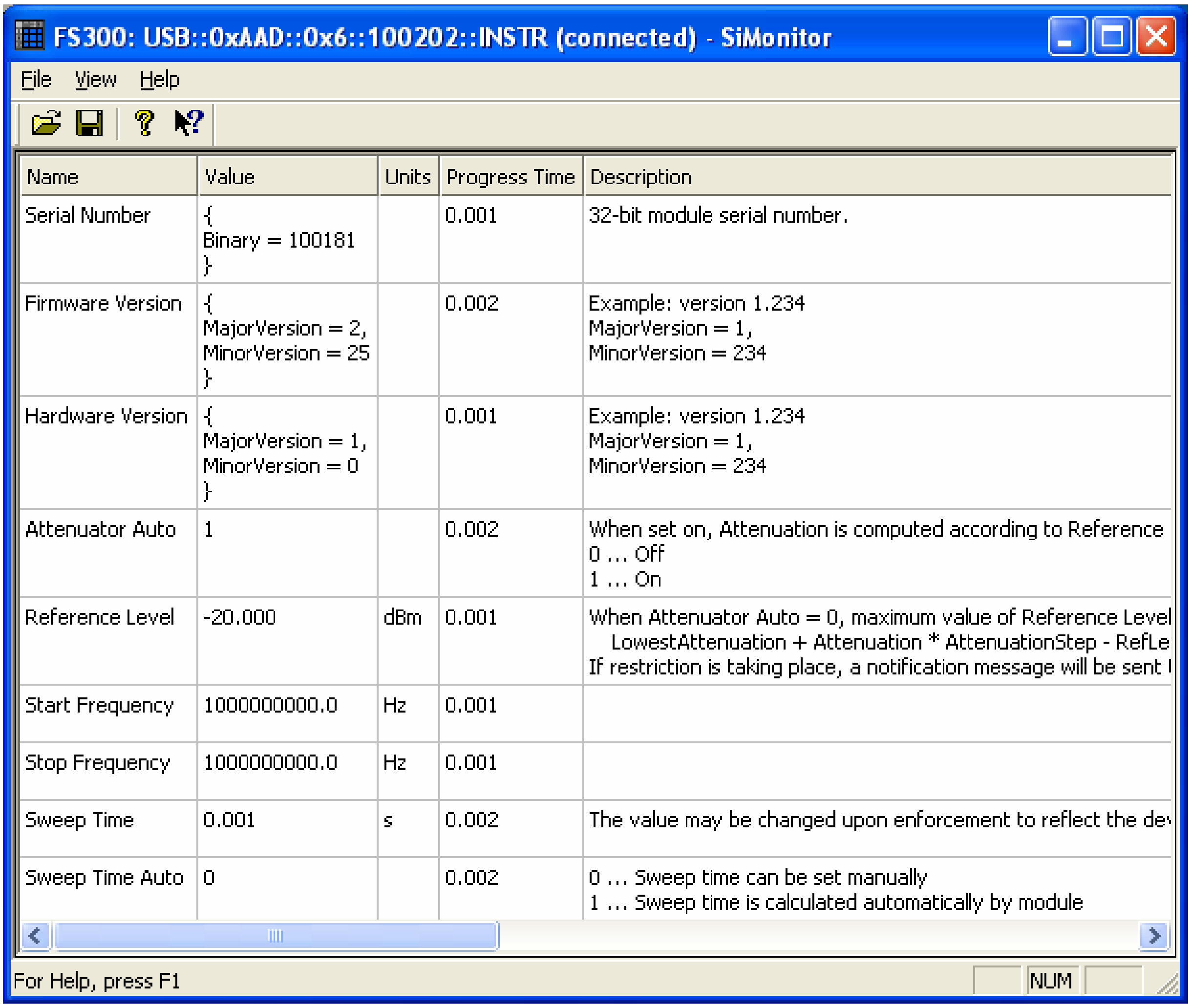
SiScan is tool providing also access to the low-level monitoring tool called ***SiMonitor***. ***SiMonitor*** can be launched from ***SiScan*** by mouse right-click on the connected device in the list.



Logical Name	Instrument	Resource Descriptor	Connected
	AM300 Arbitrary Generator	USB::0x0AAD::0x0005::100011	YES
	FS300 Spectrum Analyzer	USB::0x0AAD::0x0006::100202	YES
	SM300 Signal Generator	001	YES

SiMonitor

The ***SiMonitor*** is used to provide informations of current device settings. All parameters accessible in the table are core configuration elements of monitored device. Since polling these parameters affects the performance of other applications communicating with the instrument, it would be better to use it only for debugging and maintenance during development period of the remote control application. More detailed information on using the ***SiMonitor*** can be found in the associated help file.


FS300: USB::0xAAD::0x6::100202::INSTR (connected) - SiMonitor
File View Help
For Help, press F1
NUM

Name	Value	Units	Progress Time	Description

Serial Number	{
	Binary = 100181
	}
			0.001	32-bit module serial number.

Firmware Version	{
	MajorVersion = 2,
	MinorVersion = 25
	}
			0.002	Example: version 1.234
				MajorVersion = 1,
				MinorVersion = 234

Hardware Version	{
	MajorVersion = 1,
	MinorVersion = 0
	}
			0.001	Example: version 1.234
				MajorVersion = 1,
				MinorVersion = 234

Attenuator Auto	1		0.002	When set on, Attenuation is computed according to Reference 0 ... Off 1 ... On
Reference Level	-20.000	dBm	0.001	When Attenuator Auto = 0, maximum value of Reference Level LowestAttenuation + Attenuation * AttenuationStep - RefLe If restriction is taking place, a notification message will be sent t
Start Frequency	1000000000.0	Hz	0.001	
Stop Frequency	1000000000.0	Hz	0.001	
Sweep Time	0.001	s	0.002	The value may be changed upon enforcement to reflect the de
Sweep Time Auto	0		0.002	0 ... Sweep time can be set manually 1 ... Sweep time is calculated automatically by module

Note

Detail description of the monitored parameters is not provided within this manual. It is part of device specific interface (DSI) AM300 Arbitrary Generator measurement module documentation.

1.3 Using An Instrument Driver in Application Development Environments

This section offers suggestions on using the rssiam_32.dll within various application development environments.

1.3.1 Microsoft Visual C++ 4.0 (or higher) and Borland C++ 4.5 (or higher)

Refer to your Microsoft Visual C++ or Borland C++ manuals for information on linking and calling DLLs.

1. The driver uses Pascal calling conventions.
2. Rebuilding the driver DLL should be done in a different directory than the one the driver was installed in order to differentiate the changes.

1.3.2 Microsoft Visual Basic 5.0 (or higher)

Refer to the Microsoft Visual BASIC manual for information on calling DLLs. The BASIC include file is rssiam.bas, which is contained in the directory ~vxipnp\winnt\include. The ~ refers to the directory in the VXIPNP variable. By default this is set to C:\. You may also need to include the visa.bas file that comes with your VISA DLL.

1.3.3 HP VEE Version 3.2 (or higher)

Your copy of HP VEE for WINDOWS contains a document titled "Using VXIplug&play Drivers with HP VEE for Windows." This document contains the detailed information you need for HP VEE applications.

1.3.4 National Instruments LabWindows/CVI(R) 4.0.1 (or higher)

The AM300 Arbitrary Generator driver is supplied as both a source code file and as a dynamic link library file (dll). There are several advantages to using the dll form of the driver. These include:

1. Transportability across different computer platforms
2. Faster load time for your project

LabWindows/CVI (R) by default will attempt to load the source version of the instrument driver. To load the dll you must include the file rssiam.fp in your project. This file can be found in the vxipnp\winnt\rssiam directory. Do not include rssiam.c in your project. You must also provide an include path for rssiam.h. This is done by adding the directory ~vxipnp\winnt\include to the include paths (CVI Project Option menu) if you have not already done so. The ~ refers to the directory in the VXIPNP variable. By default this is set to C:\.

1.3.5 National Instruments LabVIEW(R) 6.1 (or higher)

If you want to use this driver as a standard LabVIEW driver, please copy the content of ~VXIpnp\GWinnt\rssiam directory into your LabVIEW directory (~LabVIEW\instr.lib\rssiam\) manually. The driver will then be directly accessible from the LabVIEW Instrument Driver function pallete menu.

1.4 VXIPNP Directory Location

The driver does not use VISA library, but it is installed to the VXIpnp directory and can be used as a standard VXIpnp instrument driver. If VISA library is not installed on the target machine, then installer will create a directory structure to fit with VXIpnp standard.

1.5 Files Installed

The install program will place the following files on the hard drive:

rssiam.h	Header file for use with C, HP VEE and LabView/LabWindows
rssiam.c	Source code for use with C
rssiam.def	Definition file for use with C++ when building the .dll file
rssiam.fp	Function Panel file for use with HP VEE and LabVIEW/LabWindows
rssiam.bas	Module file for use with Visual BASIC
rssiam.vb	Module file for use with .NET BASIC
rssiam.hlp	Help file for use with VB
rssiam.chm	Compressed HTML help for use with C and Visual Basic
rssiam.lib	Library file for use with C++
rssiam_32.dll	Dynamic Link Library file for use with all platforms
instrsup.dll	Instrument support Dynamic Link Library file from LabWindows/CVI.
SiControl.dll	Dynamic Link Library file for use with all platforms
SiControl.lib	Library file for use with C/C++
SiControl.h	Header file for use with C and LabWindows
rssitype.h	Header file for use with C and LabWindows (VISA data types)
rssi.inf	RSSI (USB I/O) Setup Information file
rssi.sys	RSSI (USB I/O) Driver file
readme.txt	File that contains general information
license.pdf	Instrument Driver License Agreement
SiTools	Set of device management utilities (SiMonitor, SiScan)

Table 1-1: Program Installation

LabVIEW installer in addition will place the following files on the hard drive:

rssiam.llb	LabVIEW library containing the driver VIs
rssiam.chm	LabVIEW Context Help (LabVIEW 6.1 or higher)
*.mnu	LabVIEW palette menu files of the driver

Table 1-2: LabVIEW Installation

The install program will also place additional support files for SiControl Library (Lite) to the following locations:

~vxipnp\winnt\SiControl
~vxipnp\gwinnt\SiControl

Table 1-3: SiControl (Lite) Installation

Note	If a particular platform is not going to be used, the corresponding platform-specific files may be deleted. Installer may bring more files than is listed in above section.

2 Programmer's Reference Manual

2.1 Instrument Driver Tree Structure

Class/Panel Name	Function Name
Initialization	rssiam_init
Application Functions	
Apply Instrument Setup	rssiam_appl
Apply Instrument Query	rssiam_appl_Q
Configuration Functions	
Channel Settings	
Output Channel Setup	rssiam_outpChannel
Output Channel Query	rssiam_outpChannel_Q
Output Channel Coupling Setup	rssiam_outpChannelCoupl
Output Channel Coupling Query	rssiam_outpChannelCoupl_Q
Output Configuration	
Output Function Shape Setup	rssiam_funcShap
Output Function Shape Query	rssiam_funcShap_Q
Output Frequency Setup	rssiam_freq
Output Frequency Query	rssiam_freq_Q
Output Voltage Setup	rssiam_volt
Output Voltage Query	rssiam_volt_Q
Output Voltage Offset Setup	rssiam_voltOffs
Output Voltage Offset Query	rssiam_voltOffs_Q
Output Voltage High Setup	rssiam_voltHigh
Output Voltage High Query	rssiam_voltHigh_Q
Output Voltage Low Setup	rssiam_voltLow
Output Voltage Low Query	rssiam_voltLow_Q
Output Voltage Unit Setup	rssiam_voltUnit
Output Voltage Unit Query	rssiam_voltUnit_Q
Output Square Duty Cycle Setup	rssiam_pulsDcyc
Output Square Duty Cycle Query	rssiam_pulsDcyc_Q
Output Ramp Symmetry Setup	rssiam_rampSymm
Output Ramp Symmetry Query	rssiam_rampSymm_Q
Output Load Setup	rssiam_outpLoad
Output Load User Setup	rssiam_outpLoadUser
Output Load Query	rssiam_outpLoad_Q

Class/Panel Name	**Function Name**
Output Sync Setup	rssiam_outpSync
Output Sync Query	rssiam_outpSync_Q
Output Setup	rssiam_outpSetup
Output Query	rssiam_outpSetup_Q
Output Start Phase Setup	rssiam_outpStartPhase
Output Start Phase Query	rssiam_outpStartPhase_Q
Output Filter Setup	rssiam_outpFilter
Output Filter Query	rssiam_outpFilter_Q
Output Filter Type Setup	rssiam_outpFilterType
Output Filter Type Query	rssiam_outpFilterType_Q
Output External Filter Setup	rssiam_extFilter
Output External Filter Query	rssiam_extFilter_Q
Output Aditional Setup	rssiam_outpAddition
Output Aditional Query	rssiam_outpAdditon_Q
Pulse Generator Settings	
Pulse Period Setup	rssiam_pulsePeriod
Pulse Period Query	rssiam_pulsePeriod_Q
Pulse Width Setup	rssiam_pulseWidth
Pulse Width Query	rssiam_pulseWidth_Q
Pulse Polarity Setup	rssiam_pulsePolarity
Pulse Polarity Query	rssiam_pulsePolarity_Q
Modulation Settings	
Modulation Points Setup	rssiam_mPoints
Modulation Points Query	rssiam_mPoints_Q
AM Modulation	
Amplitude Mod Depth Setup	rssiam_amDept
Amplitude Mod Depth Query	rssiam_amDept_Q
Amplitude Mod Int Func Setup	rssiam_amIntFunc
Amplitude Mod Int Func Query	rssiam_amIntFunc_Q
Amplitude Mod Int Freq Setup	rssiam_amIntFreq
Amplitude Mod Int Freq Query	rssiam_amIntFreq_Q
Amplitude Mod Stat Setup	rssiam_amState
Amplitude Mod Stat Query	rssiam_amState_Q
FM Modulation	
Frequency Mod Dev Setup	rssiam_fmDev
Frequency Mod Dev Query	rssiam_fmDev_Q
Frequency Mod Int Func Setup	rssiam_fmIntFunc

Class/Panel Name	Function Name
Frequency Mod Int Func Query	rssiam_fmIntFunc_Q
Frequency Mod Int Freq Setup	rssiam_fmIntFreq
Frequency Mod Int Freq Query	rssiam_fmIntFreq_Q
Frequency Mod Stat Setup	rssiam_fmStat
Frequency Mod Stat Query	rssiam_fmStat_Q
PM Modulation	
Phase Mod Dev Setup	rssiam_pmDev
Phase Mod Dev Query	rssiam_pmDev_Q
Phase Mod Int Func Setup	rssiam_pmIntFunc
Phase Mod Int Func Query	rssiam_pmIntFunc_Q
Phase Mod Int Freq Setup	rssiam_pmIntFreq
Phase Mod Int Freq Query	rssiam_pmIntFreq_Q
Phase Mod Stat Setup	rssiam_pmStat
Phase Mod Stat Query	rssiam_pmStat_Q
FSK Modulation	
FSK Frequency Setup	rssiam_fskFreq
FSK Frequency Query	rssiam_fskFreq_Q
FSK Internal Rate Setup	rssiam_fskIntRate
FSK Internal Rate Query	rssiam_fskIntRate_Q
FSK Source Setup	rssiam_fskSour
FSK Source Query	rssiam_fskSour_Q
FSK Stat Setup	rssiam_fskStat
FSK Stat Query	rssiam_fskStat_Q
PSK Modulation	
PSK Phase Setup	rssiam_pskPhase
PSK Phase Query	rssiam_pskPhase_Q
PSK Internal Rate Setup	rssiam_pskIntRate
PSK Internal Rate Query	rssiam_pskIntRate_Q
PSK Source Setup	rssiam_pskSour
PSK Source Query	rssiam_pskSour_Q
PSK Stat Setup	rssiam_pskStat
PSK Stat Query	rssiam_pskStat_Q
Arbitrary Waveform Settings	
Data Arb Points Setup	rssiam_dataArbPoin
Data Arb Points Query	rssiam_dataArbPoin_Q
Data Arb Start Setup	rssiam_dataArbStart
Data Arb Start Query	rssiam_dataArbStart_Q

Class/Panel Name	Function Name
Data Arb Stop Setup	rssiam_dataArbStop
Data Arb Stop Query	rssiam_dataArbStop_Q
Data Arb Rate Setup	rssiam_dataArbRate
Data Arb Rate Query	rssiam_dataArbRate_Q
Data Arb Rate Mode Setup	rssiam_dataArbRateMode
Data Arb Rate Mode Query	rssiam_dataArbRateMode_Q
Data Arb Volatile Setup	rssiam_dataVolatile
Data Arb Dac Volatile Setup	rssiam_dataDacVolatile
Sweep Settings	
Sweep Freq Start Setup	rssiam_freqStar
Sweep Freq Start Query	rssiam_freqStar_Q
Sweep Freq Stop Setup	rssiam_freqStop
Sweep Freq Stop Query	rssiam_freqStop_Q
Sweep Freq Center Setup	rssiam_freqCenter
Sweep Freq Center Query	rssiam_freqCenter_Q
Sweep Freq Span Setup	rssiam_freqSpan
Sweep Freq Span Query	rssiam_freqSpan_Q
Sweep Direction Setup	rssiam_sweDirection
Sweep Direction Query	rssiam_sweDirection_Q
Sweep Points Setup	rssiam_swePoin
Sweep Points Query	rssiam_swePoin_Q
Sweep Spacing Setup	rssiam_sweSpac
Sweep Spacing Query	rssiam_sweSpac_Q
Sweep Time Setup	rssiam_sweTime
Sweep Time Query	rssiam_sweTime_Q
Sweep Stat Setup	rssiam_sweStat
Sweep Stat Query	rssiam_sweStat_Q
Frequency Marker Settings	
Marker Frequency Setup	rssiam_mkrFreq
Marker Frequency Query	rssiam_mkrFreq_Q
Marker Stat Setup	rssiam_mkrStat
Marker Stat Query	rssiam_mkrStat_Q
Trigger Subsystem Settings	
Trigger Settings	
Trigger Source Setup	rssiam_trigSour
Trigger Source Query	rssiam_trigSour_Q
Trigger Polarity Setup	rssiam_trigPolarity

Class/Panel Name	**Function Name**
Trigger Polarity Query	rssiam_trigPolarity_Q
Trigger Delay Query	rssiam_trigDelay_Q
Trigger Frequency Setup	rssiam_trigFrequency
Trigger Frequency Query	rssiam_trigFrequency_Q
Trigger Period Setup	rssiam_trigPeriod
Trigger Period Query	rssiam_trigPeriod_Q
Burst Mod	
Burst Mod Ncycles Setup	rssiam_bmNcyc
Burst Mod Ncycles Query	rssiam_bmNcyc_Q
Burst Mod Int Rate Setup	rssiam_bmIntRate
Burst Mod Int Rate Query	rssiam_bmIntRate_Q
Burst Mod Sour Setup	rssiam_bmSour
Burst Mod Sour Query	rssiam_bmSour_Q
Burst Mod Stat Setup	rssiam_bmStat
Burst Mod Stat Query	rssiam_bmStat_Q
Sync Settings	
Sync Type Setup	rssiam_syncType
Sync Type Query	rssiam_syncType_Q
Sync Polarity Setup	rssiam_syncPolarity
Sync Polarity Query	rssiam_syncPolarity_Q
Gate Settings	
Gate Function Setup	rssiam_gateFunction
Gate Function Query	rssiam_gateFunction_Q
Gate Length Setup	rssiam_gateLength
Gate Length Query	rssiam_gateLength_Q
Gate Source Setup	rssiam_gateSource
Gate Source Query	rssiam_gateSource_Q
Gate Polarity Setup	rssiam_gatePolarity
Gate Polarity Query	rssiam_gatePolarity_Q
Action/Status Functions	
Send Trigger	rssiam_sendTrigger
Abort Trigger	rssiam_abortTrigger
Utility Functions	
Time Out	
Time Out Setup	rssiam_timeOut
Time Out Query	rssiam_timeOut_Q
Reference Oscillator	

Class/Panel Name	**Function Name**
Reference Oscillator Source Setup	rssiam_roscSour
Reference Oscillator Source Query	rssiam_roscSour_Q
Reference Oscillator Stat Setup	rssiam_roscStat
Reference Oscillator Stat Query	rssiam_roscStat_Q
Flush Error Queue	rssiam_flushErrorQueue
State Checking	rssiam_errorCheckState
Reset	rssiam_reset
Self-Test	rssiam_self_test
Error-Query	rssiam_error_query
Error Message	rssiam_error_message
Revision Query	rssiam_revision_query
Obsolete	
Trigger Slope Setup	rssiam_trigSlope
Trigger Slope Query	rssiam_trigSlope_Q
Output Summary Setup	rssiam_outpSum
Output Summary Query	rssiam_outpSum_Q
Gate Time Setup	rssiam_gateTime
Gate Time Query	rssiam_gateTime_Q
Close	rssiam_close

Table 2-1: Instrument driver tree structure

2.2 Function Tree Layout of the AM300 Arbitrary Generator

Description

This instrument module provides programming support for the Arbitrary Generator R&S AM300. The module is divided into the following functions:

Functions/Classes:

1. **Initialize:**

   This function initializes the instrument and sets it to a default configuration.

2. **Application Functions: (Class)**

   This class contains high-level, test and measurement routines. These example(s) call other instrument driver functions to configure, start, and read from the instrument.

3. **Configuration Functions: (Class)**

   This class of functions configures the instrument by setting acquisition and system configuration parameters.

4. **Action/Status Functions: (Class)**

   This class of functions begins or terminates an acquisition. It also provides functions which allow the user to determine the current status of the instrument.

5. **Utility Functions: (Class)**

   This class of functions provides lower level functions to communicate with the instrument, and change instrument parameters.

6. **Close:**

   This function takes the instrument offline.

2.2.1 Initialization

C Function Prototype

```
ViStatus rssiam_init (
    ViRsrc resourceName,
    ViBoolean idQuery,
    ViBoolean resetDevice,
    ViSession* instrumentHandle);
```

Basic Function Prototype

```
Function rssiam_init (
    ByVal resourceName As ViRsrc,
    ByVal idQuery As ViBoolean,
    ByVal resetDevice As ViBoolean,
    instrumentHandle As ViSession) As ViStatus
```

Purpose

This function performs the following initialization actions:

- Opens a session to the specified device using the interface and address specified in the Resource_Name control.
- Performs an identification query on the Instrument.
- Resets the instrument to a known state.
- Sends initialization commands to the instrument.
- Returns an Instrument Handle, which is used to differentiate between different sessions of this instrument driver.
- Each time this function is invoked a Unique Session is opened. It is possible to have more than one session open for the same resource.

Parameters List

1. **ViRsrc resourceName [in]**

   This control specifies the interface and address of the device that is to be initialized (Instrument Descriptor). The exact grammar to be used in this control is shown in the note below.

   Default Value: "USB::0xAAD::0x5::100011"

Note

Based on the Instrument Descriptor, this operation establishes a communication session with a device. The grammar for the Instrument Descriptor is shown below. Optional parameters are shown in square brackets ([]).

```
Interface   Grammar
------------------------------------------------------
USB::manufacturer_ID::model_code::serial_number
```

The USB keyword is used for USB interface. The Serial number is the (Model) Serial Number printed on the instrument box.

The driver also supports logical names. You can pass a logical name instead of instrument descriptor.

Example:"device_1" instead of USB::0x0aad::0x5::123456".

Logical names can be configured with the SiScan application.

2. **ViBoolean idQuery [in]**

   This control specifies if an ID Query is sent to the instrument during the initialization procedure.

   Valid Range:

   - VI_FALSE (0) - Skip Query
   - VI_TRUE (1) - Do Query (Default Value)

Note Under normal circumstances the ID Query ensures that the instrument initialized is the type supported by this driver. However circumstances may arise where it is undesirable to send an ID Query to the instrument. In those cases; set this control to "Skip Query" and this function will initialize the selected interface, without doing an ID Query.

3. **ViBoolean resetDevice [in]**

   This control specifies if the instrument is to be reset to its power-on settings during the initialization procedure.

   Valid Range:

   - VI_FALSE (0) - Don't Reset
   - VI_TRUE (1) - Reset Device (Default Value)

Note If you do not want the instrument to reset, set this control to "Don't Reset" while initializing the instrument.

4. **ViSession instrumentHandle [out]**

   This control returns an Instrument Handle that is used in all subsequent function calls to differentiate between different sessions of this instrument driver.

Note Each time this function is invoked a Unique Session is opened. It is possible to have more than one session open for the same resource.

Return Value Returns the status code of this operation.

The meaning of the status code is described in section Error (Status) Codes.

2.2.2 Application Functions

Description

This class contains high-level, test and measurement routines. These examples call other instrument driver functions to configure, start, and read from the instrument.

Functions/SubClasses:

1. **Apply Instrument Setup:**

   The Apply function provides the most straightforward method to program the function generator via the remote interface. You can select the function, frequency, amplitude and offset, all in one function.

2. **Apply Instrument Query:**

   Returns the parameters such as function, amplitude, and offset of the arbitrary generator.

2.2.2.1 Apply Instrument Setup

C Function Prototype

```
ViStatus rssiam_appl (
    ViSession instrumentHandle,
    ViInt16 function,
    ViReal64 frequency,
    ViReal64 amplitude,
    ViReal64 offset);
```

Basic Function Prototype

```
Function rssiam_appl (
    ByVal instrumentHandle As ViSession,
    ByVal function As ViInt16,
    ByVal frequency As ViReal64,
    ByVal amplitude As ViReal64,
    ByVal offset As ViReal64) As ViStatus
```

Purpose

This function provides the most straightforward method to program the function generator over the remote interface. You can select the function, frequency, amplitude, and offset all in one function.

Parameters List

1. **ViSession instrumentHandle [in]**

   This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.

   Default Value: None

2. **ViInt16 function [in]**

   Output waveform

   Valid Values:

▪ RSSIAM_OUTPUT_FUNC_SIN	(0)	- Sinusoid
▪ RSSIAM_OUTPUT_FUNC_SQU	(1)	- Square
▪ RSSIAM_OUTPUT_FUNC_TRI	(2)	- Triangle
▪ RSSIAM_OUTPUT_FUNC_RAMP	(3)	- Ramp (variable symmetry)
▪ RSSIAM_OUTPUT_FUNC_NOIS	(4)	- Noise
▪ RSSIAM_OUTPUT_FUNC_DC	(5)	- DC
▪ RSSIAM_OUTPUT_FUNC_USER	(6)	- User
▪ RSSIAM_OUTPUT_FUNC_SQUARE	(7)	- Square (variable duty cycle)
▪ RSSIAM_OUTPUT_FUNC_REXP	(8)	- Reverse Exponential
▪ RSSIAM_OUTPUT_FUNC_FEXP	(9)	- Forward Exponential
▪ RSSIAM_OUTPUT_FUNC_PULSE	(10)	- Pulse

Default Value: 0

3. **ViReal64 frequency [in]**
   Carrier frequency of the output signal

   Valid Range: 10.0e-6 Hz to 50.0e6 Hz

   Default Value: 1000.0 Hz

Note

Carrier frequency range depends on the function currently selected.

4. **ViReal64 amplitude [in]**
   Sets the output amplitude.

   Valid Ranges:

   Peak-to-peak: 0.001 V to 10 V

   DC function only: -5.0 V to 5.0 V

   Default Value: 1.0 V

Note

The relationship between Voltage Amplitude (Vpp) and Voltage Offset (Voffset) is

Vmin <= (Voffset + Vpp/2) <= Vmax

where Vmax is set by function Output Voltage High Setup (rssiam_voltHigh) and Vmin is set by function Output Voltage Low Setup (rssiam_voltLow).

5. **ViReal64 offset [in]**
   Sets the DC offset voltage.

   Valid Range: -5.0 V to 5.0 V

   Default Value: 0.0 V

Note

The relationship between Voltage Amplitude (Vpp) and Voltage Offset (Voffset) is

Vmin <= (Voffset + Vpp/2) <= Vmax

where Vmax is set by function Output Voltage High Setup (rssiam_voltHigh) and Vmin is set by function Output Voltage Low Setup (rssiam_voltLow).

Return Value

Returns the status code of this operation.

The meaning of the status code is described in section Error (Status) Codes.

2.2.2.2 Apply Instrument Query

C Function Prototype

```
ViStatus rssiam_appl_Q (
    ViSession instrumentHandle,
    ViChar[] queryResult);
```

Basic Function Prototype

```
Function rssiam_appl_Q (
    ByVal instrumentHandle As ViSession,
    queryResult As ViChar) As ViStatus
```

Purpose

This function returns the parameters such as function, frequency, amplitude, and offset currently active in the arbitrary generator.

Parameters List

1. **ViSession instrumentHandle [in]**

   This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.

   Default Value: None

2. **ViChar[] queryResult [out]**

   The function, frequency, amplitude, and offset are returned in this parameter.

Note

The array must contain at least 256 elements ViChar[256].

Return Value

Returns the status code of this operation.

The meaning of the status code is described in section Error (Status) Codes.

2.2.3 Configuration Functions

Description

This class of functions configures the instrument by setting acquisition and system configuration parameters.

2.2.3.1 Channel Settings

Description

Active channel and channel coupling operations are handled within this group of functions.

2.2.3.1.1 Output Channel Setup

C Function Prototype

```
ViStatus rssiam_outpChannel (
      ViSession instrumentHandle,
      ViInt16 activeChannel);
```

Basic Function Prototype

```
Function rssiam_outpChannel (
      ByVal instrumentHandle As ViSession,
      ByVal activeChannel As ViInt16) As ViStatus
```

Purpose

This function selects an active channel for which the settings are applied.

Note

Channel based operations are noted in the descriptions (applicable channels are listed).

Parameters List

1. **ViSession instrumentHandle [in]**

   This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.

   Default Value: None

2. **ViInt16 activeChannel [in]**

   Selects active channel.

   Valid Values:

   - RSSIAM_CH1 (1) - CH1 (Channel 1)
   - RSSIAM_CH2 (2) - CH2 (Channel 2)

   Default Value: 1

Return Value

Returns the status code of this operation.

The meaning of the status code is described in section Error (Status) Codes.

2.2.3.1.2 Output Channel Query

C Function Prototype

```
ViStatus rssiam_outpChannel_Q (
      ViSession instrumentHandle,
      ViInt16* activeChannel);
```

Basic Function Prototype

```
Function rssiam_outpChannel_Q (
      ByVal instrumentHandle As ViSession,
      activeChannel As ViInt16) As ViStatus
```

Purpose

This function returns the currently active channel for which the settings are applied.

Parameters List

1. **ViSession instrumentHandle [in]**

   This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.

   Default Value: None

2. **ViInt16 activeChannel [out]**

   Returns active channel.

   Valid Values:

RSSIAM_CH1	(1) - Channel 1 (CH1)
RSSIAM_CH2	(2) - Channel 2 (CH2)

Return Value

Returns the status code of this operation.

The meaning of the status code is described in section Error (Status) Codes.

2.2.3.1.3 Output Channel Coupling Setup

C Function Prototype

```
ViStatus rssiam_outpChannelCoupl (
      ViSession instrumentHandle,
      ViInt16 channelCoupling);
```

Basic Function Prototype

```
Function rssiam_outpChannelCoupl (
      ByVal instrumentHandle As ViSession,
      ByVal channelCoupling As ViInt16) As ViStatus
```

Purpose

This function select output channel(s) coupling.

Parameters List

1. **ViSession instrumentHandle [in]**

   This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.

   Default Value: None

2. **ViInt16 channelCoupling [in]**

   Select output channel(s) coupling.

   Valid Values:

RSSIAM_CHANNEL_COUPLING_OFF	(0x0) - Coupling Off
RSSIAM_CHANNEL_COUPLING_FREQ	(0x1) - Frequency
RSSIAM_CHANNEL_COUPLING_AMPT	(0x2) - Amplitude
RSSIAM_CHANNEL_COUPLING_OUTPUT	(0x4) - Output
RSSIAM_CHANNEL_COUPLING_PHASE	(0x8) - Phase

   Default Value: 0x1

Note

A Bit mask is used to set channel coupling.

A carrier frequency is common for both channels if any of the coupling functions are activated.

If coupling is set to Off, instrument acts with two independent carrier frequencies.

Example:

To set frequency, amplitude and phase channel coupling pass following

- RSSIAM_CHANNEL_COUPLING_FREQ ||
- RSSIAM_CHANNEL_COUPLING_AMPT ||
- RSSIAM_CHANNEL_COUPLING_PHASE

which is equal to (0x1) || (0x2) || (0x8) = 0xB

Return Value

Returns the status code of this operation.

The meaning of the status code is described in section Error (Status) Codes.

2.2.3.1.4 Output Channel Coupling Query

C Function Prototype

```
ViStatus rssiam_outpChannelCoupl_Q (
      ViSession instrumentHandle,
      ViInt16* channelCoupling);
```

Basic Function Prototype

```
Function rssiam_outpChannelCoupl_Q (
      ByVal instrumentHandle As ViSession,
      channelCoupling As ViInt16) As ViStatus
```

Purpose

This function returns output channel(s) coupling.

Parameters List

1. **ViSession instrumentHandle [in]**

   This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.

   Default Value: None

2. **ViInt16 channelCoupling [out]**

   Returns output channel(s) coupling.

   Valid Values:

RSSIAM_CHANNEL_COUPLING_OFF	(0x0) - Coupling Off
RSSIAM_CHANNEL_COUPLING_FREQ	(0x1) - Frequency
RSSIAM_CHANNEL_COUPLING_AMPT	(0x2) - Amplitude
RSSIAM_CHANNEL_COUPLING_OUTPUT	(0x4) - Output
RSSIAM_CHANNEL_COUPLING_PHASE	(0x8) - Phase

Note

Bit mask is used to set channel coupling.

Return Value

Returns the status code of this operation.

The meaning of the status code is described in section Error (Status) Codes.

2.2.3.2 Output Configuration

Description

This group of functions contains information to help you configure the arbitrary generator for outputting waveforms.

2.2.3.2.1 Output Function Shape Setup

C Function Prototype

```
ViStatus rssiam_funcShap (
      ViSession instrumentHandle,
      ViInt16 functionShape);
```

Basic Function Prototype

```
Function rssiam_funcShap (
      ByVal instrumentHandle As ViSession,
      ByVal functionShape As ViInt16) As ViStatus
```

Purpose

This function selects the function and outputs the selected waveform shape. The selected waveform is output using the previously selected frequency, amplitude, and offset settings.

Applicable channels: CH1 or CH2

Parameters List

1. **ViSession instrumentHandle [in]**

   This control accepts the Instrument Handle returned by the initialize function to select the desired instrument driver session.

   Default Value: None

2. **ViInt16 functionShape [in]**

   Selects carrier waveform shape of the standard output signal.

   Valid Values:

▪ RSSIAM_OUTPUT_FUNC_SIN	(0)	- Sinusoid
▪ RSSIAM_OUTPUT_FUNC_SQU	(1)	- Square
▪ RSSIAM_OUTPUT_FUNC_TRI	(2)	- Triangle
▪ RSSIAM_OUTPUT_FUNC_RAMP	(3)	- Ramp (variable symmetry)
▪ RSSIAM_OUTPUT_FUNC_NOIS	(4)	- Noise
▪ RSSIAM_OUTPUT_FUNC_DC	(5)	- DC
▪ RSSIAM_OUTPUT_FUNC_USER	(6)	- User
▪ RSSIAM_OUTPUT_FUNC_SQUARE	(7)	- Square (variable duty cycle)
▪ RSSIAM_OUTPUT_FUNC_REXP	(8)	- Reverse Exponential
▪ RSSIAM_OUTPUT_FUNC_FEXP	(9)	- Forward Exponential
▪ RSSIAM_OUTPUT_FUNC_PULSE	(10)	- Pulse

   Default Value: 0

Return Value

Returns the status code of this operation.

The meaning of the status code is described in section Error (Status) Codes.

2.2.3.2.2 Output Function Shape Query

C Function Prototype

```
ViStatus rssiam_funcShap_Q (
      ViSession instrumentHandle,
      ViInt16* functionShape);
```

Basic Function Prototype

```
Function rssiam_funcShap_Q (
      ByVal instrumentHandle As ViSession,
      functionShape As ViInt16) As ViStatus
```

Purpose

This function returns a numeric value indicating the current waveform shape.

Applicable channels: CH1 or CH2

Parameters List

1. **ViSession instrumentHandle [in]**

   This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.

   Default Value: None

2. **ViInt16 functionShape [out]**

   Carrier waveform shape of the standard output signal is returned in this parameter.

   Valid Values:

   - RSSIAM_OUTPUT_FUNC_SIN (0) - Sinusoid
   - RSSIAM_OUTPUT_FUNC_SQU (1) - Square
   - RSSIAM_OUTPUT_FUNC_TRI (2) - Triangle
   - RSSIAM_OUTPUT_FUNC_RAMP (3) - Ramp (variable symmetry)
   - RSSIAM_OUTPUT_FUNC_NOIS (4) - Noise
   - RSSIAM_OUTPUT_FUNC_DC (5) - DC
   - RSSIAM_OUTPUT_FUNC_USER (6) - User
   - RSSIAM_OUTPUT_FUNC_SQUARE (7) - Square (variable duty cycle)
   - RSSIAM_OUTPUT_FUNC_REXP (8) - Reverse Exponential
   - RSSIAM_OUTPUT_FUNC_FEXP (9) - Forward Exponential
   - RSSIAM_OUTPUT_FUNC_PULSE (10) - Pulse

Return Value

Returns the status code of this operation.

The meaning of the status code is described in section Error (Status) Codes.

2.2.3.2.3 Output Frequency Setup

C Function Prototype

```
ViStatus rssiam_freq (
      ViSession instrumentHandle,
      ViReal64 frequency);
```

Basic Function Prototype

```
Function rssiam_freq (
      ByVal instrumentHandle As ViSession,
      ByVal frequency As ViReal64) As ViStatus
```

Purpose

This function sets the carrier frequency of the output signal (Hz).

Applicable channels:

- CH1, CH2 if device config selects two independent frequencies

Note

If device config selects common frequency (see function Output Channel Coupling (rssiam_outpChannelCoupl())), channel does not apply.

Parameters List

1. **ViSession instrumentHandle [in]**

   This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.

   Default Value: None

2. **ViReal64 frequency [in]**

   Carrier frequency of the output signal

   Valid Range: 10.0e-6 Hz to 50.0e6 Hz

   Default Value: 1000.0 Hz

Note Carrier frequency range depends on the function currently selected.

Return Value

Returns the status code of this operation.

The meaning of the status code is described in section Error (Status) Codes.

2.2.3.2.4 Output Frequency Query

C Function Prototype

```
ViStatus rssiam_freq_Q (
      ViSession instrumentHandle,
      ViInt16 range,
      ViReal64* frequency);
```

Basic Function Prototype

```
Function rssiam_freq_Q (
      ByVal instrumentHandle As ViSession,
      ByVal range As ViInt16,
      frequency As ViReal64) As ViStatus
```

Purpose

This function returns the carrier frequency setting of the output signal (Hz).

Applicable channels:

- CH1, CH2 if device config selects two independent frequencies

Note If device config selects common frequency (see function Output Channel Coupling (rssiam_outpChannelCoupl())), channel does not apply.

Parameters List

1. **ViSession instrumentHandle [in]**

   This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.

   Default Value: None

2. **ViInt16 range [in]**

   Specifies whether the current, minimum or maximum setting is to be queried.

   Valid Values:

▪ RSSIAM_CURRENT_RANGE	(0) - Current
▪ RSSIAM_MIN_RANGE	(1) - Minimum
▪ RSSIAM_MAX_RANGE	(2) - Maximum

   Default Value: 0

3. **ViReal64 frequency [out]**

   Returns carrier frequency setting of the output signal (Hz).

Return Value

Returns the status code of this operation.

The meaning of the status code is described in section Error (Status) Codes.

2.2.3.2.5 Output Voltage Setup

C Function Prototype

```
ViStatus rssiam_volt (
      ViSession instrumentHandle,
      ViReal64 voltage);
```

Basic Function Prototype

```
Function rssiam_volt (
      ByVal instrumentHandle As ViSession,
      ByVal voltage As ViReal64) As ViStatus
```

Purpose

This function sets the output amplitude for currently active function.

Applicable channels: CH1 or CH2

Parameters List

1. **ViSession instrumentHandle [in]**

   This control accepts the Instrument Handle returned by the Initialize function select the desired instrument driver session.

   Default Value: None

2. **ViReal64 voltage [in]**

   Sets the output amplitude.

   Valid Ranges (for terminating resistance of 50 Ohms):

   - Peak-to-peak: 0.001 V to 10 V
   - DC function only: -5.0 V to 5.0 V

   and for Sinewave (only) applies also following:

   - Vpp: 0.001 V to 10 V
   - Vrms: 0.0003 Vrms to 3.5355 Vrms
   - dBm: -56.0206 dBm to 23.9794 dBm

   Default Value: 1.0 V

Note

The relationship between Voltage Amplitude (Vpp) and Voltage Offset (Voffset) is

Vmin <= (Voffset + Vpp/2) <= Vmax

where Vmax is set by function Output Voltage High Setup (rssiam_voltHigh) and Vmin is set by function Output Voltage Low Setup (rssiam_voltLow).

The currently active units are set calling function Output Voltage Unit Setup (rssiam_voltUnit).

Return Value

Returns the status code of this operation.

The meaning of the status code is described in section Error (Status) Codes.

2.2.3.2.6 Output Voltage Query

C Function Prototype

```
ViStatus rssiam_volt_Q (
      ViSession instrumentHandle,
      ViInt16 range,
      ViReal64* voltage);
```

Basic Function Prototype

```
Function rssiam_volt_Q (
    ByVal instrumentHandle As ViSession,
    ByVal range As ViInt16,
    voltage As ViReal64) As ViStatus
```

Purpose

This function returns the output amplitude for the currently selected function.

Applicable channels: CH1 or CH2

Parameters List

1. **ViSession instrumentHandle [in]**

   This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.

   Default Value: None

2. **ViInt16 range [in]**

   Specifies whether the current, minimum or maximum setting is to be queried.

   Valid Values:

RSSIAM_CURRENT_RANGE	(0) - Current
RSSIAM_MIN_RANGE	(1) - Minimum
RSSIAM_MAX_RANGE	(2) - Maximum

   Default Value: 0

3. **ViReal64 voltage [out]**

   Returns amplitude of output signal in selected units for Sinewave (Vpp ,Vrms, dBm) or as Vpp.

Return Value

Returns the status code of this operation.

The meaning of the status code is described in section Error (Status) Codes.

2.2.3.2.7 Output Voltage Offset Setup

C Function Prototype

```
ViStatus rssiam_voltOffs (
    ViSession instrumentHandle,
    ViReal64 voltageOffset);
```

Basic Function Prototype

```
Function rssiam_voltOffs (
    ByVal instrumentHandle As ViSession,
    ByVal voltageOffset As ViReal64) As ViStatus
```

Purpose

This function sets the DC offset voltage for currently active function.

Applicable channels: CH1 or CH2

Parameters List

1. **ViSession instrumentHandle [in]**

   This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.

   Default Value: None

2. **ViReal64 voltageOffset [in]**

   Sets the DC offset voltage.

   Valid Range: -5.0 V to 5.0 V

   Default Value: 0.0 V

Note

The relationship between Voltage Amplitude (Vpp) and Voltage Offset (Voffset) is

Vmin <= (Voffset + Vpp/2) <= Vmax

where Vmax is set by function Output Voltage High Setup (rssiam_voltHigh) and Vmin is set by function Output Voltage Low Setup (rssiam_voltLow).

Return Value

Returns the status code of this operation.

The meaning of the status code is described in section Error (Status) Codes.

2.2.3.2.8 Output Voltage Offset Query

C Function Prototype

```
ViStatus rssiam_voltOffs_Q (
      ViSession instrumentHandle,
      ViInt16 range,
      ViReal64* voltageOffset);
```

Basic Function Prototype

```
Function rssiam_voltOffs_Q (
      ByVal instrumentHandle As ViSession,
      ByVal range As ViInt16,
      voltageOffset As ViReal64) As ViStatus
```

Purpose

This function returns the DC offset voltage in volts for the currently selected function.

Applicable channels: CH1 or CH2

Parameters List

1. **ViSession instrumentHandle [in]**

   This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.

   Default Value: None

2. **ViInt16 range [in]**

   Specifies whether the current, minimum or maximum setting is to be queried.

   Valid Values:

   - RSSIAM_CURRENT_RANGE (0) - Current
   - RSSIAM_MIN_RANGE (1) - Minimum
   - RSSIAM_MAX_RANGE (2) - Maximum

   Default Value: 0

3. **ViReal64 voltageOffset [out]**

   Returns DC offset voltage value (V).

Return Value

Returns the status code of this operation.

The meaning of the status code is described in section Error (Status) Codes.

2.2.3.2.9 Output Voltage High Setup

C Function Prototype

```
ViStatus rssiam_voltHigh (
      ViSession instrumentHandle,
      ViReal64 voltageHigh);
```

Basic Function Prototype

```
Function rssiam_voltHigh (
      ByVal instrumentHandle As ViSession,
      ByVal voltageHigh As ViReal64) As ViStatus
```

Purpose

This function sets the high voltage level (related to offset). For all functions, the default high level is +500 mV.

You can set the levels to any valid positive or negative value, but note that the low level must always be lower than the high level. If you specify a high level that is lower than the low level, this function will automatically set the high level equal to the low level.

Applicable channels: CH1 or CH2

Note

If the current settings of Voltage Amplitude (Vpp) and Voltage Offset (Voffset) is in conflict with passed value, this function returns an error.

RSSIAM_ERROR_SETTINGS_CONFLICT.

Parameters List

1. **ViSession instrumentHandle [in]**

   This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.

   Default Value: None

2. **ViReal64 voltageHigh [in]**

   Sets the high voltage level related to offset.

   Valid Range: -5.0 V to 5.0 V

   Default Value: 0.5 V

Return Value

Returns the status code of this operation.

The meaning of the status code is described in section Error (Status) Codes.

2.2.3.2.10 Output Voltage High Query

C Function Prototype

```
ViStatus rssiam_voltHigh_Q (
      ViSession instrumentHandle,
      ViInt16 range,
      ViReal64* voltageHigh);
```

Basic Function Prototype

```
Function rssiam_voltHigh_Q (
      ByVal instrumentHandle As ViSession,
      ByVal range As ViInt16,
      voltageHigh As ViReal64) As ViStatus
```

Purpose

This function returns the high voltage level (related to offset) in DC volts for the currently selected function.

Applicable channels: CH1 or CH2

Parameters List

1. **ViSession instrumentHandle [in]**

   This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.

   Default Value: None

2. **ViInt16 range [in]**

   Specifies whether the current, minimum or maximum setting is to be queried.

   Valid Values:

▪ RSSIAM_CURRENT_RANGE	(0) - Current
▪ RSSIAM_MIN_RANGE	(1) - Minimum
▪ RSSIAM_MAX_RANGE	(2) - Maximum

   Default Value: 0

3. **ViReal64 voltageHigh [out]**

   Returns high voltage level value (V).

Return Value

Returns the status code of this operation.

The meaning of the status code is described in section Error (Status) Codes.

2.2.3.2.11 Output Voltage Low Setup

C Function Prototype

```
ViStatus rssiam_voltLow (
    ViSession instrumentHandle,
    ViReal64 voltageLow);
```

Basic Function Prototype

```
Function rssiam_voltLow (
    ByVal instrumentHandle As ViSession,
    ByVal voltageLow As ViReal64) As ViStatus
```

Purpose

This function sets the low voltage level (related to offset). For all functions, the default low level is -500 mV.

It is possible to set the levels to any valid positive or negative value, but note that the high level must always be higher than the low level. If you specify a low level that is higher than the high level, this function will automatically set the low level equal to the high level.

Applicable channels: CH1 or CH2

Note

If the current settings of Voltage Amplitude (Vpp) and Voltage Offset (Voffset) is in conflict with passed value, this function returns an error RSSIAM_ERROR_SETTINGS_CONFLICT.

Parameters List

1. **ViSession instrumentHandle [in]**

   This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.

   Default Value: None

2. **ViReal64 voltageLow [in]**

   Sets the low voltage level related to offset.

   Valid Range: -5.0 V to 5.0 V

   Default Value: -0.5 V

Return Value

Returns the status code of this operation.

The meaning of the status code is described in section Error (Status) Codes.

2.2.3.2.12 Output Voltage Low Query

C Function Prototype

```
ViStatus rssiam_voltLow_Q (
      ViSession instrumentHandle,
      ViInt16 range,
      ViReal64* voltageLow);
```

Basic Function Prototype

```
Function rssiam_voltLow_Q (
      ByVal instrumentHandle As ViSession,
      ByVal range As ViInt16,
      voltageLow As ViReal64) As ViStatus
```

Purpose

This function returns the low voltage level (related to offset) in DC volts for the currently selected function.

Applicable channels: CH1 or CH2

Parameters List

1. ViSession instrumentHandle [in]

   This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.

   Default Value: None

2. **ViInt16 range [in]**

   Specifies whether the current, minimum or maximum setting is to be queried.

   Valid Values:

RSSIAM_CURRENT_RANGE	(0) - Current
RSSIAM_MIN_RANGE	(1) - Minimum
RSSIAM_MAX_RANGE	(2) - Maximum

   Default Value: 0

3. **ViReal64 voltageLow [out]**

   Returns low voltage level value (V).

Return Value

Returns the status code of this operation.

The meaning of the status code is described in section Error (Status) Codes.

2.2.3.2.13 Output Voltage Unit Setup

C Function Prototype

```
ViStatus rssiam_voltUnit (
      ViSession instrumentHandle,
      ViInt16 voltageUnit);
```

Basic Function Prototype

```
Function rssiam_voltUnit (
      ByVal instrumentHandle As ViSession,
      ByVal voltageUnit As ViInt16) As ViStatus
```

Purpose

This function selects the output units for amplitude only.

Parameters List

1. **ViSession instrumentHandle [in]**

   This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.

   Default Value: None

2. **ViInt16 voltageUnit [in]**

   Selects the output units for amplitude.

   Valid Values:

   - RSSIAM_OUTPUT_VOLT_UNIT_VPP (0) – Vpp
   - RSSIAM_OUTPUT_VOLT_UNIT_VRMS (1) – Vrms
   - RSSIAM_OUTPUT_VOLT_UNIT_DBM (2) – dBm
   - RSSIAM_OUTPUT_VOLT_UNIT_DEF (3) - Default

   Default Value: 3

Note

Default amplitude units are Volts.

Unit dBm are allowed to be used only when output load is set to 50.0 ohms.

Return Value

Returns the status code of this operation.

The meaning of the status code is described in section Error (Status) Codes.

2.2.3.2.14 Output Voltage Unit Query

C Function Prototype

```
ViStatus rssiam_voltUnit_Q (
      ViSession instrumentHandle,
      ViInt16* voltageUnit);
```

Basic Function Prototype

```
Function rssiam_voltUnit_Q (
      ByVal instrumentHandle As ViSession,
      voltageUnit As ViInt16) As ViStatus
```

Purpose

This function returns a numeric indicating the current output units.

Parameters List

1. **ViSession instrumentHandle [in]**

   This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.

   Default Value: None

2. **ViInt16 voltageUnit [out]**

   Returns a numeric indicating of the current output units.

   Valid Values:

   - RSSIAM_OUTPUT_VOLT_UNIT_DEF (3) – Default

Note

Default amplitude units are Volts.

Return Value

Returns the status code of this operation.

The meaning of the status code is described in section Error (Status) Codes.

2.2.3.2.15 Output Square Duty Cycle Setup

C Function Prototype

```
ViStatus rssiam_pulsDcyc (
      ViSession instrumentHandle,
      ViReal64 dutyCycle);
```

Basic Function Prototype

```
Function rssiam_pulsDcyc (
      ByVal instrumentHandle As ViSession,
      ByVal dutyCycle As ViReal64) As ViStatus
```

Purpose

This function sets the duty cycle in percent for variable square waves only. Duty cycle represents the amount of time per cycle that the square wave is high.

Applicable channels: CH1 or CH2

Parameters List

1. **ViSession instrumentHandle [in]**

   This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.

   Default Value: None

2. **ViReal64 dutyCycle [in]**

   Sets the duty cycle in percent for variable square waves only.

   Valid Range: 0.0 % to 100.0 %

   Default Value: 50.0 %

Return Value

Returns the status code of this operation.

The meaning of the status code is described in section Error (Status) Codes.

2.2.3.2.16 Output Square Duty Cycle Query

C Function Prototype

```
ViStatus rssiam_pulsDcyc_Q (
      ViSession instrumentHandle,
      ViInt16 range,
      ViReal64* dutyCycle);
```

Basic Function Prototype

```
Function rssiam_pulsDcyc_Q (
      ByVal instrumentHandle As ViSession,
      ByVal range As ViInt16,
      dutyCycle As ViReal64) As ViStatus
```

Purpose

This function returns the variable square wave duty cycle in percent.

Applicable channels: CH1 or CH2

Parameters List

1. **ViSession instrumentHandle [in]**

   This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.

   Default Value: None

2. **ViInt16 range [in]**

   Specifies whether the current, minimum or maximum setting is to be queried.

   Valid Values:

▪ RSSIAM_CURRENT_RANGE	(0) - Current
▪ RSSIAM_MIN_RANGE	(1) - Minimum
▪ RSSIAM_MAX_RANGE	(2) - Maximum

   Default Value: 0

3. **ViReal64 dutyCycle [out]**

   Returns the variable square wave duty cycle in percent.

Return Value

Returns the status code of this operation.

The meaning of the status code is described in section Error (Status) Codes.

2.2.3.2.17 Output Ramp Symmetry Setup

C Function Prototype

```
ViStatus rssiam_rampSymm (
    ViSession instrumentHandle,
    ViReal64 rampSymmetry);
```

Basic Function Prototype

```
Function rssiam_rampSymm (
    ByVal instrumentHandle As ViSession,
    ByVal rampSymmetry As ViReal64) As ViStatus
```

Purpose

This function sets the symmetry percentage for ramp waves. Symmetry represents the amount of time per cycle that the ramp wave is rising.

Applicable channels: CH1 or CH2

Parameters List

1. **ViSession instrumentHandle [in]**

   This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.

   Default Value: None

2. **ViReal64 rampSymmetry [in]**

   Sets the symmetry percentage for ramp waves.

   Valid Range: 0.0 % to 100.0 %

   Default Value: 100.0 %

Return Value

Returns the status code of this operation.

The meaning of the status code is described in section Error (Status) Codes.

2.2.3.2.18 Output Ramp Symmetry Query

C Function Prototype

```
ViStatus rssiam_rampSymm_Q (
    ViSession instrumentHandle,
    ViInt16 range,
    ViReal64* rampSymmetry);
```

Basic Function Prototype

```
Function rssiam_rampSymm_Q (
    ByVal instrumentHandle As ViSession,
    ByVal range As ViInt16,
    rampSymmetry As ViReal64) As ViStatus
```

Purpose

This function returns the ramp wave symmetry in percent.

Applicable channels: CH1 or CH2

Parameters List

1. **ViSession instrumentHandle [in]**

   This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.

   Default Value: None

2. **ViInt16 range [in]**

   Specifies whether the current, minimum or maximum setting is to be queried.

   Valid Values:

▪ RSSIAM_CURRENT_RANGE	(0) - Current
▪ RSSIAM_MIN_RANGE	(1) - Minimum
▪ RSSIAM_MAX_RANGE	(2) - Maximum

   Default Value: 0

3. **ViReal64 rampSymmetry [out]**

   Returns the ramp wave symmetry value in percent.

Return Value

Returns the status code of this operation.

The meaning of the status code is described in section Error (Status) Codes.

2.2.3.2.19 Output Load Setup

C Function Prototype

```
ViStatus rssiam_outpLoad (
      ViSession instrumentHandle,
      ViInt16 outputLoad);
```

Basic Function Prototype

```
Function rssiam_outpLoad (
      ByVal instrumentHandle As ViSession,
      ByVal outputLoad As ViInt16) As ViStatus
```

Purpose

Selects the output termination for output amplitude and offset voltage.

Applicable channels: CH1 or CH2

The function generator has a fixed output impedance of 50 ohms on the OUTPUT terminals. The user value passed to the instrument driver is used only for calculation of the impedance matching, thus output voltage level is adjusted to fit the best match.

Incorrect impedance matching between the function generator and your load will result in amplitude or offset which does not match the specified signal level.

Parameters List

1. **ViSession instrumentHandle [in]**

   This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.

   Default Value: None

2. **ViInt16 outputLoad [in]**

   Sets the output termination for output amplitude and offset voltage.

   Valid Values:

   - RSSIAM_OUTPUT_LOAD_50 (0) - 50 ohms
   - RSSIAM_OUTPUT_LOAD_USER (1) - User

   Default Value: 0

Note

When user output load is selected, call function Output Load User Setup (rssiam_OutpLoadUser) to change its value.

Return Value

Returns the status code of this operation.

The meaning of the status code is described in section Error (Status) Codes.

2.2.3.2.20 Output Load User Setup

C Function Prototype

```
ViStatus rssiam_outpLoadUser (
   ViSession instrumentHandle,
   ViReal64 outputLoad);
```

Basic Function Prototype

```
Function rssiam_outpLoadUser (
   ByVal instrumentHandle As ViSession,
   ByVal outputLoad As ViReal64) As ViStatus
```

Purpose

Selects user value of the output termination for output amplitude and offset voltage.

Applicable channels: CH1 or CH2

The function generator has a fixed output impedance of 50 ohms on the OUTPUT terminals. The user value passed to the instrument driver is used only for calculation of the impedance matching, thus output voltage level is adjusted to fit the best match.

Incorrect impedance matching between the function generator and your load will result in amplitude or offset which does not match the specified signal level.

Applicable only when function Output Load Setup (rssiam_outpLoad) is set to RSSIAM_OUTPUT_LOAD_USER (1).

Parameters List

1. ViSession instrumentHandle [in]

This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.

Default Value: None

2. ViReal64 outputLoad [in]

Sets the output termination for output amplitude and offset voltage.

Valid Values: 1.0 ohms to 1.0e6 ohms

Default Value: 50.0 ohms

When units are dBm the only allowed output load is 50.0 ohms.

Return Value

Returns the status code of this operation.

The meaning of the status code is described in section Error (Status) Codes.

2.2.3.2.21 Output Load Query

C Function Prototype

```
ViStatus rssiam_outpLoad_Q (
      ViSession instrumentHandle,
      ViInt16 range,
      ViReal64* outputLoad);
```

Basic Function Prototype

```
Function rssiam_outpLoad_Q (
      ByVal instrumentHandle As ViSession,
      ByVal range As ViInt16,
      outputLoad As ViReal64) As ViStatus
```

Purpose

This function returns the output impedance.

Applicable channels: CH1 or CH2

Parameters List

1. **ViSession instrumentHandle [in]**

   This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.

   Default Value: None

2. **ViInt16 range [in]**

   Specifies whether the current, minimum or maximum setting is to be queried.

   Valid Values:

▪ RSSIAM_CURRENT_RANGE	(0) - Current
▪ RSSIAM_MIN_RANGE	(1) - Minimum
▪ RSSIAM_MAX_RANGE	(2) - Maximum

   Default Value: 0

3. **ViReal64 outputLoad [out]**

   Returns the output impedance in ohms.

Return Value

Returns the status code of this operation.

The meaning of the status code is described in section Error (Status) Codes.

2.2.3.2.22 Output Sync Setup

C Function Prototype

```
ViStatus rssiam_outpSync (
      ViSession instrumentHandle,
      ViInt16 outputSync);
```

Basic Function Prototype

```
Function rssiam_outpSync (
      ByVal instrumentHandle As ViSession,
      ByVal outputSync As ViInt16) As ViStatus
```

Purpose

This function enables or disables output from the SYNC terminal. The default is "OFF" When the sync signal is disabled, the output level on the SYNC terminal is indeterminate (it might be a TTL "high" or a TTL "low").

Applicable channels: CH1 or CH2

Parameters List

1. **ViSession instrumentHandle [in]**

   This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.

   Default Value: None

2. **ViInt16 outputSync [in]**

   Enables or disables output from the SYNC terminal.

   Valid Values:

   - RSSIAM_OUTPUT_SYNC_OFF (0) - Off
   - RSSIAM_OUTPUT_SYNC_ON (1) - On

   Default Value: 0

Return Value

Returns the status code of this operation.

The meaning of the status code is described in section Error (Status) Codes.

2.2.3.2.23 Output Sync Query

C Function Prototype

```
ViStatus rssiam_outpSync_Q (
      ViSession instrumentHandle,
      ViInt16* outputSync);
```

Basic Function Prototype

```
Function rssiam_outpSync_Q (
      ByVal instrumentHandle As ViSession,
      outputSync As ViInt16) As ViStatus
```

Purpose

This function returns whether the SYNC terminal output is enabled ("1") or disabled ("0").

Applicable channels: CH1 or CH2

Parameters List

1. **ViSession instrumentHandle [in]**

   This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.

   Default Value: None

2. **ViInt16 outputSync [out]**

   Returns whether the SYNC terminal output is enabled or disabled.

   Valid Values:

   - RSSIAM_OUTPUT_SYNC_OFF (0) - Off
   - RSSIAM_OUTPUT_SYNC_ON (1) - On

Return Value

Returns the status code of this operation.

The meaning of the status code is described in section Error (Status) Codes.

2.2.3.2.24 Output Setup

C Function Prototype

```
ViStatus rssiam_outpSetup (
      ViSession instrumentHandle,
      ViInt16 outputSetup);
```

Basic Function Prototype

```
Function rssiam_outpSetup (
      ByVal instrumentHandle As ViSession,
      ByVal outputSetup As ViInt16) As ViStatus
```

Purpose

This function enables or disables the front-panel output connector.

Applicable channels: CH1 or CH2

Parameters List

1. **ViSession instrumentHandle [in]**

   This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.

   Default Value: None

2. **ViInt16 outputSetup [in]**

   Enables or disables the front-panel output connector.

   Valid Values:

   - RSSIAM_OUTPUT_SETUP_OFF (0) - Off
   - RSSIAM_OUTPUT_SETUP_ON (1) - On

   Default Value: 0

Return Value

Returns the status code of this operation.

The meaning of the status code is described in section Error (Status) Codes.

2.2.3.2.25 Output Query

C Function Prototype

```
ViStatus rssiam_outpSetup_Q (
    ViSession instrumentHandle,
    ViInt16* outputSetup);
```

Basic Function Prototype

```
Function rssiam_outpSetup_Q (
    ByVal instrumentHandle As ViSession,
    outputSetup As ViInt16) As ViStatus
```

Purpose

This function returns whether the front-panel output connector is enabled or disabled.

Applicable channels: CH1 or CH2

Parameters List

1. **ViSession instrumentHandle [in]**

   This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.

   Default Value: None

2. **ViInt16 outputSetup [out]**

   Returns whether the front-panel output connector is enabled or disabled.

   Valid Values:

   - RSSIAM_OUTPUT_SETUP_OFF (0) - Off
   - RSSIAM_OUTPUT_SETUP_ON (1) - On

Return Value

Returns the status code of this operation.

The meaning of the status code is described in section Error (Status) Codes.

2.2.3.2.26 Output Start Phase Setup

C Function Prototype

```
ViStatus rssiam_outpStartPhase (
    ViSession instrumentHandle,
    ViReal64 startPhase);
```

Basic Function Prototype

```
Function rssiam_outpStartPhase (
    ByVal instrumentHandle As ViSession,
    ByVal startPhase As ViReal64) As ViStatus
```

Purpose

This function selects the start phase on the output of a channel.

Applicable channels: CH1 or CH2

Parameters List

1. **ViSession instrumentHandle [in]**

   This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.

   Default Value: None

2. **ViReal64 startPhase [in]**

   Selects the start phase on the output of a channel.

   Valid Range: -180.0 deg to 180.0 deg

   Default Value: 0.0 deg

Return Value

Returns the status code of this operation.

The meaning of the status code is described in section Error (Status) Codes.

2.2.3.2.27 Output Start Phase Query

C Function Prototype

```
ViStatus rssiam_outpStartPhase_Q (
      ViSession instrumentHandle,
      ViReal64* startPhase);
```

Basic Function Prototype

```
Function rssiam_outpStartPhase_Q (
      ByVal instrumentHandle As ViSession,
      startPhase As ViReal64) As ViStatus
```

Purpose

This function returns the start phase on the output of a channel.

Applicable channels: CH1 or CH2

Parameters List

1. **ViSession instrumentHandle [in]**

   This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.

   Default Value: None

2. **ViReal64 startPhase [out]**

   Returns the start phase on the output of a channel (deg).

Return Value

Returns the status code of this operation.

The meaning of the status code is described in section Error (Status) Codes.

2.2.3.2.28 Output Filter Setup

C Function Prototype

```
ViStatus rssiam_outpFilter (
      ViSession instrumentHandle,
      ViInt16 outputFilter);
```

Basic Function Prototype

```
Function rssiam_outpFilter (
      ByVal instrumentHandle As ViSession,
      ByVal outputFilter As ViInt16) As ViStatus
```

Purpose

This function selects the filter on the output of a channel.

Applicable channels: CH1 or CH2

Parameters List

1. **ViSession instrumentHandle [in]**

   This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.

   Default Value: None

2. **ViInt16 outputFilter [in]**

   Selects the filter on the output of a channel.

   Valid Values:

   - RSSIAM_OUTPUT_FILTER_OFF (0) - Off
   - RSSIAM_OUTPUT_FILTER_ON (1) - On
   - RSSIAM_OUTPUT_FILTER_AUTO (2) - Auto

   Default Value: 2

Return Value

Returns the status code of this operation.

The meaning of the status code is described in section Error (Status) Codes.

2.2.3.2.29 Output Filter Query

C Function Prototype

```
ViStatus rssiam_outpFilter_Q (
      ViSession instrumentHandle,
      ViInt16* outputFilter);
```

Basic Function Prototype

```
Function rssiam_outpFilter_Q (
      ByVal instrumentHandle As ViSession,
      outputFilter As ViInt16) As ViStatus
```

Purpose

This function returns the filter on the output of a channel.

Applicable channels: CH1 or CH2

Parameters List

1. **ViSession instrumentHandle [in]**

   This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.

   Default Value: None

2. **ViInt16 outputFilter [out]**

   Returns the filter on the output of a channel.

   Valid Values:

   - RSSIAM_OUTPUT_FILTER_OFF (0) - Off
   - RSSIAM_OUTPUT_FILTER_ON (1) - On
   - RSSIAM_OUTPUT_FILTER_AUTO (2) - Auto

Return Value

Returns the status code of this operation.

The meaning of the status code is described in section Error (Status) Codes.

2.2.3.2.30 Output Filter Type Setup

C Function Prototype

```
ViStatus rssiam_outpFilterType (
      ViSession instrumentHandle,
      ViInt16 outputFilterType);
```

Basic Function Prototype

```
Function rssiam_outpFilterType (
      ByVal instrumentHandle As ViSession,
      ByVal outputFilterType As ViInt16) As ViStatus
```

Purpose

This function selects the characteristic of the low pass filter.

Applicable channels: CH1 or CH2

Parameters List

1. **ViSession instrumentHandle [in]**

   This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.

   Default Value: None

2. **ViInt16 outputFilterType [in]**

   Selects the characteristic of the low pass filter.

   Valid Values:

   - RSSIAM_OUTPUT_FILTER_TYPE_CAUER (0) - Cauer 35 MHz
   - RSSIAM_OUTPUT_FILTER_TYPE_BESSEL10 (1) - Bessel 37 MHz
   - RSSIAM_OUTPUT_FILTER_TYPE_BESSEL75 (2) - Bessel 75 MHz

   Default Value: 0

Return Value

Returns the status code of this operation.

The meaning of the status code is described in section Error (Status) Codes.

2.2.3.2.31 Output Filter Type Query

C Function Prototype

```
ViStatus rssiam_outpFilterType_Q (
      ViSession instrumentHandle,
      ViInt16* outputFilterType);
```

Basic Function Prototype

```
Function rssiam_outpFilterType_Q (
      ByVal instrumentHandle As ViSession,
      outputFilterType As ViInt16) As ViStatus
```

Purpose

This function returns the characteristic of the low pass filter.

Applicable channels: CH1 or CH2

Parameters List

1. **ViSession instrumentHandle [in]**

   This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.

   Default Value: None

2. **ViInt16 outputFilterType [out]**

   Returns the characteristic of the low pass filter.

   Valid Values:

   - RSSIAM_OUTPUT_FILTER_TYPE_CAUER (0) - Cauer 35 MHz
   - RSSIAM_OUTPUT_FILTER_TYPE_BESSEL10 (1) - Bessel 37 MHz
   - RSSIAM_OUTPUT_FILTER_TYPE_BESSEL75 (2) - Bessel 75 MHz

Return Value

Returns the status code of this operation.

The meaning of the status code is described in section Error (Status) Codes.

2.2.3.2.32 Output External Filter Setup

C Function Prototype

```
ViStatus rssiam_extFilter (
      ViSession instrumentHandle,
      ViInt16 externalFilter);
```

Basic Function Prototype

```
Function rssiam_extFilter (
      ByVal instrumentHandle As ViSession,
      ByVal externalFilter As ViInt16) As ViStatus
```

Purpose

This function selects if external filter is connected.

Applicable channels: CH1 or CH2

Parameters List

1. **ViSession instrumentHandle [in]**

   This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.

   Default Value: None

2. **ViInt16 externalFilter [in]**

   Selects if external filter is connected.

   Valid Values:

   - RSSIAM_OUTPUT_EXT_FILTER_OFF (0) - Off
   - RSSIAM_OUTPUT_EXT_FILTER_ON (1) - On

   Default Value: 0

Return Value

Returns the status code of this operation.

The meaning of the status code is described in section Error (Status) Codes.

2.2.3.2.33 Output External Filter Query

C Function Prototype

```
ViStatus rssiam_extFilter_Q (
      ViSession instrumentHandle,
      ViInt16* externalFilter);
```

Basic Function Prototype

```
Function rssiam_extFilter_Q (
      ByVal instrumentHandle As ViSession,
      externalFilter As ViInt16) As ViStatus
```

Purpose

This function returns if external filter is connected.

Applicable channels: CH1 or CH2

Parameters List

1. **ViSession instrumentHandle [in]**

   This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.

   Default Value: None

2. **ViInt16 externalFilter [out]**

   Returns if external filter is connected.

   Valid Values:

   - RSSIAM_OUTPUT_EXT_FILTER_OFF (0) - Off
   - RSSIAM_OUTPUT_EXT_FILTER_ON (1) - On

Return Value

Returns the status code of this operation.

The meaning of the status code is described in section Error (Status) Codes.

2.2.3.2.34 Output Aditional Setup

C Function Prototype

```
ViStatus rssiam_outpAddition  (
    ViSession instrumentHandle,
    ViInt16 outputAddition);
```

Basic Function Prototype

```
Function rssiam_outpAddition (
    ByVal instrumentHandle As ViSession,
    ByVal outputAddition As ViInt16) As ViStatus
```

Purpose

Sets whether the output signal Ch1 = (Ch1 + Ch2) is enabled or disabled.

Parameters List

1. **ViSession instrumentHandle [in]**

   This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.

   Default Value: None

2. **ViInt16 outputAddition [in]**

   Sets whether the output signal Ch1 = (Ch1 + Ch2) is enabled or disabled.

   Valid Values:

   - RSSIAM_OUTPUT_SUM_OFF (0) – Disabled
   - RSSIAM_OUTPUT_SUM_ON (1) - Enabled

   Default Value: 0

Return Value

Returns the status code of this operation.

The meaning of the status code is described in section Error (Status) Codes.

2.2.3.2.35 Output Aditional Query

C Function Prototype

```
ViStatus rssiam_outpAdditon_Q  (
    ViSession instrumentHandle,
    ViInt16* outputSummary);
```

Basic Function Prototype

```
Function rssiam_outpAdditon_Q (
    ByVal instrumentHandle As ViSession,
    outputSummary As ViInt16) As ViStatus
```

Purpose

Returns whether the output signal Ch1 = (Ch1 + Ch2) is enabled or disabled.

Parameters List

1. **ViSession instrumentHandle [in]**

   This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.

   Default Value: None

2. **ViInt16 outputSummary [out]**

   Returns whether the output signal Ch1 = (Ch1 + Ch2) is enabled or disabled.

   Valid Values:
   - RSSIAM_OUTPUT_SUM_OFF (0) – Disabled
   - RSSIAM_OUTPUT_SUM_ON (1) - Enabled

Return Value

Returns the status code of this operation.

The meaning of the status code is described in section Error (Status) Codes.

2.2.3.3 Pulse Generator Settings

Description

This group of functions contains information to help you configure the function generator for outputting pulse waveforms.

2.2.3.3.1 Pulse Period Setup

C Function Prototype

```
ViStatus rssiam_pulsePeriod (
      ViSession instrumentHandle,
      ViReal64 pulsePeriod);
```

Basic Function Prototype

```
Function rssiam_pulsePeriod (
      ByVal instrumentHandle As ViSession,
      ByVal pulsePeriod As ViReal64) As ViStatus
```

Purpose

This function sets period of a pulsed waveform.

Parameters List

1. **ViSession instrumentHandle [in]**

   This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.

   Default Value: None

2. **ViReal64 pulsePeriod [in]**

   Sets the period for pulses.

   Valid Range: 20.0e-9 s to 9999.0 s

   Default Value: 0.001 s

Return Value

Returns the status code of this operation.

The meaning of the status code is described in section Error (Status) Codes.

2.2.3.3.2 Pulse Period Query

C Function Prototype

```
ViStatus rssiam_pulsePeriod_Q (
    ViSession instrumentHandle,
    ViInt16 range,
    ViReal64* pulsePeriod);
```

Basic Function Prototype

```
Function rssiam_pulsePeriod_Q (
    ByVal instrumentHandle As ViSession,
    ByVal range As ViInt16,
    pulsePeriod As ViReal64) As ViStatus
```

Purpose

This function returns the period of the pulse waveform in seconds.

Parameters List

1. **ViSession instrumentHandle [in]**

   This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.

   Default Value: None

2. **ViInt16 range [in]**

   Specifies whether the current, minimum or maximum setting is to be queried.

   Valid Values:

   - RSSIAM_CURRENT_RANGE (0) - Current
   - RSSIAM_MIN_RANGE (1) - Minimum
   - RSSIAM_MAX_RANGE (2) - Maximum

   Default Value: 0

3. **ViReal64 pulsePeriod [out]**

   Returns period of a pulsed waveform (s).

Return Value

Returns the status code of this operation.

The meaning of the status code is described in section Error (Status) Codes.

2.2.3.3.3 Pulse Width Setup

C Function Prototype

```
ViStatus rssiam_pulseWidth (
    ViSession instrumentHandle,
    ViReal64 pulseWidth);
```

Basic Function Prototype

```
Function rssiam_pulseWidth (
    ByVal instrumentHandle As ViSession,
    ByVal pulseWidth As ViReal64) As ViStatus
```

Purpose

This function sets the width or duration of the pulse in seconds. The pulse width represents the time from the 50% threshold of the rising edge of the pulse to the 50% threshold of the next falling edge.

Applicable channels:

- CH1, CH2 if device config selects two independent frequencies

Note

If device config selects common frequency (see function Output Channel Coupling (rssiam_outpChannelCoupl())), channel does not apply.

Parameters List

1. **ViSession instrumentHandle [in]**

   This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.

   Default Value: None

2. **ViReal64 pulseWidth [in]**

   Sets the width or duration of the pulse in seconds.

   Valid Range: 10.0e-9 s to 9999.0 s

   Default Value: 5.0e-4 s

Return Value

Returns the status code of this operation.

The meaning of the status code is described in section Error (Status) Codes.

2.2.3.3.4 Pulse Width Query

C Function Prototype

```
ViStatus rssiam_pulseWidth_Q (
      ViSession instrumentHandle,
      ViInt16 range,
      ViReal64* pulseWidth);
```

Basic Function Prototype

```
Function rssiam_pulseWidth_Q (
      ByVal instrumentHandle As ViSession,
      ByVal range As ViInt16,
      pulseWidth As ViReal64) As ViStatus
```

Purpose

This function returns the width or duration of the pulse in seconds.

Applicable channels:

- CH1, CH2 if device config selects two independent frequencies

Note

If device config selects common frequency (see function Output Channel Coupling (rssiam_outpChannelCoupl())), channel does not apply.

Parameters List

1. **ViSession instrumentHandle [in]**

   This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.

   Default Value: None

2. **ViInt16 range [in]**

   Specifies whether the current, minimum or maximum setting is to be queried.

   Valid Values:

RSSIAM_CURRENT_RANGE	(0) - Current
RSSIAM_MIN_RANGE	(1) - Minimum
RSSIAM_MAX_RANGE	(2) - Maximum

   Default Value: 0

3. **ViReal64 pulseWidth [out]**

   Returns the width or duration of the pulse in seconds.

Return Value

Returns the status code of this operation.

The meaning of the status code is described in section Error (Status) Codes.

2.2.3.3.5 Pulse Polarity Setup

C Function Prototype

```
ViStatus rssiam_pulsePolarity (
      ViSession instrumentHandle,
      ViInt16 pulsePolarity);
```

Basic Function Prototype

```
Function rssiam_pulsePolarity (
      ByVal instrumentHandle As ViSession,
      ByVal pulsePolarity As ViInt16) As ViStatus
```

Purpose

This function sets the polarity of pulses.

Applicable channels: CH1 or CH2

Parameters List

1. **ViSession instrumentHandle [in]**

   This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.

   Default Value: None

2. **ViInt16 pulsePolarity [in]**

   Sets the polarity of pulses.

   Valid Values:

   - RSSIAM_PULSE_POLARITY_NORMAL (0) - Normal
   - RSSIAM_PULSE_POLARITY_INVERTED (1) - Inverted

   Default Value: 0

Return Value

Returns the status code of this operation.

The meaning of the status code is described in section Error (Status) Codes.

2.2.3.3.6 Pulse Polarity Query

C Function Prototype

```
ViStatus rssiam_pulsePolarity_Q (
      ViSession instrumentHandle,
      ViInt16* pulsePolarity);
```

Basic Function Prototype

```
Function rssiam_pulsePolarity_Q (
      ByVal instrumentHandle As ViSession,
      pulsePolarity As ViInt16) As ViStatus
```

Purpose

This function returns the polarity of pulses.

Applicable channels: CH1 or CH2

Parameters List

1. **ViSession instrumentHandle [in]**

   This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.

   Default Value: None

2. **ViInt16 pulsePolarity [out]**

   Returns the polarity of pulses.

   Valid Values:

   - RSSIAM_PULSE_POLARITY_NORMAL (0) - Normal
   - RSSIAM_PULSE_POLARITY_INVERTED (1) - Inverted

Return Value

Returns the status code of this operation.

The meaning of the status code is described in section Error (Status) Codes.

2.2.3.4 Modulation Settings

Description

This group of functions can be used to setup and query the modulation parameters. A modulated Waveform consists of a carrier and a modulating waveform.

2.2.3.4.1 Modulation Points Setup

C Function Prototype

```
ViStatus rssiam_mPoints (
      ViSession instrumentHandle,
      ViInt32 modulationPoints);
```

Basic Function Prototype

```
Function rssiam_mPoints (
      ByVal instrumentHandle As ViSession,
      ByVal modulationPoints As ViInt32) As ViStatus
```

Purpose

This function sets additional parameters to AM, FM, PM (number of modulation points to calculate envelope of the modulated signal).

Parameters List

1. **ViSession instrumentHandle [in]**

   This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.

   Default Value: None

2. **ViInt32 modulationPoints [in]**

   Sets the number of modulation points.

   Valid Range: 2 to 65535

   Default Value: 64

Return Value

Returns the status code of this operation.

The meaning of the status code is described in section Error (Status) Codes.

2.2.3.4.2 Modulation Points Query

C Function Prototype

```
ViStatus rssiam_mPoints_Q (
      ViSession instrumentHandle,
      ViInt16 range,
      ViInt32* modulationPoints);
```

Basic Function Prototype

```
Function rssiam_mPoints_Q (
      ByVal instrumentHandle As ViSession,
      ByVal range As ViInt16,
      modulationPoints As ViInt32) As ViStatus
```

Purpose

This function returns additional parameter of AM, FM, PM (number of modulation points used to calculate the envelope of the modulated signal).

Parameters List

1. **ViSession instrumentHandle [in]**

   This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.

   Default Value: None

2. **ViInt16 range [in]**

   Specifies whether the current, minimum or maximum setting is to be queried.

   Valid Values:

RSSIAM_CURRENT_RANGE	(0) - Current
RSSIAM_MIN_RANGE	(1) - Minimum
RSSIAM_MAX_RANGE	(2) - Maximum

   Default Value: 0

3. **ViInt32 modulationPoints [out]**

   Returns number of modulation points.

Return Value

Returns the status code of this operation.

The meaning of the status code is described in section Error (Status) Codes.

2.2.3.4.3 AM Modulation

Description

In AM, the amplitude of the carrier is varied by the amplitude of the modulating waveform. These functions can be used to setup and query the amplitude modulated parameters.

2.2.3.4.3.1 Amplitude Mod Depth Setup

C Function Prototype

```
ViStatus rssiam_amDept (
      ViSession instrumentHandle,
      ViReal64 amplitudeDepth);
```

Basic Function Prototype

```
Function rssiam_amDept (
      ByVal instrumentHandle As ViSession,
      ByVal amplitudeDepth As ViReal64) As ViStatus
```

Purpose

This function sets the internal modulation depth in percent.

Applicable channels: CH1 or CH2

Parameters List

1. **ViSession instrumentHandle [in]**

   This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.

   Default Value: None

2. **ViReal64 amplitudeDepth [in]**

   Sets the internal modulation depth in percent.

   Valid Range: 0.0 % to 100.0 %

   Default Value: 100.0 %

Return Value

Returns the status code of this operation.

The meaning of the status code is described in section Error (Status) Codes.

2.2.3.4.3.2 Amplitude Mod Depth Query

C Function Prototype

```
ViStatus rssiam_amDept_Q (
      ViSession instrumentHandle,
      ViInt16 range,
      ViReal64* amplitudeDepth);
```

Basic Function Prototype

```
Function rssiam_amDept_Q (
      ByVal instrumentHandle As ViSession,
      ByVal range As ViInt16,
      amplitudeDepth As ViReal64) As ViStatus
```

Purpose

This function returns the internal modulation depth.

Applicable channels: CH1 or CH2

Parameters List

1. **ViSession instrumentHandle [in]**

   This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.

   Default Value: None

2. **ViInt16 range [in]**

   Specifies whether the current, minimum or maximum setting is to be queried.

   Valid Values:

▪ RSSIAM_CURRENT_RANGE	(0) - Current
▪ RSSIAM_MIN_RANGE	(1) - Minimum
▪ RSSIAM_MAX_RANGE	(2) - Maximum

   Default Value: 0

3. **ViReal64 amplitudeDepth [out]**

   Returns the internal modulation depth in percent.

Return Value

Returns the status code of this operation.

The meaning of the status code is described in section Error (Status) Codes.

2.2.3.4.3.3 Amplitude Mod Int Func Setup

C Function Prototype

```
ViStatus rssiam_amIntFunc (
      ViSession instrumentHandle,
      ViInt16 amIntFunction);
```

Basic Function Prototype

```
Function rssiam_amIntFunc (
      ByVal instrumentHandle As ViSession,
      ByVal amIntFunction As ViInt16) As ViStatus
```

Purpose

This function selects the shape of the modulating waveform.

Applicable channels: CH1 or CH2

Parameters List

1. **ViSession instrumentHandle [in]**

   This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.

   Default Value: None

2. **ViInt16 amIntFunction [in]**

   Selects the shape of the modulating waveform.

   Valid Values:

RSSIAM_AMP_INT_FUNC_SIN	(0) - Sinusoid
RSSIAM_AMP_INT_FUNC_SQU	(1) - Square
RSSIAM_AMP_INT_FUNC_TRI	(2) - Triangle
RSSIAM_AMP_INT_FUNC_URAMP	(3) - Ramp (Up)
RSSIAM_AMP_INT_FUNC_DRAMP	(4) - Ramp (Down)
RSSIAM_AMP_INT_FUNC_USER	(5) - User
RSSIAM_AMP_INT_FUNC_REXP	(8) - Reverse Exponential
RSSIAM_AMP_INT_FUNC_FEXP	(9) - Forward Exponential

   Default Value: 0

Return Value

Returns the status code of this operation.

The meaning of the status code is described in section Error (Status) Codes.

2.2.3.4.3.4 Amplitude Mod Int Func Query

C Function Prototype

```
ViStatus rssiam_amIntFunc_Q (
      ViSession instrumentHandle,
      ViInt16* amIntFunction);
```

Basic Function Prototype

```
Function rssiam_amIntFunc_Q (
      ByVal instrumentHandle As ViSession,
      amIntFunction As ViInt16) As ViStatus
```

Purpose

This function returns a numeric value indicating the shape of the internal modulating waveform.

Applicable channels: CH1 or CH2

Parameters List

1. **ViSession instrumentHandle [in]**

   This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.

   Default Value: None

2. **ViInt16 amIntFunction [out]**

   Returns the shape of the modulating waveform.

   Valid Values:

RSSIAM_AMP_INT_FUNC_SIN	(0) - Sinusoid
RSSIAM_AMP_INT_FUNC_SQU	(1) - Square
RSSIAM_AMP_INT_FUNC_TRI	(2) - Triangle
RSSIAM_AMP_INT_FUNC_URAMP	(3) - Ramp (Up)
RSSIAM_AMP_INT_FUNC_DRAMP	(4) - Ramp (Down)
RSSIAM_AMP_INT_FUNC_USER	(5) - User
RSSIAM_AMP_INT_FUNC_REXP	(8) - Reverse Exponential
RSSIAM_AMP_INT_FUNC_FEXP	(9) - Forward Exponential

Return Value

Returns the status code of this operation.

The meaning of the status code is described in section Error (Status) Codes.

2.2.3.4.3.5 Amplitude Mod Int Freq Setup

C Function Prototype

```
ViStatus rssiam_amIntFreq (
      ViSession instrumentHandle,
      ViReal64 amFrequency);
```

Basic Function Prototype

```
Function rssiam_amIntFreq (
      ByVal instrumentHandle As ViSession,
      ByVal amFrequency As ViReal64) As ViStatus
```

Purpose

This function sets the frequency of the modulating waveform.

Parameters List

1. **ViSession instrumentHandle [in]**

   This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.

   Default Value: None

2. **ViReal64 amFrequency [in]**

   Sets the frequency of the modulating waveform.

   Valid Range: 0.01 Hz to 100.0e3 Hz

   Default Value: 1000.0 Hz

Return Value

Returns the status code of this operation.

The meaning of the status code is described in section Error (Status) Codes.

2.2.3.4.3.6 Amplitude Mod Int Freq Query

C Function Prototype

```
ViStatus rssiam_amIntFreq_Q (
      ViSession instrumentHandle,
      ViInt16 range,
      ViReal64* amFrequency);
```

Basic Function Prototype

```
Function rssiam_amIntFreq_Q (
      ByVal instrumentHandle As ViSession,
      ByVal range As ViInt16,
      amFrequency As ViReal64) As ViStatus
```

Purpose

This function returns the internal modulating frequency.

Parameters List

1. **ViSession instrumentHandle [in]**

   This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.

   Default Value: None

2. **ViInt16 range [in]**

   Specifies whether the current, minimum or maximum setting is to be queried.

   Valid Values:

   - RSSIAM_CURRENT_RANGE (0) - Current
   - RSSIAM_MIN_RANGE (1) - Minimum
   - RSSIAM_MAX_RANGE (2) - Maximum

   Default Value: 0

3. **ViReal64 amFrequency [out]**

   Returns the internal modulating frequency (Hz).

Return Value

Returns the status code of this operation.

The meaning of the status code is described in section Error (Status) Codes.

2.2.3.4.3.7 Amplitude Mod Stat Setup

C Function Prototype

```
ViStatus rssiam_amState (
      ViSession instrumentHandle,
      ViInt16 amState);
```

Basic Function Prototype

```
Function rssiam_amState (
      ByVal instrumentHandle As ViSession,
      ByVal amState As ViInt16) As ViStatus
```

Purpose

This function enables or disables amplitude modulation. Only one modulation mode can be enabled at a time. When AM is enabled, the previous modulation mode is turned off.

Applicable channels: CH1 or CH2

Parameters List

1. **ViSession instrumentHandle [in]**

   This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.

   Default Value: None

2. **ViInt16 amState [in]**

   Enables or disables AM.

   Valid Values:

   - RSSIAM_AMP_STAT_OFF (0) - Off
   - RSSIAM_AMP_STAT_ON (1) - On

   Default Value: 0

Return Value

Returns the status code of this operation.

The meaning of the status code is described in section Error (Status) Codes.

2.2.3.4.3.8 Amplitude Mod Stat Query

C Function Prototype

```
ViStatus rssiam_amState_Q (
      ViSession instrumentHandle,
      ViInt16* amState);
```

Basic Function Prototype

```
Function rssiam_amState_Q (
      ByVal instrumentHandle As ViSession,
      amState As ViInt16) As ViStatus
```

Purpose

This function returns whether AM is enabled or disabled.

Applicable channels: CH1 or CH2

Parameters List

1. **ViSession instrumentHandle [in]**

   This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.

   Default Value: None

2. **ViInt16 amState [out]**

   Returns whether AM is enabled or disabled.

   Valid Values:

   - RSSIAM_AMP_STAT_OFF (0) - Off
   - RSSIAM_AMP_STAT_ON (1) - On

Return Value

Returns the status code of this operation.

The meaning of the status code is described in section Error (Status) Codes.

2.2.3.4.4 FM Modulation

Description

In FM, the frequency of the carrier is varied by the amplitude of the modulating waveform. These functions can be used to setup and query the frequency modulated parameters.

2.2.3.4.4.1 Frequency Mod Dev Setup

C Function Prototype

```
ViStatus rssiam_fmDev (
      ViSession instrumentHandle,
      ViReal64 fmDeviation);
```

Basic Function Prototype

```
Function rssiam_fmDev (
      ByVal instrumentHandle As ViSession,
      ByVal fmDeviation As ViReal64) As ViStatus
```

Purpose

This function sets the peak frequency deviation in hertz. This value represents the variation in frequency of the modulating waveform from the carrier frequency.

The sum of the Carrier Frequency and FM Peak Deviation must be less than the maximum frequency for the selected function. The difference of the Carrier Frequency and FM Peak Deviation must be greater than the minimum frequency for the selected function.

Parameters List

1. **ViSession instrumentHandle [in]**

   This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.

   Default Value: None

2. **ViReal64 fmDeviation [in]**

   Sets the peak frequency deviation.

   Valid Range:

   The sum of the Carrier Frequency and FM Peak Deviation must be less than the maximum frequency for the selected function. The difference of the Carrier Frequency and FM Peak Deviation must be greater than the minimum frequency for the selected function.

   Default Value: 999.9 Hz

Return Value

Returns the status code of this operation.

The meaning of the status code is described in section Error (Status) Codes.

2.2.3.4.4.2 Frequency Mod Dev Query

C Function Prototype

```
ViStatus rssiam_fmDev_Q (
      ViSession instrumentHandle,
      ViInt16 range,
      ViReal64* fmDeviation);
```

Basic Function Prototype

```
Function rssiam_fmDev_Q (
      ByVal instrumentHandle As ViSession,
      ByVal range As ViInt16,
      fmDeviation As ViReal64) As ViStatus
```

Purpose

This function returns the peak frequency deviation in hertz. This value represents the variation in frequency of the modulating waveform from the carrier frequency.

Parameters List

1. **ViSession instrumentHandle [in]**

   This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.

   Default Value: None

2. **ViInt16 range [in]**

   Specifies whether the current, minimum or maximum setting is to be queried.

   Valid Values:

▪ RSSIAM_CURRENT_RANGE	(0) - Current
▪ RSSIAM_MIN_RANGE	(1) - Minimum
▪ RSSIAM_MAX_RANGE	(2) - Maximum

   Default Value: 0

3. **ViReal64 fmDeviation [out]**

   Returns the peak frequency deviation in hertz.

Return Value

Returns the status code of this operation.

The meaning of the status code is described in section Error (Status) Codes.

2.2.3.4.4.3 Frequency Mod Int Func Setup

C Function Prototype

```
ViStatus rssiam_fmIntFunc (
      ViSession instrumentHandle,
      ViInt16 fmIntFunction);
```

Basic Function Prototype

```
Function rssiam_fmIntFunc (
      ByVal instrumentHandle As ViSession,
      ByVal fmIntFunction As ViInt16) As ViStatus
```

Purpose

This function selects the shape of the modulating waveform.

Parameters List

1. **ViSession instrumentHandle [in]**

   This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.

   Default Value: None

2. **ViInt16 fmIntFunction [in]**

   Selects the shape of the modulating waveform.

   Valid Values:

RSSIAM_FREQ_INT_FUNC_SIN	(0) - Sinusoid
RSSIAM_FREQ_INT_FUNC_SQU	(1) - Square
RSSIAM_FREQ_INT_FUNC_TRI	(2) - Triangle
RSSIAM_FREQ_INT_FUNC_URAMP	(3) - Ramp (Up)
RSSIAM_FREQ_INT_FUNC_DRAMP	(4) - Ramp (Down)
RSSIAM_FREQ_INT_FUNC_USER	(5) - User
RSSIAM_FREQ_INT_FUNC_REXP	(8) - Reverse Exponential
RSSIAM_FREQ_INT_FUNC_FEXP	(9) - Forward Exponential

   Default Value: 0

Return Value

Returns the status code of this operation.

The meaning of the status code is described in section Error (Status) Codes.

2.2.3.4.4.4 Frequency Mod Int Func Query

C Function Prototype

```
ViStatus rssiam_fmIntFunc_Q (
      ViSession instrumentHandle,
      ViInt16* fmIntFunction);
```

Basic Function Prototype

```
Function rssiam_fmIntFunc_Q (
      ByVal instrumentHandle As ViSession,
      fmIntFunction As ViInt16) As ViStatus
```

Purpose

This function returns the shape of the modulating waveform.

Parameters List

1. **ViSession instrumentHandle [in]**

   This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.

   Default Value: None

2. **ViInt16 fmIntFunction [out]**

   Returns the shape of the modulating waveform.

   Valid Values:

RSSIAM_FREQ_INT_FUNC_SIN	(0) - Sinusoid
RSSIAM_FREQ_INT_FUNC_SQU	(1) - Square
RSSIAM_FREQ_INT_FUNC_TRI	(2) - Triangle
RSSIAM_FREQ_INT_FUNC_URAMP	(3) - Ramp (Up)
RSSIAM_FREQ_INT_FUNC_DRAMP	(4) - Ramp (Down)
RSSIAM_FREQ_INT_FUNC_USER	(5) - User
RSSIAM_FREQ_INT_FUNC_REXP	(8) - Reverse Exponential
RSSIAM_FREQ_INT_FUNC_FEXP	(9) - Forward Exponential

Return Value

Returns the status code of this operation.

The meaning of the status code is described in section Error (Status) Codes.

2.2.3.4.4.5 Frequency Mod Int Freq Setup

C Function Prototype

```
ViStatus rssiam_fmIntFreq (
      ViSession instrumentHandle,
      ViReal64 fmFrequency);
```

Basic Function Prototype

```
Function rssiam_fmIntFreq (
      ByVal instrumentHandle As ViSession,
      ByVal fmFrequency As ViReal64) As ViStatus
```

Purpose

This function sets the frequency of the modulating waveform.

Parameters List

1. **ViSession instrumentHandle [in]**

   This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.

   Default Value: None

2. **ViReal64 fmFrequency [in]**

   Sets the frequency of the modulating waveform.

   Valid Range: 0.01 Hz to 100.0e3 Hz

   Default Value: 1000.0 Hz

Return Value

Returns the status code of this operation.

The meaning of the status code is described in section Error (Status) Codes.

2.2.3.4.4.6 Frequency Mod Int Freq Query

C Function Prototype

```
ViStatus rssiam_fmIntFreq_Q (
      ViSession instrumentHandle,
      ViInt16 range,
      ViReal64* fmFrequency);
```

Basic Function Prototype

```
Function rssiam_fmIntFreq_Q (
      ByVal instrumentHandle As ViSession,
      ByVal range As ViInt16,
      fmFrequency As ViReal64) As ViStatus
```

Purpose

This function returns the frequency of the modulating waveform.

Parameters List

1. **ViSession instrumentHandle [in]**

   This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.

   Default Value: None

2. **ViInt16 range [in]**

   Specifies whether the current, minimum or maximum setting is to be queried.

   Valid Values:

RSSIAM_CURRENT_RANGE	(0) - Current
RSSIAM_MIN_RANGE	(1) - Minimum
RSSIAM_MAX_RANGE	(2) - Maximum

   Default Value: 0

3. **ViReal64 fmFrequency [out]**

   Returns the frequency of the modulating waveform in hertz.

Return Value

Returns the status code of this operation.

The meaning of the status code is described in section Error (Status) Codes.

2.2.3.4.4.7 Frequency Mod Stat Setup

C Function Prototype

```
ViStatus rssiam_fmStat (
      ViSession instrumentHandle,
      ViInt16 fmState);
```

Basic Function Prototype

```
Function rssiam_fmStat (
      ByVal instrumentHandle As ViSession,
      ByVal fmState As ViInt16) As ViStatus
```

Purpose

This function enables or disables the frequency modulation. Only one modulation mode can be enabled at a time. When FM is enabled, the previous modulation mode is turned off.

Parameters List

1. **ViSession instrumentHandle [in]**

   This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.

   Default Value: None

2. **ViInt16 fmState [in]**

   Enables or disables the frequency modulation.

   Valid Values:

   - RSSIAM_FREQ_STAT_OFF (0) - Off
   - RSSIAM_FREQ_STAT_ON (1) - On

   Default Value: 0

Return Value

Returns the status code of this operation.

The meaning of the status code is described in section Error (Status) Codes.

2.2.3.4.4.8 Frequency Mod Stat Query

C Function Prototype

```
ViStatus rssiam_fmStat_Q (
      ViSession instrumentHandle,
      ViInt16* fmState);
```

Basic Function Prototype

```
Function rssiam_fmStat_Q (
      ByVal instrumentHandle As ViSession,
       fmState As ViInt16) As ViStatus
```

Purpose

This function returns whether the frequency modulation is enabled or disabled.

Parameters List

1. **ViSession instrumentHandle [in]**

   This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.

   Default Value: None

2. **ViInt16 fmState [out]**

   Returns whether the frequency modulation is enabled or disabled.

   Valid Values:

   - RSSIAM_FREQ_STAT_OFF (0) - Off
   - RSSIAM_FREQ_STAT_ON (1) - On

Return Value

Returns the status code of this operation.

The meaning of the status code is described in section Error (Status) Codes.

2.2.3.4.5 PM Modulation

Description

In PM, the phase of the carrier is varied by the amplitude of the modulating waveform. These functions can be used to setup and query the phase modulated parameters.

2.2.3.4.5.1 Phase Mod Dev Setup

C Function Prototype

```
ViStatus rssiam_pmDev (
      ViSession instrumentHandle,
      ViReal64 pmDeviation);
```

Basic Function Prototype

```
Function rssiam_pmDev (
      ByVal instrumentHandle As ViSession,
      ByVal pmDeviation As ViReal64) As ViStatus
```

Purpose

This function sets the modulation deviation of a PM signal. The phase of the carrier is varied by the amplitude of the modulating waveform.

Applicable channels: CH1 or CH2

Parameters List

1. **ViSession instrumentHandle [in]**

   This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.

   Default Value: None

2. **ViReal64 pmDeviation [in]**

   Sets the modulation deviation of a PM signal.

   Valid Range: 0.0 deg to 180.0 deg

   Default Value: 180.0 deg

Return Value

Returns the status code of this operation.

The meaning of the status code is described in section Error (Status) Codes.

2.2.3.4.5.2 Phase Mod Dev Query

C Function Prototype

```
ViStatus rssiam_pmDev_Q (
      ViSession instrumentHandle,
      ViInt16 range,
      ViReal64* pmDeviation);
```

Basic Function Prototype

```
Function rssiam_pmDev_Q (
      ByVal instrumentHandle As ViSession,
      ByVal range As ViInt16,
      pmDeviation As ViReal64) As ViStatus
```

Purpose

This function returns the modulation deviation of a PM signal. The phase of the carrier is varied by the amplitude of the modulating waveform.

Applicable channels: CH1 or CH2

Parameters List

1. **ViSession instrumentHandle [in]**

   This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.

   Default Value: None

2. **ViInt16 range [in]**

   Specifies whether the current, minimum or maximum setting is to be queried.

   Valid Values:

RSSIAM_CURRENT_RANGE	(0) - Current
RSSIAM_MIN_RANGE	(1) - Minimum
RSSIAM_MAX_RANGE	(2) - Maximum

   Default Value: 0

3. **ViReal64 pmDeviation [out]**

   Returns the phase deviation in degree.

Return Value

Returns the status code of this operation.

The meaning of the status code is described in section Error (Status) Codes.

2.2.3.4.5.3 Phase Mod Int Func Setup

C Function Prototype

```
ViStatus rssiam_pmIntFunc (
      ViSession instrumentHandle,
       ViInt16 pmIntFunction);
```

Basic Function Prototype

```
Function rssiam_pmIntFunc (
      ByVal instrumentHandle As ViSession,
      ByVal pmIntFunction As ViInt16) As ViStatus
```

Purpose

This function selects the shape of the modulating waveform.

Applicable channels: CH1 or CH2

Parameters List

1. **ViSession instrumentHandle [in]**

   This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.

   Default Value: None

2. **ViInt16 pmIntFunction [in]**

   Selects the shape of the modulating waveform.

   Valid Values:

RSSIAM_PHASE_INT_FUNC_SIN	(0) - Sinusoid
RSSIAM_PHASE_INT_FUNC_SQU	(1) - Square
RSSIAM_PHASE_INT_FUNC_TRI	(2) - Triangle
RSSIAM_PHASE_INT_FUNC_URAMP	(3) - Ramp (Up)
RSSIAM_PHASE_INT_FUNC_DRAMP	(4) - Ramp (Down)
RSSIAM_PHASE_INT_FUNC_USER	(5) - User
RSSIAM_PHASE_INT_FUNC_REXP	(8) - Reverse Exponential
RSSIAM_PHASE_INT_FUNC_FEXP	(9) - Forward Exponential

   Default Value: 0

Return Value

Returns the status code of this operation.

The meaning of the status code is described in section Error (Status) Codes.

2.2.3.4.5.4 Phase Mod Int Func Query

C Function Prototype

```
ViStatus rssiam_pmIntFunc_Q (
      ViSession instrumentHandle,
      ViInt16* pmIntFunction);
```

Basic Function Prototype

```
Function rssiam_pmIntFunc_Q (
      ByVal instrumentHandle As ViSession,
      pmIntFunction As ViInt16) As ViStatus
```

Purpose

This function returns the shape of the modulating waveform.

Applicable channels: CH1 or CH2

Parameters List

1. **ViSession instrumentHandle [in]**

   This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.

   Default Value: None

2. **ViInt16 pmIntFunction [out]**

   Returns the shape of the modulating waveform.

   Valid Values:

   - RSSIAM_PHASE_INT_FUNC_SIN (0) - Sinusoid
   - RSSIAM_PHASE_INT_FUNC_SQU (1) - Square
   - RSSIAM_PHASE_INT_FUNC_TRI (2) - Triangle
   - RSSIAM_PHASE_INT_FUNC_URAMP (3) - Ramp (Up)
   - RSSIAM_PHASE_INT_FUNC_DRAMP (4) - Ramp (Down)
   - RSSIAM_PHASE_INT_FUNC_USER (5) - User
   - RSSIAM_PHASE_INT_FUNC_REXP (8) - Reverse Exponential
   - RSSIAM_PHASE_INT_FUNC_FEXP (9) - Forward Exponential

Return Value

Returns the status code of this operation.

The meaning of the status code is described in section Error (Status) Codes.

2.2.3.4.5.5 Phase Mod Int Freq Setup

C Function Prototype

```
ViStatus rssiam_pmIntFreq (
      ViSession instrumentHandle,
      ViReal64 pmFrequency);
```

Basic Function Prototype

```
Function rssiam_pmIntFreq (
      ByVal instrumentHandle As ViSession,
      ByVal pmFrequency As ViReal64) As ViStatus
```

Purpose

This function sets the frequency of the modulating waveform.

Applicable channels: CH1 or CH2

Parameters List

1. **ViSession instrumentHandle [in]**

   This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.

   Default Value: None

2. **ViReal64 pmFrequency [in]**

   Sets the frequency of the modulating waveform.

   Valid Range: 0.01 Hz to 100.0e3 Hz

   Default Value: 1000.0 Hz

Return Value

Returns the status code of this operation.

The meaning of the status code is described in section Error (Status) Codes.

2.2.3.4.5.6 Phase Mod Int Freq Query

C Function Prototype

```
ViStatus rssiam_pmIntFreq_Q (
    ViSession instrumentHandle,
    ViInt16 range,
    ViReal64* pmFrequency);
```

Basic Function Prototype

```
Function rssiam_pmIntFreq_Q (
    ByVal instrumentHandle As ViSession,
    ByVal range As ViInt16,
    pmFrequency As ViReal64) As ViStatus
```

Purpose

This function returns the frequency of the modulating waveform.

Applicable channels: CH1 or CH2

Parameters List

1. **ViSession instrumentHandle [in]**

   This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.

   Default Value: None

2. **ViInt16 range [in]**

   Specifies whether the current, minimum or maximum setting is to be queried.

   Valid Values:

▪ RSSIAM_CURRENT_RANGE	(0) - Current
▪ RSSIAM_MIN_RANGE	(1) - Minimum
▪ RSSIAM_MAX_RANGE	(2) - Maximum

   Default Value: 0

3. **ViReal64 pmFrequency [out]**

   Returns the frequency of the modulating waveform in hertz.

Return Value

Returns the status code of this operation.

The meaning of the status code is described in section Error (Status) Codes.

2.2.3.4.5.7 Phase Mod Stat Setup

C Function Prototype

```
ViStatus rssiam_pmStat (
    ViSession instrumentHandle,
    ViInt16 pmState);
```

Basic Function Prototype

```
Function rssiam_pmStat (
    ByVal instrumentHandle As ViSession,
    ByVal pmState As ViInt16) As ViStatus
```

Purpose

This function enables or disables the phase modulation. Only one modulation mode can be enabled at a time. When PM is enabled, the previous modulation mode is turned off.

Applicable channels: CH1 or CH2

Parameters List

1. **ViSession instrumentHandle [in]**

   This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.

   Default Value: None

2. **ViInt16 pmState [in]**

   Enables or disables the phase modulation.

   Valid Values:

   - RSSIAM_PHASE_STAT_OFF (0) - Off
   - RSSIAM_PHASE_STAT_ON (1) - On

   Default Value: 0

Return Value

Returns the status code of this operation.

The meaning of the status code is described in section Error (Status) Codes.

2.2.3.4.5.8 Phase Mod Stat Query

C Function Prototype

```
ViStatus rssiam_pmStat_Q (
      ViSession instrumentHandle,
      ViInt16* pmState);
```

Basic Function Prototype

```
Function rssiam_pmStat_Q (
      ByVal instrumentHandle As ViSession,
      pmState As ViInt16) As ViStatus
```

Purpose

This function returns whether the phase modulation is enabled or disabled.

Applicable channels: CH1 or CH2

Parameters List

1. **ViSession instrumentHandle [in]**

   This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.

   Default Value: None

2. **ViInt16 pmState [out]**

   Returns whether the phase modulation is enabled or disabled.

   Valid Values:

   - RSSIAM_PHASE_STAT_OFF (0) - Off
   - RSSIAM_PHASE_STAT_ON (1) - On

Return Value

Returns the status code of this operation.

The meaning of the status code is described in section Error (Status) Codes.

2.2.3.4.6 FSK Modulation

Description

It is possible to configure the function generator to "shift" its output frequency between two preset values using the FSK modulation. The rate at which the output shifts between the two frequencies (called the "carrier frequency" and the "hop frequency") is determined by the internal rate generator or the signal level on the rear-panel FSK terminal.

2.2.3.4.6.1 FSK Frequency Setup

C Function Prototype

```
ViStatus rssiam_fskFreq (
      ViSession instrumentHandle,
      ViReal64 fskFrequency);
```

Basic Function Prototype

```
Function rssiam_fskFreq (
      ByVal instrumentHandle As ViSession,
      ByVal fskFrequency As ViReal64) As ViStatus
```

Purpose

This function sets the FSK "hop" frequency.

Parameters List

1. **ViSession instrumentHandle [in]**

   This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.

   Default Value: None

2. **ViReal64 fskFrequency [in]**

   Sets the FSK "hop" frequency.

   Range of FSK frequency depends on function currently selected:

SINUSOID	10.0e-6 Hz to 35.0e6 Hz
TRIANGLE	10.0e-6 Hz to 1.0e6 Hz
RAMP	10.0e-6 Hz to 1.0e6 Hz
SQUARE (var duty cycle)	10.0e-6 Hz to 1.0e6 Hz
REXP	10.0e-6 Hz to 1.0e6 Hz
FEXP	10.0e-6 Hz to 1.0e6 Hz
SQUARE	10.0e-6 Hz to 50.0e6 Hz
PULSE	10.0e-6 Hz to 50.0e6 Hz
USER	10.0e-6 Hz to 6.25e6 Hz

   Default Value: 10.0e3 Hz

Return Value

Returns the status code of this operation.

The meaning of the status code is described in section Error (Status) Codes.

2.2.3.4.6.2 FSK Frequency Query

C Function Prototype

```
ViStatus rssiam_fskFreq_Q (
      ViSession instrumentHandle,
      ViInt16 range,
      ViReal64* fskFrequency);
```

Basic Function Prototype

```
Function rssiam_fskFreq_Q (
      ByVal instrumentHandle As ViSession,
      ByVal range As ViInt16,
      fskFrequency As ViReal64) As ViStatus
```

Purpose

This function returns the FSK "hop" frequency.

Parameters List

1. **ViSession instrumentHandle [in]**

   This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.

   Default Value: None

2. **ViInt16 range [in]**

   Specifies whether the current, minimum or maximum setting is to be queried.

   Valid Values:

▪ RSSIAM_CURRENT_RANGE	(0) - Current
▪ RSSIAM_MIN_RANGE	(1) - Minimum
▪ RSSIAM_MAX_RANGE	(2) - Maximum

   Default Value: 0

3. **ViReal64 fskFrequency [out]**

   Returns the FSK "hop" frequency in hertz.

Return Value

Returns the status code of this operation.

The meaning of the status code is described in section Error (Status) Codes.

2.2.3.4.6.3 FSK Internal Rate Setup

C Function Prototype

```
ViStatus rssiam_fskIntRate (
      ViSession instrumentHandle,
      ViReal64 fskIntRate);
```

Basic Function Prototype

```
Function rssiam_fskIntRate (
      ByVal instrumentHandle As ViSession,
      ByVal fskIntRate As ViReal64) As ViStatus
```

Purpose

This function sets the rate at which the output frequency "shifts" between the carrier and hop frequency when the internal FSK source is selected.

Parameters List

1. **ViSession instrumentHandle [in]**

   This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.

   Default Value: None

2. **ViReal64 fskIntRate [in]**

   Sets the rate at which the output frequency "shifts".

   Valid Range: 0.0001 Hz to 2.0e6 Hz

   Default Value: 1000.0 Hz

Return Value

Returns the status code of this operation.

The meaning of the status code is described in section Error (Status) Codes.

2.2.3.4.6.4 FSK Internal Rate Query

C Function Prototype

```
ViStatus rssiam_fskIntRate_Q (
      ViSession instrumentHandle,
      ViInt16 range,
      ViReal64* fskIntRate);
```

Basic Function Prototype

```
Function rssiam_fskIntRate_Q (
      ByVal instrumentHandle As ViSession,
      ByVal range As ViInt16,
      fskIntRate As ViReal64) As ViStatus
```

Purpose

This function returns the FSK rate.

Parameters List

1. **ViSession instrumentHandle [in]**

   This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.

   Default Value: None

2. **ViInt16 range [in]**

   Specifies whether the current, minimum or maximum setting is to be queried.

   Valid Values:

▪ RSSIAM_CURRENT_RANGE	(0) - Current
▪ RSSIAM_MIN_RANGE	(1) - Minimum
▪ RSSIAM_MAX_RANGE	(2) - Maximum

   Default Value: 0

3. **ViReal64 fskIntRate [out]**

   Returns the FSK rate in hertz.

Return Value

Returns the status code of this operation.

The meaning of the status code is described in section Error (Status) Codes.

2.2.3.4.6.5 FSK Source Setup

C Function Prototype

```
ViStatus rssiam_fskSour (
      ViSession instrumentHandle,
      ViInt16 fskSource);
```

Basic Function Prototype

```
Function rssiam_fskSour (
      ByVal instrumentHandle As ViSession,
      ByVal fskSource As ViInt16) As ViStatus
```

Purpose

This function selects the internal or external FSK source.

When the internal source is selected, the rate at which the output frequency "shifts" between the carrier frequency and hop frequency is determined by the FSK rate specified.

When the external source is selected, the signal level on the rear-panel FSK terminal determines the output frequency. When a "low" TTL level is present, the carrier frequency is output. When a "high" TTL level is present, the hop frequency is output.

Parameters List

1. **ViSession instrumentHandle [in]**

   This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.

   Default Value: None

2. **ViInt16 fskSource [in]**

   Selects the internal or external FSK source.

   Valid Values:

   - RSSIAM_FSK_SOUR_INT (0) - Internal
   - RSSIAM_FSK_SOUR_EXT (1) - External

   Default Value: 0

Return Value

Returns the status code of this operation.

The meaning of the status code is described in section Error (Status) Codes.

2.2.3.4.6.6 FSK Source Query

C Function Prototype

```
ViStatus rssiam_fskSour_Q (
      ViSession instrumentHandle,
      ViInt16* fskSource);
```

Basic Function Prototype

```
Function rssiam_fskSour_Q (
      ByVal instrumentHandle As ViSession,
      fskSource As ViInt16) As ViStatus
```

Purpose

This function returns the internal or external FSK source.

Parameters List

1. **ViSession instrumentHandle [in]**

   This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.

   Default Value: None

2. **ViInt16 fskSource [out]**

   Returns the internal or external FSK source.

   Valid Values:

   - RSSIAM_FSK_SOUR_INT (0) - Internal
   - RSSIAM_FSK_SOUR_EXT (1) - External

Return Value

Returns the status code of this operation.

The meaning of the status code is described in section Error (Status) Codes.

2.2.3.4.6.7 FSK Stat Setup

C Function Prototype

```
ViStatus rssiam_fskStat (
      ViSession instrumentHandle,
      ViInt16 fskState);
```

Basic Function Prototype

```
Function rssiam_fskStat (
      ByVal instrumentHandle As ViSession,
      ByVal fskState As ViInt16) As ViStatus
```

Purpose

This function enables or disables the FSK modulation. Only one modulation mode can be enabled at a time. When the FSK is enabled, the previous modulation mode is turned off.

Parameters List

1. **ViSession instrumentHandle [in]**

   This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.

   Default Value: None

2. **ViInt16 fskState [in]**

   Enables or disables the FSK modulation.

   Valid Values:

   - RSSIAM_FSK_STAT_OFF (0) - Off
   - RSSIAM_FSK_STAT_ON (1) - On

   Default Value: 0

Return Value

Returns the status code of this operation.

The meaning of the status code is described in section Error (Status) Codes.

2.2.3.4.6.8 FSK Stat Query

C Function Prototype

```
ViStatus rssiam_fskStat_Q (
      ViSession instrumentHandle,
      ViInt16* fskState);
```

Basic Function Prototype

```
Function rssiam_fskStat_Q (
      ByVal instrumentHandle As ViSession,
      fskState As ViInt16) As ViStatus
```

Purpose

This function returns whether the FSK modulation is enabled or disabled.

Parameters List

1. **ViSession instrumentHandle [in]**

   This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.

   Default Value: None

2. **ViInt16 fskState [out]**

   Returns whether the FSK modulation is enabled or disabled.

   Valid Values:

   - RSSIAM_FSK_STAT_OFF (0) - Off
   - RSSIAM_FSK_STAT_ON (1) - On

Return Value

Returns the status code of this operation.

The meaning of the status code is described in section Error (Status) Codes.

2.2.3.4.7 PSK Modulation

Description

It is possible to configure the function generator to "shift" its output phase using the PSK modulation. The rate at which the output shifts the phase is determined by the internal rate generator or the signal level on the rear-panel PSK terminal.

2.2.3.4.7.1 PSK Phase Setup

C Function Prototype

```
ViStatus rssiam_pskPhase (
      ViSession instrumentHandle,
      ViReal64 pskPhase);
```

Basic Function Prototype

```
Function rssiam_pskPhase (
      ByVal instrumentHandle As ViSession,
      ByVal pskPhase As ViReal64) As ViStatus
```

Purpose

This function sets the PSK "hop" phase.

Applicable channels: CH1 or CH2

Parameters List

1. **ViSession instrumentHandle [in]**

   This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.

   Default Value: None

2. **ViReal64 pskPhase [in]**

   Sets the PSK "hop" phase.

   Valid Range: 0.0 deg to 360.0 deg

   Default Value: 180.0 deg

Return Value

Returns the status code of this operation.

The meaning of the status code is described in section Error (Status) Codes.

2.2.3.4.7.2 PSK Phase Query

C Function Prototype

```
ViStatus rssiam_pskPhase_Q (
    ViSession instrumentHandle,
    ViInt16 range,
    ViReal64* pskPhase);
```

Basic Function Prototype

```
Function rssiam_pskPhase_Q (
    ByVal instrumentHandle As ViSession,
    ByVal range As ViInt16,
    pskPhase As ViReal64) As ViStatus
```

Purpose

This function returns the PSK "hop" phase.

Applicable channels: CH1 or CH2

Parameters List

1. **ViSession instrumentHandle [in]**

   This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.

   Default Value: None

2. **ViInt16 range [in]**

   Specifies whether the current, minimum or maximum setting is to be queried.

   Valid Values:

▪ RSSIAM_CURRENT_RANGE	(0) - Current
▪ RSSIAM_MIN_RANGE	(1) - Minimum
▪ RSSIAM_MAX_RANGE	(2) - Maximum

   Default Value: 0

3. **ViReal64 pskPhase [out]**

   Returns the PSK "hop" phase in degree.

Return Value

Returns the status code of this operation.

The meaning of the status code is described in section Error (Status) Codes.

2.2.3.4.7.3 PSK Internal Rate Setup

C Function Prototype

```
ViStatus rssiam_pskIntRate (
      ViSession instrumentHandle,
      ViReal64 pskIntRate);
```

Basic Function Prototype

```
Function rssiam_pskIntRate (
      ByVal instrumentHandle As ViSession,
      ByVal pskIntRate As ViReal64) As ViStatus
```

Purpose

This function sets the rate at which the output phase "shifts" at the carrier frequency and when the internal FSK source is selected.

Parameters List

1. **ViSession instrumentHandle [in]**

   This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.

   Default Value: None

2. **ViReal64 pskIntRate [in]**

   Sets the rate at which the output phase "shifts" at the carrier frequency.

   Valid Range: 0.0001 Hz to 2.0e6 Hz

   Default Value: 1000.0 Hz

Return Value

Returns the status code of this operation.

The meaning of the status code is described in section Error (Status) Codes.

2.2.3.4.7.4 PSK Internal Rate Query

C Function Prototype

```
ViStatus rssiam_pskIntRate_Q (
      ViSession instrumentHandle,
      ViInt16 range,
      ViReal64* pskIntRate);
```

Basic Function Prototype

```
Function rssiam_pskIntRate_Q (
      ByVal instrumentHandle As ViSession,
      ByVal range As ViInt16,
      pskIntRate As ViReal64) As ViStatus
```

Purpose

This function returns the rate at which the output phase "shifts" at the carrier frequency.

Parameters List

1. **ViSession instrumentHandle [in]**

   This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.

   Default Value: None

2. **ViInt16 range [in]**

   Specifies whether the current, minimum or maximum setting is to be queried.

   Valid Values:

▪ RSSIAM_CURRENT_RANGE	(0) - Current
▪ RSSIAM_MIN_RANGE	(1) - Minimum
▪ RSSIAM_MAX_RANGE	(2) - Maximum

   Default Value: 0

3. **ViReal64 pskIntRate [out]**

   Returns the rate at which the output phase "shifts" at the carrier frequency (deg).

Return Value

Returns the status code of this operation.

The meaning of the status code is described in section Error (Status) Codes.

2.2.3.4.7.5 PSK Source Setup

C Function Prototype

```
ViStatus rssiam_pskSour (
    ViSession instrumentHandle,
    ViInt16 pskSource);
```

Basic Function Prototype

```
Function rssiam_pskSour (
    ByVal instrumentHandle As ViSession,
    ByVal pskSource As ViInt16) As ViStatus
```

Purpose

This function selects the internal or external PSK source.

When the internal source is selected, the rate at which the phase of output frequency "shifts" is determined by the PSK rate specified.

When the external source is selected, the signal level on the rear-panel PSK terminal determines phase of the output frequency.

Parameters List

1. **ViSession instrumentHandle [in]**

   This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.

   Default Value: None

2. **ViInt16 pskSource [in]**

   Selects the internal or external PSK source.

   Valid Values:

   - RSSIAM_PSK_SOUR_INT (0) - Internal
   - RSSIAM_PSK_SOUR_EXT (1) - External

   Default Value: 0

Return Value

Returns the status code of this operation.

The meaning of the status code is described in section Error (Status) Codes.

2.2.3.4.7.6 PSK Source Query

C Function Prototype

```
ViStatus rssiam_pskSour_Q (
      ViSession instrumentHandle,
      ViInt16* pskSource);
```

Basic Function Prototype

```
Function rssiam_pskSour_Q (
      ByVal instrumentHandle As ViSession,
      pskSource As ViInt16) As ViStatus
```

Purpose

This function returns the internal or external PSK source.

Parameters List

1. **ViSession instrumentHandle [in]**

   This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.

   Default Value: None

2. **ViInt16 pskSource [out]**

   Returns the internal or external PSK source.

   Valid Values:

   - RSSIAM_PSK_SOUR_INT (0) - Internal
   - RSSIAM_PSK_SOUR_EXT (1) - External

Return Value

Returns the status code of this operation.

The meaning of the status code is described in section Error (Status) Codes.

2.2.3.4.7.7 PSK Stat Setup

C Function Prototype

```
ViStatus rssiam_pskStat (
      ViSession instrumentHandle,
      ViInt16 pskState);
```

Basic Function Prototype

```
Function rssiam_pskStat (
      ByVal instrumentHandle As ViSession,
      ByVal pskState As ViInt16) As ViStatus
```

Purpose

This function enables or disables the PSK modulation. Only one modulation mode can be enabled at a time. When PSK is enabled, the previous modulation mode is turned off.

Parameters List

1. **ViSession instrumentHandle [in]**

   This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.

   Default Value: None

2. **ViInt16 pskState [in]**

   Enables or disables the PSK modulation.

   Valid Values:

   - RSSIAM_PSK_STAT_OFF (0) - Off
   - RSSIAM_PSK_STAT_ON (1) - On

   Default Value: 0

Return Value

Returns the status code of this operation.

The meaning of the status code is described in section Error (Status) Codes.

2.2.3.4.7.8 PSK Stat Query

C Function Prototype

ViStatus rssiam_pskStat_Q (
ViSession instrumentHandle,
ViInt16* pskState);

Basic Function Prototype

Function rssiam_pskStat_Q (
ByVal instrumentHandle As ViSession,
pskState As ViInt16) As ViStatus

Purpose

This function returns whether the PSK modulation is enabled or disabled.

Parameters List

1. **ViSession instrumentHandle [in]**

   This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.

   Default Value: None

2. **ViInt16 pskState [out]**

   Returns whether the PSK modulation is enabled or disabled.

   Valid Values:

   - RSSIAM_PSK_STAT_OFF (0) - Off
   - RSSIAM_PSK_STAT_ON (1) - On

Return Value

Returns the status code of this operation.

The meaning of the status code is described in section Error (Status) Codes.

2.2.3.5 Arbitrary Waveform Settings

Description Arbitrary waveform settings.

2.2.3.5.1 Data Arb Points Setup

C Function Prototype

```
ViStatus rssiam_dataArbPoin (
      ViSession instrumentHandle,
      ViInt32 dataPoints);
```

Basic Function Prototype

```
Function rssiam_dataArbPoin (
      ByVal instrumentHandle As ViSession,
      ByVal dataPoints As ViInt32) As ViStatus
```

Purpose This function sets number of points to be used from the arbitrary waveform memory when Data Arb Rate Mode is set to Sample Accurate (fixed settings).

Note Output Function Shape should be set to User.

Parameters List

1. **ViSession instrumentHandle [in]**

   This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.

   Default Value: None

2. **ViInt32 dataPoints [in]**

   Sets number of points to be used from the arbitrary waveform memory.

   Valid Range: 16 to 262144

   Default Value: 16

Return Value Returns the status code of this operation.

The meaning of the status code is described in section Error (Status) Codes.

2.2.3.5.2 Data Arb Points Query

C Function Prototype

```
ViStatus rssiam_dataArbPoin_Q (
    ViSession instrumentHandle,
    ViInt32* dataPoints);
```

Basic Function Prototype

```
Function rssiam_dataArbPoin_Q (
    ByVal instrumentHandle As ViSession,
    dataPoints As ViInt32) As ViStatus
```

Purpose

This function returns number of points used from the arbitrary waveform memory when Data Arb Rate Mode is set to Sample Accurate (fixed settings).

Parameters List

1. **ViSession instrumentHandle [in]**

   This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.

   Default Value: None

2. **ViInt32 dataPoints [out]**

   Returns number of points used from the arbitrary waveform memory.

Return Value

Returns the status code of this operation.

The meaning of the status code is described in section Error (Status) Codes.

2.2.3.5.3 Data Arb Start Setup

C Function Prototype

```
ViStatus rssiam_dataArbStart (
    ViSession instrumentHandle,
    ViInt32 dataStart);
```

Basic Function Prototype

```
Function rssiam_dataArbStart (
    ByVal instrumentHandle As ViSession,
    ByVal dataStart As ViInt32) As ViStatus
```

Purpose

This function sets the start point in waveform memory when Data Arb Rate Mode is set to Sample Accurate (fixed settings).

Applicable channels: CH1 or CH2

Note

Output Function Shape should be set to User.

Parameters List

1. **ViSession instrumentHandle [in]**

   This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.

   Default Value: None

2. **ViInt32 dataStart [in]**

   Sets the start point in waveform memory.

   Valid Range: 0 to 262143

   Default Value: 0

Return Value

Returns the status code of this operation.

The meaning of the status code is described in section Error (Status) Codes.

2.2.3.5.4 Data Arb Start Query

C Function Prototype

```
ViStatus rssiam_dataArbStart_Q (
      ViSession instrumentHandle,
      ViInt32* dataStart);
```

Basic Function Prototype

```
Function rssiam_dataArbStart_Q (
      ByVal instrumentHandle As ViSession,
      dataStart As ViInt32) As ViStatus
```

Purpose

This function returns the start point in waveform memory when Data Arb Rate Mode is set to Sample Accurate (fixed settings).

Applicable channels: CH1 or CH2

Parameters List

1. **ViSession instrumentHandle [in]**

   This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.

   Default Value: None

2. **ViInt32 dataStart [out]**

   Returns the start point in waveform memory.

Return Value

Returns the status code of this operation.

The meaning of the status code is described in section Error (Status) Codes.

2.2.3.5.5 Data Arb Stop Setup

C Function Prototype

```
ViStatus rssiam_dataArbStop (
      ViSession instrumentHandle,
      ViInt32 dataStop);
```

Basic Function Prototype

```
Function rssiam_dataArbStop (
      ByVal instrumentHandle As ViSession,
      ByVal dataStop As ViInt32) As ViStatus
```

Purpose

This function sets the stop point in waveform memory when Data Arb Rate Mode is set to Sample Accurate (fixed settings).

Applicable channels: CH1 or CH2

Note

Output Function Shape should be set to User.

Parameters List

1. **ViSession instrumentHandle [in]**

   This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.

   Default Value: None

2. **ViInt32 dataStop [in]**

   Sets the stop point in waveform memory.

   Valid Range: 0 to 262143

   Default Value: 262143

Return Value

Returns the status code of this operation.

The meaning of the status code is described in section Error (Status) Codes.

2.2.3.5.6 Data Arb Stop Query

C Function Prototype

```
ViStatus rssiam_dataArbStop_Q (
     ViSession instrumentHandle,
     ViInt32* dataStop);
```

Basic Function Prototype

```
Function rssiam_dataArbStop_Q (
     ByVal instrumentHandle As ViSession,
     dataStop As ViInt32) As ViStatus
```

Purpose

This function returns the stop point in waveform memory when Data Arb Rate Mode is set to Sample Accurate (fixed settings).

Applicable channels: CH1 or CH2

Parameters List

1. **ViSession instrumentHandle [in]**

   This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.

   Default Value: None

2. **ViInt32 dataStop [out]**

   Returns the stop point in waveform memory.

Return Value

Returns the status code of this operation.

The meaning of the status code is described in section Error (Status) Codes.

2.2.3.5.7 Data Arb Rate Setup

C Function Prototype

```
ViStatus rssiam_dataArbRate (
     ViSession instrumentHandle,
     ViReal64 dataSampleRate);
```

Basic Function Prototype

```
Function rssiam_dataArbRate (
     ByVal instrumentHandle As ViSession,
     ByVal dataSampleRate As ViReal64) As ViStatus
```

Purpose

This function sets the sample rate for the arbitrary waveform when Data Arb Rate Mode is set to Sample Accurate (fixed settings).

Note

Output Function Shape should be set to User.

Parameters List

1. **ViSession instrumentHandle [in]**

   This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.

   Default Value: None

2. **ViReal64 dataSampleRate [in]**

   Sets the sample rate for the arbitrary waveform.

   Valid Range: 1.0e-5 Hz to 100.0e6 Hz

   Default Value: 100.0e6 Hz

Return Value

Returns the status code of this operation.

The meaning of the status code is described in section Error (Status) Codes.

2.2.3.5.8 Data Arb Rate Query

C Function Prototype

```
ViStatus rssiam_dataArbRate_Q (
      ViSession instrumentHandle,
      ViReal64* dataSampleRate);
```

Basic Function Prototype

```
Function rssiam_dataArbRate_Q (
      ByVal instrumentHandle As ViSession,
      dataSampleRate As ViReal64) As ViStatus
```

Purpose

This function returns the sample rate of the arbitrary waveform when Data Arb Rate Mode is set to Sample Accurate (fixed settings).

Parameters List

1. **ViSession instrumentHandle [in]**

   This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.

   Default Value: None

2. **ViReal64 dataSampleRate [out]**

   Returns the sample rate of the arbitrary waveform in hertz.

Return Value

Returns the status code of this operation.

The meaning of the status code is described in section Error (Status) Codes.

2.2.3.5.9 Data Arb Rate Mode Setup

C Function Prototype

```
ViStatus rssiam_dataArbRateMode (
      ViSession instrumentHandle,
      ViInt16 dataSampleRateMode);
```

Basic Function Prototype

```
Function rssiam_dataArbRateMode (
      ByVal instrumentHandle As ViSession,
      ByVal dataSampleRateMode As ViInt16) As ViStatus
```

Purpose

This function sets the arbitrary waveform sample rate mode to Normal or customized Sample Accurate setting.

Note

Output Function Shape should be set to User.

Parameters List

1. **ViSession instrumentHandle [in]**

   This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.

   Default Value: None

2. **ViInt16 dataSampleRateMode [in]**

   Sets the arbitrary waveform sample rate mode to Normal or customized Sample Accurate setting.

   Valid Values:

   - RSSIAM_ARB_RATE_MODE_NORMAL (0) - Normal
   - RSSIAM_ARB_RATE_MODE_ACCURATE (1) - Sample Accurate

   Default Value: 0

Return Value

Returns the status code of this operation.

The meaning of the status code is described in section Error (Status) Codes.

2.2.3.5.10 Data Arb Rate Mode Query

C Function Prototype

```
ViStatus rssiam_dataArbRateMode_Q (
      ViSession instrumentHandle,
      ViInt16* dataSampleRateMode);
```

Basic Function Prototype

```
Function rssiam_dataArbRateMode_Q (
      ByVal instrumentHandle As ViSession,
      dataSampleRateMode As ViInt16) As ViStatus
```

Purpose

This function returns the arbitrary waveform sample rate mode.

Parameters List

1. **ViSession instrumentHandle [in]**

   This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.

   Default Value: None

2. **ViInt16 dataSampleRateMode [out]**

   Returns the arbitrary waveform sample rate mode.

   Valid Values:

   - RSSIAM_ARB_RATE_MODE_NORMAL (0) - Normal
   - RSSIAM_ARB_RATE_MODE_ACCURATE (1) - Sample Accurate

Return Value

Returns the status code of this operation.

The meaning of the status code is described in section Error (Status) Codes.

2.2.3.5.11 Data Arb Volatile Setup

C Function Prototype

```
ViStatus rssiam_dataVolatile (
      ViSession instrumentHandle,
      ViInt32 noofPoints,
      ViReal64[] dataArray);
```

Basic Function Prototype

```
Function rssiam_dataVolatile (
      ByVal instrumentHandle As ViSession,
      ByVal noofPoints As ViInt32,
      dataArray As ViReal64) As ViStatus
```

Purpose

This function uploads floating-point values between -1 and +1 into volatile memory. Out of range values are coerced to the lower (upper) range limit.

The values -1 and +1 correspond to the peak values of the waveform.

Downloading the floating-point values is slower than downloading binary values but is more convenient when using the trigonometric functions, which return values between -1 and +1.

This function overwrites the previous waveform in VOLATILE memory.

Applicable channels: CH1 or CH2

Note

Output Function Shape should be set to User.

Parameters List

1. **ViSession instrumentHandle [in]**

   This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.

   Default Value: None

2. **ViInt32 noofPoints [in]**

   Number of floating-point values to be uploaded.

   Valid Range: 16, 32, ... 262144 (in steps of power of two)

   Default Value: 16

3. **ViReal64[] dataArray [in]**

   Floating-point values between -1 and +1 to upload into volatile memory.

Return Value

Returns the status code of this operation.

The meaning of the status code is described in section Error (Status) Codes.

2.2.3.5.12 Data Arb Dac Volatile Setup

C Function Prototype

```
ViStatus rssiam_dataDacVolatile (
      ViSession instrumentHandle,
      ViInt32 noofPoints,
      ViInt16[] dataArray);
```

Basic Function Prototype

```
Function rssiam_dataDacVolatile (
      ByVal instrumentHandle As ViSession,
      ByVal noofPoints As ViInt32,
      dataArray As ViInt16) As ViStatus
```

Purpose

This function uploads binary integer values between -16383 and 16383 into volatile memory. The binary range of values corresponds to the values available using internal 14-bit DAC codes.

The values -16383 and 16383 correspond to the peak values of the waveform.

MSB (Most Significant Bit at position 15) represents marker state, where On (1) and Off (0).

This function overwrites the previous waveform in VOLATILE memory.

Applicable channels: CH1 or CH2

Note

Output Function Shape should be set to User.

Parameters List

1. **ViSession instrumentHandle [in]**

   This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.

   Default Value: None

2. **ViInt32 noofPoints [in]**

   Number of binary integer values to upload into volatile memory.

   Valid Range: 16, 32, ... 262144 (in steps of power of two)

   Default Value: 16

3. **ViInt16[] dataArray [in]**

   Binary integer values between -16383 and 16383 to upload into volatile memory.

Return Value

Returns the status code of this operation.

The meaning of the status code is described in section Error (Status) Codes.

2.2.3.6 Sweep Settings

Description

In the frequency sweep mode, the function generator "steps" from the start frequency to the stop frequency at a sweep rate which you specify. It is possible to sweep up or down in frequency, and with either linear or logarithmic spacing. It is also possible to configure the function generator to output a single sweep by applying an external trigger.

2.2.3.6.1 Sweep Freq Start Setup

C Function Prototype

```
ViStatus rssiam_freqStar (
      ViSession instrumentHandle,
      ViReal64 frequencyStart);
```

Basic Function Prototype

```
Function rssiam_freqStar (
      ByVal instrumentHandle As ViSession,
      ByVal frequencyStart As ViReal64) As ViStatus
```

Purpose

This function sets the sweep start frequency.

Parameters List

1. **ViSession instrumentHandle [in]**

   This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.

   Default Value: None

2. **ViReal64 frequencyStart [in]**

   Sets the sweep start frequency.

   Range of this frequency depends on function currently selected:

SINUSOID	0.1 Hz to 35.0e6 Hz
TRIANGLE	0.1 Hz to 250.0e3 Hz
RAMP	0.1 Hz to 250.0e3 Hz
SQUARE	0.1 Hz to 250.0e3 Hz
REXP	0.1 Hz to 250.0e3 Hz
FEXP	0.1 Hz to 250.0e3 Hz
SQUARE	0.1 Hz to 50.0e6 Hz
PULSE	0.1 Hz to 50.0e6 Hz
USER	0.1 Hz to 6.25e6 Hz

   Default Value: 1.0 Hz

Return Value

Returns the status code of this operation.

The meaning of the status code is described in section Error (Status) Codes.

2.2.3.6.2 Sweep Freq Start Query

C Function Prototype

```
ViStatus rssiam_freqStar_Q (
      ViSession instrumentHandle,
      ViInt16 range,
      ViReal64* frequencyStart);
```

Basic Function Prototype

```
Function rssiam_freqStar_Q (
      ByVal instrumentHandle As ViSession,
      ByVal range As ViInt16,
      frequencyStart As ViReal64) As ViStatus
```

Purpose

This function returns the sweep start frequency.

Parameters List

1. **ViSession instrumentHandle [in]**

   This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.

   Default Value: None

2. **ViInt16 range [in]**

   Specifies whether the current, minimum or maximum setting is to be queried.

   Valid Values:

RSSIAM_CURRENT_RANGE	(0) - Current
RSSIAM_MIN_RANGE	(1) - Minimum
RSSIAM_MAX_RANGE	(2) - Maximum

   Default Value: 0

3. **ViReal64 frequencyStart [out]**

   Returns the sweep start frequency in hertz.

Return Value

Returns the status code of this operation.

The meaning of the status code is described in section Error (Status) Codes.

2.2.3.6.3 Sweep Freq Stop Setup

C Function Prototype

```
ViStatus rssiam_freqStop (
      ViSession instrumentHandle,
      ViReal64 frequencyStop);
```

Basic Function Prototype

```
Function rssiam_freqStop (
      ByVal instrumentHandle As ViSession,
      ByVal frequencyStop As ViReal64) As ViStatus
```

Purpose

This function sets the sweep stop frequency.

Parameters List

1. **ViSession instrumentHandle [in]**

   This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.

   Default Value: None

2. **ViReal64 frequencyStop [in]**

   Sets the sweep stop frequency.

   Range of this frequency depends on function currently selected:

SINUSOID	0.1 Hz to 35.0e6 Hz
TRIANGLE	0.1 Hz to 250.0e3 Hz
RAMP	0.1 Hz to 250.0e3 Hz
SQUARE	0.1 Hz to 250.0e3 Hz
REXP	0.1 Hz to 250.0e3 Hz
FEXP	0.1 Hz to 250.0e3 Hz
SQUARE	0.1 Hz to 50.0e6 Hz
PULSE	0.1 Hz to 50.0e6 Hz
USER	0.1 Hz to 6.25e6 Hz

   Default Value: 250.0e3 Hz

Return Value

Returns the status code of this operation.

The meaning of the status code is described in section Error (Status) Codes.

2.2.3.6.4 Sweep Freq Stop Query

C Function Prototype

```
ViStatus rssiam_freqStop_Q (
      ViSession instrumentHandle,
      ViInt16 range,
      ViReal64* frequencyStop);
```

Basic Function Prototype

```
Function rssiam_freqStop_Q (
      ByVal instrumentHandle As ViSession,
      ByVal range As ViInt16,
      frequencyStop As ViReal64) As ViStatus
```

Purpose

This function returns the sweep stop frequency.

Parameters List

1. **ViSession instrumentHandle [in]**

   This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.

   Default Value: None

2. **ViInt16 range [in]**

   Specifies whether the current, minimum or maximum setting is to be queried.

   Valid Values:

▪ RSSIAM_CURRENT_RANGE	(0) - Current
▪ RSSIAM_MIN_RANGE	(1) - Minimum
▪ RSSIAM_MAX_RANGE	(2) - Maximum

   Default Value: 0

3. **ViReal64 frequencyStop [out]**

   Returns the sweep stop frequency in hertz.

Return Value

Returns the status code of this operation.

The meaning of the status code is described in section Error (Status) Codes.

2.2.3.6.5 Sweep Freq Center Setup

C Function Prototype

```
ViStatus rssiam_freqCenter  (
   ViSession instrumentHandle,
   ViReal64 frequencyCenter);
```

Basic Function Prototype

```
Function rssiam_freqCenter (
   ByVal instrumentHandle As ViSession,
   ByVal frequencyCenter As ViReal64) As ViStatus
```

Purpose

Sets the sweep center frequency.

Parameters List

1. **ViSession instrumentHandle [in]**

   This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.

   Default Value: None

2. **ViReal64 frequencyCenter [in]**

   Sets the sweep center frequency.

   Range of this frequency depends on function currently selected:

▪ SINUSOID	0.1 Hz to 35.0e6 Hz
▪ TRIANGLE	0.1 Hz to 250.0e3 Hz

- RAMP 0.1 Hz to 250.0e3 Hz
- SQUARE 0.1 Hz to 250.0e3 Hz
- REXP 0.1 Hz to 250.0e3 Hz
- FEXP 0.1 Hz to 250.0e3 Hz
- SQUARE 0.1 Hz to 50.0e6 Hz
- PULSE 0.1 Hz to 50.0e6 Hz
- USER 0.1 Hz to 6.25e6 Hz

Default Value: 125.0e3 Hz

Return Value

Returns the status code of this operation.

The meaning of the status code is described in section Error (Status) Codes.

2.2.3.6.6 Sweep Freq Center Query

C Function Prototype

```
ViStatus rssiam_freqCenter_Q  (
   ViSession instrumentHandle,
   ViInt16 range,
   ViReal64* frequencyCenter);
```

Basic Function Prototype

```
Function rssiam_freqCenter_Q (
   ByVal instrumentHandle As ViSession,
   ByVal range As ViInt16,
   frequencyCenter As ViReal64) As ViStatus
```

Purpose

Returns the sweep center frequency.

Parameters List

1. **ViSession instrumentHandle [in]**

   This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.

   Default Value: None

2. **ViInt16 range [in]**

   Specifies whether the current, minimum or maximum setting is to be queried.

   Valid Values:

RSSIAM_CURRENT_RANGE	(0) - Current
RSSIAM_MIN_RANGE	(1) - Minimum
RSSIAM_MAX_RANGE	(2) - Maximum

   Default Value: 0

3. **ViReal64 frequencyCenter [out]**

   Returns the sweep center frequency in hertz.

Return Value

Returns the status code of this operation.

The meaning of the status code is described in section Error (Status) Codes.

2.2.3.6.7 Sweep Freq Span Setup

C Function Prototype

```
ViStatus rssiam_freqSpan  (
   ViSession instrumentHandle,
   ViReal64 frequencySpan);
```

Basic Function Prototype

```
Function rssiam_freqSpan (
   ByVal instrumentHandle As ViSession,
   ByVal frequencySpan As ViReal64) As ViStatus
```

Purpose

Sets the sweep span frequency.

Parameters List

1. **ViSession instrumentHandle [in]**

   This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.

   Default Value: None

2. **ViReal64 frequencySpan [in]**

   Sets the sweep span frequency.

   Range of this frequency depends on function currently selected:

   - SINUSOID 0.1 Hz to 35.0e6 Hz
   - TRIANGLE 0.1 Hz to 250.0e3 Hz
   - RAMP 0.1 Hz to 250.0e3 Hz
   - SQUARE 0.1 Hz to 250.0e3 Hz
   - REXP 0.1 Hz to 250.0e3 Hz
   - FEXP 0.1 Hz to 250.0e3 Hz
   - SQUARE 0.1 Hz to 50.0e6 Hz
   - PULSE 0.1 Hz to 50.0e6 Hz
   - USER 0.1 Hz to 6.25e6 Hz

   Default Value: 250.0e3 Hz

Return Value

Returns the status code of this operation.

The meaning of the status code is described in section Error (Status) Codes.

2.2.3.6.8 Sweep Freq Span Query

C Function Prototype

```
ViStatus rssiam_freqSpan_Q  (
   ViSession instrumentHandle,
   ViInt16 range,
   ViReal64* frequencySpan);
```

Basic Function Prototype

```
Function rssiam_freqSpan_Q (
   ByVal instrumentHandle As ViSession,
   ByVal range As ViInt16,
   frequencySpan As ViReal64) As ViStatus
```

Purpose

Returns the sweep span frequency.

Parameters List

1. **ViSession instrumentHandle [in]**

   This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.

Default Value: None

2. **ViInt16 range [in]**

   Specifies whether the current, minimum or maximum setting is to be queried.

   Valid Values:

   - RSSIAM_CURRENT_RANGE (0) - Current
   - RSSIAM_MIN_RANGE (1) - Minimum
   - RSSIAM_MAX_RANGE (2) - Maximum

   Default Value: 0

3. **ViReal64 frequencySpan [out]**

   Returns the sweep span frequency in hertz.

Return Value

Returns the status code of this operation.

The meaning of the status code is described in section Error (Status) Codes.

2.2.3.6.9 Sweep Direction Setup

C Function Prototype

```
ViStatus rssiam_sweDirection (
      ViSession instrumentHandle,
      ViInt16 sweepDirection);
```

Basic Function Prototype

```
Function rssiam_sweDirection (
      ByVal instrumentHandle As ViSession,
      ByVal sweepDirection As ViInt16) As ViStatus
```

Purpose

This function selects the direction of the sweep.

Parameters List

1. **ViSession instrumentHandle [in]**

   This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.

   Default Value: None

2. **ViInt16 sweepDirection [in]**

   Selects the direction of the sweep.

   Valid Values:

   - RSSIAM_SWEEP_DIRECTION_UP (0) - Up
   - RSSIAM_SWEEP_DIRECTION_DOWN (1) - Down

   Default Value: 0

Return Value

Returns the status code of this operation.

The meaning of the status code is described in section Error (Status) Codes.

2.2.3.6.10 Sweep Direction Query

C Function Prototype

```
ViStatus rssiam_sweDirection_Q (
      ViSession instrumentHandle,
      ViInt16* sweepDirection);
```

Basic Function Prototype

```
Function rssiam_sweDirection_Q (
      ByVal instrumentHandle As ViSession,
      sweepDirection As ViInt16) As ViStatus
```

Purpose This function returns the direction of the sweep.

Parameters List

1. **ViSession instrumentHandle [in]**

   This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.

   Default Value: None

2. **ViInt16 sweepDirection [out]**

   Returns the direction of the sweep.

   Valid Values:

   - RSSIAM_SWEEP_DIRECTION_UP (0) - Up
   - RSSIAM_SWEEP_DIRECTION_DOWN (1) - Down

Return Value Returns the status code of this operation.

The meaning of the status code is described in section Error (Status) Codes.

2.2.3.6.11 Sweep Points Setup

C Function Prototype

```
ViStatus rssiam_swePoin (
      ViSession instrumentHandle,
      ViInt32 sweepPoints);
```

Basic Function Prototype

```
Function rssiam_swePoin (
      ByVal instrumentHandle As ViSession,
      ByVal sweepPoints As ViInt32) As ViStatus
```

Purpose This function sets number of points in a stepped sweep.

Parameters List

1. **ViSession instrumentHandle [in]**

   This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.

   Default Value: None

2. **ViInt32 sweepPoints [in]**

   Sets number of points in a stepped sweep.

   Valid Range: 1 to 4096

   Default Value: 1024

Return Value Returns the status code of this operation.

The meaning of the status code is described in section Error (Status) Codes.

2.2.3.6.12 Sweep Points Query

C Function Prototype

```
ViStatus rssiam_swePoin_Q (
      ViSession instrumentHandle,
      ViInt32* dataPoints);
```

Basic Function Prototype

```
Function rssiam_swePoin_Q (
      ByVal instrumentHandle As ViSession,
      dataPoints As ViInt32) As ViStatus
```

Purpose This function returns number of points in the arbitrary waveform.

Parameters List

1. **ViSession instrumentHandle [in]**

   This control accepts the Instrument Handle returned by the Initialize

function to select the desired instrument driver session.

Default Value: None

2. **ViInt32 dataPoints [out]**

   Returns number of points in the arbitrary waveform.

Return Value

Returns the status code of this operation.

The meaning of the status code is described in section Error (Status) Codes.

2.2.3.6.13 Sweep Spacing Setup

C Function Prototype

```
ViStatus rssiam_sweSpac (
      ViSession instrumentHandle,
      ViInt16 sweepSpacing);
```

Basic Function Prototype

```
Function rssiam_sweSpac (
      ByVal instrumentHandle As ViSession,
      ByVal sweepSpacing As ViInt16) As ViStatus
```

Purpose

This function selects the linear or logarithmic spacing for the sweep.

Parameters List

1. **ViSession instrumentHandle [in]**

   This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.

   Default Value: None

2. **ViInt16 sweepSpacing [in]**

   Selects the linear or logarithmic spacing for the sweep.

   Valid Values:

   - RSSIAM_SWEEP_SPAC_LIN (0) - Linear
   - RSSIAM_SWEEP_SPAC_LOG (1) - Logarithmic

   Default Value: 0

Return Value

Returns the status code of this operation.

The meaning of the status code is described in section Error (Status) Codes.

2.2.3.6.14 Sweep Spacing Query

C Function Prototype

```
ViStatus rssiam_sweSpac_Q (
      ViSession instrumentHandle,
      ViInt16* sweepSpacing);
```

Basic Function Prototype

```
Function rssiam_sweSpac_Q (
      ByVal instrumentHandle As ViSession,
      sweepSpacing As ViInt16) As ViStatus
```

Purpose

This function returns whether the linear or logarithmic spacing is used for the sweep.

Parameters List

1. **ViSession instrumentHandle [in]**

   This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.

   Default Value: None

2. **ViInt16 sweepSpacing [out]**

Returns whether the linear or logarithmic spacing is used for the sweep.

Valid Values:

- RSSIAM_SWEEP_SPAC_LIN (0) - Linear
- RSSIAM_SWEEP_SPAC_LOG (1) - Logarithmic

Return Value

Returns the status code of this operation.

The meaning of the status code is described in section Error (Status) Codes.

2.2.3.6.15 Sweep Time Setup

C Function Prototype

```
ViStatus rssiam_sweTime (
      ViSession instrumentHandle,
      ViReal64 sweepTime);
```

Basic Function Prototype

```
Function rssiam_sweTime (
      ByVal instrumentHandle As ViSession,
      ByVal sweepTime As ViReal64) As ViStatus
```

Purpose

This function sets number of seconds required to sweep from the start frequency to the stop frequency.

Parameters List

1. **ViSession instrumentHandle [in]**

   This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.

   Default Value: None

2. **ViReal64 sweepTime [in]**

   Time required to sweep from start to stop frequency.

   Valid Range: 0.001 s to 999.0 s

   Default Value: 0.001 s

Return Value

Returns the status code of this operation.

The meaning of the status code is described in section Error (Status) Codes.

2.2.3.6.16 Sweep Time Query

C Function Prototype

```
ViStatus rssiam_sweTime_Q (
      ViSession instrumentHandle,
      ViInt16 range,
      ViReal64* sweepTime);
```

Basic Function Prototype

```
Function rssiam_sweTime_Q (
      ByVal instrumentHandle As ViSession,
      ByVal range As ViInt16,
      sweepTime As ViReal64) As ViStatus
```

Purpose

This function returns number of seconds required to sweep from the start frequency to the stop frequency.

Parameters List

1. **ViSession instrumentHandle [in]**

   This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.

   Default Value: None

2. **ViInt16 range [in]**

   Specifies whether the current, minimum or maximum setting is to be queried.

Valid Values:

RSSIAM_CURRENT_RANGE	(0) - Current
RSSIAM_MIN_RANGE	(1) - Minimum
RSSIAM_MAX_RANGE	(2) - Maximum

Default Value: 0

3. **ViReal64 sweepTime [out]**

   Returns the sweep time in seconds.

Return Value

Returns the status code of this operation.

The meaning of the status code is described in section Error (Status) Codes.

2.2.3.6.17 Sweep Stat Setup

C Function Prototype

```
ViStatus rssiam_sweStat (
      ViSession instrumentHandle,
      ViInt16 sweepState);
```

Basic Function Prototype

```
Function rssiam_sweStat (
      ByVal instrumentHandle As ViSession,
      ByVal sweepState As ViInt16) As ViStatus
```

Purpose

This function enables or disables the sweep mode. Only one modulation mode can be enabled at a time. When the sweep mode is enabled, the previous modulation mode is turned off.

Parameters List

1. **ViSession instrumentHandle [in]**

   This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.

   Default Value: None

2. **ViInt16 sweepState [in]**

   Enables or disables the sweep mode.

   Valid Values:

RSSIAM_SWEEP_STAT_OFF	(0) - Off
RSSIAM_SWEEP_STAT_ON	(1) - On

Default Value: 0

Return Value

Returns the status code of this operation.

The meaning of the status code is described in section Error (Status) Codes.

2.2.3.6.18 Sweep Stat Query

C Function Prototype

```
ViStatus rssiam_sweStat_Q (
      ViSession instrumentHandle,
      ViInt16* sweepState);
```

Basic Function Prototype

```
Function rssiam_sweStat_Q (
      ByVal instrumentHandle As ViSession,
      sweepState As ViInt16) As ViStatus
```

Purpose

This function returns whether the sweep mode is enabled or disabled.

Parameters List

1. **ViSession instrumentHandle [in]**

   This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.

   Default Value: None

2. **ViInt16 sweepState [out]**

   Returns whether the sweep mode is enabled or disabled.

   Valid Values:

   - RSSIAM_SWEEP_STAT_OFF (0) - Off
   - RSSIAM_SWEEP_STAT_ON (1) - On

Return Value

Returns the status code of this operation.

The meaning of the status code is described in section Error (Status) Codes.

2.2.3.7 Frequency Marker Settings

Description

Marker settings.

2.2.3.7.1 Marker Frequency Setup

C Function Prototype

```
ViStatus rssiam_mkrFreq (
      ViSession instrumentHandle,
      ViReal64 markerFrequency);
```

Basic Function Prototype

```
Function rssiam_mkrFreq (
      ByVal instrumentHandle As ViSession,
      ByVal markerFrequency As ViReal64) As ViStatus
```

Purpose

This function sets the marker frequency.

Parameters List

1. **ViSession instrumentHandle [in]**

   This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.

   Default Value: None

2. **ViReal64 markerFrequency [in]**

   Sets the marker frequency.

   Frequency range depends on function currently selected:

SINUSOID	0.1 Hz to 35.0e6 Hz
TRIANGLE	0.1 Hz to 250.0e3 Hz
RAMP	0.1 Hz to 250.0e3 Hz
SQUARE	0.1 Hz to 250.0e3 Hz
REXP	0.1 Hz to 250.0e3 Hz
FEXP	0.1 Hz to 250.0e3 Hz
SQUARE	0.1 Hz to 50.0e6 Hz
PULSE	0.1 Hz to 50.0e6 Hz
USER	0.1 Hz to 6.25e6 Hz

   Default Value: 1.0 Hz

Return Value

Returns the status code of this operation.

The meaning of the status code is described in section Error (Status) Codes.

2.2.3.7.2 Marker Frequency Query

C Function Prototype

```
ViStatus rssiam_mkrFreq_Q (
      ViSession instrumentHandle,
      ViReal64* markerFrequency);
```

Basic Function Prototype

```
Function rssiam_mkrFreq_Q (
      ByVal instrumentHandle As ViSession,
      markerFrequency As ViReal64) As ViStatus
```

Purpose

This function returns the marker frequency.

Parameters List

1. **ViSession instrumentHandle [in]**

   This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.

   Default Value: None

2. **ViReal64 markerFrequency [out]**

   Returns the marker frequency in hertz.

Return Value

Returns the status code of this operation.

The meaning of the status code is described in section Error (Status) Codes.

2.2.3.7.3 Marker Stat Setup

C Function Prototype

```
ViStatus rssiam_mkrStat (
      ViSession instrumentHandle,
      ViInt16 markerState);
```

Basic Function Prototype

```
Function rssiam_mkrStat (
      ByVal instrumentHandle As ViSession,
      ByVal markerState As ViInt16) As ViStatus
```

Purpose

This function sets the marker state.

Parameters List

1. **ViSession instrumentHandle [in]**

   This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.

   Default Value: None

2. **ViInt16 markerState [in]**

   Sets the marker state.

   Valid Values:

   - RSSIAM_MARKER_STAT_OFF (0) - Off
   - RSSIAM_MARKER_STAT_ON (1) - On

   Default Value: 0

Return Value

Returns the status code of this operation.

The meaning of the status code is described in section Error (Status) Codes.

2.2.3.7.4 Marker Stat Query

C Function Prototype

```
ViStatus rssiam_mkrStat_Q (
      ViSession instrumentHandle,
      ViInt16* markerState);
```

Basic Function Prototype

```
Function rssiam_mkrStat_Q (
      ByVal instrumentHandle As ViSession,
      markerState As ViInt16) As ViStatus
```

Purpose

This function returns the marker state.

Parameters List

1. **ViSession instrumentHandle [in]**

   This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.

   Default Value: None

2. **ViInt16 markerState [out]**

   Returns the marker state.

   Valid Values:

   - RSSIAM_MARKER_STAT_OFF (0) - Off
   - RSSIAM_MARKER_STAT_ON (1) - On

Return Value

Returns the status code of this operation.

The meaning of the status code is described in section Error (Status) Codes.

2.2.3.8 Trigger Subsystem Settings

2.2.3.8.1 Trigger Settings

Description

Applies to burst modulation and frequency modulation. It is possible to issue triggers for bursts and sweeps using an immediate trigger, an external trigger, or a bus trigger.

2.2.3.8.1.1 Trigger Source Setup

C Function Prototype

```
ViStatus rssiam_trigSour (
    ViSession instrumentHandle,
    ViInt16 triggerSource);
```

Basic Function Prototype

```
Function rssiam_trigSour (
    ByVal instrumentHandle As ViSession,
    ByVal triggerSource As ViInt16) As ViStatus
```

Purpose

This function selects the source from which the function generator will accept a trigger. The function generator will accept an immediate internal trigger, a hardware trigger from the rear-panel Ext Trig terminal, or a software (bus) trigger.

Parameters List

1. **ViSession instrumentHandle [in]**

   This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.

   Default Value: None

2. **ViInt16 triggerSource [in]**

   Selects the trigger source.

   Valid Values:

   - RSSIAM_TRIGGER_SOUR_IMM (0) - Immediate
   - RSSIAM_TRIGGER_SOUR_EXT (1) - External
   - RSSIAM_TRIGGER_SOUR_BUS (2) - Manual
   - RSSIAM_TRIGGER_SOUR_SYS (3) – System

   Default Value: 0

Return Value

Returns the status code of this operation.

The meaning of the status code is described in section Error (Status) Codes.

2.2.3.8.1.2 Trigger Source Query

C Function Prototype

```
ViStatus rssiam_trigSour_Q (
    ViSession instrumentHandle,
    ViInt16* triggerSource);
```

Basic Function Prototype

```
Function rssiam_trigSour_Q (
    ByVal instrumentHandle As ViSession,
    triggerSource As ViInt16) As ViStatus
```

Purpose

This function returns the source from which the function generator will accept a trigger.

Parameters List

1. **ViSession instrumentHandle [in]**

   This control accepts the Instrument Handle returned by the Initialize

function to select the desired instrument driver session.

Default Value: None

2. **ViInt16 triggerSource [out]**

   Returns the trigger source.

   Valid Values:

   - RSSIAM_TRIGGER_SOUR_IMM (0) - Immediate
   - RSSIAM_TRIGGER_SOUR_EXT (1) - External
   - RSSIAM_TRIGGER_SOUR_BUS (2) - Manual
   - RSSIAM_TRIGGER_SOUR_SYS (3) - System

Return Value

Returns the status code of this operation.

The meaning of the status code is described in section Error (Status) Codes.

2.2.3.8.1.3 Trigger Polarity Setup

C Function Prototype

```
ViStatus rssiam_trigPolarity  (
   ViSession instrumentHandle,
   ViInt16 triggerPolarity);
```

Basic Function Prototype

```
Function rssiam_trigPolarity (
   ByVal instrumentHandle As ViSession,
   ByVal triggerPolarity As ViInt16) As ViStatus
```

Purpose

Sets the trigger polarity (edge which the event occurs on).

Parameters List

1. **ViSession instrumentHandle [in]**

   This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.

   Default Value: None

2. **ViInt16 triggerPolarity [in]**

   Sets the trigger polarity (edge which the event occurs on).

   Valid Values:

   - RSSIAM_TRIGGER_SLOPE_POS (0) – Positive
   - RSSIAM_TRIGGER_SLOPE_NEG (1) - Negative

   Default Value: 0

Return Value

Returns the status code of this operation.

The meaning of the status code is described in section Error (Status) Codes.

2.2.3.8.1.4 Trigger Polarity Query

C Function Prototype

```
ViStatus rssiam_trigPolarity_Q  (
   ViSession instrumentHandle,
   ViInt16* triggerPolarity);
```

Basic Function Prototype

```
Function rssiam_trigPolarity_Q (
   ByVal instrumentHandle As ViSession,
   triggerPolarity As ViInt16) As ViStatus
```

Purpose

Returns the trigger polarity (edge which the event occurs on).

Parameters List

1. **ViSession instrumentHandle [in]**

   This control accepts the Instrument Handle returned by the Initialize

function to select the desired instrument driver session.

Default Value: None

2. **ViInt16 triggerPolarity [out]**

   Returns the trigger polarity (edge which the event occurs on).

   Valid Values:

   - RSSIAM_TRIGGER_SLOPE_POS (0) - Positive
   - RSSIAM_TRIGGER_SLOPE_NEG (1) - Negative

Return Value

Returns the status code of this operation.

The meaning of the status code is described in section Error (Status) Codes.

2.2.3.8.1.5 Trigger Delay Setup

C Function Prototype

```
ViStatus rssiam_trigDelay (
      ViSession instrumentHandle,
      ViReal64 triggerDelay);
```

Basic Function Prototype

```
Function rssiam_trigDelay (
      ByVal instrumentHandle As ViSession,
      ByVal triggerDelay As ViReal64) As ViStatus
```

Purpose

This function sets the trigger delay (time duration between the recognition of an event(s) and the start of the action).

Applicable channels:

- CH1, CH2 if device config selects two independent frequencies

Note

If device config selects common frequency (see function Output Channel Coupling (rssiam_outpChannelCoupl())), channel does not apply.

Parameters List

1. **ViSession instrumentHandle [in]**

   This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.

   Default Value: None

2. **ViReal64 triggerDelay [in]**

   Sets the trigger delay.

   Valid Range: 0.0 s to 9999.0 s

   Default Value: 0.0 s

Return Value

Returns the status code of this operation.

The meaning of the status code is described in section Error (Status) Codes.

2.2.3.8.1.6 Trigger Delay Query

C Function Prototype

```
ViStatus rssiam_trigDelay_Q (
      ViSession instrumentHandle,
      ViReal64* triggerDelay);
```

Basic Function Prototype

```
Function rssiam_trigDelay_Q (
      ByVal instrumentHandle As ViSession,
      triggerDelay As ViReal64) As ViStatus
```

Purpose

This function returns the trigger delay (time duration between the recognition of an event(s) and the start of the action).

Applicable channels:

- CH1, CH2 if device config selects two independent frequencies

Note

If device config selects common frequency (see function Output Channel Coupling (rssiam_outpChannelCoupl())), channel does not apply.

Parameters List

1. **ViSession instrumentHandle [in]**

   This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.

   Default Value: None

2. **ViReal64 triggerDelay [out]**

   Returns the trigger delay in seconds.

Return Value

Returns the status code of this operation.

The meaning of the status code is described in section Error (Status) Codes.

2.2.3.8.1.7 Trigger Frequency Setup

C Function Prototype

```
ViStatus rssiam_trigFrequency  (
   ViSession instrumentHandle,
   ViReal64 triggerFrequency);
```

Basic Function Prototype

```
Function rssiam_trigFrequency (
   ByVal instrumentHandle As ViSession,
   ByVal triggerFrequency As ViReal64) As ViStatus
```

Purpose

Sets the internal trigger generator frequency.

Parameters List

1. **ViSession instrumentHandle [in]**

   This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.

   Default Value: None

2. **ViReal64 triggerFrequency [in]**

   Sets the internal trigger generator frequency.

   Valid Range: 101.0e-6 Hz to 2.0e6 Hz

   Default Value: 1000.0 Hz

Return Value

Returns the status code of this operation.

The meaning of the status code is described in section Error (Status) Codes.

2.2.3.8.1.8 Trigger Frequency Query

C Function Prototype

```
ViStatus rssiam_trigFrequency_Q  (
   ViSession instrumentHandle,
   ViReal64* triggerFrequency);
```

Basic Function Prototype

```
Function rssiam_trigFrequency_Q (
   ByVal instrumentHandle As ViSession,
   triggerFrequency As ViReal64) As ViStatus
```

Purpose

Returns the internal trigger generator frequency.

Parameters List

1. **ViSession instrumentHandle [in]**

   This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.

   Default Value: None

2. **ViReal64 triggerFrequency [out]**

   Returns the internal trigger generator frequency in Hz.

Return Value

Returns the status code of this operation.

The meaning of the status code is described in section Error (Status) Codes.

2.2.3.8.1.9 Trigger Period Setup

C Function Prototype

```
ViStatus rssiam_trigPeriod  (
   ViSession instrumentHandle,
   ViReal64 triggerPeriod);
```

Basic Function Prototype

```
Function rssiam_trigPeriod (
   ByVal instrumentHandle As ViSession,
   ByVal triggerPeriod As ViReal64) As ViStatus
```

Purpose

Sets the internal trigger generator period.

Parameters List

1. **ViSession instrumentHandle [in]**

   This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.

   Default Value: None

2. **ViReal64 triggerPeriod [in]**

   Sets the internal trigger generator period.

   Valid Range: 500.0e-9 s to 9901.0 s

   Default Value: 0.001 s

Return Value

Returns the status code of this operation.

The meaning of the status code is described in section Error (Status) Codes.

2.2.3.8.1.10 Trigger Period Query

C Function Prototype

```
ViStatus rssiam_trigPeriod_Q  (
   ViSession instrumentHandle,
   ViReal64* triggerPeriod);
```

Basic Function Prototype

```
Function rssiam_trigPeriod_Q (
   ByVal instrumentHandle As ViSession,
   triggerPeriod As ViReal64) As ViStatus
```

Purpose

Returns the internal trigger generator period.

Parameters List

1. **ViSession instrumentHandle [in]**

   This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.

   Default Value: None

2. **ViReal64 triggerPeriod [out]**

Returns the internal trigger generator period in seconds.

Return Value

Returns the status code of this operation.

The meaning of the status code is described in section Error (Status) Codes.

2.2.3.8.2 Burst Mode

Description

It is possible to configure the function generator to output a waveform with a specified number of cycles, called burst. It is possible to output the burst at a rate determined by the internal rate generator or an external signal applied to the rear-panel connector. These functions can be used to setup and query the burst modulated parameters.

2.2.3.8.2.1 Burst Mod Ncycles Setup

C Function Prototype

```
ViStatus rssiam_bmNcyc (
      ViSession instrumentHandle,
      ViInt32 noofCycles);
```

Basic Function Prototype

```
Function rssiam_bmNcyc (
      ByVal instrumentHandle As ViSession,
      ByVal noofCycles As ViInt32) As ViStatus
```

Purpose

This function sets number of cycles to be output per burst.

Applicable channels:

- CH1, CH2 if device config selects two independent frequencies

Note

If device config selects common frequency (see function Output Channel Coupling (rssiam_outpChannelCoupl())), channel does not apply.

Parameters List

3. **ViSession instrumentHandle [in]**

   This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.

   Default Value: None

4. **ViInt32 noofCycles [in]**

   Sets number of cycles to be output per burst.

   Valid Range: 1 to 65535

   Default Value: 1

Return Value

Returns the status code of this operation.

The meaning of the status code is described in section Error (Status) Codes.

2.2.3.8.2.2 Burst Mod Ncycles Query

C Function Prototype

```
ViStatus rssiam_bmNcyc_Q (
      ViSession instrumentHandle,
      ViInt16 range,
      ViInt32* noofCycles);
```

Basic Function Prototype

```
Function rssiam_bmNcyc_Q (
     ByVal instrumentHandle As ViSession,
     ByVal range As ViInt16,
     noofCycles As ViInt32) As ViStatus
```

Purpose

This function returns number of cycles outputed per burst.

Applicable channels:

- CH1, CH2 if device config selects two independent frequencies

Note If device config selects common frequency (see function Output Channel Coupling (rssiam_outpChannelCoupl())), channel does not apply.

Parameters List

5. **ViSession instrumentHandle [in]**

   This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.

   Default Value: None

6. **ViInt16 range [in]**

   Specifies whether the current, minimum or maximum setting is to be queried.

   Valid Values:

RSSIAM_CURRENT_RANGE	(0) - Current
RSSIAM_MIN_RANGE	(1) - Minimum
RSSIAM_MAX_RANGE	(2) - Maximum

   Default Value: 0

7. **ViInt32 noofCycles [out]**

   Returns number of cycles outputed per burst.

Return Value

Returns the status code of this operation.

The meaning of the status code is described in section Error (Status) Codes.

2.2.3.8.2.3 Burst Mod Int Rate Setup

C Function Prototype

```
ViStatus rssiam_bmIntRate (
     ViSession instrumentHandle,
     ViReal64 bmIntRate);
```

Basic Function Prototype

```
Function rssiam_bmIntRate (
     ByVal instrumentHandle As ViSession,
     ByVal bmIntRate As ViReal64) As ViStatus
```

Purpose

This function sets the burst rate for internally triggered bursts. The burst rate frequency defines the interval between bursts.

The burst rate setting is used only when internal triggering is enabled. The burst rate is ignored when manual triggering or external triggering is enabled.

It is possible to specify a burst rate which is too fast for the function generator to output with the specified carrier frequency and burst count. If the burst rate is too high, the function generator will internally adjust it as needed to continuously re-trigger the burst. The adjustment is handled internally by the function generator.

Parameters List

8. **ViSession instrumentHandle [in]**

   This control accepts the Instrument Handle returned by the Initialize

function to select the desired instrument driver session.

Default Value: None

9. **ViReal64 bmIntRate [in]**

   Sets the burst rate for internally triggered bursts.

   Valid Range: 0.0001 Hz to 2.0e6 Hz

   Default Value: 1000.0 Hz

Return Value

Returns the status code of this operation.

The meaning of the status code is described in section Error (Status) Codes.

2.2.3.8.2.4 Burst Mod Int Rate Query

C Function Prototype

```
ViStatus rssiam_bmIntRate_Q (
      ViSession instrumentHandle,
      ViInt16 range,
      ViReal64* bmIntRate);
```

Basic Function Prototype

```
Function rssiam_bmIntRate_Q (
      ByVal instrumentHandle As ViSession,
      ByVal range As ViInt16,
      bmIntRate As ViReal64) As ViStatus
```

Purpose

This function returns the burst rate frequency.

Parameters List

10. **ViSession instrumentHandle [in]**

    This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.

    Default Value: None

11. **ViInt16 range [in]**

    Specifies whether the current, minimum or maximum setting is to be queried.

    Valid Values:

▪ RSSIAM_CURRENT_RANGE	(0) - Current
▪ RSSIAM_MIN_RANGE	(1) - Minimum
▪ RSSIAM_MAX_RANGE	(2) - Maximum

    Default Value: 0

12. **ViReal64 bmIntRate [out]**

    Returns the burst rate frequency in hertz.

Return Value

Returns the status code of this operation.

The meaning of the status code is described in section Error (Status) Codes.

2.2.3.8.2.5 Burst Mod Sour Setup

C Function Prototype

```
ViStatus rssiam_bmSour (
      ViSession instrumentHandle,
      ViInt16 bmSource);
```

Basic Function Prototype

```
Function rssiam_bmSour (
      ByVal instrumentHandle As ViSession,
      ByVal bmSource As ViInt16) As ViStatus
```

Purpose

Selects the burst mode. In the triggered mode, the function generator outputs a waveform with a specified number of cycles (burst count) each time a trigger is received. In the gated mode, the output waveform is either "on" or "off" based on the level of the external signal applied to the rear panel Burst In connector. When the internal burst source is selected, the external gated mode is disabled.

When the gated source is selected, the burst count, burst rate and trigger source are ignored.

This function affects settings of Gate Source Setup (rssiam_gateSource) function.

Parameters List

13. **ViSession instrumentHandle [in]**

    This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.

    Default Value: None

14. **ViInt16 bmSource [in]**

    Selects the burst modulation source.

    Valid Values:

    - RSSIAM_BM_SOUR_TRIG (0) - Triggered
    - RSSIAM_BM_SOUR_INT (1) - Int Gated
    - RSSIAM_BM_SOUR_EXT (2) - Ext Gated

    Default Value: 0

Return Value

Returns the status code of this operation.

The meaning of the status code is described in section Error (Status) Codes.

2.2.3.8.2.6 Burst Mod Sour Query

C Function Prototype

```
ViStatus rssiam_bmSour_Q (
      ViSession instrumentHandle,
      ViInt16* bmSource);
```

Basic Function Prototype

```
Function rssiam_bmSour_Q (
      ByVal instrumentHandle As ViSession,
      bmSource As ViInt16) As ViStatus
```

Purpose

This function returns the burst modulation source.

Parameters List

15. **ViSession instrumentHandle [in]**

    This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.

    Default Value: None

16. **ViInt16 bmSource [out]**

    Returns the burst modulation source.

    Valid Values:

    - RSSIAM_BM_MOD_SOUR_TRIG (0) - Triggered
    - RSSIAM_BM_MOD_SOUR_INT (1) - Int Gated
    - RSSIAM_BM_MOD_SOUR_EXT (2) - Ext Gated

Return Value

Returns the status code of this operation.

The meaning of the status code is described in section Error (Status) Codes.

2.2.3.8.2.7 Burst Mod Stat Setup

C Function Prototype

```
ViStatus rssiam_bmStat (
      ViSession instrumentHandle,
      ViInt16 bmState);
```

Basic Function Prototype

```
Function rssiam_bmStat (
      ByVal instrumentHandle As ViSession,
      ByVal bmState As ViInt16) As ViStatus
```

Purpose

This function enables or disables the burst modulation. Only one modulation mode can be enabled at a time. When BM is enabled, the previous modulation mode is turned off.

Parameters List

17. **ViSession instrumentHandle [in]**

    This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.

    Default Value: None

18. **ViInt16 bmState [in]**

    Enables or disables the burst modulation.

    Valid Values:

    - RSSIAM_BM_STAT_OFF (0) - Off
    - RSSIAM_BM_STAT_ON (1) - On

    Default Value: 0

Return Value

Returns the status code of this operation.

The meaning of the status code is described in section Error (Status) Codes.

2.2.3.8.2.8 Burst Mod Stat Query

C Function Prototype

```
ViStatus rssiam_bmStat_Q (
      ViSession instrumentHandle,
      ViInt16* bmState);
```

Basic Function Prototype

```
Function rssiam_bmStat_Q (
      ByVal instrumentHandle As ViSession,
      bmState As ViInt16) As ViStatus
```

Purpose

This function returns whether the burst modulation is enabled or disabled.

Parameters List

19. **ViSession instrumentHandle [in]**

    This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.

    Default Value: None

20. **ViInt16 bmState [out]**

    Returns whether the burst modulation is enabled or disabled.

    Valid Values:

    - RSSIAM_BM_STAT_OFF (0) - Off
    - RSSIAM_BM_STAT_ON (1) – On

Return Value

Returns the status code of this operation.

The meaning of the status code is described in section Error (Status) Codes.

2.2.3.8.3 Sync Settings

Description

This group of functions contains information to help you configure the sync signal parameters.

2.2.3.8.3.1 Sync Type Setup

C Function Prototype

```
ViStatus rssiam_syncType (
      ViSession instrumentHandle,
      ViInt16 syncType);
```

Basic Function Prototype

```
Function rssiam_syncType (
      ByVal instrumentHandle As ViSession,
      ByVal syncType As ViInt16) As ViStatus
```

Purpose

This function selects the type of the sync signal.

Applicable channels: CH1 or CH2

Parameters List

1. **ViSession instrumentHandle [in]**

   This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.

   Default Value: None

2. **ViInt16 syncType [in]**

   Selects the type of the sync signal.

   Valid Values:

▪ RSSIAM_SYNC_TYPE_COMP	(0) - COMP
▪ RSSIAM_SYNC_TYPE_CARRY	(1) - CARRY
▪ RSSIAM_SYNC_TYPE_TRIG	(2) - TRIG
▪ RSSIAM_SYNC_TYPE_MOD	(3) - MOD
▪ RSSIAM_SYNC_TYPE_MFRQ	(4) - MFRQ
▪ RSSIAM_SYNC_TYPE_MWFM	(5) - MWFM

   Default Value: 0

Return Value

Returns the status code of this operation.

The meaning of the status code is described in section Error (Status) Codes.

2.2.3.8.3.2 Sync Type Query

C Function Prototype

```
ViStatus rssiam_syncType_Q (
      ViSession instrumentHandle,
      ViInt16* syncType);
```

Basic Function Prototype

```
Function rssiam_syncType_Q (
      ByVal instrumentHandle As ViSession,
      syncType As ViInt16) As ViStatus
```

Purpose

This function returns the type of the sync signal.

Applicable channels: CH1 or CH2

Parameters List

1. **ViSession instrumentHandle [in]**

   This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.

   Default Value: None

2. **ViInt16 syncType [out]**

Returns the type of the sync signal.

Valid Values:

RSSIAM_SYNC_TYPE_COMP	(0) - COMP
RSSIAM_SYNC_TYPE_CARRY	(1) - CARRY
RSSIAM_SYNC_TYPE_TRIG	(2) - TRIG
RSSIAM_SYNC_TYPE_MOD	(3) - MOD
RSSIAM_SYNC_TYPE_MFRQ	(4) - MFRQ
RSSIAM_SYNC_TYPE_MWFM	(5) - MWFM

Return Value

Returns the status code of this operation.

The meaning of the status code is described in section Error (Status) Codes.

2.2.3.8.3.3 Sync Polarity Setup

C Function Prototype

```
ViStatus rssiam_syncPolarity (
      ViSession instrumentHandle,
      ViInt16 syncPolarity);
```

Basic Function Prototype

```
Function rssiam_syncPolarity (
      ByVal instrumentHandle As ViSession,
      ByVal syncPolarity As ViInt16) As ViStatus
```

Purpose

This function selects the polarity of the sync signal.

Applicable channels: CH1 or CH2

Parameters List

1. **ViSession instrumentHandle [in]**

   This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.

   Default Value: None

2. **ViInt16 syncPolarity [in]**

   Selects the polarity of the sync signal.

   Valid Values:

RSSIAM_SYNC_POLARITY_NORMAL	(0) - Normal
RSSIAM_SYNC_POLARITY_INVERTED	(1) - Inverted

   Default Value: 0

Return Value

Returns the status code of this operation.

The meaning of the status code is described in section Error (Status) Codes.

2.2.3.8.3.4 Sync Polarity Query

C Function Prototype

```
ViStatus rssiam_syncPolarity_Q (
      ViSession instrumentHandle,
      ViInt16* syncPolarity);
```

Basic Function Prototype

```
Function rssiam_syncPolarity_Q (
      ByVal instrumentHandle As ViSession,
      syncPolarity As ViInt16) As ViStatus
```

Purpose

This function returns the polarity of the sync signal.

Applicable channels: CH1 or CH2

Parameters List

1. **ViSession instrumentHandle [in]**

   This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.

   Default Value: None

2. **ViInt16 syncPolarity [out]**

   Returns the polarity of the sync signal.

   Valid Values:

   - RSSIAM_SYNC_POLARITY_NORMAL (0) - Normal
   - RSSIAM_SYNC_POLARITY_INVERTED (1) - Inverted

Return Value

Returns the status code of this operation.

The meaning of the status code is described in section Error (Status) Codes.

2.2.3.8.4 Gate Settings

Description

Gate settings.

2.2.3.8.4.1 Gate Function Setup

C Function Prototype

```
ViStatus rssiam_gateFunction (
      ViSession instrumentHandle,
      ViInt16 gateFunction);
```

Basic Function Prototype

```
Function rssiam_gateFunction (
      ByVal instrumentHandle As ViSession,
      ByVal gateFunction As ViInt16) As ViStatus
```

Purpose

This function sets the gate function.

Parameters List

1. **ViSession instrumentHandle [in]**

   This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.

   Default Value: None

2. **ViInt16 gateFunction [in]**

   Sets the gate function.

   Valid Values:

▪ RSSIAM_GATE_SAMPLE_HOLD	(0) - Sample & Hold
▪ RSSIAM_GATE_BLOCK_END	(1) - Block End

   Default Value: 0

Return Value

Returns the status code of this operation.

The meaning of the status code is described in section Error (Status) Codes.

2.2.3.8.4.2 Gate Function Query

C Function Prototype

```
ViStatus rssiam_gateFunction_Q (
      ViSession instrumentHandle,
      ViInt16* gateFunction);
```

Basic Function Prototype

```
Function rssiam_gateFunction_Q (
      ByVal instrumentHandle As ViSession,
      gateFunction As ViInt16) As ViStatus
```

Purpose

This function returns the gate function.

Parameters List

1. **ViSession instrumentHandle [in]**

   This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.

   Default Value: None

2. **ViInt16 gateFunction [out]**

   Returns the gate function.

   Valid Values:

▪ RSSIAM_GATE_SAMPLE_HOLD	(0) - Sample & Hold
▪ RSSIAM_GATE_BLOCK_END	(1) - Block End

Return Value

Returns the status code of this operation.

The meaning of the status code is described in section Error (Status) Codes.

2.2.3.8.4.3 Gate Length Setup

C Function Prototype

```
ViStatus rssiam_gateLength (
   ViSession instrumentHandle,
   ViReal64 gateLength);
```

Basic Function Prototype

```
Function rssiam_gateLength (
   ByVal instrumentHandle As ViSession,
   ByVal gateLength As ViReal64) As ViStatus
```

Purpose

Sets the internal gate time (the time for which generator outputs a continuous waveform).

Parameters List

1. **ViSession instrumentHandle [in]**

   This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.

   Default Value: None

2. **ViReal64 gateLength [in]**

   Sets the internal gate time.

   Valid Range: 0.0 s to 9999.0 s

   Default Value: 100.0e-9 s

Return Value

Returns the status code of this operation.

The meaning of the status code is described in section Error (Status) Codes.

2.2.3.8.4.4 Gate Length Query

C Function Prototype

```
ViStatus rssiam_gateLength_Q (
   ViSession instrumentHandle,
   ViReal64* gateLength);
```

Basic Function Prototype

```
Function rssiam_gateLength_Q (
   ByVal instrumentHandle As ViSession,
   gateLength As ViReal64) As ViStatus
```

Purpose

Returns the internal gate time (the time for which generator outputs a continuous waveform).

Parameters List

1. **ViSession instrumentHandle [in]**

   This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.

   Default Value: None

2. **ViReal64 gateLength [out]**

   Returns the internal gate time.

Return Value

Returns the status code of this operation.

The meaning of the status code is described in section Error (Status) Codes.

2.2.3.8.4.5 Gate Source Setup

C Function Prototype

```
ViStatus rssiam_gateSource  (
   ViSession instrumentHandle,
   ViInt16 gateSource);
```

Basic Function Prototype

```
Function rssiam_gateSource (
   ByVal instrumentHandle As ViSession,
   ByVal gateSource As ViInt16) As ViStatus
```

Purpose

Selects the gate source. In the triggered mode, the function generator outputs a waveform with a specified number of cycles (burst count) each time a trigger is received. In the gated mode, the output waveform is either "on" or "off" based on the level of the external signal applied to the rear panel Burst In connector. When the internal burst source is selected, the external gated mode is disabled.

When the gated source is selected, the burst count, burst rate and trigger source are ignored.

This function affects settings of Burst Mod Sour Setup (rssiam_bmSour) function.

Parameters List

1. **ViSession instrumentHandle [in]**

   This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.

   Default Value: None

2. **ViInt16 gateSource [in]**

   Selects the burst source.

   Valid Values:

   - RSSIAM_BM_SOUR_TRIG (0) – Triggered
   - RSSIAM_BM_SOUR_INT (1) - Int Gated
   - RSSIAM_BM_SOUR_EXT (2) - Ext Gated

   Default Value: 0

Return Value

Returns the status code of this operation.

The meaning of the status code is described in section Error (Status) Codes.

2.2.3.8.4.6 Gate Source Query

C Function Prototype

```
ViStatus rssiam_gateSource_Q  (
   ViSession instrumentHandle,
   ViInt16* gateSource);
```

Basic Function Prototype

```
Function rssiam_gateSource_Q (
   ByVal instrumentHandle As ViSession,
   gateSource As ViInt16) As ViStatus
```

Purpose

Returns the gate source.

Parameters List

1. **ViSession instrumentHandle [in]**

   This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.

   Default Value: None

2. **ViInt16 gateSource [out]**

   Returns the gate source.

   Valid Values:

   - RSSIAM_BM_SOUR_TRIG (0) – Triggered
   - RSSIAM_BM_SOUR_INT (1) - Int Gated
   - RSSIAM_BM_SOUR_EXT (2) - Ext Gated

Return Value

Returns the status code of this operation.

The meaning of the status code is described in section Error (Status) Codes.

2.2.3.8.4.7 Gate Polarity Setup

C Function Prototype

```
ViStatus rssiam_gatePolarity  (
   ViSession instrumentHandle,
   ViInt16 gatePolarity);
```

Basic Function Prototype

```
Function rssiam_gatePolarity (
   ByVal instrumentHandle As ViSession,
   ByVal gatePolarity As ViInt16) As ViStatus
```

Purpose

Sets the gate polarity (edge which the event occurs on).

Parameters List

1. **ViSession instrumentHandle [in]**

   This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.

   Default Value: None

2. **ViInt16 gatePolarity [in]**

   Sets the gate polarity (edge which the event occurs on).

   Valid Values:

   - RSSIAM_TRIGGER_SLOPE_POS (0) – Positive
   - RSSIAM_TRIGGER_SLOPE_NEG (1) – Negative

   Default Value: 0

Return Value

Returns the status code of this operation.

The meaning of the status code is described in section Error (Status) Codes.

2.2.3.8.4.8 Gate Polarity Query

C Function Prototype

```
ViStatus rssiam_gatePolarity_Q  (
   ViSession instrumentHandle,
   ViInt16* gatePolarity);
```

Basic Function Prototype

```
Function rssiam_gatePolarity_Q (
   ByVal instrumentHandle As ViSession,
   gatePolarity As ViInt16) As ViStatus
```

Purpose

Returns the gate polarity (edge which the event occurs on).

Parameters List

1. **ViSession instrumentHandle [in]**

   This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.

Default Value: None

2. **ViInt16 gatePolarity [out]**
   Returns the gate polarity (edge which the event occurs on).

   Valid Values:

   - RSSIAM_TRIGGER_SLOPE_POS (0) – Positive
   - RSSIAM_TRIGGER_SLOPE_NEG (1) – Negative

Return Value

Returns the status code of this operation.

The meaning of the status code is described in section Error (Status) Codes.

2.2.4 Action/Status Functions

Description

This class of functions begins or terminates an acquisition. It also provides functions which allow the user to determine the current status of the instrument.

Functions/SubClasses:

1. **Send Trigger**
   This function sends a trigger.
2. **Abort Trigger**
   This function aborts the sweep system (stops sweeping).

2.2.4.1 Send Trigger

C Function Prototype

```
ViStatus rssiam_sendTrigger (
      ViSession instrumentHandle);
```

Basic Function Prototype

```
Function rssiam_sendTrigger (
      ByVal instrumentHandle As ViSession) As ViStatus
```

Purpose

This function sends a trigger.

Note

This function is available in case of Manual Trigger Source.

Parameters List

1. **ViSession instrumentHandle [in]**
   This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.

   Default Value: None

Return Value

Returns the status code of this operation.

The meaning of the status code is described in section Error (Status) Codes.

2.2.4.2 Abort Trigger

C Function Prototype

```
ViStatus rssiam_abortTrigger (
    ViSession instrumentHandle);
```

Basic Function Prototype

```
Function rssiam_abortTrigger (
    ByVal instrumentHandle As ViSession) As ViStatus
```

Purpose

This function aborts (resets) the trigger system (stops sweeping).

Parameters List

1. **ViSession instrumentHandle [in]**

   This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.

   Default Value: None

Return Value

Returns the status code of this operation.

The meaning of the status code is described in section Error (Status) Codes.

2.2.5 Utility Functions

Description

This class of functions provides lower level functions to communicate with the instrument, and change instrument's parameters.

2.2.5.1 Time Out

Description

This class of function enables you to set and query the timeout parameter of the instrument.

2.2.5.1.1 Time Out Setup

C Function Prototype

```
ViStatus rssiam_timeOut (
      ViSession instrumentHandle,
      ViInt32 timeout);
```

Basic Function Prototype

```
Function rssiam_timeOut (
      ByVal instrumentHandle As ViSession,
      ByVal timeout As ViInt32) As ViStatus
```

Purpose

This function sets a minimum timeout value for driver I/O transactions in milliseconds. The timeout period may vary on computer platforms.

Parameters List

1. **ViSession instrumentHandle [in]**

   This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.

   Default Value: None

2. **ViInt32 timeout [in]**

   Sets the I/O timeout for all functions in the driver. It is specified in milliseconds.

   Valid Range: > 0 ms

   Default Value: 15000 ms

Return Value

Returns the status code of this operation.

The meaning of the status code is described in section Error (Status) Codes.

2.2.5.1.2 Time Out Query

C Function Prototype

```
ViStatus rssiam_timeOut_Q (
      ViSession instrumentHandle,
      ViInt32* timeout);
```

Basic Function Prototype

```
Function rssiam_timeOut_Q (
      ByVal instrumentHandle As ViSession,
      timeout As ViInt32) As ViStatus
```

Purpose

This function returns the timeout value for driver I/O transactions in milliseconds.

The timeout period may vary on computer platforms.

Parameters List

1. **ViSession instrumentHandle [in]**

   This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.

   Default Value: None

2. **ViInt32 timeout [out]**

   Returns the timeout value for driver I/O transactions in milliseconds.

Return Value

Returns the status code of this operation.

The meaning of the status code is described in section Error (Status) Codes.

2.2.5.2 Reference Oscillator

Description Reference oscillator settings.

2.2.5.2.1 Reference Oscillator Source Setup

C Function Prototype

```
ViStatus rssiam_roscSour (
      ViSession instrumentHandle,
      ViInt16 roscSource);
```

Basic Function Prototype

```
Function rssiam_roscSour (
      ByVal instrumentHandle As ViSession,
      ByVal roscSource As ViInt16) As ViStatus
```

Purpose This function selects the reference oscillator source.

Parameters List

1. **ViSession instrumentHandle [in]**

   This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.

   Default Value: None

2. **ViInt16 roscSource [in]**

   Selects the reference oscillator source.

   Valid Values:

   - RSSIAM_ROSC_TCXO (0) - TCXO
   - RSSIAM_ROSC_EXT (2) - External

   Default Value: 0

Return Value Returns the status code of this operation.

The meaning of the status code is described in section Error (Status) Codes.

2.2.5.2.2 Reference Oscillator Source Query

C Function Prototype

```
ViStatus rssiam_roscSour_Q (
      ViSession instrumentHandle,
      ViInt16* roscSource);
```

Basic Function Prototype

```
Function rssiam_roscSour_Q (
      ByVal instrumentHandle As ViSession,
      roscSource As ViInt16) As ViStatus
```

Purpose This function returns the reference oscillator source.

Parameters List

1. **ViSession instrumentHandle [in]**

   This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.

   Default Value: None

2. **ViInt16 roscSource [out]**

   Returns the reference oscillator source.

   Valid Values:

   - RSSIAM_ROSC_TCXO (0) - TCXO
   - RSSIAM_ROSC_EXT (2) - External

Return Value Returns the status code of this operation.

The meaning of the status code is described in section Error (Status) Codes.

2.2.5.2.3 Reference Oscillator Stat Setup

C Function Prototype

```
ViStatus rssiam_roscStat (
      ViSession instrumentHandle,
      ViInt16 roscState);
```

Basic Function Prototype

```
Function rssiam_roscStat (
      ByVal instrumentHandle As ViSession,
      ByVal roscState As ViInt16) As ViStatus
```

Purpose

This function sets the state of the reference oscillator output.

Parameters List

1. **ViSession instrumentHandle [in]**

   This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.

   Default Value: None

2. **ViInt16 roscState [in]**

   Sets the state of the reference oscillator output.

   Valid Values:

   - RSSIAM_ROSC_STAT_OFF (0) - Off
   - RSSIAM_ROSC_STAT_ON (1) - On

   Default Value: 1

Return Value

Returns the status code of this operation.

The meaning of the status code is described in section Error (Status) Codes.

2.2.5.2.4 Reference Oscillator Stat Query

C Function Prototype

```
ViStatus rssiam_roscStat_Q (
      ViSession instrumentHandle,
      ViInt16* roscState);
```

Basic Function Prototype

```
Function rssiam_roscStat_Q (
      ByVal instrumentHandle As ViSession,
      roscState As ViInt16) As ViStatus
```

Purpose

This function returns the state of the reference oscillator output.

Parameters List

1. **ViSession instrumentHandle [in]**

   This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.

   Default Value: None

2. **ViInt16 roscState [out]**

   Returns the state of the reference oscillator output.

   Valid Values:

   - RSSIAM_ROSC_STAT_OFF (0) - Off
   - RSSIAM_ROSC_STAT_ON (1) - On

Return Value

Returns the status code of this operation.

The meaning of the status code is described in section Error (Status) Codes.

2.2.5.3 Flush Error Queue

C Function Prototype

```
ViStatus rssiam_flushErrorQueue (
      ViSession instrumentHandle);
```

Basic Function Prototype

```
Function rssiam_flushErrorQueue (
      ByVal instrumentHandle As ViSession) As ViStatus
```

Purpose

This function deletes the error queue.

Parameters List

1. **ViSession instrumentHandle [in]**

   This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.

   Default Value: None

Return Value

Returns the status code of this operation.

The meaning of the status code is described in section Error (Status) Codes.

2.2.5.4 State Checking

C Function Prototype

```
ViStatus rssiam_errorCheckState (
      ViSession instrumentHandle,
      ViBoolean stateChecking);
```

Basic Function Prototype

```
Function rssiam_errorCheckState (
      ByVal instrumentHandle As ViSession,
      ByVal stateChecking As ViBoolean) As ViStatus
```

Purpose

This function enables (disables) the status checking. The status checking is by default enabled.

Note

When disabled, status checking is not performed and function rssiam_error_query does not provide any error message information.

Parameters List

1. **ViSession instrumentHandle [in]**

   This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.

   Default Value: None

2. **ViBoolean stateChecking [in]**

   This control switches the instrument status checking On or Off.

   Valid Range:

   - VI_FALSE (0) - Off
   - VI_TRUE (1) - On

   Default Value: VI_TRUE (1)

Return Value

Returns the status code of this operation.

The meaning of the status code is described in section Error (Status) Codes.

2.2.5.5 Reset

C Function Prototype	ViStatus rssiam_reset (ViSession instrumentHandle);
Basic Function Prototype	Function rssiam_reset (ByVal instrumentHandle As ViSession) As ViStatus
Purpose	This function resets the instrument to a known state.

Note If this instrument does not support a Reset, this function should return the Warning Code VI_WARN_NSUP_RESET (0x3FFC0102).

Parameters List

1. **ViSession instrumentHandle [in]**

   This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.

   Default Value: None

Return Value

Returns the status code of this operation.

The meaning of the status code is described in section Error (Status) Codes.

2.2.5.6 Self-Test

C Function Prototype	ViStatus rssiam_self_test (ViSession instrumentHandle, ViInt16* selfTestResult, ViChar[] selfTestMessage);
Basic Function Prototype	Function rssiam_self_test (ByVal instrumentHandle As ViSession, selfTestResult As ViInt16, selfTestMessage As ViChar) As ViStatus
Purpose	This function runs the instrument's self test routine and returns the test result(s).

Parameters List

1. **ViSession instrumentHandle [in]**

   This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.

   Default Value: None

2. **ViInt16 selfTestResult [out]**

   This control contains the value returned from the instrument self test. Zero means success. For any other code, see the device's operator's manual.

3. **ViChar[] selfTestMessage [out]**

   This control contains the string returned from the self test. See the device's operation manual for an explanation of the string's contents.

Note The array must contain at least 256 elements ViChar[256].

Return Value

Returns the status code of this operation.

The meaning of the status code is described in section Error (Status) Codes.

2.2.5.7 Error-Query

C Function Prototype

```
ViStatus rssiam_error_query (
      ViSession instrumentHandle,
      ViInt32* errorCode,
      ViChar[] errorMessage);
```

Basic Function Prototype

```
Function rssiam_error_query (
      ByVal instrumentHandle As ViSession,
      errorCode As ViInt32,
      errorMessage As ViChar) As ViStatus
```

Purpose

This function reads an error code from the instrument's error queue.

Note

If this instrument does not support an Error Query, this function should return the Warning Code VI_WARN_NSUP_ERROR_QUERY (0x3FFC0104).

Parameters List

1. **ViSession instrumentHandle [in]**

   This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.

   Default Value: None

2. **ViInt32 errorCode [out]**

   This control returns the error code read from the instrument's error queue.

Note

Error code is represented by an Event Id.

3. **ViChar[] errorMessage [out]**

   This control returns the error message string read from the instrument's error message queue.

Note

The array must contain at least 256 elements ViChar[256].

Return Value

Returns the status code of this operation.

The meaning of the status code is described in section Error (Status) Codes.

2.2.5.8 Error Message

C Function Prototype

```
ViStatus rssiam_error_message (
     ViSession instrumentHandle,
     ViStatus statusCode,
     ViChar[] message);
```

Basic Function Prototype

```
Function rssiam_error_message (
     ByVal instrumentHandle As ViSession,
     ByVal statusCode As ViStatus,
     message As ViChar) As ViStatus
```

Purpose

This function takes the Status Code returned by the instrument driver functions, interprets it and returns as a user readable string.

Parameters List

1. **ViSession instrumentHandle [in]**

   This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.

   Default Value: None

2. **ViStatus statusCode [in]**

   This control accepts the Status Code returned from the instrument driver functions.

   Default Value: 0 - VI_SUCCESS

3. **ViChar[] message [out]**

   This control returns the interpreted Status Code as a user readable message string.

Note

The array must contain at least 256 elements ViChar[256].

Return Value

Returns the status code of this operation.

The meaning of the status code is described in section Error (Status) Codes.

2.2.5.9 Revision Query

C Function Prototype

```
ViStatus rssiam_revision_query (
      ViSession instrumentHandle,
      ViChar[] instrumentDriverRevision,
      ViChar[] firmwareRevision);
```

Basic Function Prototype

```
Function rssiam_revision_query (
      ByVal instrumentHandle As ViSession,
      instrumentDriverRevision As ViChar,
      firmwareRevision As ViChar) As ViStatus
```

Purpose

This function returns the revision numbers of the instrument driver and instrument module firmware, and informs the user with which instrument firmware this revision of the driver is compatible.

Parameters List

1. **ViSession instrumentHandle [in]**

   This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.

   Default Value: None

2. **ViChar[] instrumentDriverRevision [out]**

   This control returns the Instrument Driver Software Revision.

Note The array must contain at least 256 elements ViChar[256].

3. **ViChar[] firmwareRevision [out]**

   This control returns the AM300 module firmware revision.

Note The array must contain at least 256 elements ViChar[256].

Return Value

Returns the status code of this operation.

The meaning of the status code is described in section Error (Status) Codes.

2.2.6 Obsolete

2.2.6.1.1 Trigger Slope Setup

C Function Prototype

```
ViStatus rssiam_trigSlope (
      ViSession instrumentHandle,
      ViInt16 triggerSlope);
```

Basic Function Prototype

```
Function rssiam_trigSlope (
      ByVal instrumentHandle As ViSession,
      ByVal triggerSlope As ViInt16) As ViStatus
```

Purpose

This function sets the trigger slope (edge which the event occurs on).

Parameters List

1. **ViSession instrumentHandle [in]**

   This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.

   Default Value: None

2. **ViInt16 triggerSlope [in]**

   Sets the trigger slope (edge which the event occurs on).

   Valid Values:

   - RSSIAM_TRIGGER_SLOPE_POS (0) - Positive
   - RSSIAM_TRIGGER_SLOPE_NEG (1) - Negative

   Default Value: 0

Return Value

Returns the status code of this operation.

The meaning of the status code is described in section Error (Status) Codes.

2.2.6.1.2 Trigger Slope Query

C Function Prototype

```
ViStatus rssiam_trigSlope_Q (
      ViSession instrumentHandle,
      ViInt16* triggerSlope);
```

Basic Function Prototype

```
Function rssiam_trigSlope_Q (
      ByVal instrumentHandle As ViSession,
      triggerSlope As ViInt16) As ViStatus
```

Purpose

This function returns the trigger slope (edge which the event occurs on).

Parameters List

1. **ViSession instrumentHandle [in]**

   This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.

   Default Value: None

2. **ViInt16 triggerSlope [out]**

   Returns the trigger slope (edge which the event occurs on).

   Valid Values:

   - RSSIAM_TRIGGER_SLOPE_POS (0) - Positive
   - RSSIAM_TRIGGER_SLOPE_NEG (1) - Negative

Return Value

Returns the status code of this operation.

The meaning of the status code is described in section Error (Status) Codes.

2.2.6.1.3 Output Summary Setup

C Function Prototype

```
ViStatus rssiam_outpSum (
      ViSession instrumentHandle,
      ViInt16 outputSummary);
```

Basic Function Prototype

```
Function rssiam_outpSum (
      ByVal instrumentHandle As ViSession,
      ByVal outputSummary As ViInt16) As ViStatus
```

Purpose

This functions sets whether the output signal I = (Q + I) is On or Off.

Parameters List

1. **ViSession instrumentHandle [in]**

   This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.

   Default Value: None

2. **ViInt16 outputSummary [in]**

   Sets whether the output signal I = (Q + I) is On or Off.

   Valid Values:

   - RSSIAM_OUTPUT_SUM_OFF (0) - Off
   - RSSIAM_OUTPUT_SUM_ON (1) - On

   Default Value: 0

Return Value

Returns the status code of this operation.

The meaning of the status code is described in section Error (Status) Codes.

2.2.6.1.4 Output Summary Query

C Function Prototype

```
ViStatus rssiam_outpSum_Q (
      ViSession instrumentHandle,
      ViInt16* outputSummary);
```

Basic Function Prototype

```
Function rssiam_outpSum_Q (
      ByVal instrumentHandle As ViSession,
      outputSummary As ViInt16) As ViStatus
```

Purpose

This function returns whether the output signal I = (Q + I) is On or Off.

Parameters List

1. **ViSession instrumentHandle [in]**

   This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.

   Default Value: None

2. **ViInt16 outputSummary [out]**

   Returns whether the output signal I = (Q + I) is On or Off.

   Valid Values:

   - RSSIAM_OUTPUT_SUM_OFF (0) - Off
   - RSSIAM_OUTPUT_SUM_ON (1) - On

Return Value

Returns the status code of this operation.

The meaning of the status code is described in section Error (Status) Codes.

2.2.6.1.5 Gate Time Setup

C Function Prototype

```
ViStatus rssiam_gateTime (
      ViSession instrumentHandle,
      ViReal64 gateTime);
```

Basic Function Prototype

```
Function rssiam_gateTime (
      ByVal instrumentHandle As ViSession,
      ByVal gateTime As ViReal64) As ViStatus
```

Purpose

This function sets the internal gate time (the time for which generator outputs a continuous waveform).

Parameters List

1. **ViSession instrumentHandle [in]**

   This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.

   Default Value: None

2. **ViReal64 gateTime [in]**

   Sets the internal gate time.

   Valid Range: 0.0 s to 9999.0 s

   Default Value: 100.0e-9 s

Return Value

Returns the status code of this operation.

The meaning of the status code is described in section Error (Status) Codes.

2.2.6.1.6 Gate Time Query

C Function Prototype

```
ViStatus rssiam_gateTime_Q (
      ViSession instrumentHandle,
      ViReal64* gateTime);
```

Basic Function Prototype

```
Function rssiam_gateTime_Q (
      ByVal instrumentHandle As ViSession,
      gateTime As ViReal64) As ViStatus
```

Purpose

This function returns the internal gate time (the time for which generator outputs a continuous waveform).

Parameters List

1. **ViSession instrumentHandle [in]**

   This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.

   Default Value: None

2. **ViReal64 gateTime [out]**

   Returns the internal gate time.

Return Value

Returns the status code of this operation.

The meaning of the status code is described in section Error (Status) Codes.

2.2.7 Close

C Function Prototype

```
ViStatus rssiam_close (
     ViSession instrumentHandle);
```

Basic Function Prototype

```
Function rssiam_close (
     ByVal instrumentHandle As ViSession) As ViStatus
```

Purpose

This function closes session to the instrument.

Note

The instrument must be reinitialized to use it again.

Parameters List

1. **ViSession instrumentHandle [in]**

   This control accepts the Instrument Handle returned by the Initialize function to select the desired instrument driver session.

   Default Value: None

Return Value

Returns the status code of this operation.

The meaning of the status code is described in section Error (Status) Codes.

2.3 Error (Status) Codes

Description The status code either indicates success or describes an error or warning condition. You are able to examine the status code from each call to an instrument driver function to determine if an error occurred. To obtain a text description of the status code, call the **rssifs_error_message** function.

Error Queue Error messages produced by device are queued and can be queried calling the **rssifs_error_query** function. Up to 512 messages is queued using FILO (First-In Last-Out) method. If the error queue is full, error RSSIFS_ERROR_QUEUE_OVERFLOW (0xBFFC09F8) is produced. All new upcoming errors are lost then.

Status Code The general meaning of the status code is as follows:

Value	Meaning
0	Success
Positive Values	Warning
Negative Values	Errors

Table 2-2: The Meaning of the Status Code

Details This instrument driver also returns errors and warnings defined by other sources. The following table defines the ranges of additional status codes that this driver can return. The table lists the different include files that contain the defined constants for the particular status codes:

Numeric Range (in Hex)	Status Code	Types
3FFF0000 to 3FFFFFFF	VISA	Warnings
3FFC0000 to 3FFCFFFF	VXIPnP	Driver Warnings
BFFF0000 to BFFFFFFF	VISA	Errors
BFFC0000 to BFFCFFFF	VXIPnP	Driver Errors

Table 2-3: Status Codes

List of all known instrument driver warnings codes:

Value	Text Description
0x3FFC0101	WARNING: ID Query not supported.
0x3FFC0102	WARNING: Reset not supported.
0x3FFC0103	WARNING: Self-test not supported.
0x3FFC0104	WARNING: Error Query not supported.
0x3FFC0105	WARNING: Revision Query not supported.

Table 2-4: Warnings Codes

List of all known instrument driver errors codes:

Value	Text Description
0xBFFC0001	ERROR: Parameter 1 out of range.
0xBFFC0002	ERROR: Parameter 2 out of range.
0xBFFC0003	ERROR: Parameter 3 out of range.
0xBFFC0004	ERROR: Parameter 4 out of range.
0xBFFC0005	ERROR: Parameter 5 out of range.
0xBFFC0006	ERROR: Parameter 6 out of range.
0xBFFC0007	ERROR: Parameter 7 out of range.
0xBFFC0008	ERROR: Parameter 8 out of range.
0xBFFC0011	ERROR: Identification query failed.
0xBFFC0012	ERROR: Interpreting instrument response.
0xBFFC0800	ERROR: Opening the specified file.
0xBFFC0801	ERROR: Writing to the specified file.
0xBFFC0803	ERROR: Interpreting the instrument's response.
0xBFFC0809	ERROR: Parameter 9 out of range.
0xBFFC080A	ERROR: Parameter 10 out of range.
0xBFFC080B	ERROR: Parameter 11 out of range.
0xBFFC080C	ERROR: Parameter 12 out of range.
0xBFFC080D	ERROR: Parameter 13 out of range.
0xBFFC080E	ERROR: Parameter 14 out of range.
0xBFFC080F	ERROR: Parameter 15 out of range.
0xBFFC09F0	ERROR: Instrument status error.
0xBFFC09F1	ERROR: Instrument configuration error.
0xBFFC09F2	ERROR: Required instrument's option is not installed.
0xBFFC09F3	ERROR: Required instrument model is not connected.
0xBFFC09F4	ERROR: Selected register name is not supported by the instrument.
0xBFFC09F5	ERROR: Invalid value (value out of range).
0xBFFC09F6	ERROR: Range table for selected register name is not available.
0xBFFC09F7	ERROR: NULL pointer passed as parameter.
0xBFFC09F8	ERROR: The error queue is overflowed.
0xBFFC09F9	ERROR: Data not available.
0xBFFC09FA	ERROR: Data are corrupted.
0xBFFC09FB	ERROR: Settings conflict. Passed parameter does not match with the current instrument's settings.
0xBFFC09FC	ERROR: Function or parameter is for any reason reserved.
0xBFFC09FD	ERROR: Current measurement has been aborted.
0xBFFC09FE	ERROR: Error on execution of the function.

Table 2-5: Error Codes

2.4 Execution Timeout

Description

Timeout value specifies the minimum time to use (in milliseconds) when accessing the device associated with the given session. A timeout value means that operations should wait for the device to respond at least defined amount of time. The timeout value is set via **rssifs_setTimeOut** function. The actual timeout value is retrieved via **rssifs_getTimeOut** function.

Note

Notice that the actual timeout value used by the driver may be higher than the requested one.

2.5 Alphabetical List of Functions

R

rssiam_abortTrigger 133
rssiam_amDept 65
rssiam_amDept_Q 66
rssiam_amIntFreq 68
rssiam_amIntFreq_Q 68
rssiam_amIntFunc 66
rssiam_amIntFunc_Q 67
rssiam_amState 69
rssiam_amState_Q 69
rssiam_appl 30
rssiam_appl_Q 32
rssiam_bmIntRate 120
rssiam_bmIntRate_Q 121
rssiam_bmNcyc 119
rssiam_bmNcyc_Q 119
rssiam_bmSour 121
rssiam_bmSour_Q 122
rssiam_bmStat 123
rssiam_bmStat_Q 123
rssiam_close 146
rssiam_dataArbPoin 93
rssiam_dataArbPoin_Q 94
rssiam_dataArbRate 96
rssiam_dataArbRate_Q 97
rssiam_dataArbRateMode 97
rssiam_dataArbRateMode_Q 98
rssiam_dataArbStart 94
rssiam_dataArbStart_Q 95
rssiam_dataArbStop 95
rssiam_dataArbStop_Q 96
rssiam_dataDacVolatile 99
rssiam_dataVolatile 98
rssiam_error_message 141
rssiam_error_query 140
rssiam_errorCheckState 138
rssiam_extFilter 57
rssiam_extFilter_Q 58
rssiam_flushErrorQueue 138
rssiam_fmDev 70
rssiam_fmDev_Q 71
rssiam_fmIntFreq 73
rssiam_fmIntFreq_Q 73
rssiam_fmIntFunc 71
rssiam_fmIntFunc_Q 72
rssiam_fmStat 74
rssiam_fmStat_Q 74
rssiam_freq 37
rssiam_freq_Q 38
rssiam_freqCenter 102
rssiam_freqCenter_Q 103
rssiam_freqSpan 104
rssiam_freqSpan_Q 104
rssiam_freqStar 100
rssiam_freqStar_Q 100

rssiam_freqStop ... 101
rssiam_freqStop_Q ... 102
rssiam_fskFreq ... 81
rssiam_fskFreq_Q ... 82
rssiam_fskIntRate ... 83
rssiam_fskIntRate_Q ... 84
rssiam_fskSour ... 84
rssiam_fskSour_Q ... 85
rssiam_fskStat ... 86
rssiam_fskStat_Q ... 86
rssiam_funcShap ... 36
rssiam_funcShap_Q ... 37
rssiam_gateFunction ... 127
rssiam_gateFunction_Q ... 127
rssiam_gateLength ... 128
rssiam_gateLength_Q ... 128
rssiam_gatePolarity ... 130
rssiam_gatePolarity_Q ... 130
rssiam_gateSource ... 129
rssiam_gateSource_Q ... 129
rssiam_gateTime ... 145
rssiam_gateTime_Q ... 145
rssiam_init ... 28
rssiam_mkrFreq ... 111
rssiam_mkrFreq_Q ... 112
rssiam_mkrStat ... 112
rssiam_mkrStat_Q ... 113
rssiam_mPoints ... 64
rssiam_mPoints_Q ... 64
rssiam_outpAddition ... 58
rssiam_outpAddition_Q ... 59
rssiam_outpChannel ... 33
rssiam_outpChannel_Q ... 34
rssiam_outpChannelCoupl ... 34
rssiam_outpChannelCoupl_Q ... 35
rssiam_outpFilter ... 55
rssiam_outpFilter_Q ... 56
rssiam_outpFilterType ... 56
rssiam_outpFilterType_Q ... 57
rssiam_outpLoad ... 49
rssiam_outpLoad_Q ... 50
rssiam_outpLoadUser ... 50
rssiam_outpSetup ... 53
rssiam_outpSetup_Q ... 54
rssiam_outpStartPhase ... 54
rssiam_outpStartPhase_Q ... 55
rssiam_outpSum ... 144
rssiam_outpSum_Q ... 144
rssiam_outpSync ... 52
rssiam_outpSync_Q ... 53
rssiam_pmDev ... 75
rssiam_pmDev_Q ... 75
rssiam_pmIntFreq ... 77
rssiam_pmIntFreq_Q ... 78
rssiam_pmIntFunc ... 76
rssiam_pmIntFunc_Q ... 77
rssiam_pmStat ... 79
rssiam_pmStat_Q ... 80
rssiam_pskIntRate ... 89
rssiam_pskIntRate_Q ... 89
rssiam_pskPhase ... 87
rssiam_pskPhase_Q ... 88

rssiam_pskSour 90
rssiam_pskSour_Q 91
rssiam_pskStat 91
rssiam_pskStat_Q 92
rssiam_pulsDcyc 47
rssiam_pulsDcyc_Q 47
rssiam_pulsePeriod 59
rssiam_pulsePeriod_Q 61
rssiam_pulsePolarity 63
rssiam_pulsePolarity_Q 63
rssiam_pulseWidth 61
rssiam_pulseWidth_Q 62
rssiam_rampSymm 48
rssiam_rampSymm_Q 48
rssiam_reset 139
rssiam_revision_query 142
rssiam_roscSour 136
rssiam_roscSour_Q 136
rssiam_roscStat 137
rssiam_roscStat_Q 137
rssiam_self_test 139
rssiam_sendTrigger 132
rssiam_sweDirection 105
rssiam_sweDirection_Q 105
rssiam_swePoin 106
rssiam_swePoin_Q 106
rssiam_sweSpac 107
rssiam_sweSpac_Q 107
rssiam_sweStat 109
rssiam_sweStat_Q 110
rssiam_sweTime 108
rssiam_sweTime_Q 108
rssiam_syncPolarity 125
rssiam_syncPolarity_Q 126
rssiam_syncType 124
rssiam_syncType_Q 124
rssiam_timeOut 134
rssiam_timeOut_Q 135
rssiam_trigDelay 116
rssiam_trigDelay_Q 116
rssiam_trigFrequency 117
rssiam_trigFrequency_Q 117
rssiam_trigPeriod 118
rssiam_trigPeriod_Q 118
rssiam_trigPolarity 115
rssiam_trigPolarity_Q 115
rssiam_trigSlope 143
rssiam_trigSlope_Q 143
rssiam_trigSour 114
rssiam_trigSour_Q 114
rssiam_volt 39
rssiam_volt_Q 39
rssiam_voltHigh 43
rssiam_voltHigh_Q 43
rssiam_voltLow 44
rssiam_voltLow_Q 45
rssiam_voltOffs 40
rssiam_voltOffs_Q 42
rssiam_voltUnit 45
rssiam_voltUnit_Q 46

2.6 Contacts

Technical support – where and when you need it

For quick, expert help with any Rohde & Schwarz equipment, contact one of our Customer Support Centers. A team of highly qualified engineers provides telephone support and will work with you to find a solution to your query on any aspect of the operation, programming or applications of Rohde & Schwarz equipment.

Up-to-date information and upgrades

To keep your Rohde & Schwarz equipment always up-to-date, lease subscribe to our electronic newsletter at http://www.rohde-schwarz.com/www/response.nsf/newsletterpreselection
or request the desired information and upgrades via email from your Customer Support Center (addresses see below).

Feedback

We want to know if we are meeting your support needs. If you have any comments please email us and let us know CustomerSupport.Feedback@rohde-schwarz.com.

USA & Canada

Monday to Friday (except US public holidays)
8:00 AM – 8:00 PM Eastern Standard Time (EST)

Tel. from USA	888-test-rsa (888-837-8772) (opt 2)
From outside USA	+1 410 910 7800 (opt 2)
Fax	+1 410 910 7801
E-mail	Customer.Support@rsa.rohde-schwarz.com

East Asia

Monday to Friday (except Singaporean public holidays)
8:30 AM – 6:00 PM Singapore Time (SGT)

Tel.	+65 6 513 0488
Fax	+65 6 846 1090
E-mail	Customersupport.asia@rohde-schwarz.com

Rest of the World

Monday to Friday (except German public holidays)
08:00 – 17:00 Central European Time (CET)

Tel. from Europe	+49 (0) 180 512 42 42
From outside Europe	+49 89 4129 13776
Fax	+49 (0) 89 41 29 637 78
E-mail	CustomerSupport@rohde-schwarz.com

2.7 Remote Control Programming Examples

Description

All the programming examples are written in ANSII C language. Examples are commented; execution results are appended after the source code and when needed trace data are displayed using embedded illustration pictures, where the pictures are not part of the examples.

2.7.1.1.1 Error Handling & Time Profiling

Source Code:

```
/***************************************************************************
*
 *
 * Title:   Error Handling & Time Profiling
 *
 * Purpose: This example shows basic principles of error handling.
 *

****************************************************************************
/

#include <ansi_c.h>
#include "rssiam.h"

/*** Macros & definitions
***************************************************/

#define CHECKERR( fCal ) \
    { \
        fCalStartTime = clock();\
        error = (fCal);\
        status = error;\
        printf(" Line %ld (%ld ms): %s\n", __LINE__,\
            (fCalTime = (clock() - fCalStartTime)), #fCal);\
        if (error != VI_SUCCESS)\
            {\
            rssiam_error_message (io, error, error_message);\
            printf("\tFunction Call Status: 0x%08X, %s\n", error,
error_message);\
            rssiam_error_query(io, &error, error_message);\
            printf("\tInstrument Error: 0x%X, %s\n", error, error_message);\
            }\
    }

#define RESOURCE_NAME   "USB::0xAAD::0x5::100011"   // Resource name

/*** Main
*******************************************************************/

int main (int argc, char *argv[])
{
    ViStatus    error               = VI_SUCCESS,
                status              = VI_SUCCESS;
    clock_t     fCalTime            = 0,
                fCalStartTime       = 0;
    ViChar      error_message[256];
    ViSession   io;
```

```
/* Define remote connection parameters (as default values) */

    ViBoolean   IDQuery             = VI_TRUE,
                resetDevice         = VI_TRUE;
    ViRsrc      resourceName        = RESOURCE_NAME;

    CHECKERR (rssiam_init (resourceName, IDQuery, resetDevice, &io));

    printf (

    "\n --- Error Checking ---\n\n"
    "\tThis example uses macro CHECKERR to cover error handling.\n"
    "\tBasic principle of error handling is as follows:\n\n"

    "\t(1) status code (status) is returned by function call (fCal)\n"
    "\t(2) if status code is <> VI_SUCCESS (0):\n"
    "\t    - translate status code to message using rssiam_error_message"
    " function\n"
    "\t    - call rssiam_error_query function to get more information from
error"
    " queue\n"
    "\t(3) if status code is equal to VI_SUCCESS (0):\n"
    "\t    - function call succeed\n\n"

    );

    /* Correct function call */
    CHECKERR (rssiam_freq (io, 1000.0));
    /* Passed wrong data to generate error */
    CHECKERR (rssiam_freq (io, 9.9e9));

    printf (

    "\n --- Disable Warning & Error Checking ---\n\n"
    "\tWarning & Error checking can be disabled by calling function"
    " rssiam_errorCheckState.\n"
    "\tSee description:\n\n"

    "\t(1) When disabled, status checking is not performed and function\n"
    "\t    rssiam_error_query do not provide any error message\n"
    "\t    information.\n"

    "\t(2) When disabled, internal event handling mechanism (interrupt\n"
    "\t    pipe checking & control transfer handshake checking) is also\n"
    "\t    disabled.\n"

    "\t(3) When disabled, interaction in between instrument driver on\n"
    "\t    the host computer and device's firmware is affected.
Performance\n"
    "\t    of the instrument driver calls might increase, but behavior is\n"
    "\t    not fully predictable (synchronization and timing mechanism is\n"
    "\t    disabled).\n\n"

    );

    printf (

    "\n --- Performance Improvement ---\n\n"

    "\tWhen the error checking is disabled, performance might increase:\n\n"

    );
    CHECKERR (rssiam_freq (io, 1000.0));
    CHECKERR (rssiam_freq (io, 1000.0));
    CHECKERR (rssiam_errorCheckState (io, VI_FALSE));
    CHECKERR (rssiam_freq (io, 1000.0));
    CHECKERR (rssiam_freq (io, 1000.0));

    printf (

    "\n\tWARNING: Use rssiam_errorCheckState function with care!\n\n"

    );

    CHECKERR (rssiam_close (io));

    return 0;
}
```

Execution Result:

```
 Line 50 (8561 ms): rssiam_init (resourceName, IDQuery, resetDevice, &io)

 --- Error Checking ---

      This example uses macro CHECKERR to cover error handling.
      Basic principle of error handling is as follows:

      (1) status code (status) is returned by function call (fCal)
      (2) if status code is <> VI_SUCCESS (0):
          - translate status code to message using rssiam_error_message
function
          - call rssiam_error_query function to get more information from
error queue
      (3) if status code is equal to VI_SUCCESS (0):
          - function call succeed

 Line 70 (21 ms): rssiam_freq (io, 1000.0)
 Line 72 (1 ms): rssiam_freq (io, 9.9e9)
      Function Call Status: 0xBFFC09F5, ERROR: Invalid value (value out of
range).
      Instrument Error: 0x0, No error.

 --- Disable Warning & Error Checking ---

      Warning & Error checking can be disabled by calling function
      rssiam_errorCheckState.

      See description:

      (1) When disabled, status checking is not performed and function
          rssiam_error_query do not provide any error message
          information.
      (2) When disabled, internal event handling mechanism (interrupt
          pipe checking & control transfer handshake checking) is also
          disabled.
      (3) When disabled, interaction in between instrument driver on
          the host computer and device's firmware is affected. Performance
          of the instrument driver calls might increase, but behavior is
          not fully predictable (synchronization and timing mechanism is
          disabled).


 --- Performance Improvement ---
```

```
     When the error checking is disabled, performance might increase:

Line 105 (23 ms): rssiam_freq (io, 1000.0)
Line 106 (25 ms): rssiam_freq (io, 1000.0)
Line 107 (0 ms): rssiam_errorCheckState (io, VI_FALSE)
Line 108 (1 ms): rssiam_freq (io, 1000.0)
Line 109 (1 ms): rssiam_freq (io, 1000.0)

     WARNING: Use rssiam_errorCheckState function with care!

Line 117 (0 ms): rssiam_close (io)
```

2.7.1.1.2 Loading Arbitrary data to Channel 1 DAC data and Channel 2 volatile data (sample accurate)

Source Code:

```
/***************************************************************************
*
 *
 * Title:   Load Arbitrary data in Sample Accurate mode
 *
 * Purpose: This example shows how to load Arbitrary data to Channel 1 DAC
 *          data and Channel 2 volatile data in sample accurate mode.
 *

****************************************************************************
/

#include <ansi_c.h>
#include "rssiam.h"

/*** Macros & definitions
***************************************************/

#define CHECKERR( fCal ) \
    { \
        fCalStartTime = clock();\
        error = (fCal);\
        status = error;\
        printf(" Line %ld (%ld ms): %s\n", __LINE__,\
            (fCalTime = (clock() - fCalStartTime)), #fCal);\
        if (error != VI_SUCCESS)\
            {\
            rssiam_error_message (io, error, error_message);\
            printf("\tFunction Call Status: 0x%08X, %s\n", error,
error_message);\
            rssiam_error_query(io, &error, error_message);\
            printf("\tInstrument Error: 0x%X, %s\n", error, error_message);\
            }\
    }

#define RESOURCE_NAME   "USB::0xAAD::0x5::100011" // Resource name

/*** Main
*******************************************************************/

int main (int argc, char *argv[])
{
    ViStatus    error               = VI_SUCCESS,
                status              = VI_SUCCESS;
    clock_t     fCalTime            = 0,
                fCalStartTime       = 0;
    ViChar      error_message[256];
    ViSession   io;

    ViReal64    volatile_data[1024],
                resolution          = 0.0,
                grad                = 0.0,
                rad                 = 0.0;
    ViInt16     DAC_data[1024],
                samples             = 0,
                max                 = 0,
                min                 = 0,
                i                   = 0;

    /* Define remote connection parameters (as default values) */

    ViBoolean   IDQuery             = VI_TRUE,
                resetDevice         = VI_TRUE;
    ViRsrc      resourceName        = RESOURCE_NAME;

    CHECKERR (rssiam_init (resourceName, IDQuery, resetDevice, &io));

    printf (

    "\n --- Load Arb Data --------------------------------------------------
---\n\n"
    "\t - Sample accurate mode\n\n"
    "\t CH1:\n\n"
    "\t - DAC Data\n\n"
    "\t CH2:\n\n"
    "\t - Volatile Data\n"
    "\n ---------------------------------------------------------------------
----\n\n"

    );

    /* Calculate Arb data (sine and cosine wave) */

    samples     = 1024;
    max         = 16383;                // DAC max
    min         = -16383;               // DAC min
    resolution  = (2.0 / (max - min));  // Volatile data range is from -1.0
to 1.0

    for (i = 0; i < samples; i++)
        {
        grad = (360.0 / (float)(samples)) * (float)i;
        rad = grad * (3.141592654/180.0);
        DAC_data[i] = (sin(rad)) / resolution;
        volatile_data[i] = cos(rad);
        }

    /* --- CH1 Function --- */
    CHECKERR (rssiam_outpChannel (io, RSSIAM_CH1));
    CHECKERR (rssiam_funcShap (io, RSSIAM_OUTPUT_FUNC_USER));

    /* --- CH2 Function --- */
    CHECKERR (rssiam_outpChannel (io, RSSIAM_CH2));
    CHECKERR (rssiam_funcShap (io, RSSIAM_OUTPUT_FUNC_USER));

    /* --- CH1 Arb Data --- */
    CHECKERR (rssiam_outpChannel (io, RSSIAM_CH1));
    CHECKERR (rssiam_dataDacVolatile (io, samples, DAC_data));

    /* --- CH2 Arb Data --- */
    CHECKERR (rssiam_outpChannel (io, RSSIAM_CH2));
    CHECKERR (rssiam_dataVolatile (io, samples, volatile_data));

    /* Set Arb parameters */
    CHECKERR (rssiam_dataArbRateMode (io, RSSIAM_ARB_RATE_MODE_ACCURATE));
    CHECKERR (rssiam_dataArbRate (io, 1.0e6));
    CHECKERR (rssiam_dataArbPoin (io, samples));

    /* --- CH1 State --- */
    CHECKERR (rssiam_outpChannel (io, RSSIAM_CH1));
    CHECKERR (rssiam_outpSetup (io, RSSIAM_OUTPUT_SETUP_ON));

    /* --- CH2 State --- */
    CHECKERR (rssiam_outpChannel (io, RSSIAM_CH2));
    CHECKERR (rssiam_outpSetup (io, RSSIAM_OUTPUT_SETUP_ON));

    printf ("\n");

    CHECKERR (rssiam_close (io));

    return 0;
}
```

Execution Result:

```
 Line 60 (8571 ms): rssiam_init (resourceName, IDQuery, resetDevice, &io)

 --- Load Arb Data -----------------------------------------------------

       - Sample accurate mode

       CH1:

       - DAC Data

       CH2:

       - Volatile Data

 ------------------------------------------------------------------------

 Line 90 (0 ms): rssiam_outpChannel (io, RSSIAM_CH1)
 Line 91 (1604 ms): rssiam_funcShap (io, RSSIAM_OUTPUT_FUNC_USER)
 Line 94 (0 ms): rssiam_outpChannel (io, RSSIAM_CH2)
 Line 95 (1599 ms): rssiam_funcShap (io, RSSIAM_OUTPUT_FUNC_USER)
 Line 98 (0 ms): rssiam_outpChannel (io, RSSIAM_CH1)
 Line 99 (1543 ms): rssiam_dataDacVolatile (io, samples, DAC_data)
 Line 102 (0 ms): rssiam_outpChannel (io, RSSIAM_CH2)
 Line 103 (1543 ms): rssiam_dataVolatile (io, samples, volatile_data)
 Line 106 (23 ms): rssiam_dataArbRateMode (io,
RSSIAM_ARB_RATE_MODE_ACCURATE)
 Line 107 (48 ms): rssiam_dataArbRate (io, 1.0e6)
 Line 108 (24 ms): rssiam_dataArbPoin (io, samples)
 Line 111 (1 ms): rssiam_outpChannel (io, RSSIAM_CH1)
 Line 112 (55 ms): rssiam_outpSetup (io, RSSIAM_OUTPUT_SETUP_ON)
 Line 115 (0 ms): rssiam_outpChannel (io, RSSIAM_CH2)
 Line 116 (55 ms): rssiam_outpSetup (io, RSSIAM_OUTPUT_SETUP_ON)

 Line 120 (1 ms): rssiam_close (io)
```

2.7.1.1.3 Loading Arbitrary data to Channel 1 DAC data and Channel 2 volatile data (normal)

Source Code:

```
/***************************************************************************
*
 *
 * Title:   Load Arbitrary data in Normal mode
 *
 * Purpose: This example shows how to load Arbitrary data to Channel 1 DAC
 *          data and Channel 2 volatile data in normal mode.
 *

****************************************************************************
/

#include <ansi_c.h>
#include "rssiam.h"

/*** Macros & definitions
**************************************************/

#define CHECKERR( fCal ) \
    { \
        fCalStartTime = clock();\
        error = (fCal);\
        status = error;\
        printf(" Line %ld (%ld ms): %s\n", __LINE__,\
            (fCalTime = (clock() - fCalStartTime)), #fCal);\
        if (error != VI_SUCCESS)\
            {\
            rssiam_error_message (io, error, error_message);\
            printf("\tFunction Call Status: 0x%08X, %s\n", error,
error_message);\
            rssiam_error_query(io, &error, error_message);\
            printf("\tInstrument Error: 0x%X, %s\n", error, error_message);\
            }\
    }

#define RESOURCE_NAME   "USB::0xAAD::0x5::100011" // Resource name

/*** Main
*****************************************************************/

int main (int argc, char *argv[])
{
    ViStatus    error               = VI_SUCCESS,
                status              = VI_SUCCESS;
    clock_t     fCalTime            = 0,
                fCalStartTime       = 0;
    ViChar      error_message[256];
    ViSession   io;

    ViReal64    volatile_data[1024],
                resolution          = 0.0,
                grad                = 0.0,
                rad                 = 0.0;
    ViInt16     DAC_data[1024],
                samples             = 0,
                max                 = 0,
            min                     = 0,
            i                       = 0;

    /* Define remote connection parameters (as default values) */

    ViBoolean   IDQuery             = VI_TRUE,
                resetDevice         = VI_TRUE;
    ViRsrc      resourceName        = RESOURCE_NAME;

    CHECKERR (rssiam_init (resourceName, IDQuery, resetDevice, &io));

    printf (

    "\n --- Load Arb Data --------------------------------------------------
---\n\n"
    "\t - Normal mode\n\n"
    "\t CH1:\n\n"
    "\t - DAC Data\n\n"
    "\t CH2:\n\n"
    "\t - Volatile Data\n"
    "\n --------------------------------------------------------------------
----\n\n"

    );

    /* Calculate Arb data (sine and cosine wave) */

    samples     = 1024;
    max         = 16383;                // DAC max
    min         = -16383;               // DAC min
    resolution  = (2.0 / (max - min));  // Volatile data range is from -1.0
to 1.0

    for (i = 0; i < samples; i++)
        {
        grad = (360.0 / (float)(samples)) * (float)i;
        rad = grad * (3.141592654/180.0);
        DAC_data[i] = (sin(rad)) / resolution;
        volatile_data[i] = cos(rad);
        }

    /* --- CH1 Function --- */
    CHECKERR (rssiam_outpChannel (io, RSSIAM_CH1));
    CHECKERR (rssiam_funcShap (io, RSSIAM_OUTPUT_FUNC_USER));

    /* --- CH2 Function --- */
    CHECKERR (rssiam_outpChannel (io, RSSIAM_CH2));
    CHECKERR (rssiam_funcShap (io, RSSIAM_OUTPUT_FUNC_USER));

    /* --- CH1 Arb Data --- */
    CHECKERR (rssiam_outpChannel (io, RSSIAM_CH1));
    CHECKERR (rssiam_dataDacVolatile (io, samples, DAC_data));

    /* --- CH2 Arb Data --- */
    CHECKERR (rssiam_outpChannel (io, RSSIAM_CH2));
    CHECKERR (rssiam_dataVolatile (io, samples, volatile_data));

    /* Set Arb parameters */
    CHECKERR (rssiam_dataArbRateMode (io, RSSIAM_ARB_RATE_MODE_NORMAL));
    CHECKERR (rssiam_freq (io, 1000.0));

    /* --- CH1 State --- */
    CHECKERR (rssiam_outpChannel (io, RSSIAM_CH1));
    CHECKERR (rssiam_outpSetup (io, RSSIAM_OUTPUT_SETUP_ON));

    /* --- CH2 State --- */
    CHECKERR (rssiam_outpChannel (io, RSSIAM_CH2));
    CHECKERR (rssiam_outpSetup (io, RSSIAM_OUTPUT_SETUP_ON));

    printf ("\n");

    CHECKERR (rssiam_close (io));

    return 0;
}
```

Execution Result:

```
 Line 60 (8523 ms): rssiam_init (resourceName, IDQuery, resetDevice, &io)

 --- Load Arb Data -----------------------------------------------------

       - Normal mode

       CH1:

       - DAC Data

       CH2:

       - Volatile Data

 -----------------------------------------------------------------------

 Line 90 (0 ms): rssiam_outpChannel (io, RSSIAM_CH1)
 Line 91 (1598 ms): rssiam_funcShap (io, RSSIAM_OUTPUT_FUNC_USER)
 Line 94 (10 ms): rssiam_outpChannel (io, RSSIAM_CH2)
 Line 95 (1609 ms): rssiam_funcShap (io, RSSIAM_OUTPUT_FUNC_USER)
 Line 98 (0 ms): rssiam_outpChannel (io, RSSIAM_CH1)
 Line 99 (1528 ms): rssiam_dataDacVolatile (io, samples, DAC_data)
 Line 102 (0 ms): rssiam_outpChannel (io, RSSIAM_CH2)
 Line 103 (1539 ms): rssiam_dataVolatile (io, samples, volatile_data)
 Line 106 (22 ms): rssiam_dataArbRateMode (io, RSSIAM_ARB_RATE_MODE_NORMAL)
 Line 107 (33 ms): rssiam_freq (io, 1000.0)
 Line 110 (0 ms): rssiam_outpChannel (io, RSSIAM_CH1)
 Line 111 (56 ms): rssiam_outpSetup (io, RSSIAM_OUTPUT_SETUP_ON)
 Line 114 (0 ms): rssiam_outpChannel (io, RSSIAM_CH2)
 Line 115 (55 ms): rssiam_outpSetup (io, RSSIAM_OUTPUT_SETUP_ON)

 Line 119 (0 ms): rssiam_close (io)
```

2.7.1.1.4 Setting Sweep Mode

Source Code:

```
/***************************************************************************
*
 *
 * Title:   Setting of Sweep Mode
 *
 * Purpose: This example shows how to set frequency sweep mode.
 *

****************************************************************************
/

#include <ansi_c.h>
#include "rssiam.h"

/*** Macros & definitions
****************************************************/

#define CHECKERR( fCal ) \
    { \
        fCalStartTime = clock();\
        error = (fCal);\
        status = error;\
        printf(" Line %ld (%ld ms): %s\n", __LINE__,\
            (fCalTime = (clock() - fCalStartTime)), #fCal);\
        if (error != VI_SUCCESS)\
            {\
            rssiam_error_message (io, error, error_message);\
            printf("\tFunction Call Status: 0x%08X, %s\n", error,
error_message);\
            rssiam_error_query(io, &error, error_message);\
            printf("\tInstrument Error: 0x%X, %s\n", error, error_message);\
            }\
    }

#define RESOURCE_NAME   "USB::0xAAD::0x5::100011" // Resource name

/*** Main
******************************************************************/

int main (int argc, char *argv[])
{
    ViStatus    error               = VI_SUCCESS,
                status              = VI_SUCCESS;
    clock_t     fCalTime            = 0,
                fCalStartTime       = 0;
    ViChar      error_message[256];
    ViSession   io;

    /* Define remote connection parameters (as default values) */

    ViBoolean   IDQuery             = VI_TRUE,
                resetDevice         = VI_TRUE;
    ViRsrc      resourceName        = RESOURCE_NAME;

    CHECKERR (rssiam_init (resourceName, IDQuery, resetDevice, &io));

    printf (
    "\n --- Set Frequency Sweep Mode ---------------------------------------
----\n\n"
    "\t - Function: Sine\n"
    "\t - Start Frequency: 1.0 kHz\n"
    "\t - Stop Frequency: 10.0 kHz\n"
    "\t - Sweep Direction: Down\n"
    "\t - Sweep Spacing: Log\n"
    "\t - Sweep Time: 1.0 s\n"
    "\t - Sweep Points: 100\n"
    "\n --------------------------------------------------------------------
----\n\n"

    );

    /* Output setup */
    CHECKERR (rssiam_funcShap (io, RSSIAM_OUTPUT_FUNC_SIN));
    CHECKERR (rssiam_outpSetup (io, RSSIAM_OUTPUT_SETUP_ON));

    /* Sweep setup */
    CHECKERR (rssiam_freqStar (io, 1.0e3));
    CHECKERR (rssiam_freqStop (io, 10.0e3));
    CHECKERR (rssiam_sweDirection (io, RSSIAM_SWEEP_DIRECTION_DOWN));
    CHECKERR (rssiam_swePoin (io, 100));
    CHECKERR (rssiam_sweSpac (io, RSSIAM_SWEEP_SPAC_LOG));
    CHECKERR (rssiam_sweTime (io, 1.0));
    CHECKERR (rssiam_sweStat (io, RSSIAM_SWEEP_STAT_ON));

    printf ("\n");

    CHECKERR (rssiam_close (io));

    return 0;
}
```

Execution Result:

```
 Line 49 (8545 ms): rssiam_init (resourceName, IDQuery, resetDevice, &io)

 --- Set Frequency Sweep Mode -------------------------------------------

       - Function: Sine
       - Start Frequency: 1.0 kHz
       - Stop Frequency: 10.0 kHz
       - Sweep Direction: Down
       - Sweep Spacing: Log
       - Sweep Time: 1.0 s
       - Sweep Points: 100

 ------------------------------------------------------------------------

 Line 66 (21 ms): rssiam_funcShap (io, RSSIAM_OUTPUT_FUNC_SIN)
 Line 67 (55 ms): rssiam_outpSetup (io, RSSIAM_OUTPUT_SETUP_ON)
 Line 70 (23 ms): rssiam_freqStar (io, 1.0e3)
 Line 71 (23 ms): rssiam_freqStop (io, 10.0e3)
 Line 72 (23 ms): rssiam_sweDirection (io, RSSIAM_SWEEP_DIRECTION_DOWN)
 Line 73 (23 ms): rssiam_swePoin (io, 100)
 Line 74 (23 ms): rssiam_sweSpac (io, RSSIAM_SWEEP_SPAC_LOG)
 Line 75 (23 ms): rssiam_sweTime (io, 1.0)
 Line 76 (2223 ms): rssiam_sweStat (io, RSSIAM_SWEEP_STAT_ON)

 Line 80 (0 ms): rssiam_close (io)
```

2.7.1.1.5 Setting two different waveforms with different frequencies on both channels

Source Code:

```
/*************************************************************************
*
 *
 * Title:   Uncoupled Channel Mode
 *
 * Purpose: This example shows how to set two different waveforms with
 *          different frequencies on both channels.
 *

*************************************************************************
/

#include <ansi_c.h>
#include "rssiam.h"

/*** Macros & definitions
*************************************************/

#define CHECKERR( fCal ) \
    { \
        fCalStartTime = clock();\
        error = (fCal);\
        status = error;\
        printf(" Line %ld (%ld ms): %s\n", __LINE__,\
            (fCalTime = (clock() - fCalStartTime)), #fCal);\
        if (error != VI_SUCCESS)\
            {\
            rssiam_error_message (io, error, error_message);\
            printf("\tFunction Call Status: 0x%08X, %s\n", error,
error_message);\
            rssiam_error_query(io, &error, error_message);\
            printf("\tInstrument Error: 0x%X, %s\n", error, error_message);\
            }\
    }

#define RESOURCE_NAME   "USB::0xAAD::0x5::100011" // Resource name

/*** Main
*****************************************************************/

int main (int argc, char *argv[])
{
    ViStatus    error               = VI_SUCCESS,
                status              = VI_SUCCESS;
    clock_t     fCalTime            = 0,
                fCalStartTime       = 0;
    ViChar      error_message[256];
    ViSession   io;

    /* Define remote connection parameters (as default values) */

    ViBoolean   IDQuery             = VI_TRUE,
                resetDevice         = VI_TRUE;
    ViRsrc      resourceName        = RESOURCE_NAME;

    CHECKERR (rssiam_init (resourceName, IDQuery, resetDevice, &io));
    printf (

    "\n --- Set Independent Channel Parameters -----------------------------
----\n\n"
    "\t CH1:\n\n"
    "\t - Frequency: 700.0 Hz\n"
    "\t - Function: Square\n\n"
    "\t CH2:\n\n"
    "\t - Frequency: 2400.0 Hz\n"
    "\t - Function: Sine\n"
    "\n --------------------------------------------------------------------
----\n\n"

    );

    CHECKERR (rssiam_outpChannelCoupl (io, RSSIAM_CHANNEL_COUPLING_OFF));

    /* --- CH1 --- */
    CHECKERR (rssiam_outpChannel (io, RSSIAM_CH1));
    CHECKERR (rssiam_freq (io, 700.0));
    CHECKERR (rssiam_funcShap (io, RSSIAM_OUTPUT_FUNC_SQU));
    CHECKERR (rssiam_outpSetup (io, RSSIAM_OUTPUT_SETUP_ON));

    /* --- CH2 --- */
    CHECKERR (rssiam_outpChannel (io, RSSIAM_CH2));
    CHECKERR (rssiam_freq (io, 2400.0));
    CHECKERR (rssiam_funcShap (io, RSSIAM_OUTPUT_FUNC_SIN));
    CHECKERR (rssiam_outpSetup (io, RSSIAM_OUTPUT_SETUP_ON));

    printf ("\n");

    CHECKERR (rssiam_close (io));

    return 0;
}
```

Execution Result:

```
 Line 50 (8560 ms): rssiam_init (resourceName, IDQuery, resetDevice, &io)

 --- Set Independent Channel Parameters ---------------------------------

       CH1:

       - Frequency: 700.0 Hz
       - Function: Square

       CH2:

       - Frequency: 2400.0 Hz
       - Function: Sine

 ------------------------------------------------------------------------

 Line 65 (9276 ms): rssiam_outpChannelCoupl (io,
RSSIAM_CHANNEL_COUPLING_OFF)
 Line 68 (0 ms): rssiam_outpChannel (io, RSSIAM_CH1)
 Line 69 (93 ms): rssiam_freq (io, 700.0)
 Line 70 (1112 ms): rssiam_funcShap (io, RSSIAM_OUTPUT_FUNC_SQU)
 Line 71 (73 ms): rssiam_outpSetup (io, RSSIAM_OUTPUT_SETUP_ON)
 Line 74 (0 ms): rssiam_outpChannel (io, RSSIAM_CH2)
 Line 75 (107 ms): rssiam_freq (io, 2400.0)
 Line 76 (23 ms): rssiam_funcShap (io, RSSIAM_OUTPUT_FUNC_SIN)
 Line 77 (56 ms): rssiam_outpSetup (io, RSSIAM_OUTPUT_SETUP_ON)

 Line 81 (0 ms): rssiam_close (io)
```

2.7.1.1.6 Setting the burst mode with external source

Source Code:

```
/*************************************************************************
*
 *
 * Title:   Setting the burst mode with external source
 *
 * Purpose: This example shows how to set burst mode with external source.
 *

**************************************************************************
/

#include <ansi_c.h>
#include "rssiam.h"

/*** Macros & definitions
**************************************************/

#define CHECKERR( fCal ) \
    { \
        fCalStartTime = clock();\
        error = (fCal);\
        status = error;\
        printf(" Line %ld (%ld ms): %s\n", __LINE__,\
            (fCalTime = (clock() - fCalStartTime)), #fCal);\
        if (error != VI_SUCCESS)\
            {\
            rssiam_error_message (io, error, error_message);\
            printf("\tFunction Call Status: 0x%08X, %s\n", error,
error_message);\
            rssiam_error_query(io, &error, error_message);\
            printf("\tInstrument Error: 0x%X, %s\n", error, error_message);\
            }\
    }

#define RESOURCE_NAME   "USB::0xAAD::0x5::100011" // Resource name

/*** Main
******************************************************************/

int main (int argc, char *argv[])
{
    ViStatus    error               = VI_SUCCESS,
                status              = VI_SUCCESS;
    clock_t     fCalTime            = 0,
                fCalStartTime       = 0;
    ViChar      error_message[256];
    ViSession   io;

    /* Define remote connection parameters (as default values) */

    ViBoolean   IDQuery             = VI_TRUE,
                resetDevice         = VI_TRUE;
    ViRsrc      resourceName        = RESOURCE_NAME;

    CHECKERR (rssiam_init (resourceName, IDQuery, resetDevice, &io));

    printf (
    "\n --- Set Burst Mode -------------------------------------------------
----\n\n"
    "\t - Carrier Frequency: 1000.0 Hz\n"
    "\t - Function: Sine\n"
    "\t - Burst Source: External (Rear Trig/Gate In connector)\n"
    "\t - No. of Bursts: 3\n"
    "\n --------------------------------------------------------------------
----\n\n"

    );

    /* Waweform setup */
    CHECKERR (rssiam_freq (io, 1000.0));
    CHECKERR (rssiam_funcShap (io, RSSIAM_OUTPUT_FUNC_SIN));
    CHECKERR (rssiam_outpSetup (io, RSSIAM_OUTPUT_SETUP_ON));

    /* Trigger setup */
    CHECKERR (rssiam_trigSour (io, RSSIAM_TRIGGER_SOUR_EXT));

    /* Burst setup */
    CHECKERR (rssiam_bmSour (io, RSSIAM_BM_SOUR_TRIG));
    CHECKERR (rssiam_bmNcyc (io, 3));
    CHECKERR (rssiam_bmStat (io, RSSIAM_BM_STAT_ON));

    printf ("\n");

    CHECKERR (rssiam_close (io));

    return 0;
}
```

Execution Result:

```
 Line 49 (8586 ms): rssiam_init (resourceName, IDQuery, resetDevice, &io)

 --- Set Burst Mode -----------------------------------------------------

       - Carrier Frequency: 1000.0 Hz
       - Function: Sine
       - Burst Source: External (Rear Trig/Gate In connector)
       - No. of Bursts: 3

 ------------------------------------------------------------------------

 Line 63 (21 ms): rssiam_freq (io, 1000.0)
 Line 64 (24 ms): rssiam_funcShap (io, RSSIAM_OUTPUT_FUNC_SIN)
 Line 65 (56 ms): rssiam_outpSetup (io, RSSIAM_OUTPUT_SETUP_ON)
 Line 68 (24 ms): rssiam_trigSour (io, RSSIAM_TRIGGER_SOUR_EXT)
 Line 71 (24 ms): rssiam_bmSour (io, RSSIAM_BM_SOUR_TRIG)
 Line 72 (24 ms): rssiam_bmNcyc (io, 3)
 Line 73 (24 ms): rssiam_bmStat (io, RSSIAM_BM_STAT_ON)

 Line 77 (0 ms): rssiam_close (io)
```

2.7.1.1.7 Setting of FM modulation

Source Code:

```
/****************************************************************************
*
 *
 * Title:   Setting the FM Modulation
 *
 * Purpose: This example shows how to set FM modulation with internal
source.
 *

****************************************************************************
/

#include <ansi_c.h>
#include "rssiam.h"

/*** Macros & definitions
**************************************************/

#define CHECKERR( fCal ) \
    { \
        fCalStartTime = clock();\
        error = (fCal);\
        status = error;\
        printf(" Line %ld (%ld ms): %s\n", __LINE__,\
            (fCalTime = (clock() - fCalStartTime)), #fCal);\
        if (error != VI_SUCCESS)\
            {\
            rssiam_error_message (io, error, error_message);\
            printf("\tFunction Call Status: 0x%08X, %s\n", error,
error_message);\
            rssiam_error_query(io, &error, error_message);\
            printf("\tInstrument Error: 0x%X, %s\n", error, error_message);\
            }\
    }

#define RESOURCE_NAME   "USB::0xAAD::0x5::100011" // Resource name

/*** Main
******************************************************************/

int main (int argc, char *argv[])
{
    ViStatus    error               = VI_SUCCESS,
                status              = VI_SUCCESS;
    clock_t     fCalTime            = 0,
                fCalStartTime       = 0;
    ViChar      error_message[256];
    ViSession   io;

    /* Define remote connection parameters (as default values) */

    ViBoolean   IDQuery             = VI_TRUE,
                resetDevice         = VI_TRUE;
    ViRsrc      resourceName        = RESOURCE_NAME;

    CHECKERR (rssiam_init (resourceName, IDQuery, resetDevice, &io));
    printf (

    "\n --- Set FM Modulation ----------------------------------------------
----\n\n"
    "\t - Carrier Frequency: 1.0 MHz\n"
    "\t - Carrier Function: Sine\n"
    "\t - Modulation: FM\n"
    "\t - Modulation Frequency: 1.0 kHz\n"
    "\t - Modulation Deviation: 1.0 kHz\n"
    "\t - Modulation Function: Sine\n"
    "\n --------------------------------------------------------------------
----\n\n"

    );

    /* Waweform setup */
    CHECKERR (rssiam_freq (io, 1.0e6));
    CHECKERR (rssiam_funcShap (io, RSSIAM_OUTPUT_FUNC_SIN));
    CHECKERR (rssiam_outpSetup (io, RSSIAM_OUTPUT_SETUP_ON));

    /* Modulation setup */
    CHECKERR (rssiam_fmDev (io, 1000.0));
    CHECKERR (rssiam_fmIntFunc (io, RSSIAM_FREQ_INT_FUNC_SIN));
    CHECKERR (rssiam_fmIntFreq (io, 1000.0));
    CHECKERR (rssiam_fmStat (io, RSSIAM_FREQ_STAT_ON));

    printf ("\n");

    CHECKERR (rssiam_close (io));

    return 0;
}
```

Execution Result:

```
 Line 49 (8577 ms): rssiam_init (resourceName, IDQuery, resetDevice, &io)

 --- Set FM Modulation --------------------------------------------------

       - Carrier Frequency: 1.0 MHz
       - Carrier Function: Sine
       - Modulation: FM
       - Modulation Frequency: 1.0 kHz
       - Modulation Deviation: 1.0 kHz
       - Modulation Function: Sine

 ------------------------------------------------------------------------

 Line 65 (94 ms): rssiam_freq (io, 1.0e6)
 Line 66 (1767 ms): rssiam_funcShap (io, RSSIAM_OUTPUT_FUNC_SIN)
 Line 67 (55 ms): rssiam_outpSetup (io, RSSIAM_OUTPUT_SETUP_ON)
 Line 70 (23 ms): rssiam_fmDev (io, 1000.0)
 Line 71 (31 ms): rssiam_fmIntFunc (io, RSSIAM_FREQ_INT_FUNC_SIN)
 Line 72 (23 ms): rssiam_fmIntFreq (io, 1000.0)
 Line 73 (6870 ms): rssiam_fmStat (io, RSSIAM_FREQ_STAT_ON)

 Line 77 (1 ms): rssiam_close (io)
```

2.7.1.1.8 Setting of FSK with internal source

Source Code:

```
/*****************************************************************************
*
 *
 * Title:   Setting the FSK Modulation
 *
 * Purpose: This example shows how to set FSK modulation with internal
source.
 *

*****************************************************************************
/

#include <ansi_c.h>
#include "rssiam.h"

/*** Macros & definitions
*************************************************/

#define CHECKERR( fCal ) \
    { \
        fCalStartTime = clock();\
        error = (fCal);\
        status = error;\
        printf(" Line %ld (%ld ms): %s\n", __LINE__,\
            (fCalTime = (clock() - fCalStartTime)), #fCal);\
        if (error != VI_SUCCESS)\
            {\
            rssiam_error_message (io, error, error_message);\
            printf("\tFunction Call Status: 0x%08X, %s\n", error,
error_message);\
            rssiam_error_query(io, &error, error_message);\
            printf("\tInstrument Error: 0x%X, %s\n", error, error_message);\
            }\
    }

#define RESOURCE_NAME   "USB::0xAAD::0x5::100011" // Resource name

/*** Main
*****************************************************************/

int main (int argc, char *argv[])
{
    ViStatus    error               = VI_SUCCESS,
                status              = VI_SUCCESS;
    clock_t     fCalTime            = 0,
                fCalStartTime       = 0;
    ViChar      error_message[256];
    ViSession   io;

    /* Define remote connection parameters (as default values) */

    ViBoolean   IDQuery             = VI_TRUE,
                resetDevice         = VI_TRUE;
    ViRsrc      resourceName        = RESOURCE_NAME;

    CHECKERR (rssiam_init (resourceName, IDQuery, resetDevice, &io));
    printf (

    "\n --- Set FSK Modulation --------------------------------------------
----\n\n"
    "\t - Carrier Frequency: 1.0 MHz\n"
    "\t - Carrier Function: Sine\n"
    "\t - Modulation: FSK\n"
    "\t - FSK Frequency: 100.0 kHz\n"
    "\t - FSK Rate: 1000.0 Hz\n"
    "\t - FSK Source Internal\n"
    "\n ------------------------------------------------------------------
----\n\n"

    );

    /* Waweform setup */
    CHECKERR (rssiam_freq (io, 1.0e6));
    CHECKERR (rssiam_funcShap (io, RSSIAM_OUTPUT_FUNC_SIN));
    CHECKERR (rssiam_outpSetup (io, RSSIAM_OUTPUT_SETUP_ON));

    /* Modulation setup */
    CHECKERR (rssiam_fskFreq (io, 100.0e3));
    CHECKERR (rssiam_fskIntRate (io, 1000.0));
    CHECKERR (rssiam_fskSour (io, RSSIAM_FSK_SOUR_INT));
    CHECKERR (rssiam_fskStat (io, RSSIAM_FSK_STAT_ON));

    printf ("\n");

    CHECKERR (rssiam_close (io));

    return 0;
}
```

Execution Result:

```
 Line 49 (8602 ms): rssiam_init (resourceName, IDQuery, resetDevice, &io)

 --- Set FSK Modulation ----------------------------------------------

       - Carrier Frequency: 1.0 MHz
       - Carrier Function: Sine
       - Modulation: FSK
       - FSK Frequency: 100.0 kHz
       - FSK Rate: 1000.0 Hz
       - FSK Source Internal

 ------------------------------------------------------------------------

 Line 65 (93 ms): rssiam_freq (io, 1.0e6)
 Line 66 (24 ms): rssiam_funcShap (io, RSSIAM_OUTPUT_FUNC_SIN)
 Line 67 (55 ms): rssiam_outpSetup (io, RSSIAM_OUTPUT_SETUP_ON)
 Line 70 (24 ms): rssiam_fskFreq (io, 100.0e3)
 Line 71 (24 ms): rssiam_fskIntRate (io, 1000.0)
 Line 72 (24 ms): rssiam_fskSour (io, RSSIAM_FSK_SOUR_INT)
 Line 73 (32 ms): rssiam_fskStat (io, RSSIAM_FSK_STAT_ON)

 Line 77 (0 ms): rssiam_close (io)
```

2.7.1.1.9 Setting of PSK with external source

Source Code:

```
/***************************************************************************
*
 *
 * Title:   Setting the PSK Modulation
 *
 * Purpose: This example shows how to set PSK modulation with external
source.
 *

***************************************************************************
/

#include <ansi_c.h>
#include "rssiam.h"

/*** Macros & definitions
*************************************************/

#define CHECKERR( fCal ) \
    { \
        fCalStartTime = clock();\
        error = (fCal);\
        status = error;\
        printf(" Line %ld (%ld ms): %s\n", __LINE__,\
            (fCalTime = (clock() - fCalStartTime)), #fCal);\
        if (error != VI_SUCCESS)\
            {\
            rssiam_error_message (io, error, error_message);\
            printf("\tFunction Call Status: 0x%08X, %s\n", error,
error_message);\
            rssiam_error_query(io, &error, error_message);\
            printf("\tInstrument Error: 0x%X, %s\n", error, error_message);\
            }\
    }

#define RESOURCE_NAME   "USB::0xAAD::0x5::100011" // Resource name

/*** Main
*****************************************************************/

int main (int argc, char *argv[])
{
    ViStatus    error               = VI_SUCCESS,
                status              = VI_SUCCESS;
    clock_t     fCalTime            = 0,
                fCalStartTime       = 0;
    ViChar      error_message[256];
    ViSession   io;

    /* Define remote connection parameters (as default values) */

    ViBoolean   IDQuery             = VI_TRUE,
                resetDevice         = VI_TRUE;
    ViRsrc      resourceName        = RESOURCE_NAME;

    CHECKERR (rssiam_init (resourceName, IDQuery, resetDevice, &io));
    printf (

    "\n --- Set PSK Modulation ---------------------------------------------
----\n\n"
    "\t - Carrier Frequency: 1.0 MHz\n"
    "\t - Carrier Function: Sine\n"
    "\t - Modulation: PSK\n"
    "\t - PSK Phase: 180.0 Deg\n"
    "\t - PSK Source: External\n"
    "\n --------------------------------------------------------------------
----\n\n"

    );

    /* Waweform setup */
    CHECKERR (rssiam_freq (io, 1.0e6));
    CHECKERR (rssiam_funcShap (io, RSSIAM_OUTPUT_FUNC_SIN));
    CHECKERR (rssiam_outpSetup (io, RSSIAM_OUTPUT_SETUP_ON));

    /* Modulation setup */
    CHECKERR (rssiam_pskPhase (io, 180.0));
    CHECKERR (rssiam_pskSour (io, RSSIAM_PSK_SOUR_EXT));
    CHECKERR (rssiam_pskStat (io, RSSIAM_PSK_STAT_ON));

    printf ("\n");

    CHECKERR (rssiam_close (io));

    return 0;
}
```

Execution Result:

```
 Line 49 (8566 ms): rssiam_init (resourceName, IDQuery, resetDevice, &io)

 --- Set PSK Modulation -------------------------------------------------

       - Carrier Frequency: 1.0 MHz
       - Carrier Function: Sine
       - Modulation: PSK
       - PSK Phase: 180.0 Deg
       - PSK Source: External

 ------------------------------------------------------------------------

 Line 64 (93 ms): rssiam_freq (io, 1.0e6)
 Line 65 (23 ms): rssiam_funcShap (io, RSSIAM_OUTPUT_FUNC_SIN)
 Line 66 (56 ms): rssiam_outpSetup (io, RSSIAM_OUTPUT_SETUP_ON)
 Line 69 (24 ms): rssiam_pskPhase (io, 180.0)
 Line 70 (25 ms): rssiam_pskSour (io, RSSIAM_PSK_SOUR_EXT)
 Line 71 (23 ms): rssiam_pskStat (io, RSSIAM_PSK_STAT_ON)

 Line 75 (1 ms): rssiam_close (io)
```

2.7.1.1.10 Setting of phase differences between Ch1 and Ch2

Source Code:

```
/***************************************************************************
*
 *
 * Title:   Setting of Coupled Channel Mode
 *
 * Purpose: This example shows how to set phase differences between Ch1 and
Ch2.
 *

***************************************************************************
/

#include <ansi_c.h>
#include "rssiam.h"

/*** Macros & definitions
****************************************************/

#define CHECKERR( fCal ) \
    { \
        fCalStartTime = clock();\
        error = (fCal);\
        status = error;\
        printf(" Line %ld (%ld ms): %s\n", __LINE__,\
            (fCalTime = (clock() - fCalStartTime)), #fCal);\
        if (error != VI_SUCCESS)\
            {\
            rssiam_error_message (io, error, error_message);\
            printf("\tFunction Call Status: 0x%08X, %s\n", error,
error_message);\
            rssiam_error_query(io, &error, error_message);\
            printf("\tInstrument Error: 0x%X, %s\n", error, error_message);\
            }\
    }

#define RESOURCE_NAME   "USB::0xAAD::0x5::100011" // Resource name

/*** Main
******************************************************************/

int main (int argc, char *argv[])
{
    ViStatus    error               = VI_SUCCESS,
                status              = VI_SUCCESS;
    clock_t     fCalTime            = 0,
                fCalStartTime       = 0;
    ViChar      error_message[256];
    ViSession   io;

    /* Define remote connection parameters (as default values) */

    ViBoolean   IDQuery             = VI_TRUE,
                resetDevice         = VI_TRUE;
    ViRsrc      resourceName        = RESOURCE_NAME;

    CHECKERR (rssiam_init (resourceName, IDQuery, resetDevice, &io));
    printf (

    "\n --- Set Channel Coupling Mode --------------------------------------
----\n\n"
    "\t - Carrier Frequency: 1000.0 Hz\n\n"
    "\t CH1:\n\n"
    "\t - Function: Sine\n"
    "\t - Start Phase: 0 Deg\n\n"
    "\t CH2:\n\n"
    "\t - Function: Sine\n"
    "\t - Start Phase: 180 Deg\n"
    "\n --------------------------------------------------------------------
----\n\n"

    );

    CHECKERR (rssiam_outpChannelCoupl (io, RSSIAM_CHANNEL_COUPLING_FREQ));

    /* Output Frequency Setup */
    CHECKERR (rssiam_freq (io, 1000.0));

    /* --- CH1 --- */
    CHECKERR (rssiam_outpChannel (io, RSSIAM_CH1));
    CHECKERR (rssiam_funcShap (io, RSSIAM_OUTPUT_FUNC_SIN));
    CHECKERR (rssiam_outpStartPhase (io, 0.0));
    CHECKERR (rssiam_outpSetup (io, RSSIAM_OUTPUT_SETUP_ON));

    /* --- CH2 --- */
    CHECKERR (rssiam_outpChannel (io, RSSIAM_CH2));
    CHECKERR (rssiam_funcShap (io, RSSIAM_OUTPUT_FUNC_SIN));
    CHECKERR (rssiam_outpStartPhase (io, 180.0));
    CHECKERR (rssiam_outpSetup (io, RSSIAM_OUTPUT_SETUP_ON));

    printf ("\n");

    CHECKERR (rssiam_close (io));

    return 0;
}
```

Execution Result:

```
 Line 49 (8544 ms): rssiam_init (resourceName, IDQuery, resetDevice, &io)

 --- Set Channel Coupling Mode ------------------------------------------

       - Carrier Frequency: 1000.0 Hz

       CH1:

       - Function: Sine
       - Start Phase: 0 Deg

       CH2:

       - Function: Sine
       - Start Phase: 180 Deg

 ------------------------------------------------------------------------

 Line 65 (5 ms): rssiam_outpChannelCoupl (io, RSSIAM_CHANNEL_COUPLING_FREQ)
 Line 68 (24 ms): rssiam_freq (io, 1000.0)
 Line 71 (0 ms): rssiam_outpChannel (io, RSSIAM_CH1)
```

```
Line 72 (23 ms): rssiam_funcShap (io, RSSIAM_OUTPUT_FUNC_SIN)
Line 73 (24 ms): rssiam_outpStartPhase (io, 0.0)
Line 74 (56 ms): rssiam_outpSetup (io, RSSIAM_OUTPUT_SETUP_ON)
Line 77 (0 ms): rssiam_outpChannel (io, RSSIAM_CH2)
Line 78 (23 ms): rssiam_funcShap (io, RSSIAM_OUTPUT_FUNC_SIN)
Line 79 (32 ms): rssiam_outpStartPhase (io, 180.0)
Line 80 (56 ms): rssiam_outpSetup (io, RSSIAM_OUTPUT_SETUP_ON)

Line 84 (1 ms): rssiam_close (io)
```