R&S Smart instruments™
The new product family from Rohde & Schwarz

Remote Control Manual
SM300 Signal Generator

VXI Plug & Play Style Instrument Driver

Version 04

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

4th edition 03/2005
Last Change: March 1st 2005

Table of Contents

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 SM300 Signal Generator via VXI Plug & 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 often demo programs are also available.

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. With complex equipment this is a time-consuming task, since just the description of the register command set may comprise several hundred pages and assumes a detailed knowledge about the instrument's hardware. This is the reason why Rohde&Schwarz provides ready-to-use software device drivers for all major interfaces, relieving designers of these efforts to a great extent. Extensive search for commands in manuals can be avoided because the work has already been invested when developing the driver.

1 Manual Concept

About this chapter Chapter 1 contains information for the user/installer of the Rohde&Schwarz SM300 Signal Generator VXI Plug & Play style Instrument Driver.

More Information Chapter 2 describes the function tree layout of the SM300 Signal Generator.

1.1 Introduction

Introduction

The Rohde&Schwarz SM300 Signal Generator drivers are single 32-bit drivers.

This Rohde&Schwarz SM300 Signal 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 SM300 Signal Generator since it is not a VXI instrument. One of the differences is, that there is no soft front panel, as the Rohde&Schwarz SM300 Signal 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 use the services provided by VISA. VISA is used for its prototype definitions and future compatibility with the other VXIplug&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 VXIplug&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 SM300 Signal 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 to establish a communication session between the host computer and device. The installation process is self-guided.

Instrument Description

The SM300 Signal Generator 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

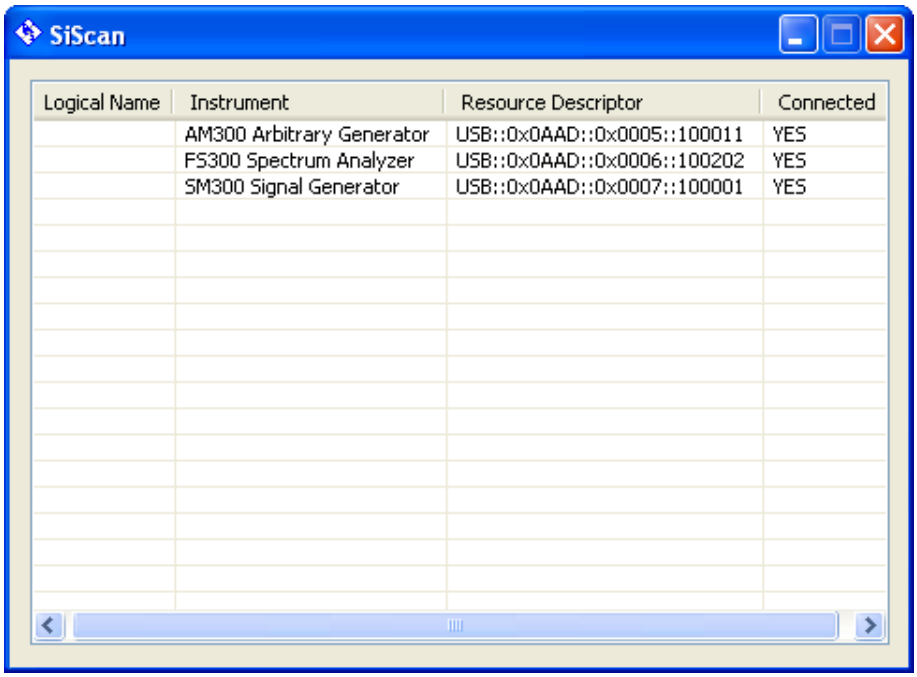
Based on the Instrument Descriptor (instrument address, resource string), the driver establishes a communication session with a device. The grammar for the Instrument Descriptor is shown below. Optional parameters are shown in square brackets ([]).

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 (0x7 for SM300 Signal Generator)
- **Serial Number** is the serial number printed on the instrument box

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



1.2.2 Using Callbacks

Caution

Callbacks are not supported with this driver.

1.2.3 Threat Safety

Caution

We recommend not to use multiple threats to communicate with the instrument in parallel.

1.2.4 Device Identification and Logical Names

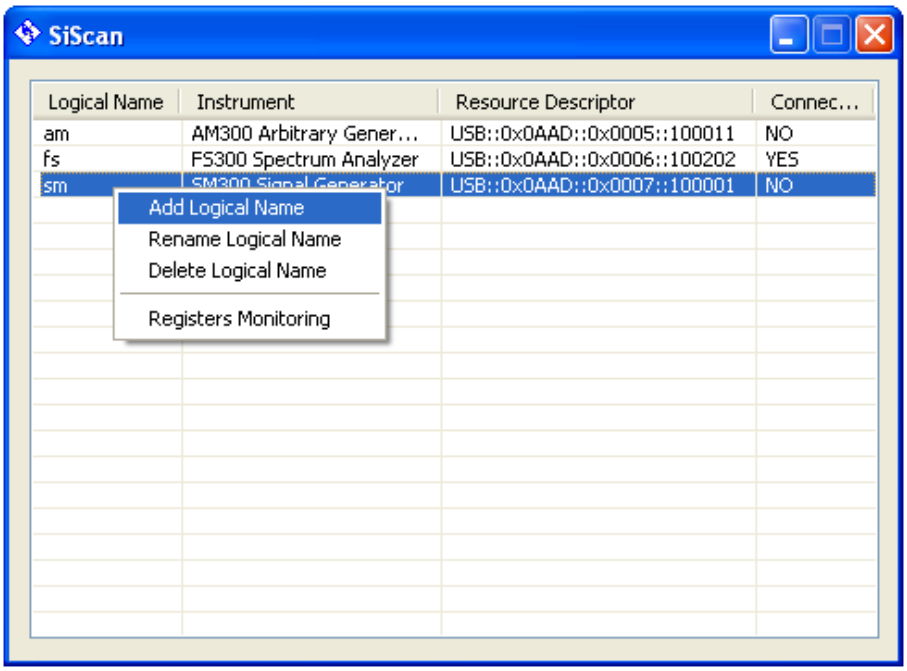
SiScan

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

The 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::0x7::123456**.

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



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 descripton 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 remote mode (plugged) without loosing the initialized communication session. Once the session is opened (rssifs_init), session based data are safe until the session is closed (rssifs_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 a path defined in the windows registry under

HKEY_LOCAL_MACHINE\SOFTWARE\Rohde&Schwarz\SiTools

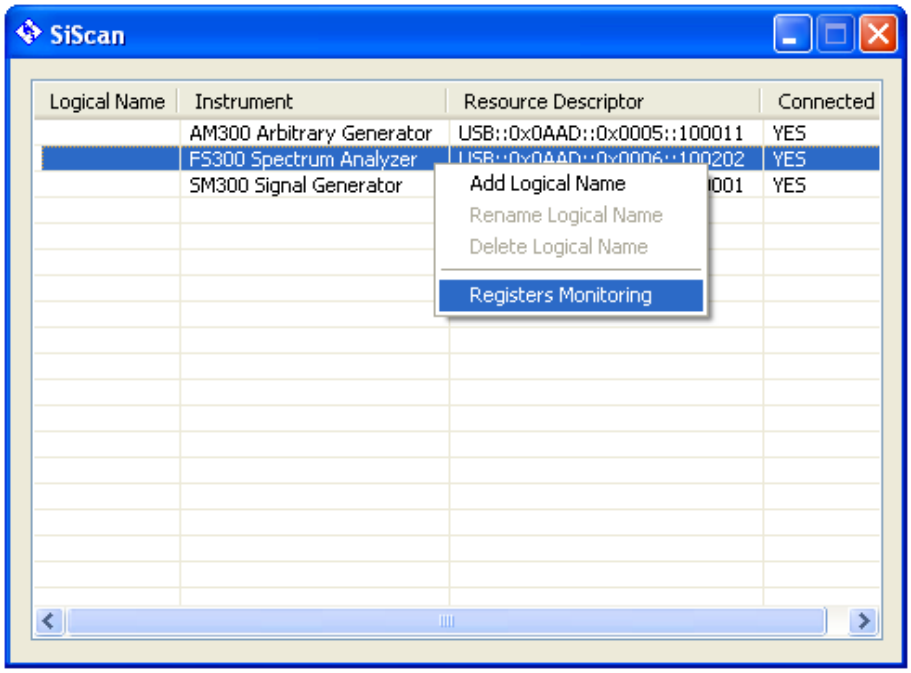
key (***SiToolsDir***). Each of ***SiTools*** provides comprehensive help how to use it. ***SiTools*** are installed with the driver and stored in the system's program folder (typically C:\Program Files\Series300\SiTools).

Note

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

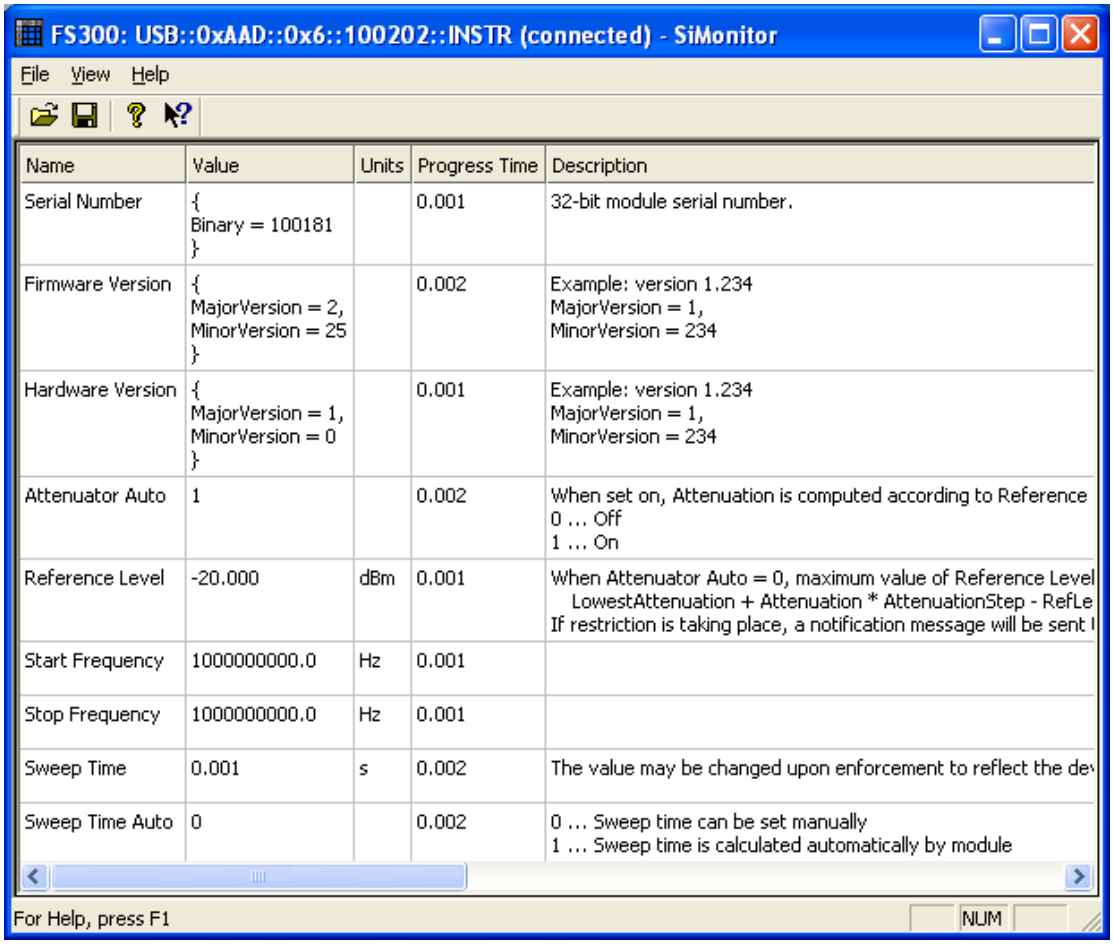
SiScan

SiScan is a 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.



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 the development period of the remote control application. More detailed information on using the ***SiMonitor*** can be found in the associated help file.



Note

The detail description of the monitored parameters is not provided within this manual. It is part of the device specific interface (DSI) SM300 Signal Generator measurement module documentation.

1.3 Using Instrument Driver in Application Development Environments

This section offers suggestions on using the rssifs_32.dll within various application development environments.

Note

The application notes "R&S SmartInstruments™ Family300 Basic Programming Guide", which is available on the Rohde&Schwarz hompage, will also give a detailed overview about who to use the drivers in different 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 rssifs.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 SM300 Signal 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 rssism.fp in your project. This file can be found in the vxipnp\winnt\rssism directory. Do not include rssism.c in your project. You must also provide an include path for rssism.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 manually the content of ~VXIpnp\GWinnt\rssism directory into your LabVIEW directory (~LabVIEW\instr.lib\rssism\). 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:

rssism.h	Header file for use with C, HP VEE and LabView/LabWindows
rssism.c	Source code for use with C
rssism.def	Definition file for use with C++ when building the .dll file
rssism.fp	Function Panel file for use with HP VEE and LabVIEW/LabWindows
rssism.bas	Module file for use with Visual BASIC
rssism.vb	Module file for use with .NET BASIC
rssism.hlp	Help file for use with VB
rssism.chm	Compressed HTML help for use with C and Visual Basic
rssism.lib	Library file for use with C++
rssism_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:

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

Table 1-2: LabVIEW 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 listed in above section.

2 Programmer's Reference Manual

2.1 Instrument Driver Tree Structure

Class/Panel Name	Function Name
Initialize	rssism_init
Function Examples	
Output AM Modulated Signal	rssism_exOutAmpl
Configuration Functions	
RF Frequency	
Configure RF Frequency	rssism_configRFFreq
Low-Level	
Set CW Frequency	rssism_setCWFreq
Get CW Frequency	rssism_getCWFreq
Set RF Frequency Offset	rssism_setRFFreqOffset
Get RF Frequency Offset	rssism_getRFFreqOffset
Set RF Frequency Mode	rssism_setRFFreqMode
Get RF Frequency Mode	rssism_getRFFreqMode
RF Frequency Sweep	
Configure RF Frequency Sweep	rssism_configRFFreqSweep
Low-Level	
Set RF Freq Sweep Mode	rssism_setRFFreqSweepMode
Get RF Freq Sweep Mode	rssism_getRFFreqSweepMode
Set RF Start Frequency	rssism_setRFStartFreq
Get RF Start Frequency	rssism_getRFStartFreq
Set RF Stop Frequency	rssism_setRFStopFreq
Get RF Stop Frequency	rssism_getRFStopFreq
Set RF Freq Sweep Dwell Time	rssism_setRFFreqSweepDwellTime
Get RF Freq Sweep Dwell Time	rssism_getRFFreqSweepDwellTime
Set RF Freq Sweep Spacing	rssism_setRFFreqSweepSpacing
Get RF Freq Sweep Spacing	rssism_getRFFreqSweepSpacing
Set RF Freq Sweep Step	rssism_setRFFreqSweepStep
Get RF Freq Sweep Step	rssism_getRFFreqSweepStep
Set RF Sweep Frequency Manual	rssism_setRFSweepFreqManual
Get RF Sweep Frequency Manual	rssism_getRFSweepFreqManual
Set RF Center Frequency	rssism_setRFCenterFreq
Get RF Center Frequency	rssism_getRFCenterFreq
Set RF Span Frequency	rssism_setRFSpanFreq

Class/Panel Name	Function Name
Get RF Span Frequency	rssism_getRFSpanFreq
RF Level	
Configure RF Level	rssism_configRFLevel
Low-Level	
Set RF Level	rssism_setRFLevel
Get RF Level	rssism_getRFLevel
Set RF Level Offset	rssism_setRFLevelOffset
Get RF Level Offset	rssism_getRFLevelOffset
Set RF Level Limit	rssism_setRFLevelLimit
Get RF Level Limit	rssism_getRFLevelLimit
Set RF Level Mode	rssism_setRFLevelMode
Get RF Level Mode	rssism_getRFLevelMode
RF Level Sweep	
Configure RF Level Sweep	rssism_configRFLevelSweep
Low-Level	
Set RF Level Sweep Mode	rssism_setRFLevelSweepMode
Get RF Level Sweep Mode	rssism_getRFLevelSweepMode
Set RF Start Level	rssism_setRFStartLevel
Get RF Start Level	rssism_getRFStartLevel
Set RF Stop Level	rssism_setRFStopFreqLevel
Get RF Stop Level	rssism_getRFStopFreqLevel
Set RF Level Sweep Step	rssism_setRFLevelSweepStep
Get RF Level Sweep Step	rssism_getRFLevelSweepStep
Set RF Level Sweep Dwell Time	rssism_setRFLevelSweepDwellTime
Get RF Level Sweep Dwell Time	rssism_getRFLevelSweepDwellTime
Set RF Level Sweep Manual	rssism_setRFLevelSweepManual
Get RF Level Sweep Manual	rssism_getRFLevelSweepManual
RF Output	
Set RF Output State	rssism_setRFOutputState
Get RF Output State	rssism_getRFOutputState
RF Modulation	
Configure AM Modulation	rssism_configAMModulation
Configure FM Modulation	rssism_configFMModulation
Configure Phase Modulation	rssism_configPHMModulation
Configure Pulse Modulation	rssism_configPulseModulation
Low-Level AM Modulation	
Set AM State	rssism_setAMState

Class/Panel Name	Function Name
Get AM State	rssism_getAMState
Set AM Depth	rssism_setAMDepth
Get AM Depth	rssism_getAMDepth
Set AM Frequency	rssism_setAMFreq
Get AM Frequency	rssism_getAMFreq
Set AM Source	rssism_setAMSource
Get AM Source	rssism_getAMSource
Set AM External Coupling	rssism_setAMExtCoupling
Get AM External Coupling	rssism_getAMExtCoupling
Set AM Polarity	rssism_setAMPolar
Get AM Polarity	rssism_getAMPolar
Low-Level FM Modulation	
Set FM State	rssism_setFMState
Get FM State	rssism_getFMState
Set FM Deviation	rssism_setFMDeviation
Get FM Deviation	rssism_getFMDeviation
Set FM Frequency	rssism_setFMFreq
Get FM Frequency	rssism_getFMFreq
Set FM Source	rssism_setFMSource
Get FM Source	rssism_getFMSource
Set FM External Coupling	rssism_setFMExtCoupling
Get FM External Coupling	rssism_getFMExtCoupling
Set FM Polarity	rssism_setFMPolar
Get FM Polarity	rssism_getFMPolar
Low-Level Phase Modulation	
Set PHM State	rssism_setPHMState
Get PHM State	rssism_getPHMState
Set PHM Deviation	rssism_setPHMDeviation
Get PHM Deviation	rssism_getPHMDeviation
Set PHM Frequency	rssism_setPHMFreq
Get PHM Frequency	rssism_getPHMFreq
Set PHM Source	rssism_setPHMSource
Get PHM Source	rssism_getPHMSource
Set PHM Polarity	rssism_setPHMPolar
Get PHM Polarity	rssism_getPHMPolar
Low-Level Pulse Modulation	
Set PM State	rssism_setPMState

Class/Panel Name	Function Name
Get PM State	rssism_getPMState
Set PM Source	rssism_setPMSource
Get PM Source	rssism_getPMSource
Set PM Polarity	rssism_setPMPolar
Get PM Polarity	rssism_getPMPolar
Set PM Pulse Off Time	rssism_setPMPulseOffTime
Get PM Pulse Off Time	rssism_getPMPulseOffTime
Set PM Pulse On Time	rssism_setPMPulseOnTime
Get PM Pulse On Time	rssism_getPMPulseOnTime
Set PM Pulse Delay Time	rssism_setPMPulseDelayTime
Get PM Pulse Delay Time	rssism_getPMPulseDelayTime
Low-Level Vector Modulation	
Set Vector State	rssism_setVectorState
Get Vector State	rssism_getVectorState
LF Output	
Configure LF Output	rssism_configLFOutput
Low-Level	
Set LF Output State	rssism_setLFOutputState
Get LF Output State	rssism_getLFOutputState
Set LF Frequency	rssism_setLFCWFreq
Get LF Frequency	rssism_getLFCWFreq
Set LF Voltage	rssism_setLFVoltage
Get LF Voltage	rssism_getLFVoltage
LF Sweep	
Configure LF Sweep	rssism_configLFSweep
Low-Level	
Set LF Sweep Mode	rssism_setLFSweepMode
Get LF Sweep Mode	rssism_getLFSweepMode
Set LF Start Frequency	rssism_setLFStartFreq
Get LF Start Frequency	rssism_getLFStartFreq
Set LF Stop Frequency	rssism_setLFStopFreq
Get LF Stop Frequency	rssism_getLFStopFreq
Set LF Frequency Manual	rssism_setLFFreqMan
Get LF Frequency Manual	rssism_getLFFreqMan
Set LF Sweep Spacing	rssism_setLFSweepSpac
Get LF Sweep Spacing	rssism_getLFSweepSpac
Set LF Sweep Step	rssism_setLFSweepStep

Class/Panel Name	Function Name
Get LF Sweep Step	rssism_getLFSweepStep
Set LF Sweep Dwell Time	rssism_setLFSweepDwellTime
Get LF Sweep Dwell Time	rssism_getLFSweepDwellTime
Reference Oscillator	
Configure Ref Oscillator	rssism_configureRefOscilator
Get Ref Oscillator	rssism_getRefOscillator
Trigger	
Set Sweep Trigger Source	rssism_setSweepTrigSource
Get Sweep Trigger Source	rssism_getSweepTrigSource
Action/Status Functions	
Send Trigger	rssism_SendTrigger
Send Trigger and Wait for OPC	rssism_SendTriggerWopc
Abort Trigger	rssism_abortTrigger
Utility Functions	
Time Out	
Set Time Out	rssism_setTimeOut
Get Time Out	rssism_getTimeOut
Flush Error Queue	rssism_FlushErrorQueue
State Checking	rssism_errorCheckState
Reset	rssism_reset
Self-Test	rssism_self_test
Error-Query	rssism_error_query
Error Message	rssism_error_message
Revision Query	rssism_revision_query
Close	rssism_close

Table 2-1: Install program

2.2 Function Tree Layout of the SM300 Signal Generator

Description

This instrument module provides programming support for the R&S SM300 Signal Generator. 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. **Function Examples: (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 Initialize

C Function Prototype

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

Basic Function Prototype

```
Function rssism_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::0x7::100015"

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::0x7::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 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 Function Examples

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 Output AM Modulated Signal:

- This function configures RF frequency output, switches it on and applies AM modulation on it with selected parameters.

2.2.2.1 Output AM Modulated Signal

C Function Prototype

```
ViStatus rssism_exOutAmpl (
    ViSession instrumentHandle,
    ViReal64 frequency,
    ViReal64 offset,
    ViReal64 level,
    ViInt32 modulationSource,
    ViReal64 modulationDepth,
    ViReal64 modulationFrequency);
```

Basic Function Prototype

```
Function rssism_exOutAmpl (
    ByVal instrumentHandle As ViSession,
    ByVal frequency As ViReal64,
    ByVal offset As ViReal64,
    ByVal level As ViReal64,
    ByVal modulationSource As ViInt32,
    ByVal modulationDepth As ViReal64,
    ByVal modulationFrequency As ViReal64) As ViStatus
```

Purpose This function represents a high-level approach to the instrument driver. It demonstrates how to create an application function which contains a set of low-level functions. E.g. this function fully configures the instrument to output an amplitude modulated signal.

This function performs the following operations:

- Resets the instrument
- Sets the RF output frequency
- Sets the RF output level
- Configures the internal LF generator (source of the modulating signal)
- Configures the amplitude modulation of the RF output signal
- Turns on the instrument's RF 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. **ViReal64 frequency [in]**

   This control sets the RF frequency.

   Valid Range (Offset = 0.0): 9.0e3 Hz to 3.0e9 Hz

   Default Value: 1.5e9 Hz

3. **ViReal64 offset [in]**

   This control sets the RF frequency offset.

   Valid Range: -50.0 GHz to 50.0 GHz

   Default Value: 0.0 Hz

Note

The RF frequency applied to the signal generator is calculated from the frequency and offset values as follows:

RF frequency = (frequency - offset)

Where RF frequency is the value sent to the instrument, frequency and offset are values entered to the functions.

4. **ViReal64 level [in]**

   This control sets the RF output level.

   Valid Range (Offset = 0.0): -127.0 dBm to 13.0 dBm

   Default Value: 0.0 dBm

5. **ViInt32 modulationSource [in]**

   This control selects the modulation source.

   Valid Range:

   - 0 - Internal
   - 1 - External
   - 2 - Internal + External

   Default Value: 0

6. **ViReal64 modulationDepth [in]**

   This control sets the modulation depth in percent.

   Valid Range: 0.0 % to 100.0 %

   Default Value: 100.0 %

7. **ViReal64 modulationFrequency [in]**

   This control sets the modulation frequency.

   Valid Range: 1.0 Hz to 80.0e3 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 Configuration Functions

Description

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

Functions/SubClasses

1. **RF Frequency: (Class)**
   This class operates RF Frequency.
2. **RF Frequency Sweep: (Class)**
   This class operates RF Frequency Sweep.
3. **RF Level: (Class)**
   This class operates RF Level.
4. **RF Level Sweep: (Class)**
   This class operates RF Level Sweep.
5. **RF Output: (Class)**
   This class operates RF Output.
6. **RF Modulation: (Class)**
   This class operates RF Modulation.
7. **LF Output: (Class)**
   This class operates LF Output.
8. **LF Sweep: (Class)**
   This class operates LF Sweep.
9. **Reference Oscillator: (Class)**
   This class operates Reference Oscillator.
10. **Trigger: (Class)**
    This class operates Trigger.

2.2.3.1 RF Frequency

Description

This class operates the RF Frequency.

Functions/SubClasses:

1. **Configure RF Frequency:**
   This function configures frequency of RF output signal.
2. **Low-Level: (Class**)
   Functions in this class operate all RF Frequency parameters.

2.2.3.1.1 Configure RF Frequency

C Function Prototype

```
ViStatus rssism_configRFFreq (
    ViSession instrumentHandle,
    ViReal64 frequency,
    ViReal64 offset,
    ViReal64 multiplier);
```

Basic Function Prototype

```
Function rssism_configRFFreq (
    ByVal instrumentHandle As ViSession,
    ByVal frequency As ViReal64,
    ByVal offset As ViReal64,
    ByVal multiplier As ViReal64) As ViStatus
```

Purpose

This function configures the frequency of RF output signal.

Note

The frequency of RF output signal is calculated from the frequency, offset and multiplier values as follows:

RF output frequency = (frequency - offset) / multiplier

Where RF output frequency is the displayed value, frequency, offset and multiplier are entered values.

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]**
   This control sets the frequency of RF output signal.
   Valid Range (Offset = 0.0): 9.0e3 Hz to 3.0e9 Hz
   Default Value: 1.5e9 Hz
3. **ViReal64 offset [in]**
   This control sets the frequency offset.
   Default Value: 0.0 Hz
   Valid Range: -50.0 GHz to 50.0 GHz
4. **ViReal64 multiplier [in]**
   Defines multiplier of the enetred frequency.
   Default Value: 1.0
   Valid Range: 1.0 to 10.0

Return Value

Returns the status code of this operation.

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

2.2.3.1.2 Low-Level

Description

Functions in this class operates all RF Frequency parameters.

2.2.3.1.2.1 Set CW Frequency

C Function Prototype

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

Basic Function Prototype

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

Purpose

This function sets the frequency of RF output signal.

Note

The frequency of RF output signal is calculated from the frequency and offset values as follows:

RF output frequency = (frequency - offset)

Where RF output frequency is the displayed value, frequency and offset are entered values.

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]**

   This control sets the frequency of RF output signal.

   Valid Range (Offset = 0.0): 9.0e3 Hz to 3.0e9 Hz

   Default Value: 1.5e9 Hz

Return Value

Returns the status code of this operation.

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

2.2.3.1.2.2 Get CW Frequency

C Function Prototype

```
ViStatus rssism_getCWFreq (
    ViSession instrumentHandle,
    ViReal64* frequency);
```

Basic Function Prototype

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

Purpose

This function returns the frequency of RF output 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. **ViReal64 frequency [out]**

   This control returns the frequency of RF output signal in Hz.

Return Value

Returns the status code of this operation.

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

2.2.3.1.2.3 Set RF Frequency Offset

C Function Prototype

```
ViStatus rssism_setRFFreqOffset (
    ViSession instrumentHandle,
    ViReal64 offset);
```

Basic Function Prototype

```
Function rssism_setRFFreqOffset (
    ByVal instrumentHandle As ViSession,
    ByVal offset As ViReal64) As ViStatus
```

Purpose

This function sets the frequency offset. The frequency offset virtually shifts resulting frequency.

The RF frequency applied to the signal generator is calculated from the frequency and offset values as follows:

RF frequency = (frequency - offset)

Where RF frequency is the value sent to the instrument, frequency and offset are values entered to the functions.

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 offset [in]**

   This control sets frequency offset.

   Valid Range: -50.0 GHz to 50.0 GHz

   Default Value: 0.0 Hz

Return Value

Returns the status code of this operation.

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

2.2.3.1.2.4 Get RF Frequency Offset

C Function Prototype

```
ViStatus rssism_getRFFreqOffset (
    ViSession instrumentHandle,
    ViReal64* offset);
```

Basic Function Prototype

```
Function rssism_getRFFreqOffset (
    ByVal instrumentHandle As ViSession,
    offset As ViReal64) As ViStatus
```

Purpose

This function returns the frequency offset.

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 offset [out]**

   This control returns the frequency offset in Hz.

Return Value

Returns the status code of this operation.

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

2.2.3.1.2.5 Set RF Frequency Mode

C Function Prototype

```
ViStatus rssism_setRFFreqMode (
    ViSession instrumentHandle,
    ViInt32 operationMode);
```

Basic Function Prototype

```
Function rssism_setRFFreqMode (
    ByVal instrumentHandle As ViSession,
    ByVal operationMode As ViInt32) As ViStatus
```

Purpose

This function sets the RF frequency operation 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. **ViInt32 operationMode [in]**

   The control specifies the RF frequency operation mode.

   Valid Range:

   - 0 - CW (Fixed)
   - 1 - Sweep

   Default Value: 0

Note When CW (Fixed) frequency mode is selected, instrument stops sweeping.

Return Value

Returns the status code of this operation.

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

2.2.3.1.2.6 Get RF Frequency Mode

C Function Prototype

```
ViStatus rssism_getRFFreqMode (
    ViSession instrumentHandle,
    ViInt32* operationMode);
```

Basic Function Prototype

```
Function rssism_getRFFreqMode (
    ByVal instrumentHandle As ViSession,
    operationMode As ViInt32) As ViStatus
```

Purpose

This function returns the RF frequency operation 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. **ViInt32 operationMode [out]**

   The control returns the RF frequency operation mode.

   Valid Range:

   - 0 - CW (Fixed)
   - 1 - Sweep

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 RF Frequency Sweep

Description

This class operates the RF Frequency Sweep.

Functions/SubClasses:

1. **Configure RF Frequency Sweep:**
   This function configures the RF Frequency Sweep.
2. **Low-Level: (Class)**
   Functions in this class operate all RF Frequency Sweep parameters.

2.2.3.2.1 Configure RF Frequency Sweep

C Function Prototype

```
ViStatus rssism_configRFFreqSweep (
    ViSession instrumentHandle,
    ViInt32 sweepMode,
    ViReal64 startFrequency,
    ViReal64 stopFrequency,
    ViReal64 stepWidth,
    ViReal64 dwellTime);
```

Basic Function Prototype

```
Function rssism_configRFFreqSweep (
    ByVal instrumentHandle As ViSession,
    ByVal sweepMode As ViInt32,
    ByVal startFrequency As ViReal64,
    ByVal stopFrequency As ViReal64,
    ByVal stepWidth As ViReal64,
    ByVal dwellTime As ViReal64) As ViStatus
```

Purpose

This function configures the RF frequency sweep parameters.

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 sweepMode [in]**
   The control specifies the run of the sweep (this parameter is used only for compatibility reason and has no influence to the instrument settings).

   Valid Range:

   - 0 - Auto (Default Value)
   - 1 - Manual
   - 2 - Step (RESERVED)

Note

- Auto:
  Each trigger triggers exactly one entire sweep cycle.
- Manual:
  Each frequency step of the sweep is provided by means of manual control.
- Step:
  Each trigger triggers only one sweep step (single-step mode).

3. **ViReal64 startFrequency [in]**
   This control sets the starting value of frequency for the sweep operation.
   Valid Range (Offset = 0.0): 9.0e3 Hz to 3.0e9 Hz
   Default Value: 100.0e6 Hz
4. **ViReal64 stopFrequency [in]**
   This control sets the stop value of frequency for the sweep operation.
   Valid Range (Offset = 0.0): 9.0e3 Hz to 3.0e9 Hz
   Default Value: 3.0e9 Hz
5. **ViReal64 stepWidth [in]**
   This control sets linear step width for the frequency setting.
   Valid Range: 0.1 Hz to 1.0e9 Hz
   Default Value: 1.0e6 Hz
6. **ViReal64 dwellTime [in]**
   This control sets the dwell time per frequency step.
   Valid Range: 0.01 s to 1.0 s
   Default Value: 0.01 s

Return Value Returns the status code of this operation.
The meaning of the status code is described in section Error (Status) Codes.

2.2.3.2.2 Low-Level

Description Functions in this class operate all RF Frequency Sweep parameters.

2.2.3.2.2.1 Set RF Freq Sweep Mode

C Function Prototype
```
ViStatus rssism_setRFFreqSweepMode (
    ViSession instrumentHandle,
    ViInt32 sweepMode);
```

Basic Function Prototype
```
Function rssism_setRFFreqSweepMode (
    ByVal instrumentHandle As ViSession,
    ByVal sweepMode As ViInt32) As ViStatus
```

Purpose This function specifies the RF sweep 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. **ViInt32 sweepMode [in]**
   The control specifies the RF sweep mode (this parameter is used only for compatibility reason and has no influence to the instrument settings).
   Valid Range:
   - 0 - Auto (Default Value)
   - 1 - Manual
   - 2 - Step (RESERVED)

Note

- Auto:
  Each trigger triggers exactly one entire sweep cycle.
- Manual:
  Each frequency step of the sweep is provided by means of manual control.
- Step:
  Each trigger triggers only one sweep step (single-step mode).

Return Value

Returns the status code of this operation.

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

2.2.3.2.2.2 Get RF Freq Sweep Mode

C Function Prototype

```
ViStatus rssism_getRFFreqSweepMode (
    ViSession instrumentHandle,
    ViInt32* sweepMode);
```

Basic Function Prototype

```
Function rssism_getRFFreqSweepMode (
    ByVal instrumentHandle As ViSession,
    sweepMode As ViInt32) As ViStatus
```

Purpose

This function returns the RF sweep 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. **ViInt32 sweepMode [out]**

   The control returns the RF sweep mode.

   Valid Range:

   - 0 - Auto
   - 1 - Manual
   - 2 - Step (RESERVED)

Note

- Auto:
  Each trigger triggers exactly one entire sweep cycle.
- Manual:
  Each frequency step of the sweep is provided by means of manual control.
- Step:
  Each trigger triggers only one sweep step (single-step mode).

Return Value

Returns the status code of this operation.

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

2.2.3.2.2.3 Set RF Start Frequency

C Function Prototype

```
ViStatus rssism_setRFStartFreq (
    ViSession instrumentHandle,
    ViReal64 startFrequency);
```

Basic Function Prototype

```
Function rssism_setRFStartFreq (
    ByVal instrumentHandle As ViSession,
    ByVal startFrequency As ViReal64) As ViStatus
```

Purpose

This function sets the starting value of frequency for the sweep operation.

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 startFrequency [in]**

   This control sets the starting value of frequency for the sweep operation.

   Valid Range (Offset = 0.0): 9.0e3 Hz to 3.0e9 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.

2.2.3.2.2.4 Get RF Start Frequency

C Function Prototype

```
ViStatus rssism_getRFStartFreq (
    ViSession instrumentHandle,
    ViReal64* startFrequency);
```

Basic Function Prototype

```
Function rssism_getRFStartFreq (
    ByVal instrumentHandle As ViSession,
    startFrequency As ViReal64) As ViStatus
```

Purpose

This function returns the starting value of frequency for the sweep operation.

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 startFrequency [out]**

   This control returns the starting value of frequency for the sweep operation in Hz.

Return Value

Returns the status code of this operation.

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

2.2.3.2.2.5 Set RF Stop Frequency

C Function Prototype

```
ViStatus rssism_setRFStopFreq (
    ViSession instrumentHandle,
    ViReal64 stopFrequency);
```

Basic Function Prototype

```
Function rssism_setRFStopFreq (
    ByVal instrumentHandle As ViSession,
    ByVal stopFrequency As ViReal64) As ViStatus
```

Purpose

This function sets the stop value of RF frequency for the sweep operation.

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 stopFrequency [in]**

   This control sets the stop value of RF frequency for the sweep operation.

   Valid Range (Offset = 0.0): 9.0e3 Hz to 3.0e9 Hz

   Default Value: 3.0e9 Hz

Return Value

Returns the status code of this operation.

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

2.2.3.2.2.6 Get RF Stop Frequency

C Function Prototype

```
ViStatus rssism_getRFStopFreq (
    ViSession instrumentHandle,
    ViReal64* stopFrequency);
```

Basic Function Prototype

```
Function rssism_getRFStopFreq (
    ByVal instrumentHandle As ViSession,
    stopFrequency As ViReal64) As ViStatus
```

Purpose

This function returns the stop value of RF frequency for the sweep operation.

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 stopFrequency [out*]***

   This control returns the stop value of RF frequency for the sweep operation in Hz.

Return Value

Returns the status code of this operation.

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

2.2.3.2.2.7 Set RF Freq Sweep Dwell Time

C Function Prototype

```
ViStatus rssism_setRFFreqSweepDwellTime (
    ViSession instrumentHandle,
    ViReal64 dwellTime);
```

Basic Function Prototype

```
Function rssism_setRFFreqSweepDwellTime (
    ByVal instrumentHandle As ViSession,
    ByVal dwellTime As ViReal64) As ViStatus
```

Purpose

This function sets the dwell time per frequency step.

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 dwellTime [in]**

   This control sets the dwell time per frequency step.

   Valid Range: 0.01 s to 1.0 s

   Default Value: 0.01 s

Return Value

Returns the status code of this operation.

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

2.2.3.2.2.8 Get RF Freq Sweep Dwell Time

C Function Prototype

```
ViStatus rssism_getRFFreqSweepDwellTime (
    ViSession instrumentHandle,
    ViReal64* dwellTime);
```

Basic Function Prototype

```
Function rssism_getRFFreqSweepDwellTime (
    ByVal instrumentHandle As ViSession,
    dwellTime As ViReal64) As ViStatus
```

Purpose

This function returns the dwell time per frequency step.

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 dwellTime [out]**

   This control returns the dwell time per frequency step in sec.

Return Value

Returns the status code of this operation.

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

2.2.3.2.2.9 Set RF Freq Sweep Spacing

C Function Prototype

```
ViStatus rssism_setRFFreqSweepSpacing (
    ViSession instrumentHandle,
    ViInt32 spacing);
```

Basic Function Prototype

```
Function rssism_setRFFreqSweepSpacing (
    ByVal instrumentHandle As ViSession,
    ByVal spacing As ViInt32) As ViStatus
```

Purpose

This function selects whether the steps have linear or logarithmic spacings.

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 spacing [in]**

   This control selects whether the steps have linear or logarithmic spacings.

   Valid Range:

   - 0 - Linear
   - 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.

2.2.3.2.2.10 Get RF Freq Sweep Spacing

C Function Prototype

```
ViStatus rssism_getRFFreqSweepSpacing (
    ViSession instrumentHandle,
    ViInt32* spacing);
```

Basic Function Prototype

```
Function rssism_getRFFreqSweepSpacing (
    ByVal instrumentHandle As ViSession,
    spacing As ViInt32) As ViStatus
```

Purpose

This function returns whether the steps have linear or logarithmic spacings.

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 spacing [out]**

   This control returns whether the steps have linear or logarithmic spacings.

   Valid Range:

   - 0 - Linear
   - 1 - Logarithmic

Return Value

Returns the status code of this operation.

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

2.2.3.2.2.11 Set RF Freq Sweep Step

C Function Prototype

```
ViStatus rssism_setRFFreqSweepStep (
    ViSession instrumentHandle,
    ViInt32 mode,
    ViReal64 step);
```

Basic Function Prototype

```
Function rssism_setRFFreqSweepStep (
    ByVal instrumentHandle As ViSession,
    ByVal mode As ViInt32,
    ByVal step As ViReal64) As ViStatus
```

Purpose

This function sets the step width with the linear sweep or logarithmic 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 mode [in]**

   This control sets the step linear or logarithmic.

   Valid Range:

   - 0 - Linear
   - 1 - Logarithmic

   Default Value: 0

3. **ViReal64 step [in]**

   This control sets the step width with the linear sweep or logarithmic sweep.

   Valid Range:

   Linear sweep: 0.1 Hz to 1.0e9 Hz

   Logarithmic sweep: 0.01 % to 100.0 %

   Default Value: 1.0

Return Value

Returns the status code of this operation.

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

2.2.3.2.2.12 Get RF Freq Sweep Step

C Function Prototype

```
ViStatus rssism_getRFFreqSweepStep (
    ViSession instrumentHandle,
    ViInt32* mode,
    ViReal64* step);
```

Basic Function Prototype

```
Function rssism_getRFFreqSweepStep (
    ByVal instrumentHandle As ViSession,
    mode As ViInt32,
    step As ViReal64) As ViStatus
```

Purpose

This function sets the step width with the linear sweep or logarithmic 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 mode [out]**
   This control returns the step linear or logarithmic.
   Valid Range:
   - 0 - Linear
   - 1 - Logarithmic
3. **ViReal64 step [out]**
   This control returns the step width with the linear sweep (in Hz) or logarithmic sweep (in %).

Return Value

Returns the status code of this operation.

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

2.2.3.2.2.13 Set RF Sweep Frequency Manual

C Function Prototype

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

Basic Function Prototype

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

Purpose

This function sets the RF frequency in manual sweep mode.

Note

This function changes settings done by Set CW Frequency (rssism_setCWFreq) 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 frequency [in]**
   This control sets the RF frequency in manual sweep mode.
   Valid Range (Offset = 0.0): 9.0e3 Hz to 3.0e9 Hz
   Default Value: 1.5e9 Hz

Return Value

Returns the status code of this operation.

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

2.2.3.2.2.14 Get RF Sweep Frequency Manual

C Function Prototype

```
ViStatus rssism_getRFSweepFreqManual (
    ViSession instrumentHandle,
    ViReal64* frequency);
```

Basic Function Prototype

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

Purpose

This function returns the RF frequency in manual sweep 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. **ViReal64 frequency [out]**

   This control returns the RF frequency in manual sweep mode in Hz.

Return Value

Returns the status code of this operation.

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

2.2.3.2.2.15 Set RF Center Frequency

C Function Prototype

```
ViStatus rssism_setRFCenterFreq (
    ViSession instrumentHandle,
    ViReal64 centerFrequency);
```

Basic Function Prototype

```
Function rssism_setRFCenterFreq (
    ByVal instrumentHandle As ViSession,
    ByVal centerFrequency As ViReal64) As ViStatus
```

Purpose

This function sets the sweep range by means of the 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 centerFrequency [in]**

   This control sets the sweep range by means of the center frequency.

   Valid Range (Offset = 0.0): 9.0e3 Hz to 3.0e9 Hz

   Default Value: 1.5e9 Hz

Return Value

Returns the status code of this operation.

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

2.2.3.2.2.16 Get RF Center Frequency

C Function Prototype

```
ViStatus rssism_getRFCenterFreq (
    ViSession instrumentHandle,
    ViReal64* centerFrequency);
```

Basic Function Prototype

```
Function rssism_getRFCenterFreq (
    ByVal instrumentHandle As ViSession,
    centerFrequency As ViReal64) As ViStatus
```

Purpose

This function returns the sweep range by means of the 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 centerFrequency [out]**

   This control returns the sweep range by means of the center 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.

2.2.3.2.2.17 Set RF Span Frequency

C Function Prototype

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

Basic Function Prototype

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

Purpose

This function specifies the frequency range 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. **ViReal64 frequencySpan [in]**

   This control specifies the frequency range for the sweep.

   Valid Range (Offset = 0.0): 9.0e3 Hz to 3.0e9 Hz

   Default Value: 3.0e9 Hz

Return Value

Returns the status code of this operation.

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

2.2.3.2.2.18 Get RF Span Frequency

C Function Prototype

```
ViStatus rssism_getRFSpanFreq (
    ViSession instrumentHandle,
    ViReal64* frequencySpan);
```

Basic Function Prototype

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

Purpose

This function returns the frequency range 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. ViReal64 frequencySpan [out]

This control returns the frequency range for the sweep in Hz.

Return Value

Returns the status code of this operation.

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

2.2.3.3 RF Level

Description

This class operates the RF Level.

Functions/SubClasses:

1. **Configure RF Level:**
   This function configures RF Level.
2. **Low-Level: (Class)**
   Functions in this class operate all RF Level parameters.

2.2.3.3.1 Configure RF Level

C Function Prototype

```
ViStatus rssism_configRFLevel (
    ViSession instrumentHandle,
    ViReal64 level,
    ViReal64 offset,
    ViReal64 limit);
```

Basic Function Prototype

```
Function rssism_configRFLevel (
    ByVal instrumentHandle As ViSession,
    ByVal level As ViReal64,
    ByVal offset As ViReal64,
    ByVal limit As ViReal64) As ViStatus
```

Purpose

This function configures the RF Level.

Note

The level of RF output signal is calculated from the amplitude and offset as follows:

RF output level = amplitude - offset

Where amplitude is the displayed value, RF output level and offset are entered values.

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 level [in]**
   This control sets level of RF output signal.
   Valid Range (Offset = 0.0): -127.0 dBm to 13.0 dBm
   Default Value: 0.0 dBm
3. **ViReal64 offset [in]**
   This control enters offset of RF output signal.
   Default Value: 0.0 dB
   Valid Range: -100.0 dB to 100.0 dB
4. **ViReal64 limit [in]**
   This control sets limitation range of the RF output level settings.
   Valid Range: -127.0 dBm to 13.0 dBm
   Default Value: 13.0 dBm

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 Low-Level

Description Functions in this class operate all RF Level parameters.

2.2.3.3.2.1 Set RF Level

C Function Prototype

```
ViStatus rssism_setRFLevel (
    ViSession instrumentHandle,
    ViReal64 level);
```

Basic Function Prototype

```
Function rssism_setRFLevel (
    ByVal instrumentHandle As ViSession,
    ByVal level As ViReal64) As ViStatus
```

Purpose This function sets level of RF output 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. **ViReal64 level [in]**

   This control sets level of RF output signal.

   Valid Range (Offset = 0.0): -127.0 dBm to 13.0 dBm

   Default Value: 0.0 dBm

Return Value Returns the status code of this operation.

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

2.2.3.3.2.2 Get RF Level

C Function Prototype

```
ViStatus rssism_getRFLevel (
    ViSession instrumentHandle,
    ViReal64* level);
```

Basic Function Prototype

```
Function rssism_getRFLevel (
    ByVal instrumentHandle As ViSession,
    level As ViReal64) As ViStatus
```

Purpose This function returns level of RF output 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. **ViReal64 level [out]**

   This control returns level of RF output signal in dBm.

Return Value Returns the status code of this operation.

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

2.2.3.3.2.3 Set RF Level Offset

C Function Prototype

```
ViStatus rssism_setRFLevelOffset (
    ViSession instrumentHandle,
    ViReal64 offset);
```

Basic Function Prototype

```
Function rssism_setRFLevelOffset (
    ByVal instrumentHandle As ViSession,
    ByVal offset As ViReal64) As ViStatus
```

Purpose

This function enters offset of RF output signal.

Note

The level of RF output signal is calculated from the amplitude and offset as follows:

RF output level = amplitude - offset

Where amplitude is the displayed value, RF output level and offset are entered values.

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 offset [in]**

   This control enters offset of RF output signal.

   Default Value: 0.0 dB

   Valid Range: -100.0 dB to 100.0 dB

Return Value

Returns the status code of this operation.

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

2.2.3.3.2.4 Get RF Level Offset

C Function Prototype

```
ViStatus rssism_getRFLevelOffset (
    ViSession instrumentHandle,
    ViReal64* offset);
```

Basic Function Prototype

```
Function rssism_getRFLevelOffset (
    ByVal instrumentHandle As ViSession,
    offset As ViReal64) As ViStatus
```

Purpose

This function returns offset of RF output 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. **ViReal64 offset [out]**

   This control returns offset of RF output signal in dB.

Return Value

Returns the status code of this operation.

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

2.2.3.3.2.5 Set RF Level Limit

C Function Prototype

```
ViStatus rssism_setRFLevelLimit (
    ViSession instrumentHandle,
    ViReal64 limit);
```

Basic Function Prototype

```
Function rssism_setRFLevelLimit (
    ByVal instrumentHandle As ViSession,
    ByVal limit As ViReal64) As ViStatus
```

Purpose

This function sets limitation range of RF output level 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 limit [in]**

   This control sets limitation range of RF output level settings.

   Valid Range: -127.0 dBm to 13.0 dBm

   Default Value: 0.0 dBm

Return Value

Returns the status code of this operation.

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

2.2.3.3.2.6 Get RF Level Limit

C Function Prototype

```
ViStatus rssism_getRFLevelLimit (
    ViSession instrumentHandle,
    ViReal64* limit);
```

Basic Function Prototype

```
Function rssism_getRFLevelLimit (
    ByVal instrumentHandle As ViSession,
    limit As ViReal64) As ViStatus
```

Purpose

This function returns limitation range of RF output level 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 limit [out]**

   This control returns limitation range of RF output level settings in dBm.

Return Value

Returns the status code of this operation.

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

2.2.3.3.2.7 Set RF Level Mode

C Function Prototype

```
ViStatus rssism_setRFLevelMode (
    ViSession instrumentHandle,
    ViInt32 operationMode);
```

Basic Function Prototype

```
Function rssism_setRFLevelMode (
    ByVal instrumentHandle As ViSession,
    ByVal operationMode As ViInt32) As ViStatus
```

Purpose

This function sets RF level operation 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. **ViInt32 operationMode [in]**

   The control specifies RF level operation mode.

   Valid Range:

   - 0 - CW(Fixed) (Default Value)
   - 1 - Sweep

Note When CW (Fixed) level mode is selected, instrument stops sweeping.

Return Value

Returns the status code of this operation.

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

2.2.3.3.2.8 Get RF Level Mode

C Function Prototype

```
ViStatus rssism_getRFLevelMode (
    ViSession instrumentHandle,
    ViInt32* operationMode);
```

Basic Function Prototype

```
Function rssism_getRFLevelMode (
    ByVal instrumentHandle As ViSession,
    operationMode As ViInt32) As ViStatus
```

Purpose

This function returns the RF level operation 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. **ViInt32 operationMode [out]**

   The control returns the RF level operation mode.

   Valid Range:

   - 0 - CW(Fixed)
   - 1 - Sweep

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 RF Level Sweep

Description

This class operates the RF Level Sweep.

Functions/SubClasses:

1. **Configure RF Level Sweep:**
   This function configures the RF Level Sweep.
2. **Low-Level: (Class)**
   Functions in this class operate all RF Level Sweep parameters.

2.2.3.4.1 Configure RF Level Sweep

C Function Prototype

```
ViStatus rssism_configRFLevelSweep (
    ViSession instrumentHandle,
    ViInt32 sweepMode,
    ViReal64 startLevel,
    ViReal64 stopLevel,
    ViReal64 step,
    ViReal64 dwellTime);
```

Basic Function Prototype

```
Function rssism_configRFLevelSweep (
    ByVal instrumentHandle As ViSession,
    ByVal sweepMode As ViInt32,
    ByVal startLevel As ViReal64,
    ByVal stopLevel As ViReal64,
    ByVal step As ViReal64,
    ByVal dwellTime As ViReal64) As ViStatus
```

Purpose

This function configures the RF Level Sweep parameters.

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 sweepMode [in]**
   This control specifies the sweep mode (this parameter is used only for compatibility reason and has no influence to the instrument settings).
   Valid Range:
   - 0 - Auto
   - 1 - Manual
   - 2 - Step (RESERVED)

Note

- Auto:
  Each trigger triggers exactly one entire sweep cycle.
- Manual:
  Each level step of the sweep is provided by means of manual control.
- Step:
  Each trigger triggers only one sweep step (single-step mode).

3. **ViReal64 startLevel [in]**
   This control sets the staring value for the RF level sweep.
   Valid Range (Offset = 0.0): -127.0 dBm to 13.0 dBm
   Default Value: -40.0 dBm
4. **ViReal64 stopLevel [in]**
   This control sets the stop value for the RF level sweep.
   Valid Range (Offset = 0.0): -127.0 dBm to 13.0 dBm
   Default Value: 0.0 dBm
5. **ViReal64 step [in]**
   This control defines the step width factor for the logarithmic sweeps.
   Valid Range: 0.1 dB to 20.0 dB
   Default Value: 1.0 dB
6. **ViReal64 dwellTime [in]**
   This control sets dwell time per level step.
   Default Value: 0.01 s
   Valid Range: 0.01 s to 1.0 s

Return Value

Returns the status code of this operation.

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

2.2.3.4.2 Low-Level

Description Functions in this class operate all RF Level Sweep parameters.

2.2.3.4.2.1 Set RF Level Sweep Mode

C Function Prototype

```
ViStatus rssism_setRFLevelSweepMode (
    ViSession instrumentHandle,
    ViInt32 sweepMode);
```

Basic Function Prototype

```
Function rssism_setRFLevelSweepMode (
    ByVal instrumentHandle As ViSession,
    ByVal sweepMode As ViInt32) As ViStatus
```

Purpose This function specifies the sweep 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. **ViInt32 sweepMode [in]**

   This control specifies the sweep mode (this parameter is used only for compatibility reason and has no influence to the instrument settings).

   Valid Range:

   - 0 - Auto
   - 1 - Manual
   - 2 - Step (RESERVED)

   Default Value: 0

Note

- Auto:
  Each trigger triggers exactly one entire sweep cycle.
- Manual:
  Each level step of the sweep is provided by means of manual control.
- Step:
  Each trigger triggers only one sweep step (single-step mode).

Return Value Returns the status code of this operation.

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

2.2.3.4.2.2 Get RF Level Sweep Mode

C Function Prototype

```
ViStatus rssism_getRFLevelSweepMode (
    ViSession instrumentHandle,
    ViInt32* sweepMode);
```

Basic Function Prototype

```
Function rssism_getRFLevelSweepMode (
    ByVal instrumentHandle As ViSession,
    sweepMode As ViInt32) As ViStatus
```

Purpose This function returns the sweep 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. **ViInt32 sweepMode [out]**
   This control returns the sweep mode.
   Valid Range:
   - 0 - Auto
   - 1 - Manual
   - 2 - Step (RESERVED)

Note

- Auto:
  Each trigger triggers exactly one entire sweep cycle.
- Manual:
  Each level step of the sweep is provided by means of manual control.
- Step:
  Each trigger triggers only one sweep step (single-step mode).

Return Value

Returns the status code of this operation.

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

2.2.3.4.2.3 Set RF Start Level

C Function Prototype

```
ViStatus rssism_setRFStartLevel (
    ViSession instrumentHandle,
    ViReal64 startLevel);
```

Basic Function Prototype

```
Function rssism_setRFStartLevel (
    ByVal instrumentHandle As ViSession,
    ByVal startLevel As ViReal64) As ViStatus
```

Purpose

This function sets the staring value for the level 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. **ViReal64 startLevel [in]**
   This control sets the staring value for the RF level sweep.
   Valid Range (Offset = 0.0): -127.0 dBm to 13.0 dBm
   Default Value: -40.0 dBm

Return Value

Returns the status code of this operation.

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

2.2.3.4.2.4 Get RF Start Level

C Function Prototype

```
ViStatus rssism_getRFStartLevel (
    ViSession instrumentHandle,
    ViReal64* startLevel);
```

Basic Function Prototype

```
Function rssism_getRFStartLevel (
    ByVal instrumentHandle As ViSession,
    startLevel As ViReal64) As ViStatus
```

Purpose

This function returns the starting value for the level 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. **ViReal64 startLevel [out]**

   This control returns the starting value for the RF level sweep in dBm.

Return Value

Returns the status code of this operation.

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

2.2.3.4.2.5 Set RF Stop Level

C Function Prototype

```
ViStatus rssism_setRFStopFreqLevel (
    ViSession instrumentHandle,
    ViReal64 stopLevel);
```

Basic Function Prototype

```
Function rssism_setRFStopFreqLevel (
    ByVal instrumentHandle As ViSession,
    ByVal stopLevel As ViReal64) As ViStatus
```

Purpose

This function sets the stop value for the level 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. **ViReal64 stopLevel [in]**

   This control sets the stop value for the level sweep.

   Valid Range (Offset = 0.0): -127.0 dBm to 13.0 dBm

   Default Value: 0.0 dBm

Return Value

Returns the status code of this operation.

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

2.2.3.4.2.6 Get RF Stop Level

C Function Prototype

```
ViStatus rssism_getRFStopFreqLevel (
    ViSession instrumentHandle,
    ViReal64* stopLevel);
```

Basic Function Prototype

```
Function rssism_getRFStopFreqLevel (
    ByVal instrumentHandle As ViSession,
    stopLevel As ViReal64) As ViStatus
```

Purpose

This function returns the stop value for the level 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. **ViReal64 stopLevel [out]**

   This control sets the stop value for the level sweep in dBm.

Return Value

Returns the status code of this operation.

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

2.2.3.4.2.7 Set RF Level Sweep Step

C Function Prototype

```
ViStatus rssism_setRFLevelSweepStep (
    ViSession instrumentHandle,
    ViReal64 step);
```

Basic Function Prototype

```
Function rssism_setRFLevelSweepStep (
    ByVal instrumentHandle As ViSession,
    ByVal step As ViReal64) As ViStatus
```

Purpose

This function defines the step width factor for the logarithmic sweeps.

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 step [in]**

   This control defines the step width factor for the logarithmic sweeps.

   Valid Range: 0.1 dB to 20.0 dB

   Default Value: 1.0 dB

Return Value

Returns the status code of this operation.

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

2.2.3.4.2.8 Get RF Level Sweep Step

C Function Prototype

```
ViStatus rssism_getRFLevelSweepStep (
    ViSession instrumentHandle,
    ViReal64* step);
```

Basic Function Prototype

```
Function rssism_getRFLevelSweepStep (
    ByVal instrumentHandle As ViSession,
    step As ViReal64) As ViStatus
```

Purpose

This function returns the step width factor for the logarithmic sweeps.

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 step [out]**

   This control returns the step width factor for the logarithmic sweeps in dB.

Return Value

Returns the status code of this operation.

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

2.2.3.4.2.9 Set RF Level Sweep Dwell Time

C Function Prototype

```
ViStatus rssism_setRFLevelSweepDwellTime (
    ViSession instrumentHandle,
    ViReal64 time);
```

Basic Function Prototype

```
Function rssism_setRFLevelSweepDwellTime (
    ByVal instrumentHandle As ViSession,
    ByVal time As ViReal64) As ViStatus
```

Purpose

This function sets the dwell time per level step.

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 time [in]**

   This control sets the dwell time per level step.

   Default Value: 0.01 s

   Valid Range: 0.01 s to 1.0 s

Return Value

Returns the status code of this operation.

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

2.2.3.4.2.10 Get RF Level Sweep Dwell Time

C Function Prototype

```
ViStatus rssism_getRFLevelSweepDwellTime (
    ViSession instrumentHandle,
    ViReal64* time);
```

Basic Function Prototype

```
Function rssism_getRFLevelSweepDwellTime (
    ByVal instrumentHandle As ViSession,
    time As ViReal64) As ViStatus
```

Purpose

This function returns the dwell time per level step in sec.

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 time [out]**

   This control returns the dwell time per level step.

Return Value

Returns the status code of this operation.

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

2.2.3.4.2.11 Set RF Level Sweep Manual

C Function Prototype

```
ViStatus rssism_setRFLevelSweepManual (
    ViSession instrumentHandle,
    ViReal64 level);
```

Basic Function Prototype

```
Function rssism_setRFLevelSweepManual (
    ByVal instrumentHandle As ViSession,
    ByVal level As ViReal64) As ViStatus
```

Purpose

This function sets the RF level value for manual sweep.

Note

This function changes settings done by Set RF Level (rssism_setRFLevel) 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 level [in]**

   This control sets the level value for manual sweep.

   Valid Range (Offset = 0.0): -127.0 dBm to 13.0 dBm

   Default Value: 0.0 dBm

Return Value

Returns the status code of this operation.

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

2.2.3.4.2.12 Get RF Level Sweep Manual

C Function Prototype

```
ViStatus rssism_getRFLevelSweepManual (
    ViSession instrumentHandle,
     ViReal64* level);
```

Basic Function Prototype

```
Function rssism_getRFLevelSweepManual (
    ByVal instrumentHandle As ViSession,
    level As ViReal64) As ViStatus
```

Purpose

This function returns the RF level value for manual 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. **ViReal64 level [out]**

   This control returns the level value for manual sweep in dBm.

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 RF Output

Description

This class operates the RF Output.

Functions/SubClasses:

1. **Set RF Output State:**
   This function switches on or off the RF output.
2. **Get RF Output State:**
   This function returns state of the RF output.

2.2.3.5.1 Set RF Output State

C Function Prototype

```
ViStatus rssism_setRFOutputState (
    ViSession instrumentHandle,
    ViBoolean outputState);
```

Basic Function Prototype

```
Function rssism_setRFOutputState (
    ByVal instrumentHandle As ViSession,
    ByVal outputState As ViBoolean) As ViStatus
```

Purpose

This function switches on or off the RF 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. **ViBoolean outputState [in]**
   This control switches on or off the RF output.
   Valid Range:
   - VI_FALSE (0) - Off (Default Value)
   - VI_TRUE (1) - On

Return Value

Returns the status code of this operation.

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

2.2.3.5.2 Get RF Output State

C Function Prototype

```
ViStatus rssism_getRFOutputState (
    ViSession instrumentHandle,
    ViBoolean* outputState);
```

Basic Function Prototype

```
Function rssism_getRFOutputState (
    ByVal instrumentHandle As ViSession,
    outputState As ViBoolean) As ViStatus
```

Purpose

This function returns state of the RF 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. **ViBoolean outputState [out]**

   This control returns state of the RF output.

   Valid Range:

   - VI_FALSE (0) - Off
   - VI_TRUE (1) - On

Return Value

Returns the status code of this operation.

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

2.2.3.6 RF Modulation

Description

This class operates the RF Modulation.

Functions/Classes:

1. **Configure AM Modulation:**
   This function configures AM modulation and switches it to on state.
2. **Configure FM Modulation:**
   This function configures FM modulation and switches it to on state.
3. **Configure Phase Modulation:**
   This function configures phase modulation and switches it to on state.
4. **Configure Pulse Modulation:**
   This function configures pulse modulation and switches it to on state.
5. **Low-Level AM Modulation: (Class)**
   Functions in this class operate all AM Modulation parameters.
6. **Low-Level FM Modulation: (Class)**
   Functions in this class operate all FM Modulation parameters.
7. **Low-Level Phase Modulation: (Class)**
   Functions in this class operate all Phase Modulation parameters.
8. **Low-Level Pulse Modulation: (Class)**
   Functions in this class operate all Pulse Modulation parameters.
9. **Low-Level Vector Modulation: (Class)**
   This subsystem contains the functions to control the vector modulation and to set the parameters of the modulation signal.

2.2.3.6.1 Configure AM Modulation

C Function Prototype

```
ViStatus rssism_configAMModulation (
    ViSession instrumentHandle,
    ViInt32 modulationSource,
    ViReal64 modulationDepth,
    ViReal64 frequency);
```

Basic Function Prototype

```
Function rssism_configAMModulation (
    ByVal instrumentHandle As ViSession,
    ByVal modulationSource As ViInt32,
    ByVal modulationDepth As ViReal64,
    ByVal frequency As ViReal64) As ViStatus
```

Purpose

This function configures AM modulation parameters and switches modulator to on 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. **ViInt32 modulationSource [in]**

   This control selects the modulation source.

   Valid Range:

   - 0 - Internal
   - 1 - External
   - 2 - Internal + External

   Default Value: 0

3. **ViReal64 modulationDepth [in]**

   This control sets the modulation depth in percent.

   Valid Range: 0.0 % to 100.0 %

   Default Value: 100.0 %

4. **ViReal64 frequency [in]**

   This control sets the modulation frequency.

   Valid Range: 1.0 Hz to 80.0e3 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.

2.2.3.6.2 Configure FM Modulation

C Function Prototype

```
ViStatus rssism_configFMModulation (
    ViSession instrumentHandle,
    ViInt32 modulationSource,
    ViReal64 deviation,
    ViReal64 frequency);
```

Basic Function Prototype

```
Function rssism_configFMModulation (
    ByVal instrumentHandle As ViSession,
    ByVal modulationSource As ViInt32,
    ByVal deviation As ViReal64,
    ByVal frequency As ViReal64) As ViStatus
```

Purpose

This function configures FM modulation parameters and switches modulator to on 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. **ViInt32 modulationSource [in]**

   This control selects the modulation source.

   Valid Range:

   - 0 - Internal
   - 1 - External
   - 2 - Internal + External

   Default Value: 0

3. **ViReal64 deviation [in]**

   This control specifies the frequency variation caused by the FM.

   Valid Range: 20.0 Hz to 100.0e3 Hz

   Default Value: 20.0 Hz

4. **ViReal64 frequency [in]**

   This control sets the modulation frequency.

   Valid Range: 1.0 Hz to 80.0e3 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.

2.2.3.6.3 Configure Phase Modulation

C Function Prototype

```
ViStatus rssism_configPHMModulation (
    ViSession instrumentHandle,
    ViInt32 modulationSource,
    ViReal64 deviation,
    ViReal64 frequency);
```

Basic Function Prototype

```
Function rssism_configPHMModulation (
    ByVal instrumentHandle As ViSession,
    ByVal modulationSource As ViInt32,
    ByVal deviation As ViReal64,
    ByVal frequency As ViReal64) As ViStatus
```

Purpose

This function configures the phase modulation and switches modulator to on 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. **ViInt32 modulationSource [in]**

   This control selects the modulation source.

   Valid Range:

   - 0 – Internal

   Default Value: 0

3. **ViReal64 deviation [in]**

   This control specifies the frequency variation caused by the phase modulation.

   Default Value: 0.0 rad

   Valid Range: 0.0 rad to 10.0 rad

4. **ViReal64 frequency [in]**

   This control sets the modulation frequency.

   Valid Range: 1.0 Hz to 80.0e3 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.

2.2.3.6.4 Configure Pulse Modulation

C Function Prototype

```
ViStatus rssism_configPulseModulation (
    ViSession instrumentHandle,
    ViInt32 signalSource,
    ViInt32 polarity);
```

Basic Function Prototype

```
Function rssism_configPulseModulation (
    ByVal instrumentHandle As ViSession,
    ByVal signalSource As ViInt32,
    ByVal polarity As ViInt32) As ViStatus
```

Purpose

This function configures the pulse modulation and switches modulator to on 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. **ViInt32 signalSource [in]**

   This control selects the source of the modulating signal.

   Valid Range:

   - 0 - Internal
   - 1 - External

   Default Value: 1

Note

- Internal:
  Internal pulse generator.
- External:
  Signal is fed externally.

3. **ViInt32 polarity [in]**

   This control specifies the polarity between modulating and modulated signal.

   Valid Range:

   - 0 - Normal (Default Value)
   - 1 - Inverted

Note

- Normal:
  The RF signal is suppressed during the interpulse period.
- Inverted:
  The RF signal is suppressed during the pulse.

Return Value

Returns the status code of this operation.

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

2.2.3.6.5 Low-Level AM Modulation

Description Functions in this class operate all AM modulation parameters.

2.2.3.6.5.1 Set AM State

C Function Prototype

```
ViStatus rssism_setAMState (
    ViSession instrumentHandle,
    ViBoolean modState);
```

Basic Function Prototype

```
Function rssism_setAMState (
    ByVal instrumentHandle As ViSession,
    ByVal modState As ViBoolean) As ViStatus
```

Purpose This function switches on or off the AM modulation.

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 modState [in]**

   This control switches on or off the AM modulation.

   Valid Range:

   - VI_FALSE (0) - Off (Default Value)
   - VI_TRUE (1) - On

Return Value Returns the status code of this operation.

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

2.2.3.6.5.2 Get AM State

C Function Prototype

```
ViStatus rssism_getAMState (
    ViSession instrumentHandle,
    ViBoolean* modState);
```

Basic Function Prototype

```
Function rssism_getAMState (
    ByVal instrumentHandle As ViSession,
    modState As ViBoolean) As ViStatus
```

Purpose This function returns state of the AM modulator.

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 modState [out]**

   This control returns state of the AM modulator.

   Valid Range:

   - VI_FALSE (0) - Off
   - VI_TRUE (1) - On

Return Value Returns the status code of this operation.

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

2.2.3.6.5.3 Set AM Depth

C Function Prototype

```
ViStatus rssism_setAMDepth (
    ViSession instrumentHandle,
    ViReal64 modulationDepth);
```

Basic Function Prototype

```
Function rssism_setAMDepth (
    ByVal instrumentHandle As ViSession,
    ByVal modulationDepth As ViReal64) As ViStatus
```

Purpose

This function sets the modulation depth in percent.

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 modulationDepth [in]**

   This control sets the 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.

2.2.3.6.5.4 Get AM Depth

C Function Prototype

```
ViStatus rssism_getAMDepth (
    ViSession instrumentHandle,
    ViReal64* modulationDepth);
```

Basic Function Prototype

```
Function rssism_getAMDepth (
    ByVal instrumentHandle As ViSession,
    modulationDepth As ViReal64) As ViStatus
```

Purpose

This function returns the modulation depth in percent.

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 modulationDepth [out]**

   This control returns the 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.

2.2.3.6.5.5 Set AM Frequency

C Function Prototype

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

Basic Function Prototype

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

Purpose

This function sets the modulation 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 frequency [in]**

   This control sets the modulation frequency.

   Valid Range: 1.0 Hz to 80.0e3 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.

2.2.3.6.5.6 Get AM Frequency

C Function Prototype

```
ViStatus rssism_getAMFreq (
    ViSession instrumentHandle,
    ViReal64* frequency);
```

Basic Function Prototype

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

Purpose

This function returns the modulation 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 frequency [out]**

   This control returns the modulation 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.

2.2.3.6.5.7 Set AM Source

C Function Prototype

```
ViStatus rssism_setAMSource (
    ViSession instrumentHandle,
    ViInt32 modulationSource);
```

Basic Function Prototype

```
Function rssism_setAMSource (
    ByVal instrumentHandle As ViSession,
    ByVal modulationSource As ViInt32) As ViStatus
```

Purpose

This function selects the modulation 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. **ViInt32 modulationSource [in]**

   This control selects the modulation source.

   Valid Range:

   - 0 - Internal
   - 1 - External
   - 2 - Internal + 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.

2.2.3.6.5.8 Get AM Source

C Function Prototype

```
ViStatus rssism_getAMSource (
    ViSession instrumentHandle,
    ViInt32* modulationSource);
```

Basic Function Prototype

```
Function rssism_getAMSource (
    ByVal instrumentHandle As ViSession,
    modulationSource As ViInt32) As ViStatus
```

Purpose

This function returns the modulation 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. **ViInt32 modulationSource [out]**

   This control returns the modulation source.

   Valid Range:

   - 0 - Internal
   - 1 - External
   - 2 - Internal + External

Return Value

Returns the status code of this operation.

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

2.2.3.6.5.9 Set AM External Coupling

C Function Prototype

```
ViStatus rssism_setAMExtCoupling (
    ViSession instrumentHandle,
    ViInt32 extInput,
    ViInt32 coupling);
```

Basic Function Prototype

```
Function rssism_setAMExtCoupling (
    ByVal instrumentHandle As ViSession,
    ByVal extInput As ViInt32,
    ByVal coupling As ViInt32) As ViStatus
```

Purpose

This function selects type of coupling for the external AM input.

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 extInput [in]**

   This control selects the external input.

   Default Value: 1

   Valid Range: 1 is only valid value

3. **ViInt32 coupling [in]**

   This control selects type of coupling for the external AM input.

   Valid Range:

   - 0 - AC (Default Value)
   - 1 - DC

Note

- AC:
  The DC voltage content is separated from the modulation signal.
- DC:
  The modulation signal is not altered.

Return Value

Returns the status code of this operation.

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

2.2.3.6.5.10 Get AM External Coupling

C Function Prototype

```
ViStatus rssism_getAMExtCoupling (
    ViSession instrumentHandle,
    ViInt32 extInput,
    ViInt32* coupling);
```

Basic Function Prototype

```
Function rssism_getAMExtCoupling (
    ByVal instrumentHandle As ViSession,
    ByVal extInput As ViInt32,
    coupling As ViInt32) As ViStatus
```

Purpose

This function returns the type of coupling of the external AM input.

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 extInput [in]**
   This control selects the external input.
   Default Value: 1
   Valid Range: 1 is only valid value
3. **ViInt32 coupling [out]**
   This control returns the type of coupling of the external AM input.
   Valid Range:
   - 0 - AC
   - 1 - DC

Note

- AC:
  The DC voltage content is separated from the modulation signal.
- DC:
  The modulation signal is not altered.

Return Value

Returns the status code of this operation.

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

2.2.3.6.5.11 Set AM Polarity

C Function Prototype

```
ViStatus rssism_setAMPolar (
    ViSession instrumentHandle,
    ViInt32 polarity);
```

Basic Function Prototype

```
Function rssism_setAMPolar (
    ByVal instrumentHandle As ViSession,
    ByVal polarity As ViInt32) As ViStatus
```

Purpose

This function specifies the polarity between the modulating and 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 polarity [in]**
   This control specifies the polarity between the modulating and modulated signal.
   Valid Range:
   - 0 - Normal (Default Value)
   - 1 - Inverted

Return Value

Returns the status code of this operation.

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

2.2.3.6.5.12 Get AM Polarity

C Function Prototype

ViStatus rssism_getAMPolar (
ViSession instrumentHandle,
ViInt32* polarity);

Basic Function Prototype

Function rssism_getAMPolar (
ByVal instrumentHandle As ViSession,
polarity As ViInt32) As ViStatus

Purpose

This function returns the polarity between the modulating and 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 polarity [out]**

   This control returns the polarity between the modulating and modulated signal.

   Valid Range:

   - 0 - Normal
   - 1 - Inverted

Return Value

Returns the status code of this operation.

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

2.2.3.6.6 Low-Level FM Modulation

Description

Functions in this class operate all FM modulation parameters.

2.2.3.6.6.1 Set FM State

C Function Prototype

ViStatus rssism_setFMState (
ViSession instrumentHandle,
ViBoolean modState);

Basic Function Prototype

Function rssism_setFMState (
ByVal instrumentHandle As ViSession,
ByVal modState As ViBoolean) As ViStatus

Purpose

This function switches on or off the FM modulation.

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 modState [in]**

   This control switches on or off the FM modulation.

   Valid Range:

   - VI_FALSE (0) - Off (Default Value)
   - VI_TRUE (1) - On

Return Value

Returns the status code of this operation.

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

2.2.3.6.6.2 Get FM State

C Function Prototype

```
ViStatus rssism_getFMState (
    ViSession instrumentHandle,
    ViBoolean* modState);
```

Basic Function Prototype

```
Function rssism_getFMState (
    ByVal instrumentHandle As ViSession,
    modState As ViBoolean) As ViStatus
```

Purpose

This function returns the state of FM modulation.

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 modState [out]**

   This control returns the state of FM modulation.

   Valid Range:

   - VI_FALSE (0) - Off
   - VI_TRUE (1) - On

Return Value

Returns the status code of this operation.

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

2.2.3.6.6.3 Set FM Deviation

C Function Prototype

```
ViStatus rssism_setFMDeviation (
    ViSession instrumentHandle,
    ViReal64 deviation);
```

Basic Function Prototype

```
Function rssism_setFMDeviation (
    ByVal instrumentHandle As ViSession,
    ByVal deviation As ViReal64) As ViStatus
```

Purpose

This function specifies the frequency variation caused by the frequency modulation.

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 deviation [in]**

   This control specifies the frequency variation caused by the FM.

   Valid Range: 20.0 Hz to 100.0e3 Hz

   Default Value: 20.0 Hz

Return Value

Returns the status code of this operation.

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

2.2.3.6.6.4 Get FM Deviation

C Function Prototype

```
ViStatus rssism_getFMDeviation (
    ViSession instrumentHandle,
    ViReal64* deviation);
```

Basic Function Prototype

```
Function rssism_getFMDeviation (
    ByVal instrumentHandle As ViSession,
    deviation As ViReal64) As ViStatus
```

Purpose

This function returns the frequency variation caused by the frequency modulation.

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 deviation [out]**

   This control returns the frequency variation caused by the frequency modulation in Hz.

Return Value

Returns the status code of this operation.

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

2.2.3.6.6.5 Set FM Frequency

C Function Prototype

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

Basic Function Prototype

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

Purpose

This function sets the modulation 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 frequency [in]**

   This control sets the modulation frequency.

   Valid Range: 1.0 Hz to 80.0e3 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.

2.2.3.6.6.6 Get FM Frequency

C Function Prototype

```
ViStatus rssism_getFMFreq (
    ViSession instrumentHandle,
    ViReal64* frequency);
```

Basic Function Prototype

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

Purpose

This function returns the modulation 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 frequency [out]**

   This control returns the modulation 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.

2.2.3.6.6.7 Set FM Source

C Function Prototype

```
ViStatus rssism_setFMSource (
    ViSession instrumentHandle,
    ViInt32 modulationSource);
```

Basic Function Prototype

```
Function rssism_setFMSource (
    ByVal instrumentHandle As ViSession,
    ByVal modulationSource As ViInt32) As ViStatus
```

Purpose

This function selects the modulation 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. **ViInt32 modulationSource [in]**

   This control selects the modulation source.

   Valid Range:

   - 0 - Internal (Default Value)
   - 1 - External
   - 2 - Internal + External

Return Value

Returns the status code of this operation.

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

2.2.3.6.6.8 Get FM Source

C Function Prototype

ViStatus rssism_getFMSource (
ViSession instrumentHandle,
ViInt32* modulationSource);

Basic Function Prototype

Function rssism_getFMSource (
ByVal instrumentHandle As ViSession,
modulationSource As ViInt32) As ViStatus

Purpose

This function returns the modulation 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. **ViInt32 modulationSource [out]**

   This control returns the modulation source.

   Valid Range:

   - 0 - Internal
   - 1 - External
   - 2 - Internal + External

Return Value

Returns the status code of this operation.

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

2.2.3.6.6.9 Set FM External Coupling

C Function Prototype

ViStatus rssism_setFMExtCoupling (
ViSession instrumentHandle,
ViInt32 extInput,
ViInt32 coupling);

Basic Function Prototype

Function rssism_setFMExtCoupling (
ByVal instrumentHandle As ViSession,
ByVal extInput As ViInt32,
ByVal coupling As ViInt32) As ViStatus

Purpose

This function selects the type of coupling for the external FM input.

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 extInput [in]**

   This control selects the external input.

   Default Value: 1

   Valid Range: 1 is only valid value

3. **ViInt32 coupling [in]**

   This control selects the type of coupling for the external FM input.

   Valid Range:

   - 0 - AC (Default Value)
   - 1 - DC

Return Value Returns the status code of this operation.

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

2.2.3.6.6.10 Get FM External Coupling

C Function Prototype

```
ViStatus rssism_getFMExtCoupling (
    ViSession instrumentHandle,
    ViInt32 extInput,
    ViInt32* coupling);
```

Basic Function Prototype

```
Function rssism_getFMExtCoupling (
    ByVal instrumentHandle As ViSession,
    ByVal extInput As ViInt32,
    coupling As ViInt32) As ViStatus
```

Purpose This function returns the type of coupling for the external FM input.

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 extInput [in]**

   This control selects the external input.

   Default Value: 1

   Valid Range: 1 is only valid value

3. **ViInt32 coupling [out]**

   This control returns the type of coupling for the external FM input.

   Valid Range:

   - 0 - AC
   - 1 - DC

Return Value Returns the status code of this operation.

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

2.2.3.6.6.11 Set FM Polarity

C Function Prototype

```
ViStatus rssism_setFMPolar (
    ViSession instrumentHandle,
    ViInt32 polarity);
```

Basic Function Prototype

```
Function rssism_setFMPolar (
    ByVal instrumentHandle As ViSession,
    ByVal polarity As ViInt32) As ViStatus
```

Purpose

This function specifies the polarity between the modulating and 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 polarity [in]**

   This control specifies the polarity between the modulating and modulated signal.

   Valid Range:

   - 0 - Normal (Default Value)
   - 1 - Inverted

Return Value

Returns the status code of this operation.

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

2.2.3.6.6.12 Get FM Polarity

C Function Prototype

```
ViStatus rssism_getFMPolar (
    ViSession instrumentHandle,
    ViInt32* polarity);
```

Basic Function Prototype

```
Function rssism_getFMPolar (
    ByVal instrumentHandle As ViSession,
    polarity As ViInt32) As ViStatus
```

Purpose

This function returns the polarity between the modulating and 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 polarity [out]**

   This control returns the polarity between the modulating and modulated signal.

   Valid Range:

   - 0 - Normal
   - 1 - Inverted

Return Value

Returns the status code of this operation.

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

2.2.3.6.7 Low-Level Phase Modulation

Description Functions in this class operate all phase modulation parameters.

2.2.3.6.7.1 Set PHM State

C Function Prototype

```
ViStatus rssism_setPHMState (
    ViSession instrumentHandle,
    ViBoolean modState);
```

Basic Function Prototype

```
Function rssism_setPHMState (
    ByVal instrumentHandle As ViSession,
    ByVal modState As ViBoolean) As ViStatus
```

Purpose This function switches on or off the phase modulation.

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 modState [in]**

   This control switches on or off the phase modulation.

   Valid Range:

   - VI_FALSE (0) - Off (Default Value)
   - VI_TRUE (1) - On

Return Value Returns the status code of this operation.

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

2.2.3.6.7.2 Get PHM State

C Function Prototype

```
ViStatus rssism_getPHMState (
    ViSession instrumentHandle,
    ViBoolean* modState);
```

Basic Function Prototype

```
Function rssism_getPHMState (
    ByVal instrumentHandle As ViSession,
    modState As ViBoolean) As ViStatus
```

Purpose This function returns state of the phase modulation.

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 modState [out]**

   This control returns state of the phase modulation.

   Valid Range:

   - VI_FALSE (0) - Off
   - VI_TRUE (1) - On

Return Value Returns the status code of this operation.

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

2.2.3.6.7.3 Set PHM Deviation

C Function Prototype

```
ViStatus rssism_setPHMDeviation (
    ViSession instrumentHandle,
    ViReal64 deviation);
```

Basic Function Prototype

```
Function rssism_setPHMDeviation (
    ByVal instrumentHandle As ViSession,
    ByVal deviation As ViReal64) As ViStatus
```

Purpose

This function specifies the phase variation caused by the phase modulation.

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 deviation [in]**

   This control specifies the frequency variation caused by the phase modulation.

   Valid Range: 0.0 rad to 10.0 rad

   Default Value: 0.0 rad

Return Value

Returns the status code of this operation.

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

2.2.3.6.7.4 Get PHM Deviation

C Function Prototype

```
ViStatus rssism_getPHMDeviation (
    ViSession instrumentHandle,
    ViReal64* deviation);
```

Basic Function Prototype

```
Function rssism_getPHMDeviation (
    ByVal instrumentHandle As ViSession,
    deviation As ViReal64) As ViStatus
```

Purpose

This function returns the phase variation caused by the phase modulation.

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 deviation [out]**

   This control returns the frequency variation caused by the phase modulation in rad.

Return Value

Returns the status code of this operation.

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

2.2.3.6.7.5 Set PHM Frequency

C Function Prototype

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

Basic Function Prototype

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

Purpose

This function sets the modulation 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 frequency [in]**

   This control sets the modulation frequency.

   Valid Range: 1.0 Hz to 80.0e3 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.

2.2.3.6.7.6 Get PHM Frequency

C Function Prototype

```
ViStatus rssism_getPHMFreq (
    ViSession instrumentHandle,
    ViReal64* frequency);
```

Basic Function Prototype

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

Purpose

This function returns the modulation 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 frequency [out]**

   This control returns the modulation 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.

2.2.3.6.7.7 Set PHM Source

C Function Prototype

```
ViStatus rssism_setPHMSource (
    ViSession instrumentHandle,
    ViInt32 modulationSource);
```

Basic Function Prototype

```
Function rssism_setPHMSource (
    ByVal instrumentHandle As ViSession,
    ByVal modulationSource As ViInt32) As ViStatus
```

Purpose

This function selects the modulation 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. **ViInt32 modulationSource [in]**

   This control selects the modulation source.

   Valid Range:

   - 0 – Internal

   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.

2.2.3.6.7.8 Get PHM Source

C Function Prototype

```
ViStatus rssism_getPHMSource (
    ViSession instrumentHandle,
    ViInt32* modulationSource);
```

Basic Function Prototype

```
Function rssism_getPHMSource (
    ByVal instrumentHandle As ViSession,
    modulationSource As ViInt32) As ViStatus
```

Purpose

This function returns the modulation 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. **ViInt32 modulationSource [out]**

   This control returns the modulation source.

   Valid Range:

   - 0 - Internal

Return Value

Returns the status code of this operation.

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

2.2.3.6.7.9 Set PHM Polarity

C Function Prototype

```
ViStatus rssism_setPHMPolar (
    ViSession instrumentHandle,
    ViInt32 polarity);
```

Basic Function Prototype

```
Function rssism_setPHMPolar (
    ByVal instrumentHandle As ViSession,
    ByVal polarity As ViInt32) As ViStatus
```

Purpose

This function specifies the polarity between the modulating and 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 polarity [in]**

   This control specifies the polarity between the modulating and modulated signal.

   Valid Range:

   - 0 - Normal (Default Value)
   - 1 - Inverted

Return Value

Returns the status code of this operation.

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

2.2.3.6.7.10 Get PHM Polarity

C Function Prototype

```
ViStatus rssism_getPHMPolar (
    ViSession instrumentHandle,
    ViInt32* polarity);
```

Basic Function Prototype

```
Function rssism_getPHMPolar (
    ByVal instrumentHandle As ViSession,
    polarity As ViInt32) As ViStatus
```

Purpose

This function returns the polarity between the modulating and 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 polarity [out]**

   This control returns the polarity between the modulating and modulated signal.

   Valid Range:

   - 0 - Normal
   - 1 - Inverted

Return Value

Returns the status code of this operation.

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

2.2.3.6.8 Low-Level Pulse Modulation

Description Functions in this class operate all pulse modulation parameters.

2.2.3.6.8.1 Set PM State

C Function Prototype

```
ViStatus rssism_setPMState (
    ViSession instrumentHandle,
    ViBoolean modState);
```

Basic Function Prototype

```
Function rssism_setPMState (
    ByVal instrumentHandle As ViSession,
    ByVal modState As ViBoolean) As ViStatus
```

Purpose This function switches on or off the pulse modulation.

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 modState [in]**

   This control switches on or off the pulse modulation.

   Valid Range:

   - VI_FALSE (0) - Off (Default Value)
   - VI_TRUE (1) - On

Return Value Returns the status code of this operation.

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

2.2.3.6.8.2 Get PM State

C Function Prototype

```
ViStatus rssism_getPMState (
    ViSession instrumentHandle,
    ViBoolean* modState);
```

Basic Function Prototype

```
Function rssism_getPMState (
    ByVal instrumentHandle As ViSession,
    modState As ViBoolean) As ViStatus
```

Purpose This function returns state of the pulse modulation.

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 modState [out]**

   This control returns state of the pulse modulation.

   Valid Range:

   - VI_FALSE (0) - Off
   - VI_TRUE (1) - On

Return Value Returns the status code of this operation.

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

2.2.3.6.8.3 Set PM Source

C Function Prototype

```
ViStatus rssism_setPMSource (
    ViSession instrumentHandle,
    ViInt32 modulationSource);
```

Basic Function Prototype

```
Function rssism_setPMSource (
    ByVal instrumentHandle As ViSession,
    ByVal modulationSource As ViInt32) As ViStatus
```

Purpose

This function selects the source of the modulating 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 modulationSource [in]**

   This control selects the source of the modulating signal.

   Valid Range:

   - 0 - Internal
   - 1 - External

   Default Value: 0

Note

- Internal:
  Internal pulse generator.
- External:
  Signal is fed externally.

Return Value

Returns the status code of this operation.

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

2.2.3.6.8.4 Get PM Source

C Function Prototype

```
ViStatus rssism_getPMSource (
    ViSession instrumentHandle,
    ViInt32* modulationSource);
```

Basic Function Prototype

```
Function rssism_getPMSource (
    ByVal instrumentHandle As ViSession,
    modulationSource As ViInt32) As ViStatus
```

Purpose

This function returns the source of the modulating 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 modulationSource [out]**

   This control returns the source of the modulating signal.

   Valid Range:

   - 0 - Internal
   - 1 - External

Note

- Internal:
  Internal pulse generator.
- External:
  Signal is fed externally.

Return Value

Returns the status code of this operation.

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

2.2.3.6.8.5 Set PM Polarity

C Function Prototype

```
ViStatus rssism_setPMPolar (
    ViSession instrumentHandle,
    ViInt32 polarity);
```

Basic Function Prototype

```
Function rssism_setPMPolar (
    ByVal instrumentHandle As ViSession,
    ByVal polarity As ViInt32) As ViStatus
```

Purpose

This function specifies the polarity between the modulating and 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 polarity [in]**

   This control specifies the polarity between the modulating and modulated signal.

   Valid Range:

   - 0 - Normal (Default Value)
   - 1 - Inverted

Note

- Normal:
  The RF signal is suppressed during the interpulse period.
- Inverted:
  The RF signal is suppressed during the pulse.

Return Value

Returns the status code of this operation.

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

2.2.3.6.8.6 Get PM Polarity

C Function Prototype

```
ViStatus rssism_getPMPolar (
    ViSession instrumentHandle,
    ViInt32* polarity);
```

Basic Function Prototype

```
Function rssism_getPMPolar (
    ByVal instrumentHandle As ViSession,
    polarity As ViInt32) As ViStatus
```

Purpose

This function returns the polarity between the modulating and 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 polarity [out]**

   This control returns the polarity between the modulating and modulated signal.

   Valid Range:

   - 0 - Normal
   - 1 - Inverted

Note

- Normal:
  The RF signal is suppressed during the interpulse period.
- Inverted:
  The RF signal is suppressed during the pulse.

Return Value

Returns the status code of this operation.

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

2.2.3.6.8.7 Set PM Pulse Off Time

C Function Prototype

```
ViStatus rssism_setPMPulseOffTime (
    ViSession instrumentHandle,
    ViReal64 time);
```

Basic Function Prototype

```
Function rssism_setPMPulseOffTime (
    ByVal instrumentHandle As ViSession,
    ByVal time As ViReal64) As ViStatus
```

Purpose

This function sets off time of the pulsed RF 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. **ViReal64 time [in]**

   This control sets off time of the pulsed RF signal.

   Valid Range: 100.0e-6 s to 1.0 s

   Default Value: 99.9e-3 s

Return Value

Returns the status code of this operation.

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

2.2.3.6.8.8 Get PM Pulse Off Time

C Function Prototype

```
ViStatus rssism_getPMPulseOffTime (
    ViSession instrumentHandle,
    ViReal64* time);
```

Basic Function Prototype

```
Function rssism_getPMPulseOffTime (
    ByVal instrumentHandle As ViSession,
    time As ViReal64) As ViStatus
```

Purpose

This function returns off time of the pulsed RF 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. **ViReal64 time [out]**

   This control returns off time of the pulsed RF signal.

Return Value

Returns the status code of this operation.

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

2.2.3.6.8.9 Set PM Pulse On Time

C Function Prototype

```
ViStatus rssism_setPMPulseOnTime (
    ViSession instrumentHandle,
    ViReal64 time);
```

Basic Function Prototype

```
Function rssism_setPMPulseOnTime (
    ByVal instrumentHandle As ViSession,
    ByVal time As ViReal64) As ViStatus
```

Purpose

This function sets on time of the pulsed RF 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. **ViReal64 time [in]**

   This control sets on time of the pulsed RF signal.

   Valid Range: 100.0e-6 s to 1.0 s

   Default Value: 100.0e-6 s

Return Value

Returns the status code of this operation.

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

2.2.3.6.8.10 Get PM Pulse On Time

C Function Prototype

```
ViStatus rssism_getPMPulseOnTime (
    ViSession instrumentHandle,
    ViReal64* time);
```

Basic Function Prototype

```
Function rssism_getPMPulseOnTime (
    ByVal instrumentHandle As ViSession,
    time As ViReal64) As ViStatus
```

Purpose

This function returns on time of the pulsed RF 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. **ViReal64 time [out]**

   This control returns on time of the pulsed RF signal.

Return Value

Returns the status code of this operation.

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

2.2.3.6.8.11 Set PM Pulse Delay Time

C Function Prototype

```
ViStatus rssism_setPMPulseDelayTime (
    ViSession instrumentHandle,
    ViReal64 delay);
```

Basic Function Prototype

```
Function rssism_setPMPulseDelayTime (
    ByVal instrumentHandle As ViSession,
    ByVal delay As ViReal64) As ViStatus
```

Purpose

If external pulse modulation is selected, pulse delay can be set between the external modulation signal and pulsed RF 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. **ViReal64 delay [in]**

   This control sets the pulse delay.

   Valid Range: 0.0 s to 1.0 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.

2.2.3.6.8.12 Get PM Pulse Delay Time

C Function Prototype

```
ViStatus rssism_getPMPulseDelayTime (
    ViSession instrumentHandle,
    ViReal64* delay);
```

Basic Function Prototype

```
Function rssism_getPMPulseDelayTime (
    ByVal instrumentHandle As ViSession,
    delay As ViReal64) As ViStatus
```

Purpose

This function returns the pulse delay between the external modulation signal and pulsed RF 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. **ViReal64 delay [out]**

   This control returns the pulse delay.

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 Low-Level Vector Modulation

Description

This subsystem contains the functions to control the vector modulation and to set the parameters of the modulation signal.

2.2.3.6.9.1 Set Vector State

C Function Prototype

```
ViStatus rssism_setVectorState (
    ViSession instrumentHandle,
    ViBoolean modState);
```

Basic Function Prototype

```
Function rssism_setVectorState (
    ByVal instrumentHandle As ViSession,
    ByVal modState As ViBoolean) As ViStatus
```

Purpose

This function switches on/off the vector modulation.

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 modState [in]**

   This control switches on or off the vector modulation.

   Valid Range:

   - VI_FALSE (0) - Off (Default Value)
   - VI_TRUE (1) - On

Return Value

Returns the status code of this operation.

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

2.2.3.6.9.2 Get Vector State

C Function Prototype

```
ViStatus rssism_getVectorState (
    ViSession instrumentHandle,
    ViBoolean* modState);
```

Basic Function Prototype

```
Function rssism_getVectorState (
    ByVal instrumentHandle As ViSession,
    modState As ViBoolean) As ViStatus
```

Purpose

This function returns state of the vector modulation.

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 modState [out]

This control returns state of the vector modulation.

Valid Range:

- VI_FALSE (0) - Off
- VI_TRUE (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 LF Output

Description

This class operates the LF Output.

Functions/SubClasses:

1. **Configure LF Output:**
   This function configures the LF output.
2. **Low-Level: (Class)**
   Functions in this class operate all LF Output parameters.

2.2.3.7.1 Configure LF Output

C Function Prototype

```
ViStatus rssism_configLFOutput (
    ViSession instrumentHandle,
    ViBoolean lfOutputState,
    ViReal64 frequency,
    ViReal64 voltage);
```

Basic Function Prototype

```
Function rssism_configLFOutput (
    ByVal instrumentHandle As ViSession,
    ByVal lfOutputState As ViBoolean,
    ByVal frequency As ViReal64,
    ByVal voltage As ViReal64) As ViStatus
```

Purpose

This function configures the LF output parameters.

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 lfOutputState [in]**
   This control switches on or off the LF output.
   Valid Range:
   - VI_FALSE (0) - Off (Default Value)
   - VI_TRUE (1) - On
3. **ViReal64 frequency [in]**
   This control sets the frequency of LF generator.
   Valid Range: 1.0 Hz to 80.0e3 Hz
   Default Value: 1000.0 Hz
4. **ViReal64 voltage [in]**
   This control sets the voltage (RMS) of LF-output. The voltage is a characteristic of the output, not of the source. I.e., the voltage is maintained even if another generator is connected to the output.
   Valid Range: 1.0e-3 V to 2.0 V
   Default Value: 0.1 V

Return Value

Returns the status code of this operation.

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

2.2.3.7.2 Low-Level

Description Functions in this class operate all LF Output parameters.

2.2.3.7.2.1 Set LF Output State

C Function Prototype

```
ViStatus rssism_setLFOutputState (
    ViSession instrumentHandle,
    ViBoolean lfOutputState);
```

Basic Function Prototype

```
Function rssism_setLFOutputState (
    ByVal instrumentHandle As ViSession,
    ByVal lfOutputState As ViBoolean) As ViStatus
```

Purpose This function switches on or off the LF 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. **ViBoolean lfOutputState [in]**

   This control switches on or off the LF output.

   Valid Range:

   - VI_FALSE (0) - Off (Default Value)
   - VI_TRUE (1) - On

Return Value Returns the status code of this operation.

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

2.2.3.7.2.2 Get LF Output State

C Function Prototype

```
ViStatus rssism_getLFOutputState (
    ViSession instrumentHandle,
    ViBoolean* lfOutputState);
```

Basic Function Prototype

```
Function rssism_getLFOutputState (
    ByVal instrumentHandle As ViSession,
    lfOutputState As ViBoolean) As ViStatus
```

Purpose This function returns state of LF 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. **ViBoolean lfOutputState [out]**

   This control returns state of LF output.

   Valid Range:

   - VI_FALSE (0) - Off
   - VI_TRUE (1) - On

Return Value Returns the status code of this operation.

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

2.2.3.7.2.3 Set LF Frequency

C Function Prototype

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

Basic Function Prototype

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

Purpose

This function sets the frequency of LF 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. **ViReal64 frequency [in]**

   This control sets the frequency of LF generator.

   Valid Range: 1.0 Hz to 80.0e3 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.

2.2.3.7.2.4 Get LF Frequency

C Function Prototype

```
ViStatus rssism_getLFCWFreq (
    ViSession instrumentHandle,
    ViReal64* frequency);
```

Basic Function Prototype

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

Purpose

This function returns the frequency of LF 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. **ViReal64 frequency [out]**

   This control returns the frequency of LF generator in Hz.

Return Value

Returns the status code of this operation.

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

2.2.3.7.2.5 Set LF Voltage

C Function Prototype

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

Basic Function Prototype

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

Purpose

This function sets the voltage of LF-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. **ViReal64 voltage [in]**

   This control sets the voltage (RMS) of LF-output. The voltage is a characteristic of the output, not of the source. I.e., the voltage is maintained even if another generator is connected to the output.

   Valid Range: 1.0e-3 V to 2.0 V

   Default Value: 0.1 V

Return Value

Returns the status code of this operation.

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

2.2.3.7.2.6 Get LF Voltage

C Function Prototype

```
ViStatus rssism_getLFVoltage (
    ViSession instrumentHandle,
    ViReal64* voltage);
```

Basic Function Prototype

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

Purpose

This function returns the voltage of LF-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. **ViReal64 voltage [out]**

   This control returns the voltage (RMS) of LF-output in volts.

Return Value

Returns the status code of this operation.

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

2.2.3.8 LF Sweep

Description

This class operates the LF Sweep.

Functions/SubClasses:

1. **Configure LF Sweep:**
   This function configures the LF sweep.
2. **Low-Level: (Class)**
   Functions in this class operate all LF Sweep parameters.

2.2.3.8.1 Configure LF Sweep

C Function Prototype

```
ViStatus rssism_configLFSweep (
    ViSession instrumentHandle,
    ViInt32 sweepMode,
    ViReal64 startFrequency,
    ViReal64 stopFrequency,
    ViReal64 step,
    ViReal64 dwellTime);
```

Basic Function Prototype

```
Function rssism_configLFSweep (
    ByVal instrumentHandle As ViSession,
    ByVal sweepMode As ViInt32,
    ByVal startFrequency As ViReal64,
    ByVal stopFrequency As ViReal64,
    ByVal step As ViReal64,
    ByVal dwellTime As ViReal64) As ViStatus
```

Purpose

This function configures the LF sweep parameters.

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 sweepMode [in]**
   This control specifies the run of the sweep (this parameter is used only for compatibility reason and has no influence to the instrument settings).

   Valid Range:

   - 0 - Auto
   - 1 - Manual
   - 2 - Step (RESERVED)

   Default Value: 0

Note

- Auto:
  Each trigger triggers exactly one entire sweep cycle.
- Manual:
  Each level step of the sweep is provided by means of manual control.
- Step:
  Each trigger triggers only one sweep step (single-step mode).

3. **ViReal64 startFrequency [in]**
   This control defines the starting value of the frequency for the sweep.
   Valid Range: 1.0 Hz to 79.999e3 Hz
   Default Value: 200.0 Hz
4. **ViReal64 stopFrequency [in]**
   This control defines the stop value of the frequency for the sweep.
   Valid Range: 2.0 Hz to 80.0e3 Hz
   Default Value: 10.0e3 Hz
5. **ViReal64 step [in]**
   This control sets the step width with the linear sweep.
   Valid Range: 0.1 Hz to 80.0e3 Hz
   Default Value: 100.0 Hz
6. **ViReal64 dwellTime [in]**
   This control sets the time per frequency step (dwell).
   Valid Range: 0.01 s to 1.0 s
   Default Value: 0.1 s

Return Value

Returns the status code of this operation.

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

2.2.3.8.2 Low-Level

Description Functions in this class operate all LF Sweep parameters.

2.2.3.8.2.1 Set LF Sweep Mode

C Function Prototype

```
ViStatus rssism_setLFSweepMode (
    ViSession instrumentHandle,
    ViInt32 sweepMode);
```

Basic Function Prototype

```
Function rssism_setLFSweepMode (
    ByVal instrumentHandle As ViSession,
    ByVal sweepMode As ViInt32) As ViStatus
```

Purpose This function sets the sweep mode of LF-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. **ViInt32 sweepMode [in]**

   This control specifies the run of the sweep (this parameter is used only for compatibility reason and has no influence to the instrument settings).

   Valid Range:

   - 0 - Auto
   - 1 - Manual
   - 2 - Step (RESERVED)

   Default Value: 0

Note

- Auto:
  Each trigger triggers exactly one entire sweep cycle.
- Manual:
  Each level step of the sweep is provided by means of manual control.
- Step:
  Each trigger triggers only one sweep step (single-step mode).

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 Get LF Sweep Mode

C Function Prototype

```
ViStatus rssism_getLFSweepMode (
    ViSession instrumentHandle,
    ViInt32* sweepMode);
```

Basic Function Prototype

```
Function rssism_getLFSweepMode (
    ByVal instrumentHandle As ViSession,
    sweepMode As ViInt32) As ViStatus
```

Purpose

This function returns the sweep mode LF-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. **ViInt32 sweepMode [out]**

   This control returns the run of the sweep.

   Valid Range:

   - 0 - Auto
   - 1 - Manual
   - 2 - Step (RESERVED)

Note

- Auto:
  Each trigger triggers exactly one entire sweep cycle.
- Manual:
  Each level step of the sweep is provided by means of manual control.
- Step:
  Each trigger triggers only one sweep step (single-step mode).

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 Set LF Start Frequency

C Function Prototype

```
ViStatus rssism_setLFStartFreq (
    ViSession instrumentHandle,
    ViReal64 startFrequency);
```

Basic Function Prototype

```
Function rssism_setLFStartFreq (
    ByVal instrumentHandle As ViSession,
    ByVal startFrequency As ViReal64) As ViStatus
```

Purpose

This function defines the starting value of frequency 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. **ViReal64 startFrequency [in]**

   This control defines the starting value of frequency for the sweep.

   Valid Range: 1.0 Hz to 79.999e3 Hz

   Default Value: 200.0 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 Get LF Start Frequency

C Function Prototype

```
ViStatus rssism_getLFStartFreq (
    ViSession instrumentHandle,
    ViReal64* startFrequency);
```

Basic Function Prototype

```
Function rssism_getLFStartFreq (
    ByVal instrumentHandle As ViSession,
    startFrequency As ViReal64) As ViStatus
```

Purpose

This function returns the starting value of frequency 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. **ViReal64 startFrequency [out]**

   This control returns the starting value of frequency for the sweep in Hz.

Return Value

Returns the status code of this operation.

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

2.2.3.8.2.5 Set LF Stop Frequency

C Function Prototype

```
ViStatus rssism_setLFStopFreq (
    ViSession instrumentHandle,
    ViReal64 stopFrequency);
```

Basic Function Prototype

```
Function rssism_setLFStopFreq (
    ByVal instrumentHandle As ViSession,
    ByVal stopFrequency As ViReal64) As ViStatus
```

Purpose

This function defines the stop value of frequency 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. **ViReal64 stopFrequency [in]**

   This control defines the stop value of frequency for the sweep.

   Valid Range: 2.0 Hz to 80.0e3 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.

2.2.3.8.2.6 Get LF Stop Frequency

C Function Prototype

```
ViStatus rssism_getLFStopFreq (
    ViSession instrumentHandle,
    ViReal64* stopFrequency);
```

Basic Function Prototype

```
Function rssism_getLFStopFreq (
    ByVal instrumentHandle As ViSession,
    stopFrequency As ViReal64) As ViStatus
```

Purpose

This function returns the stop value of frequency 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. **ViReal64 stopFrequency [out]**

   This control returns the stop value of frequency for the sweep in Hz.

Return Value

Returns the status code of this operation.

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

2.2.3.8.2.7 Set LF Frequency Manual

C Function Prototype

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

Basic Function Prototype

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

Purpose

This function sets the LF-output frequency in manual sweep mode.

Note

This function changes settings done by Set LF Frequency (rssism_setLFCWFreq) 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 frequency [in]**

   This control sets the frequency of LF generator.

   Valid Range: 1.0 Hz to 80.0e3 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.

2.2.3.8.2.8 Get LF Frequency Manual

C Function Prototype

```
ViStatus rssism_getLFFreqMan (
    ViSession instrumentHandle,
    ViReal64* frequency);
```

Basic Function Prototype

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

Purpose

This function returns the LF-output frequency in manual sweep 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. **ViReal64 frequency [out]**

   This control returns the frequency LF generator in Hz.

Return Value

Returns the status code of this operation.

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

2.2.3.8.2.9 Set LF Sweep Spacing

C Function Prototype

```
ViStatus rssism_setLFSweepSpac (
    ViSession instrumentHandle,
    ViInt32 spacing);
```

Basic Function Prototype

```
Function rssism_setLFSweepSpac (
    ByVal instrumentHandle As ViSession,
    ByVal spacing As ViInt32) As ViStatus
```

Purpose

This function selects whether the steps have linear or logarithmic spacings.

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 spacing [in]**

   This control selects whether the steps have linear or logarithmic spacings.

   Valid Range:

   - 0 - Linear
   - 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.

2.2.3.8.2.10 Get LF Sweep Spacing

C Function Prototype

```
ViStatus rssism_getLFSweepSpac (
    ViSession instrumentHandle,
    ViInt32* spacing);
```

Basic Function Prototype

```
Function rssism_getLFSweepSpac (
    ByVal instrumentHandle As ViSession,
    spacing As ViInt32) As ViStatus
```

Purpose

This function returns whether the steps have linear or logarithmic spacings.

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 spacing [out]**

   This control returns whether the steps have linear or logarithmic spacings.

   Valid Range:

   - 0 - Linear
   - 1 - Logarithmic

Return Value

Returns the status code of this operation.

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

2.2.3.8.2.11 Set LF Sweep Step

C Function Prototype

```
ViStatus rssism_setLFSweepStep (
    ViSession instrumentHandle,
    ViInt32 mode,
    ViReal64 step);
```

Basic Function Prototype

```
Function rssism_setLFSweepStep (
    ByVal instrumentHandle As ViSession,
    ByVal mode As ViInt32,
    ByVal step As ViReal64) As ViStatus
```

Purpose

This function sets the step width with the linear or logarithmic 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 mode [in]**

   This control selects mode of step to define.

   Valid Range:

   - 0 - Linear
   - 1 - Logarithmic

   Default Value: 0

3. **ViReal64 step [in]**

   This control sets the step width with the linear or logarithmic sweep.

   Default Value: 100.0

   Valid Range:

   Linear Sweep: 0.1 Hz to 80.0e3 Hz

   Logarithmic Sweep: 0.01 % to 100.0 %

Return Value

Returns the status code of this operation.

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

2.2.3.8.2.12 Get LF Sweep Step

C Function Prototype

```
ViStatus rssism_getLFSweepStep (
    ViSession instrumentHandle,
    ViInt32* mode,
    ViReal64* step);
```

Basic Function Prototype

```
Function rssism_getLFSweepStep (
    ByVal instrumentHandle As ViSession,
    mode As ViInt32,
    step As ViReal64) As ViStatus
```

Purpose

This function returns the step width with the linear or logarithmic 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 mode [out]**

   This control returns the step mode.

   Valid Range:

   - 0 - Linear
   - 1 - Logarithmic

3. **ViReal64 step [out]**

   This control returns the step width in Hz (linear sweep) or percent (logarithmic sweep).

Return Value

Returns the status code of this operation.

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

2.2.3.8.2.13 Set LF Sweep Dwell Time

C Function Prototype

```
ViStatus rssism_setLFSweepDwellTime (
    ViSession instrumentHandle,
    ViReal64 dwellTime);
```

Basic Function Prototype

```
Function rssism_setLFSweepDwellTime (
    ByVal instrumentHandle As ViSession,
    ByVal dwellTime As ViReal64) As ViStatus
```

Purpose

This function sets the time per frequency step (dwell).

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 dwellTime [in]**

   This control sets the time per frequency step (dwell).

   Valid Range: 0.01 s to 1.0 s

   Default Value: 0.1 s

Return Value

Returns the status code of this operation.

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

2.2.3.8.2.14 Get LF Sweep Dwell Time

C Function Prototype

```
ViStatus rssism_getLFSweepDwellTime (
    ViSession instrumentHandle,
    ViReal64* dwellTime);
```

Basic Function Prototype

```
Function rssism_getLFSweepDwellTime (
    ByVal instrumentHandle As ViSession,
    dwellTime As ViReal64) As ViStatus
```

Purpose

This function returns the time per frequency step (dwell).

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 dwellTime [out]

This control returns the time per frequency step (dwell) in sec.

Return Value

Returns the status code of this operation.

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

2.2.3.9 Reference Oscillator

Description

This class operates the Reference Oscillator.

Functions/SubClasses:

1. **Configure Ref Oscillator:**
   This function configures the Reference Oscillator.
2. **Get Ref Oscillator:**
   This function returns Reference Oscillator parameters.

2.2.3.9.1 Configure Ref Oscillator

C Function Prototype

```
ViStatus rssism_configureRefOscillator (
    ViSession instrumentHandle,
    ViBoolean reftoOutput,
    ViInt32 refSource,
    ViInt32 refFrequency);
```

Basic Function Prototype

```
Function rssism_configureRefOscillator (
    ByVal instrumentHandle As ViSession,
    ByVal reftoOutput As ViBoolean,
    ByVal refSource As ViInt32,
    ByVal refFrequency As ViInt32) As ViStatus
```

Purpose

This function configures the Reference Oscillator parameters.

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 reftoOutput [in]**
   This control switches if reference is connected to output connector or not.
   Valid Range:
   - VI_FALSE (0) - Off (Default Value)
   - VI_TRUE (1) - On

Note

- Off:
  Reference is not connected to output connector.
- On:
  Reference is connected to output connector.

3. **ViInt32 refSource [in]**
   This control selects the reference oscillator source.
   Valid Range:
   - 0 - TCXO 10MHz
   - 1 - External
   - 2 - Reserved

   Default Value: 0

Note

- TCXO 10MHz:
  The internal oscillator is used.
- External:
  The reference signal is fed externally.

4. ViInt32 refFrequency [in]

This control selects external reference frequency.

Valid Range:

- 0 - 2 MHz
- 1 - 5 MHz
- 2 - 10 MHz

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.

2.2.3.9.2 Get Ref Oscillator

C Function Prototype

```
ViStatus rssism_getRefOscillator (
    ViSession instrumentHandle,
    ViBoolean* reftoOutput,
    ViInt32* refSource,
    ViInt32* refFrequency);
```

Basic Function Prototype

```
Function rssism_getRefOscillator (
    ByVal instrumentHandle As ViSession,
    reftoOutput As ViBoolean,
    refSource As ViInt32,
    refFrequency As ViInt32) As ViStatus
```

Purpose

This function returns Reference Oscillator parameters.

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 reftoOutput [out]

This control returns if the reference is connected to output connector or not.

Valid Range:

- VI_FALSE (0) - Off
- VI_TRUE (1) - On

Note

- Off:
  Reference is not connected to output connector.
- On:
  Reference is connected to output connector.

3. ViInt32 refSource [out]

This control returns configured reference oscillator source.

Valid Range:

- 0 - TCXO 10MHz
- 1 - External
- 2 - Reserved

Note

- TCXO 10MHz:
  The internal oscillator is used.
- External:
  The reference signal is fed externally.

4. ViInt32 refFrequency [out]

This control returns configured external reference frequency.

Valid Range:

- 0 - 2 MHz
- 1 - 5 MHz
- 2 - 10 MHz

Return Value

Returns the status code of this operation.

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

2.2.3.10 Trigger

Description

Functions in this class operate the Trigger.

Functions/SubClasses:

1. **Set Sweep Trigger Source:**
   This function specifies the sweep trigger source.
2. **Get Sweep Trigger Source:**
   This function returns the trigger sweep source.

2.2.3.10.1 Set Sweep Trigger Source

C Function Prototype

```
ViStatus rssism_setSweepTrigSource (
    ViSession instrumentHandle,
    ViInt32 sweepSource,
    ViInt32 triggerSource);
```

Basic Function Prototype

```
Function rssism_setSweepTrigSource (
    ByVal instrumentHandle As ViSession,
    ByVal sweepSource As ViInt32,
    ByVal triggerSource As ViInt32) As ViStatus
```

Purpose

This function specifies the sweep 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
2. **ViInt32 sweepSource [in]**
   This control selects the sweep source.
   Valid Range:
   - 1 - RF Sweep
   - 2 - LF Sweep
   - 3 - RF Level Sweep

   Default Value: 1
3. **ViInt32 triggerSource [in]**
   This control specifies the sweep trigger source.
   Valid Range:
   - 0 - Single
   - 1 - External (RESERVED)
   - 2 - Auto

   Default Value: 0

Note

- Single:
  Triggering is affected by calling Send Trigger function.
- External:
  Triggering is effected from outside via the EXT TRIG socket.
- Auto:
  The trigger is free-running, i.e., the trigger requirement is permanently met. As soon as a sweep has been terminated, the next one is started.

Return Value Returns the status code of this operation.

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

2.2.3.10.2 Get Sweep Trigger Source

C Function Prototype

```
ViStatus rssism_getSweepTrigSource (
    ViSession instrumentHandle,
    ViInt32* sweepSource);
```

Basic Function Prototype

```
Function rssism_getSweepTrigSource (
    ByVal instrumentHandle As ViSession,
    sweepSource As ViInt32) As ViStatus
```

Purpose This function returns the trigger sweep 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. **ViInt32 sweepSource [out]**

   This control returns the sweep source.

   Valid Range:

   - 1 - RF Sweep
   - 2 - LF Sweep
   - 3 - RF Level Sweep

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 the trigger.

2. **Send Trigger and Wait for OPC:**

   This function triggers all actions waiting for a trigger event and waits for operation completed (OPC) before returning the status code.

3. **Abort Trigger:**

   This function aborts the sweep system (stops sweeping).

2.2.4.1 Send Trigger

C Function Prototype

```
ViStatus rssism_SendTrigger (
    ViSession instrumentHandle,
    ViInt32 sweepSource,
    ViInt32 systemToTrigger);
```

Basic Function Prototype

```
Function rssism_SendTrigger (
    ByVal instrumentHandle As ViSession,
    ByVal sweepSource As ViInt32,
    ByVal systemToTrigger As ViInt32) As ViStatus
```

Purpose

This function sends the trigger.

Note

This function performs single sweep regardless which sweep source or system triggering is selected. Parameters of this function are ignored.

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 sweepSource [in]**

   This control selects sweep source to send a trigger.

   Valid Range:

   - 1 - RF Sweep
   - 2 - LF Sweep
   - 3 - RF Level Sweep

   Default Value: 0

3. **ViInt32 systemToTrigger [in]**

   This control selects the system to trigger.

   Valid Range:

   - 0 - Sweep

   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.

2.2.4.2 Send Trigger and Wait for OPC

C Function Prototype

ViStatus rssism_SendTriggerWopc (
ViSession instrumentHandle,
ViInt32 timeout);

Basic Function Prototype

Function rssism_SendTriggerWopc (
ByVal instrumentHandle As ViSession,
ByVal timeout As ViInt32) As ViStatus

Purpose

This function triggers all actions waiting for the trigger event and waits for operation completed (OPC) before returning the status code.

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 timeout for the triggering routine to be finished. If the length of time required for triggering exceeds the timeout value, then the function will return with a timeout error.

   Valid Range: 0 ms to 600000 ms

   Default Value: 10000 ms

Return Value

Returns the status code of this operation.

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

2.2.4.3 Abort Trigger

C Function Prototype

ViStatus rssism_abortTrigger (
ViSession instrumentHandle,
ViInt32 systemtoAbort);

Basic Function Prototype

Function rssism_abortTrigger (
ByVal instrumentHandle As ViSession,
ByVal systemtoAbort As ViInt32) As ViStatus

Purpose

This function aborts the sweep system (stops sweeping).

Note

This function aborts the sweeping regardless which system is selected. Parameters of this function are ignored.

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 systemtoAbort [in]**

   This control selects the system to abort.

   Valid Range:

   - 0 - Sweep

   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 Utility Functions

Description

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

Functions:

1. **Time Out: (Class)**

   This class of function enable you to set and query the timeout parameter of the instrument.

2. **Flush Error Queue:**

   This function deletes error queue.

3. **State Checking:**

   This function switches ON/OFF state checking of the instrument (reading of the Standard Event Register and checking it for error).

4. **Reset:**

   This function resets the instrument to its default state.

5. **Self-Test:**

   This function runs the instrument self test and returns the test code.

6. **Error Query:**

   This function reads an error code from the instrument error queue.

7. **Error Message:**

   This function takes the Status Code and returns it as a user readable string.

8. **Revision Query:**

   This function returns the revision numbers of the instrument driver and instrument firmware.

2.2.5.1 Time Out

Description This class of function enable you to set and query the timeout parameter of the instrument.

2.2.5.1.1 Set Time Out

C Function Prototype

```
ViStatus rssism_setTimeOut (
    ViSession instrumentHandle,
    ViInt32 timeout);
```

Basic Function Prototype

```
Function rssism_setTimeOut (
    ByVal instrumentHandle As ViSession,
    ByVal timeout As ViInt32) As ViStatus
```

Purpose Sets the 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: 10000 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 Get Time Out

C Function Prototype

```
ViStatus rssism_getTimeOut (
    ViSession instrumentHandle,
    ViInt32* timeout);
```

Basic Function Prototype

```
Function rssism_getTimeOut (
    ByVal instrumentHandle As ViSession,
    timeout As ViInt32) As ViStatus
```

Purpose 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 Flush Error Queue

C Function Prototype

```
ViStatus rssism_FlushErrorQueue (
    ViSession instrumentHandle);
```

Basic Function Prototype

```
Function rssism_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.

2.2.5.3 State Checking

C Function Prototype

```
ViStatus rssism_errorCheckState (
    ViSession instrumentHandle,
    ViBoolean stateChecking);
```

Basic Function Prototype

```
Function rssism_errorCheckState (
    ByVal instrumentHandle As ViSession,
    ByVal stateChecking As ViBoolean) As ViStatus
```

Purpose

This function enables (disables) operation(s) status checking.

Warning

Status checking is by default enabled. It is not recommended to disable it.

Note

- When disabled, status checking is not performed and function rssism_error_query does not provide any error message information.
- When disabled, internal event handling mechanism (interrupt pipe checking & control transfer handshake checking) is also disabled.
- 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).

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 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.

2.2.5.4 Reset

C Function Prototype

```
ViStatus rssism_reset (
    ViSession instrumentHandle);
```

Basic Function Prototype

```
Function rssism_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.

2.2.5.5 Self-Test

C Function Prototype

```
ViStatus rssism_self_test (
    ViSession instrumentHandle,
    ViInt16* selfTestResult,
    ViChar[] selfTestMessage);
```

Basic Function Prototype

```
Function rssism_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.

2.2.5.6 Error-Query

C Function Prototype

```
ViStatus rssism_error_query (
    ViSession instrumentHandle,
    ViInt32* errorCode,
    ViChar[] errorMessage);
```

Basic Function Prototype

```
Function rssism_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.

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.

2.2.5.7 Error Message

C Function Prototype

```
ViStatus rssism_error_message (
    ViSession instrumentHandle,
    ViStatus statusCode,
    ViChar[] message);
```

Basic Function Prototype

```
Function rssism_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: VI_NULL
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.

2.2.5.8 Revision Query

C Function Prototype

```
ViStatus rssism_revision_query (
    ViSession instrumentHandle,
    ViChar[] instrumentDriverRevision,
    ViChar[] firmwareRevision);
```

Basic Function Prototype

```
Function rssism_revision_query (
    ByVal instrumentHandle As ViSession,
    instrumentDriverRevision As ViChar,
    firmwareRevision As ViChar) As ViStatus
```

Purpose This function returns revision numbers of the instrument driver and instrument firmware, and tells the user with which instrument module 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 Instrument 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 Close

C Function Prototype

```
ViStatus rssism_close (
    ViSession instrumentHandle);
```

Basic Function Prototype

```
Function rssism_close (
    ByVal instrumentHandle As ViSession) As ViStatus
```

Purpose

This function closes the 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 **rssism_error_query** function. Up to 512 messages is queued using FILO (First-In Last-Out) method. If the error queue is full, error RSSISM_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.
0x3FFC09F0	WARNING: Pre-enforcement validation process changed value of a facultative register.
0x3FFC09F1	WARNING: Data retrieved from trace cache.

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.

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 rssism_setTimeOut function. The actual timeout value is retrieved via rssism_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

2.6 Contacts

List of your Rohde&Schwarz Partners

- For comprehensive information about Rohde&Schwarz, please visit our Rohde&Schwarz Homepage (http://www.rohde-schwarz.com/).
- For queries regarding technical aspects of our products, please contact our Customer Support (http://www.rohde-schwarz.com/www/dev_center.nsf/html/service_customerservicehotline).
- For international services, please contact our Service Partners (http://www.services.rohde-schwarz.com/).
- For information on training and seminars, please visit our Training Center (http://www.training.rohde-schwarz.com/).
- For latest instrument driver updates, please see our Rohde&Schwarz Drivers (http://www.rohde-schwarz.com/www/download.nsf/driver_frameset).

More Information

Should you have any questions or ideas concerning the instrument please contact our hotline:

Phone: **+49 18 05 12 42 12**

Fax: **+49 (89) 41 29 13 777**

E-mail: **CustomerSupport@rsd.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 Error Handling & Time Profiling

2.7.1.1 Source Code

```
/*****************************************************************************
 *
 * Title:   Error Handling & Time Profiling
 *
 * Purpose: This example shows basic principles of error handling.
 *
 *****************************************************************************/

#include <ansi_c.h>
#include "rssism.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)\
            {\
            rssism_error_message (io, error, error_message);\
            printf("\tFunction Call Status: 0x%08X, %s\n", error, error_message);\
            rssism_error_query(io, &error, error_message);\
            printf("\tInstrument Error: 0x%X, %s\n", error, error_message);\
            }\
    }

#define RESOURCE_NAME   "USB::0xAAD::0x7::100001"   // 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 (rssism_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 rssism_error_message"
    " function\n"
    "\t    - call rssism_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 (rssism_setCWFreq (io, 1.5e9));
    /* Passed wrong data to generate error */
    CHECKERR (rssism_setCWFreq (io, 9.9e99));

    printf (

    "\n --- Disable Warning & Error Checking ---\n\n"
    "\tWarning & Error checking can be disabled by calling function"
    " rssism_errorCheckState.\n"
    "\tSee description:\n\n"

    "\t(1) When disabled, status checking is not performed and function\n"
    "\t    rssism_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 (rssism_setCWFreq (io, 1.5e9));
    CHECKERR (rssism_setCWFreq (io, 1.5e9));
    CHECKERR (rssism_errorCheckState (io, VI_FALSE));
    CHECKERR (rssism_setCWFreq (io, 1.5e9));
    CHECKERR (rssism_setCWFreq (io, 1.5e9));

    printf (

    "\n\tWARNING: Use rssism_errorCheckState function with care!\n\n"

    );

    CHECKERR (rssism_close (io));
    return 0;
}
```

2.7.1.2 Execution Result

```
Line 50 (1897 ms): rssism_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 rssism_error_message function
          - call rssism_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 (53 ms): rssism_setCWFreq (io, 1.5e9)
 Line 72 (15 ms): rssism_setCWFreq (io, 9.9e99)
      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
      rssism_errorCheckState.

      See description:

      (1) When disabled, status checking is not performed and function
          rssism_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 (55 ms): rssism_setCWFreq (io, 1.5e9)
 Line 106 (55 ms): rssism_setCWFreq (io, 1.5e9)
 Line 107 (0 ms): rssism_errorCheckState (io, VI_FALSE)
 Line 108 (3 ms): rssism_setCWFreq (io, 1.5e9)
 Line 109 (3 ms): rssism_setCWFreq (io, 1.5e9)

      WARNING: Use rssism_errorCheckState function with care!

 Line 117 (1 ms): rssism_close (io)
```

2.7.2 Setting of RF Frequency and Level

2.7.2.1 Source Code

```
/*****************************************************************************
 *
 * Title:   Setting of RF Frequency and Level
 *
 * Purpose: This example shows how to set RF frequency and level parameters.
 *
 *****************************************************************************/

#include <ansi_c.h>
#include "rssism.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)\
            {\
            rssism_error_message (io, error, error_message);\
            printf("\tFunction Call Status: 0x%08X, %s\n", error, error_message);\
            rssism_error_query(io, &error, error_message);\
            printf("\tInstrument Error: 0x%X, %s\n", error, error_message);\
            }\
    }

#define RESOURCE_NAME   "USB::0xAAD::0x7::100001" // 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 (rssism_init (resourceName, IDQuery, resetDevice, &io));

    printf (

    "\n --- Set RF Frequency and Level ----------------------------------------\n\n"
    "\t- RF Frequency: 100 MHz, offset 10.0 MHz (90.0 MHz)\n"
    "\t- RF Level: -30.0 dBm, offset -10 dB (-20.0 dBm)\n"
```

```
    "\n ------------------------------------------------------------------------\n\n"

    );

    /* Configure RF Frequency */
    CHECKERR (rssism_configRFFreq (io, 1.0e8, 1.0e7, 1.0));
    /* Configure RF Level */
    CHECKERR (rssism_configRFLevel (io, -30.0, -10.0, 0.0));
    /* Set RF Output State */
    CHECKERR (rssism_setRFOutputState (io, VI_TRUE));

    printf ("\n");

    CHECKERR (rssism_close (io));

    return 0;
}
```

2.7.2.2 Execution Result

```
Line 49 (1807 ms): rssism_init (resourceName, IDQuery, resetDevice, &io)

 --- Set RF Frequency and Level ---------------------------------------

      - RF Frequency: 100 MHz, offset 10.0 MHz (90.0 MHz)
      - RF Level: -30.0 dBm, offset -10 dB (-20.0 dBm)

 ----------------------------------------------------------------------

 Line 61 (53 ms): rssism_configRFFreq (io, 1.0e8, 1.0e7, 1.0)
 Line 63 (56 ms): rssism_configRFLevel (io, -30.0, -10.0, 0.0)
 Line 65 (23 ms): rssism_setRFOutputState (io, VI_TRUE)

 Line 69 (1 ms): rssism_close (io)
```

2.7.3 Setting of LF Frequency and Level

2.7.3.1 Source Code

```
/*****************************************************************************
 *
 * Title:   Setting of LF Frequency and Level
 *
 * Purpose: This example shows how to set LF frequency and level parameters.
 *
 *****************************************************************************/

#include <ansi_c.h>
#include "rssism.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)\
            {\
            rssism_error_message (io, error, error_message);\
            printf("\tFunction Call Status: 0x%08X, %s\n", error, error_message);\
            rssism_error_query(io, &error, error_message);\
            printf("\tInstrument Error: 0x%X, %s\n", error, error_message);\
            }\
    }

#define RESOURCE_NAME   "USB::0xAAD::0x7::100001" // 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 (rssism_init (resourceName, IDQuery, resetDevice, &io));

    printf (

    "\n --- Set LF Frequency and Level ----------------------------------------\n\n"
    "\t- LF Frequency: 50 kHz\n"
    "\t- LF Level: 1.0 V\n"
    "\n ----------------------------------------------------------------------\n\n"
```

```
    );

    /* Configure LF Output */
    CHECKERR (rssism_configLFOutput (io, VI_TRUE, 50000.0, 1.0));

    printf ("\n");

    CHECKERR (rssism_close (io));

    return 0;
}
```

2.7.3.2 Execution Result

```
Line 49 (1789 ms): rssism_init (resourceName, IDQuery, resetDevice, &io)

 --- Set LF Frequency and Level ----------------------------------------

      - LF Frequency: 50 kHz
      - LF Level: 1.0 V

 ------------------------------------------------------------------------

 Line 61 (70 ms): rssism_configLFOutput (io, VI_TRUE, 50000.0, 1.0)

 Line 65 (1 ms): rssism_close (io)
```

2.7.4 Setting of AM Modulation with all Parameters and Internal Source

2.7.4.1 Source Code

```
/*****************************************************************************
 *
 * Title:   Setting of AM Modulation with all parameters and internal source
 *
 * Purpose: This example shows how to set AM parameters.
 *
 *****************************************************************************/

#include <ansi_c.h>
#include "rssism.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)\
            {\
            rssism_error_message (io, error, error_message);\
            printf("\tFunction Call Status: 0x%08X, %s\n", error, error_message);\
            rssism_error_query(io, &error, error_message);\
            printf("\tInstrument Error: 0x%X, %s\n", error, error_message);\
            }\
    }

#define RESOURCE_NAME   "USB::0xAAD::0x7::100001" // 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 (rssism_init (resourceName, IDQuery, resetDevice, &io));

    printf (

    "\n --- Set AM Parameters ------------------------------------------------\n\n"
    "\t- RF Frequency: 100 MHz\n"
```

```
    "\t- RF Level: -20.0 dBm\n"
    "\t- Modulation: AM\n"
    "\t- Modulation Source: Internal\n"
    "\t- Modulation Frequency: 1 kHz\n"
    "\t- Modulation Depth: 100.0%%\n"
    "\n ------------------------------------------------------------------------\n\n"

    );

    /* Configure RF Frequency */
    CHECKERR (rssism_configRFFreq (io, 1.0e8, 0.0, 1.0));
    /* Configure RF Level */
    CHECKERR (rssism_configRFLevel (io, -20.0, 0.0, 0.0));
    /* Set AM Source */
    CHECKERR (rssism_setAMSource (io, 0));
    /* Set AM Frequency */
    CHECKERR (rssism_setAMFreq (io, 1000.0));
    /* Set AM Depth */
    CHECKERR (rssism_setAMDepth (io, 100.0));
    /* Set AM State */
    CHECKERR (rssism_setAMState (io, VI_TRUE));
    /* Set RF Output State */
    CHECKERR (rssism_setRFOutputState (io, VI_TRUE));

    printf ("\n");

    CHECKERR (rssism_close (io));

    return 0;
}
```

2.7.4.2 Execution Result

```
Line 49 (1884 ms): rssism_init (resourceName, IDQuery, resetDevice, &io)

 --- Set AM Parameters ------------------------------------------------

      - RF Frequency: 100 MHz
      - RF Level: -20.0 dBm
      - Modulation: AM
      - Modulation Source: Internal
      - Modulation Frequency: 1 kHz
      - Modulation Depth: 100.0%

 ----------------------------------------------------------------------

 Line 65 (54 ms): rssism_configRFFreq (io, 1.0e8, 0.0, 1.0)
 Line 67 (56 ms): rssism_configRFLevel (io, -20.0, 0.0, 0.0)
 Line 69 (40 ms): rssism_setAMSource (io, 0)
 Line 71 (40 ms): rssism_setAMFreq (io, 1000.0)
 Line 73 (40 ms): rssism_setAMDepth (io, 100.0)
 Line 75 (39 ms): rssism_setAMState (io, VI_TRUE)
 Line 77 (23 ms): rssism_setRFOutputState (io, VI_TRUE)

 Line 81 (1 ms): rssism_close (io)
```

2.7.5 Setting of FM Modulation with External Source

2.7.5.1 Source Code

```
/*****************************************************************************
 *
 * Title:   Setting of FM Modulation with all parameters and external source
 *
 * Purpose: This example shows how to set AM parameters.
 *
 *****************************************************************************/

#include <ansi_c.h>
#include "rssism.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)\
            {\
            rssism_error_message (io, error, error_message);\
            printf("\tFunction Call Status: 0x%08X, %s\n", error, error_message);\
            rssism_error_query(io, &error, error_message);\
            printf("\tInstrument Error: 0x%X, %s\n", error, error_message);\
            }\
    }

#define RESOURCE_NAME   "USB::0xAAD::0x7::100001" // 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 (rssism_init (resourceName, IDQuery, resetDevice, &io));

    printf (

    "\n --- Set FM Parameters ------------------------------------------------\n\n"
    "\t- RF Frequency: 100 MHz\n"
    "\t- RF Level: -20.0 dBm\n"
    "\t- Modulation: FM\n"
    "\t- Modulation Source: External (AC Coupling)\n"
```

```
    "\t- Modulation Frequency: 1 kHz\n"
    "\t- Modulation Deviation: 1 kHz (Internal source only)\n"
    "\t- Signal Polarity: Inverted\n"
    "\n ------------------------------------------------------------------------\n\n"

    );

    /* Configure RF Frequency */
    CHECKERR (rssism_configRFFreq (io, 1.0e8, 0.0, 1.0));
    /* Configure RF Level */
    CHECKERR (rssism_configRFLevel (io, -20.0, 0.0, 0.0));
    /* Set FM Source */
    CHECKERR (rssism_setFMSource (io, 1));
    /* Set FM External Coupling */
    CHECKERR (rssism_setFMExtCoupling (io, 1, 0));
    /* Set FM Frequency */
    CHECKERR (rssism_setFMFreq (io, 1000.0));
    /* Set FM Deviation */
    CHECKERR (rssism_setFMDeviation (io, 1000.0));
    /* Set FM Polarity */
    CHECKERR (rssism_setFMPolar (io, 1));
    /* Set FM State */
    CHECKERR (rssism_setFMState (io, VI_TRUE));
    /* Set RF Output State */
    CHECKERR (rssism_setRFOutputState (io, VI_TRUE));

    printf ("\n");

    CHECKERR (rssism_close (io));

    return 0;
}
```

2.7.5.2 Execution Result

```
Line 49 (1841 ms): rssism_init (resourceName, IDQuery, resetDevice, &io)

 --- Set FM Parameters ------------------------------------------------

      - RF Frequency: 100 MHz
      - RF Level: -20.0 dBm
      - Modulation: FM
      - Modulation Source: External (AC Coupling)
      - Modulation Frequency: 1 kHz
      - Modulation Deviation: 1 kHz (Internal source only)
      - Signal Polarity: Inverted

 ----------------------------------------------------------------------

 Line 66 (54 ms): rssism_configRFFreq (io, 1.0e8, 0.0, 1.0)
 Line 68 (56 ms): rssism_configRFLevel (io, -20.0, 0.0, 0.0)
 Line 70 (39 ms): rssism_setFMSource (io, 1)
 Line 72 (40 ms): rssism_setFMExtCoupling (io, 1, 0)
 Line 74 (40 ms): rssism_setFMFreq (io, 1000.0)
 Line 76 (39 ms): rssism_setFMDeviation (io, 1000.0)
 Line 78 (40 ms): rssism_setFMPolar (io, 1)
 Line 80 (40 ms): rssism_setFMState (io, VI_TRUE)
 Line 82 (24 ms): rssism_setRFOutputState (io, VI_TRUE)

 Line 86 (6 ms): rssism_close (io)
```

2.7.6 Setting of RF Level Sweep

2.7.6.1 Source Code

```
/*****************************************************************************
 *
 * Title:   Setting of RF Level Sweep
 *
 * Purpose: This example shows how to set RF level sweep parameters.
 *
 *****************************************************************************/

#include <ansi_c.h>
#include "rssism.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)\
            {\
            rssism_error_message (io, error, error_message);\
            printf("\tFunction Call Status: 0x%08X, %s\n", error, error_message);\
            rssism_error_query(io, &error, error_message);\
            printf("\tInstrument Error: 0x%X, %s\n", error, error_message);\
            }\
    }

#define RESOURCE_NAME   "USB::0xAAD::0x7::100001" // 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 (rssism_init (resourceName, IDQuery, resetDevice, &io));

    printf (

    "\n --- Set RF Level Sweep ------------------------------------------------\n\n"
    "\t- RF Frequency: 100.0 MHz\n"
    "\t- RF Start Level: -40.0 dBm\n"
    "\t- RF Stop Level: 0.0 dBm\n"
    "\t- Step Width: 1.0 dB\n"
```

```
    "\t- Dwell Time: 0.01 s\n"
    "\n ------------------------------------------------------------------------\n\n"

    );

    /* Configure RF Frequency */
    CHECKERR (rssism_configRFFreq (io, 1.0e8, 0.0, 1.0));
    /* Configure RF Level Sweep */
    CHECKERR (rssism_configRFLevelSweep (io, 0, -40.0, 0.0, 1.0, 0.01));
    /* Set Sweep Trigger Source */
    CHECKERR (rssism_setSweepTrigSource (io, 3, 2));
    /* Set RF Output State */
    CHECKERR (rssism_setRFOutputState (io, VI_TRUE));

    printf ("\n");

    CHECKERR (rssism_close (io));

    return 0;
}
```

2.7.6.2 Execution Result

```
Line 49 (1881 ms): rssism_init (resourceName, IDQuery, resetDevice, &io)

 --- Set RF Level Sweep ------------------------------------------------

      - RF Frequency: 100.0 MHz
      - RF Start Level: -40.0 dBm
      - RF Stop Level: 0.0 dBm
      - Step Width: 1.0 dB
      - Dwell Time: 0.01 s

 ------------------------------------------------------------------------

 Line 64 (54 ms): rssism_configRFFreq (io, 1.0e8, 0.0, 1.0)
 Line 66 (120 ms): rssism_configRFLevelSweep (io, 0, -40.0, 0.0, 1.0, 0.01)
 Line 68 (56 ms): rssism_setSweepTrigSource (io, 3, 2)
 Line 70 (24 ms): rssism_setRFOutputState (io, VI_TRUE)

 Line 74 (1 ms): rssism_close (io)
```

2.7.7 Setting of RF Frequency Sweep

2.7.7.1 Source Code

```
/*****************************************************************************
 *
 * Title:   Setting of RF Frequency Sweep
 *
 * Purpose: This example shows how to set RF frequency sweep parameters.
 *
 *****************************************************************************/

#include <ansi_c.h>
#include "rssism.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)\
            {\
            rssism_error_message (io, error, error_message);\
            printf("\tFunction Call Status: 0x%08X, %s\n", error, error_message);\
            rssism_error_query(io, &error, error_message);\
            printf("\tInstrument Error: 0x%X, %s\n", error, error_message);\
            }\
    }

#define RESOURCE_NAME   "USB::0xAAD::0x7::100001" // 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 (rssism_init (resourceName, IDQuery, resetDevice, &io));

    printf (

    "\n --- Set RF Frequency Sweep ----------------------------------------\n\n"
    "\t- RF Start Frequency: 100 MHz\n"
    "\t- RF Stop Frequency: 500 MHz\n"
    "\t- Step Width: 1 MHz\n"
```

```
    "\t- Dwell Time: 0.01 s\n"
    "\t- RF Level: -20.0 dBm\n"
    "\n ------------------------------------------------------------------------\n\n"

    );

    /* Configure RF Level */
    CHECKERR (rssism_configRFLevel (io, -20.0, 0.0, 0.0));
    /* Configure RF Frequency Sweep */
    CHECKERR (rssism_configRFFreqSweep (io, 0, 100.0e6, 500.0e6, 1.0e6, 0.01));
    /* Set Sweep Trigger Source */
    CHECKERR (rssism_setSweepTrigSource (io, 1, 2));
    /* Set RF Output State */
    CHECKERR (rssism_setRFOutputState (io, VI_TRUE));

    printf ("\n");

    CHECKERR (rssism_close (io));

    return 0;
}
```

2.7.7.2 Execution Result

```
Line 49 (1755 ms): rssism_init (resourceName, IDQuery, resetDevice, &io)

 --- Set RF Frequency Sweep ------------------------------------------

      - RF Start Frequency: 100 MHz
      - RF Stop Frequency: 500 MHz
      - Step Width: 1 MHz
      - Dwell Time: 0.01 s
      - RF Level: -20.0 dBm

 ---------------------------------------------------------------------

 Line 64 (54 ms): rssism_configRFLevel (io, -20.0, 0.0, 0.0)
 Line 66 (136 ms): rssism_configRFFreqSweep (io, 0, 100.0e6, 500.0e6, 1.0e6, 0.01)
 Line 68 (56 ms): rssism_setSweepTrigSource (io, 1, 2)
 Line 70 (24 ms): rssism_setRFOutputState (io, VI_TRUE)

 Line 74 (1 ms): rssism_close (io)
```

2.7.8 Setting of Pulse Modulation with Internal Source

2.7.8.1 Source Code

```
/*****************************************************************************
 *
 * Title:   Setting of Pulse Modulation with internal source
 *
 * Purpose: This example shows how to set Pulse Modulation parameters.
 *
 *****************************************************************************/

#include <ansi_c.h>
#include "rssism.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)\
            {\
            rssism_error_message (io, error, error_message);\
            printf("\tFunction Call Status: 0x%08X, %s\n", error, error_message);\
            rssism_error_query(io, &error, error_message);\
            printf("\tInstrument Error: 0x%X, %s\n", error, error_message);\
            }\
    }

#define RESOURCE_NAME   "USB::0xAAD::0x7::100001" // 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 (rssism_init (resourceName, IDQuery, resetDevice, &io));

    printf (

    "\n --- Set Pulse Modulation  Parameters ----------------------------------\n\n"
    "\t- RF Frequency: 100 MHz\n"
    "\t- RF Level: -20.0 dBm\n"
    "\t- Modulation: Pulse\n"
```

```
    "\t- Modulation Source: Internal\n"
    "\t- Pulse Parameters: Off-Time = 10 ms, OnTime = 100 ms\n"
    "\t- Pulse Delay: 0 s\n"
    "\n ------------------------------------------------------------------------\n\n"

    );

    /* Configure RF Frequency */
    CHECKERR (rssism_configRFFreq (io, 1.0e8, 0.0, 1.0));
    /* Configure RF Level */
    CHECKERR (rssism_configRFLevel (io, -20.0, 0.0, 0.0));
    /* Set PM Source */
    CHECKERR (rssism_setPMSource (io, 0));
    /* Set PM Polarity */
    CHECKERR (rssism_setPMPolar (io, 0));
    /* Set PM Pulse Off Time */
    CHECKERR (rssism_setPMPulseOffTime (io, 10.0e-3));
    /* Set PM Pulse On Time */
    CHECKERR (rssism_setPMPulseOnTime (io, 100.0e-3));
    /* Set PM Pulse Delay Time */
    CHECKERR (rssism_setPMPulseDelayTime (io, 0.0));
    /* Set PM State */
    CHECKERR (rssism_setPMState (io, VI_TRUE));
    /* Set RF Output State */
    CHECKERR (rssism_setRFOutputState (io, VI_TRUE));

    printf ("\n");

    CHECKERR (rssism_close (io));

    return 0;
}
```

2.7.8.2 Execution Result

```
Line 49 (1935 ms): rssism_init (resourceName, IDQuery, resetDevice, &io)

 --- Set Pulse Modulation  Parameters ----------------------------------

      - RF Frequency: 100 MHz
      - RF Level: -20.0 dBm
      - Modulation: Pulse
      - Modulation Source: Internal
      - Pulse Parameters: Off-Time = 10 ms, OnTime = 100 ms
      - Pulse Delay: 0 s

 ------------------------------------------------------------------------

 Line 65 (53 ms): rssism_configRFFreq (io, 1.0e8, 0.0, 1.0)
 Line 67 (56 ms): rssism_configRFLevel (io, -20.0, 0.0, 0.0)
 Line 69 (39 ms): rssism_setPMSource (io, 0)
 Line 71 (39 ms): rssism_setPMPolar (io, 0)
 Line 73 (39 ms): rssism_setPMPulseOffTime (io, 10.0e-3)
 Line 75 (39 ms): rssism_setPMPulseOnTime (io, 100.0e-3)
 Line 77 (39 ms): rssism_setPMPulseDelayTime (io, 0.0)
 Line 79 (39 ms): rssism_setPMState (io, VI_TRUE)
 Line 81 (23 ms): rssism_setRFOutputState (io, VI_TRUE)

 Line 85 (0 ms): rssism_close (io)
```