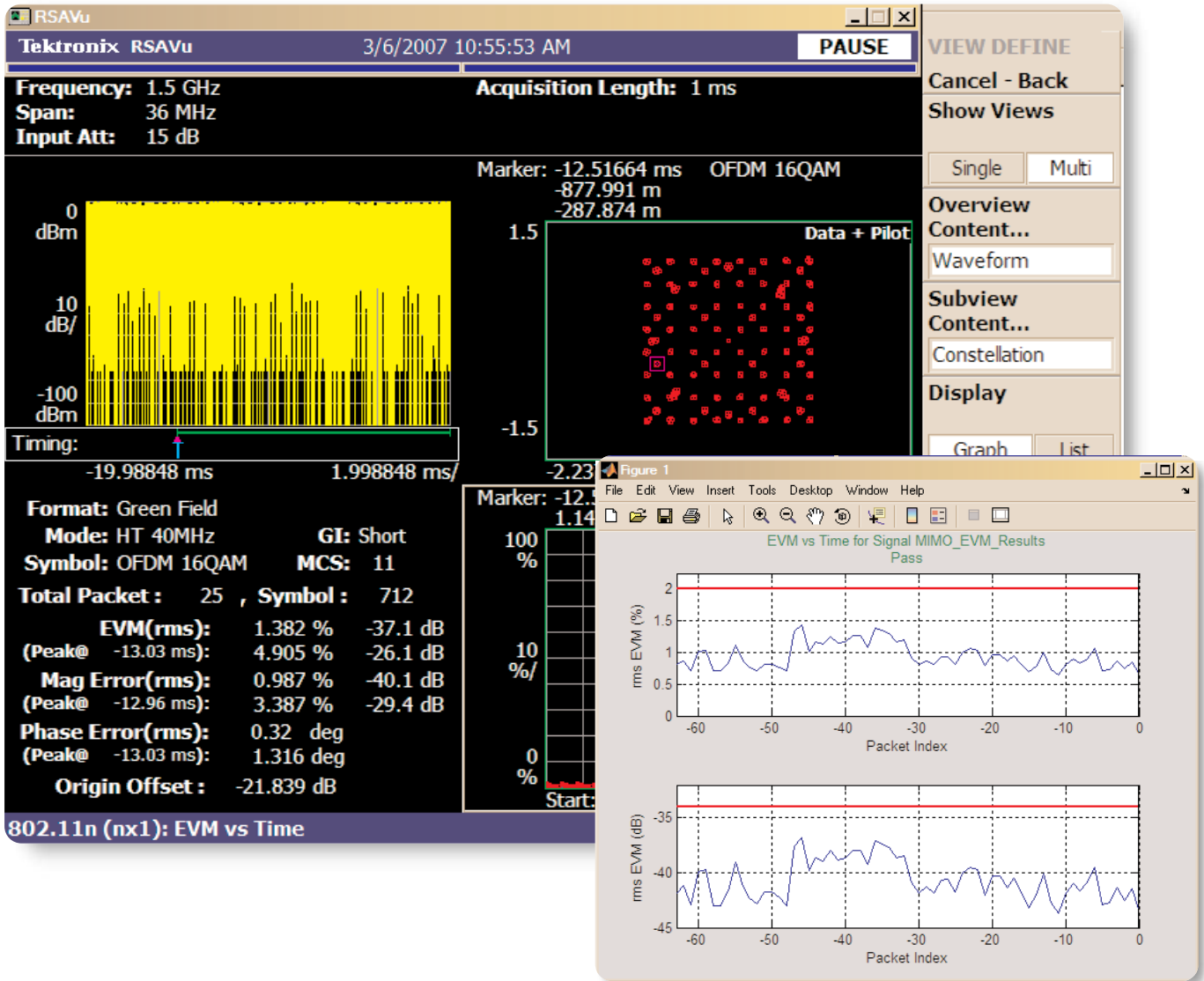


# RSAVu Remote Control Demos



## Overview

This document describes how to communicate remotely with RSAVu via SCPI commands by the use of two examples, one in C++ and one in MATLAB<sup>1</sup>. The examples analyze an 802.11n MIMO 2x2 signal and create a text report based on the measurement

results acquired programmatically from RSAVu. A 2x2 MIMO channel has two transmitters and two receivers, so users will need two real-time spectrum analyzers to capture the MIMO signals. The signals provided for this demo were captured with two Tektronix RSA6114A real-time spectrum analyzers.

<sup>1</sup>MATLAB is a registered trademark of The MathWorks Inc. ©1994-2007

## RSAVu Remote Control Demos

### ► Reference Guide

The examples perform the following tasks to analyze the 802.11n MIMO signal:

1. Make the connection to RSAVu using TekVISA
2. Setup the 802.11n MIMO 2x2 measurement mode in RSAVu
3. Load stored waveforms into RSAVu
4. Run the analysis on RSAVu
5. Retrieve measured results from the subcarrier constellation (SCC) measurement for 64 packets using the programmatic interface
6. Close the connection to RSAVu
7. Process the results to create a simple text report of rms EVM values

The MATLAB example also creates a plot that compares the measured EVM values against a threshold, which would be useful for determining if the device under test is meeting its EVM specifications.

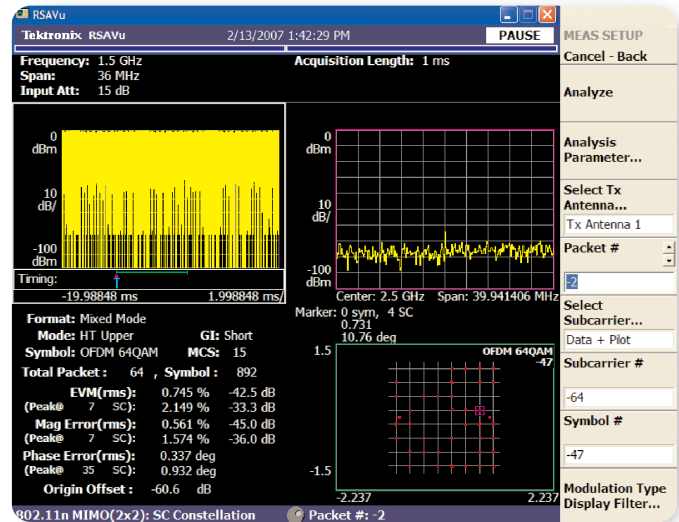
A screenshot of the 802.11n MIMO 2x2 measurement in RSAVu is shown in Figure 1, with the subcarrier constellation plot for this packet shown in the lower right hand corner and the numerical readouts in the lower left hand corner. For this packet, the signal had a 64-QAM constellation. The two captured waveforms have no fading added, so the estimated channel response will be flat. A variety of signal formats and modulation types is used over 64 packets, with the 64 packet sequence repeating over and over.

### Demo Requirements

To run the demos, you must meet the following requirements:

1. RSAVu installed on your PC with the remote interface turned on
2. TekVISA v3.0 installed on your PC (must be installed after RSAVu)
3. A license to run the 802.11n MIMO 2x2 option
4. For the MATLAB demo, you must have a valid MATLAB license.

Source code is also provided for both the C++ and MATLAB example, and the same principles used in this



► Figure 1. Example of an 802.11n MIMO 2x2 Demod.

MIMO example can be used for the other measurement modes of RSAVu.

### RSAVu Setup

To enable the programmatic interface on RSAVu:

1. After installing RSAVu, install TekVISA v3.0. If you have a previously installed version of TekVISA, you will need to re-install TekVISA after the RSAVu install to enable the remote control functionality.
2. After installing TekVISA, load RSAVu and turn on the remote control capability by clicking on the RSAVu title bar to bring up the front panel display, select “System”, and then on the soft key menus select “Remote Setup ... > Remote Interface > On”.
3. Right-click on the yellow TekVISA icon in the Windows tray and select “Instrument Manager...”. When the TekVISA Instrument Manager loads, you should see the following in your list of available instruments: GPIB8::1::INSTR. This is the address you will use to communicate with RSAVu on your PC if you are writing your own programs. The examples are already programmed to know this address.

Once the remote functionality is turned on in RSAVu, you are ready to run the examples.

## C++ Example

### Installation

Unzip the file RSAVu\_MIMO\_Demo.zip. It will create a directory called RSAVu\_MIMO\_Demo which must be copied to the root of your C: drive for the C++ example to run (some directory names are hard-coded in the example). The example can be launched by double clicking the executable in C:\RSAVu\_MIMO\_Demo\C\C\_RSAVu\_Demo.exe. The source code is located in C\_RSAVu\_Demo.cpp in the same directory.

### Running the Example

Once you double-click the executable, a console window will come up as shown in Figure 2. The message “Starting the 802.11n MIMO measurements in RSAVu.” will be displayed when the console is first launched. The basic steps in the example are:

1. VISA Setup
2. RSA Setup
3. Retrieve the measurements from RSAVu
4. Close the session
5. Process the results

#### VISA Setup

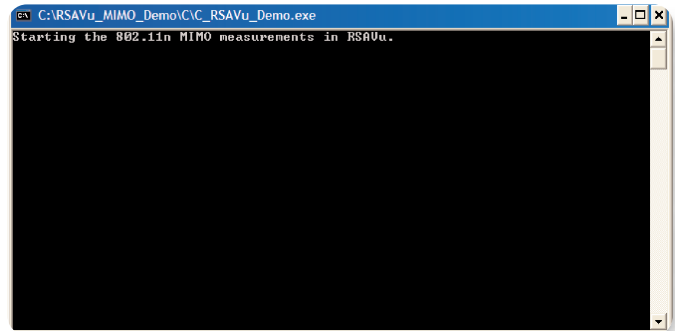
As can be seen from the source code, the first step in the example is to initialize the VISA system and open a connection to RSAVu. The example uses the built-in functions and types available by including the header file visa.h, which is installed during the TekVISA installation. For more information on the TekVISA functions, see the TekVISA Programmer Manual that is installed with TekVISA.

The steps to make the connection to RSAVu are shown below (the parameter setup and error checking that is present in the full source code is omitted here for clarity).

```
status = viOpenDefaultRM( &rm );
status = viOpen( rm, "GPIB8::1::INSTR", VI_NULL,
VI_NULL, &vi );
```

To access data on RSAVu, this example uses the formatted I/O services viPrintf and viQueryf, so the viSetAttribute function is used first to setup the connection to enable these services.

```
status = viSetAttribute( vi, VI_ATTR_WR_BUF_OPER_
MODE, VI_FLUSH_ON_ACCESS );
status = viSetAttribute( vi, VI_ATTR_RD_BUF_OPER_
MODE, VI_FLUSH_ON_ACCESS );
```



▶ Figure 2. C++ Demo Startup Screen.

Next, the timeout is set to 10 seconds (the attribute is set in milliseconds and thus we send 10000 to indicate a 10 second timeout).

```
status = viSetAttribute( vi, VI_ATTR_TMO_VALUE, 10000 );
```

The connection to RSAVu is now open and we are ready to begin sending SCPI commands to RSAVu.

#### RSA Setup

The first step is to put RSAVu into the 802.11n MIMO 2x2 measurement mode, set the default settings, and then select the subcarrier constellation (SCC) measurement. The SCC measurement includes the rms EVM values we will collect for the demo, as well as other measurements related to modulation quality.

```
status = viPrintf( vi, ":INST:SEL \"DEMM2WLAN\"\n" );
status = viPrintf( vi, ":CONF:M2WLAN\n" );
status = viPrintf( vi, ":SENS:M2WLAN:MEAS SCC\n");
```

The next step is to load the two MIMO waveforms, the signals acquired by the two RSA6114A's. These are stored in the TIQ\_Files directory as MIMO\_RX1.tiq and MIMO\_RX2.tiq. The first MIMO signal must be loaded before the second.

```
status = viPrintf( vi, ":MMEM:LOAD:RX1
\"C:\\\\RSAVu_MIMO_Demo\\\\TIQ_Files\\\\MIMO_
RX1.TIQ\"\n" );
status = viPrintf( vi, ":MMEM:LOAD:RX2
\"C:\\\\RSAVu_MIMO_Demo\\\\TIQ_Files\\\\MIMO_
RX2.TIQ\"\n" );
```

Next we set the analysis length to be 6.5ms, which for this signal represents 64 packets.

```
status = viPrintf( vi, ":SENS:M2WLAN:ANAL:
LENG 6.5E-3\n" );
```

## RSAVu Remote Control Demos

### ► Reference Guide

Next the demo program instructs RSAVu to analyze the two MIMO waveforms and produce SCC measurements for the 64 packets. Because this operation will take longer than most, we increase the timeout to be 4 minutes, and then reset it to 10 seconds after the operation is complete. We then scan for the operation to be completed by waiting for the OPC to be set to 1.

```
status = viSetAttribute( vi, VI_ATTR_TMO_
VALUE, 240000 );
status = viPrintf( vi, "SENS:M2WLAN:IMM\n" );
while ( intOPC != 1) status = viQueryf( vi, "*OPC?\n",
"%d",&intOPC );
status = viSetAttribute( vi, VI_ATTR_TMO_VALUE, 10000 );
```

### Retrieve the Measurement Results from RSAVu

Now that the analysis is complete, you can retrieve the measurement data from the 64 packets. The SCC measurement returns multiple values when called via the SCPI command, so the code parses the returned text string into the various values of the SCC measurement. This example only uses the rms EVM values when creating the text report, but all of the results are first parsed from the string returned by the SCC measurement.

The console window will display "Retrieving 802.11n MIMO results from RSAVu." when this section of the example is reached. Since RSAVu will index the packets from -63 to 0, we create an index called `iPacket` that will also increment from -63 to 0 as the loop counter `i` increments from 0 to 63.

```
int numResults = 64;           // Number of packets to
                               // read from RSAVu
int iPacket = -numResults;     // Match RSAVu's packet
                               // index (negative indexing)
for (i=0; i < numResults; ++i)
{
    // Advance to the next packet in RSAVu
    iPacket++;
    status = viPrintf( vi, "SENS:M2WLAN:PACK:
NUMB %i\n", iPacket );
    // Retrieve the measurement data
    status = viQueryf( vi, "FETCH:M2WLAN? SCC\n",
"%s", buffer );
}
```

```
peakEVM_per[i] = atof( pch );
peakEVM_dB[i] = atof( pch );
rmsEVM_per[i] = atof( pch );
rmsEVM_dB[i] = atof( pch );
SC_number_EVM[i] = atoi( pch );
peak_MError_per[i] = atof( pch );
peak_MError_dB[i] = atof( pch );
rms_MError_per[i] = atof( pch );
rms_MError_dB[i] = atof( pch );
SC_number_MError[i] = atoi( pch );
peak_PError_deg[i] = atof( pch );
rms_PError_deg[i] = atof( pch );
SC_number_PError[i] = atoi( pch );
}
```

### Close the Session

Now that we have acquired all of the measurement results, we can close the session before we process the results we have acquired.

```
viClose( vi );
viClose( rm );
```

### Process the Results

The rest of the demo illustrates one small example of what could be done with the data acquired programmatically from RSAVu. In this instance, you are creating a text file that summarizes some basic test information (model number and serial number of the RSA's), the max and mean values of the rms EVM values

of the 64 packets as both a percentage and in dB, and then the rms EVM of each packet as both a percentage and in dB. Note that the mean value expressed in dB is the dB value of the linear EVM values and not the mean of the dB values, as the logarithm is not a linear process and thus the averaging should occur before the logarithm is taken.

```
GenerateReport( resultsFilename, serial_RSA1, serial_
RSA2, numResults, rmsEVM_per, rmsEVM_dB );
```

The generated report will have the name MIMO\_EVM\_Results.txt and look like this:

```
Filename: MIMO_EVM_Results.txt
RSA #1:  RSA6114A_B000001
RSA #2:  RSA6114A_B000002

Max EVM (%):  1.438987
Mean EVM (%): 0.943646

Max EVM (dB): -36.838865
Mean EVM (dB): -40.503814
```

```
=====
```

Packet	rms EVM (%)	rms EVM (dB)
-63	0.8068	-41.86
-62	0.8695	-41.21
-61	0.7174	-42.88
-60	1.0089	-39.92
(Packets -59 through -6 snipped for brevity)		
-5	0.7185	-42.87
-4	0.7296	-42.74
-3	0.8570	-41.34
-2	0.7453	-42.55
-1	0.8489	-41.42
0	0.6822	-43.32

**MATLAB Example**

The MATLAB example is very similar to the C++ example, with a few minor additions. First, it saves the minimum EVM values to the text report in addition to the

mean and max values. Second, it stores all of the SCC measurement results into a MATLAB file. Finally, it creates a plot of rms EVM vs time with comparisons to a limit, flagging any points that occur above limit and showing a PASS/FAIL indicator on the plot.

Also, the SCPI commands used in the MATLAB example use the full version of the commands, whereas the C++ example used the abbreviated version. The full versions are a little easier to read and interpret if a person is reading the code, but the abbreviated versions require a few less bytes to be sent. It is up to the user to choose a format they prefer, both versions are functionally equivalent.

**Installation**

The MATLAB example has no limitations on its location on the hard disk. To run the example, launch MATLAB and then change to the directory RSAVu\_MIMO\_Demo\MATLAB and run the MATLAB script MATLAB\_RSAVu\_Demo.m.

**User Parameter Setup**

Since the MATLAB example is not compiled, you can change a few settings if you'd like. The most likely ones to be changed by the user are in the "User Setup Parameters" section at the top of the file. The filenames in fileRX1 and fileRX2 should point to the location of the MIMO signals that will be loaded later. By default, these locations point to the TIQ\_Files directory included in the zip file.

The filename that will be used for creating the text report and the saved variables is set by the variable resultsFilename, which should not have an extension. By default, it is set to MIMO\_EVM\_Results. The program will later add the appropriate .txt extension for the text report it creates and the .mat extension for the saved variables.

Next, the model and serial numbers of the two RSA's used to capture the MIMO waveforms are stored in rsa1 and rsa2. The program only uses these to write the information into the top of the text report.

Finally, the rms EVM limit can be changed (by default it is 2.0), this is the EVM value in percent that the program will use to determine if the EVM results are passing or failing relative to the limit.

## RSAVu Remote Control Demos

### ► Reference Guide

#### VISA Setup

As with the C++ example, the first step is to setup the VISA connection. As before, we will setup and open a connection to GPIB8::1::INSTR and then set the timeout to 10 seconds. Note that in the C++ example, the attribute is set in milliseconds, whereas the MATLAB command uses seconds.

```
instrVISA = visa( 'tek', ['GPIB8::1::INSTR'] );
fopen( instrVISA );
set( instrVISA, 'Timeout', 10.0 );
```

#### RSA Setup

Next, RSAVu queries for the current mode and switch to the 802.11n MIMO 2x2 mode if it is not already selected. If it cannot be selected (typically because a valid license cannot be found), the program will exit. You can use the `fprintf` and query commands to send and retrieve data from RSAVu. As with the C++ example, only the main commands will be reproduced here, commands like initialization and error checking will not be included here for clarity.

```
strMode = query( instrVISA, ':INSTrument:SElect?' );
if length(strMode) < 11 ||
strcmp(strMode(1:11),"DEMM2WLAN") == 0,
    fprintf( instrVISA, ':INSTrument:SElect
    "DEMM2WLAN" );
    strMode = query( instrVISA, ':INSTrument:SElect?' );
    if length(strMode) < 11 || strcmp(strMode(1:11),
    "DEMM2WLAN") == 0,
        % Can't set the proper mode, so time to quit.
        fclose(instrVISA);
        return;
    end
end
```

Next, set the default parameters for this mode and then select the subcarrier constellation (SCC) measurement.

```
fprintf( instrVISA, ':CONFigure:M2WLAN');
fprintf( instrVISA, ':SENSe:M2WLAN:MEASurement
SCConste' );
```

Now that you have setup RSAVu for 802.11n MIMO 2x2 analysis, you can load in the two MIMO waveforms.

```
fprintf( instrVISA, [':MMEMory:LOAD:RX1 "'
fullpathRX1 "' ] );
fprintf( instrVISA, [':MMEMory:LOAD:RX2 "'
fullpathRX2 "' ] );
```

As with the C++ example, it is required that the RX1 file be loaded first. Next, we set the analysis length to 6.5ms, which is the length of the 64 packets we wish to analyze.

```
fprintf( instrVISA, ':SENSe:M2WLAN:ANALysis:
LENGth 6.5E-3' );
```

Next, set the timeout to 4 minutes, start the analysis, wait for operation complete, and then set the timeout back to 10 seconds.

```
set(instrVISA,'Timeout',240.0);    % Timeout = 4 minutes
fprintf( instrVISA, ':SENSe:M2WLAN:IMMediate' );
Wait4OPC( instrVISA );            % Wait for
                                   operation complete
set(instrVISA,'Timeout',10.0);    % Timeout = 10 seconds
```

When operation complete is detected, we are ready to start retrieving the measurement results.

#### Retrieve the Measurement Results from RSAVu

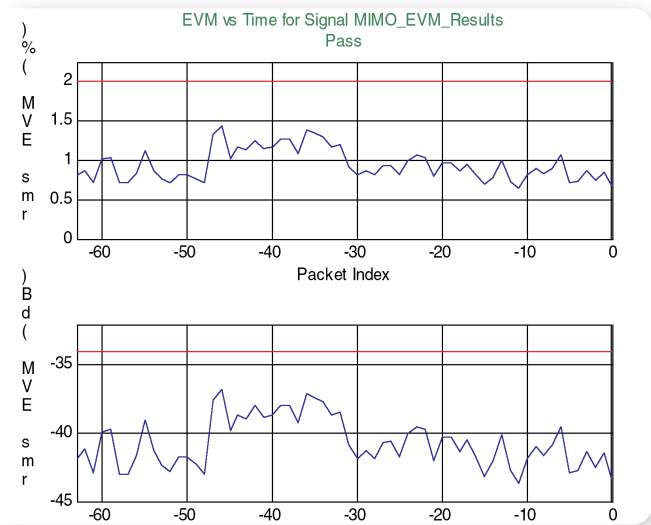
As in the C++ example, the next step is to request the measurement results for each of the 64 packets analyzed in the previous section. We will store all of the SCC measurements into structure called SCC, with each variable in the structure storing the 64 results of the associated measurement, such as rms EVM in percent and rms EVM in dB. Because the SCC measurement returns a number of results in the text string, the demo example parses the string to find the individual results of the SCC measurement. As before, since RSAVu indexes the 64 analyzed packets from -63 to 0, the example creates a variable called `iPacket` that will increment from -63 to 0 as the loop counter `i` increments from 1 to 64. We start the loop counter at 1 in the MATLAB version instead of 0 like in the C++ version since MATLAB starts its vector indexes at 1 and C++ starts at 0.

```

numResults = 64;           % Number of packets to read
                           % from RSAVu
iPacket = -numResults;    % Match RSAVu's packet
                           % index (negative indexing)
for i=1:numResults,
    % Advance to the next packet in RSAVu.
    iPacket = iPacket + 1;
    fprintf( instrVISA, [':SENSe:M2WLAN:PACKet:
NUMBER ' num2str(iPacket)]);

    % Retrieve the measurement data for this packet
    str_SCC = query( instrVISA, [ ':FETCH:
M2WLAN? SCConste' ] );
    [temp,rem] = strtok(str_SCC,',' );

    SCC.peakEVM_per(i)     = str2num(temp);
                           [temp,rem] = strtok(rem,',' );
    SCC.peakEVM_dB(i)      = str2num(temp);
                           [temp,rem] = strtok(rem,',' );
    SCC.rmsEVM_per(i)      = str2num(temp);
                           [temp,rem] = strtok(rem,',' );
    SCC.rmsEVM_dB(i)       = str2num(temp);
                           [temp,rem] = strtok(rem,',' );
    SCC.SC_number_EVM(i)   = str2num(temp);
                           [temp,rem] = strtok(rem,',' );
    SCC.peak_MError_per(i) = str2num(temp);
                           [temp,rem] = strtok(rem,',' );
    SCC.peak_MError_dB(i)  = str2num(temp);
                           [temp,rem] = strtok(rem,',' );
    SCC.rms_MError_per(i)  = str2num(temp);
                           [temp,rem] = strtok(rem,',' );
    SCC.rms_MError_dB(i)   = str2num(temp);
                           [temp,rem] = strtok(rem,',' );
    SCC.SC_number_MError(i) = str2num(temp);
                           [temp,rem] = strtok(rem,',' );
    SCC.peak_PError_deg(i) = str2num(temp);
                           [temp,rem] = strtok(rem,',' );
    SCC.rms_PError_deg(i)  = str2num(temp);
                           [temp,rem] = strtok(rem,',' );
    SCC.SC_number_PError(i) = str2num(temp);
end
  
```



▶ **Figure 3.** A Pass Condition for the rms EVM compared to a 2.0% Limit.

**Close the Session**

Now that you have acquired all of the data we need from RSAVu, you can close down the connection.

```
fclose( instrVISA );
```

**Process the Results**

The first step is to save all of the SCC measurement data collected into a MATLAB data file so it can be retrieved later if desired.

```
save(resultsFilename,'SCC'); % Save all SCC results
                               % into a MAT file
```

The filename will be based off of the filename we set in the “User Setup Parameters” at the top of the file and will automatically have the .mat extension added.

Next, create a text report much like the C++ version, with the only difference here being that we also save the minimum EVM values (in percent and dB) in addition to the maximum and mean values.

Finally, a graphic is created that compares the rms EVM results to a limit and indicates a PASS or FAIL condition. Each point that fails will be flagged with a red diamond on the trace. A solid red trace indicates the limit, with the blue trace indicating the rms EVM values that were acquired from RSAVu. The title will show PASS (in green) if all points are below the limit and FAIL (in red) if one or more points is above the limit. An example of a plot showing a passing condition is shown in Figure 3.

## Contact Tektronix:

ASEAN / Australasia (65) 6356 3900  
Austria +41 52 675 3777  
Balkan, Israel, South Africa and other ISE Countries +41 52 675 3777  
Belgium 07 81 60166  
Brazil & South America (11) 40669400  
Canada 1 (800) 661-5625  
Central East Europe, Ukraine and the Baltics +41 52 675 3777  
Central Europe & Greece +41 52 675 3777  
Denmark +45 80 88 1401  
Finland +41 52 675 3777  
France +33 (0) 1 69 86 81 81  
Germany +49 (221) 94 77 400  
Hong Kong (852) 2585-6688  
India (91) 80-22275577  
Italy +39 (02) 25086 1  
Japan 81 (3) 6714-3010  
Luxembourg +44 (0) 1344 392400  
Mexico, Central America & Caribbean 52 (55) 5424700  
Middle East, Asia and North Africa +41 52 675 3777  
The Netherlands 090 02 021797  
Norway 800 16098  
People's Republic of China 86 (10) 6235 1230  
Poland +41 52 675 3777  
Portugal 80 08 12370  
Republic of Korea 82 (2) 528-5299  
Russia & CIS +7 (495) 7484900  
South Africa +27 11 254 8360  
Spain (+34) 901 988 054  
Sweden 020 08 80371  
Switzerland +41 52 675 3777  
Taiwan 886 (2) 2722-9622  
United Kingdom & Eire +44 (0) 1344 392400  
USA 1 (800) 426-2200

For other areas contact Tektronix, Inc. at: 1 (503) 627-7111  
Updated 15 September 2006

Our most up-to-date product information is available at: [www.tektronix.com](http://www.tektronix.com)



Copyright © 2007, Tektronix. All rights reserved. Tektronix products are covered by U.S. and foreign patents, issued and pending. Information in this publication supersedes that in all previously published material. Specification and price change privileges reserved. TEKTRONIX and TEK are registered trademarks of Tektronix, Inc. All other trade names referenced are the service marks, trademarks or registered trademarks of their respective companies.

2/07 FLG/WOW

37W-20552-0

**Tektronix**  
Enabling Innovation

