Application Note

AUTOMATED EMITTER CREATION FROM USER DATA FOR MISSION DATA FILE TESTING

Products:

► R&S® Pulse Sequencer Software

Walter Schulte | 1GP131 | Version 0E | 07.2021

https://www.rohde-schwarz.com/appnote/1gp131

Contents

1 Overview

As explained in the application note "Threat Simulation and Verification for Radar Warning Receiver Testing" [1], EW receiver mission data files can contain hundreds of emitters with thousands of modes and beams that must be tested by simulation at RF.

Often, these emitters, their modes and beams are listed on intelligence databases and must be imported from a spreadsheet into to an emitter simulation application such as the R&S®Pulse Sequencer software. For this purpose, the R&S Pulse Sequencer software offers an internal script editor which allows to import user data and automatically generate emitter, sequence or platform configurations that can then immediately be played into RF without any additional software.

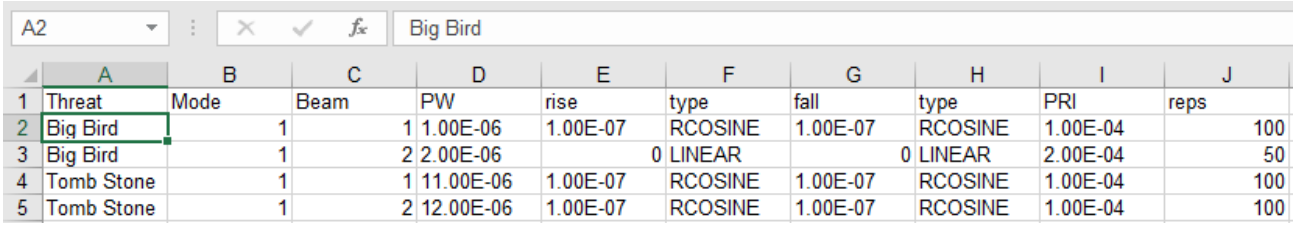
Furthermore, the R&S Pulse Sequencer offers a SCPI recorder tool to collect manual entries during scenario data creation into a list of corresponding SCPI commands. This list of commands can easily be used to create user defined scripts that can either run in the internal script editor or an external software (e.g. Matlab, Python).

2 Import Threat Data from a Spreadsheet

The JavaScript based Script Editor can be used to import threat data from a csv-spreadsheet using the “CSVToArray” method defined in Appendix B: Creating Threats From User . In the example below, data from each column of the spreadsheet is matched to the appropriate SCPI command.

```
58      ps.print('Reading csv File...');
59      file.readLine(0); // Skip first row "Column Names"
60      var prevEmitter = '';
61      var prevMode = 0;
62      var prevBeam = 0;
63      while (!file.atEnd())
64      {
65          line = file.readLine(0);
66          var arr = CSVToArray(line, ';');
67          ps.print('\nLine:\n'+arr+'\n');
68
69          // the translation of the parameters has be be performed by each customer, since it's propietary
70          // Col 0 is the name
71
72          //if emitter already exists, do not create a new one.
73          if (prevEmitter != arr[0][0])
74          {
75            ps.scpi('EMITter:CREate "' +arr[0][0]+'"');
76            ps.print('EMITter:CREate "' +arr[0][0]+'"');
77          }
78          else
79          {
80            // if mode exists but for different beam, create a new beam.
81            if (prevMode == arr[0][1] && prevBeam != arr[0][2])
82            {
83              // create a new beam
84              ps.scpi('EMITter:MODE:BEAM:ADD')
85              ps.print('EMITter:MODE:BEAM:ADD')
86            }
87
88            // if new mode, create one.
89            if (prevMode != arr[0][1])
90            {
91              // create new mode
92              ps.scpi('EMITter:MODE:ADD')
93              ps.print('EMITter:MODE:ADD')
94            }
```

Figure 1 - A 'while' loop to import a threat table from a spreadsheet.



	A	B	C	D	E	F	G	H	I	J
1	Threat	Mode	Beam	PW	rise	type	fall	type	PRI	reps
2	Big Bird	1	1	1.00E-06	1.00E-07	RCOSINE	1.00E-07	RCOSINE	1.00E-04	100
3	Big Bird	1	2	2.00E-06	0	LINEAR	0	LINEAR	2.00E-04	50
4	Tomb Stone	1	1	11.00E-06	1.00E-07	RCOSINE	1.00E-07	RCOSINE	1.00E-04	100
5	Tomb Stone	1	2	12.00E-06	1.00E-07	RCOSINE	1.00E-07	RCOSINE	1.00E-04	100

Figure 2 - A spreadsheet with the columns used in the script in Figure 1.

A complete script, including file IO, is provided in the appendix. Results of running the script on the spreadsheet above are shown below.

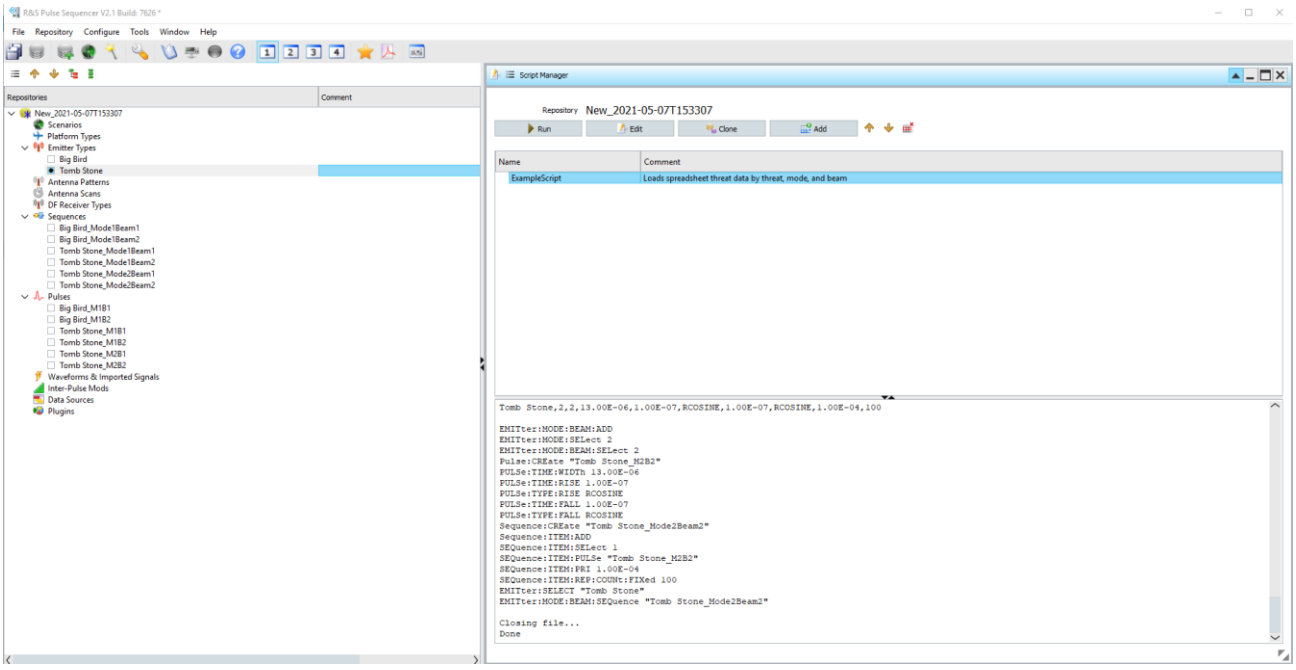


Figure 3 - Pulse Sequencer workspace after running the script in Figure 1.

3 SCPI Recorder

Since version 1.11, the R&S®Pulse Sequencer software has a SCPI recorder feature to capture UI interactions as SCPI sequences which can be exported to a Java script to make programming easy.

In the following, the operation of the SCPI recorder is explained.

Start by clicking the SCPI recorder icon in the upper taskbar.

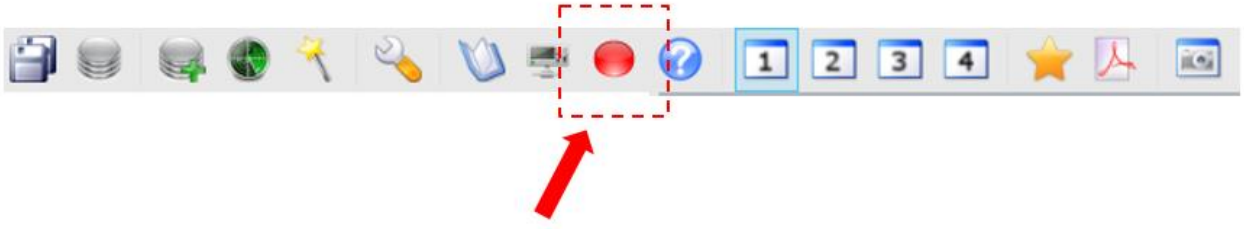


Figure 4 - Activate the SCPI Log

Add pulses, sequences, antenna patterns and scans, emitters, and scenarios. Examples of SCPI commands for most software features are given in Appendix A: SCPI for Emitter .

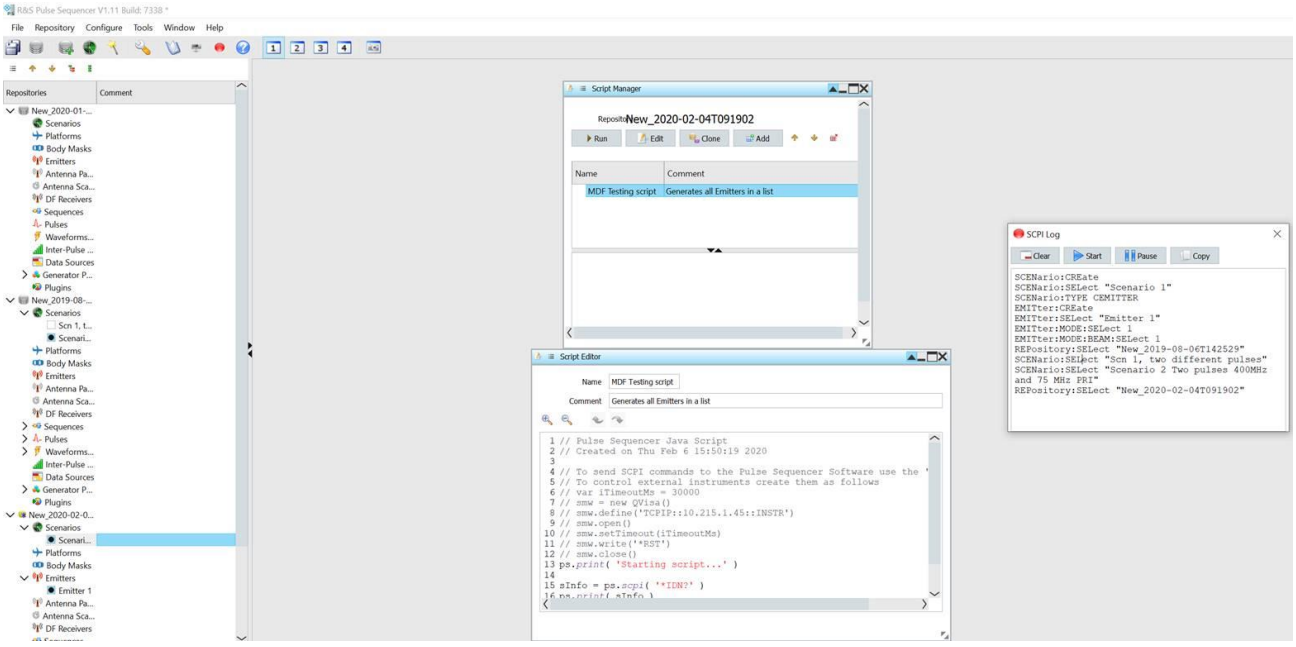


Figure 5 - With the SCPI log running, use features to capture SCPI

The SCPI command corresponding to each action is appended in the SCPI log.

4 Wrap the SCPI in Code

Use the built-in JavaScript editor and console to send SCPI commands to the R&S®Pulse Sequencer and the instrument. Add the SCPI command to the editor and wrap it in the 'ps.scpi()' method as shown in line 15 of Figure 6 below.

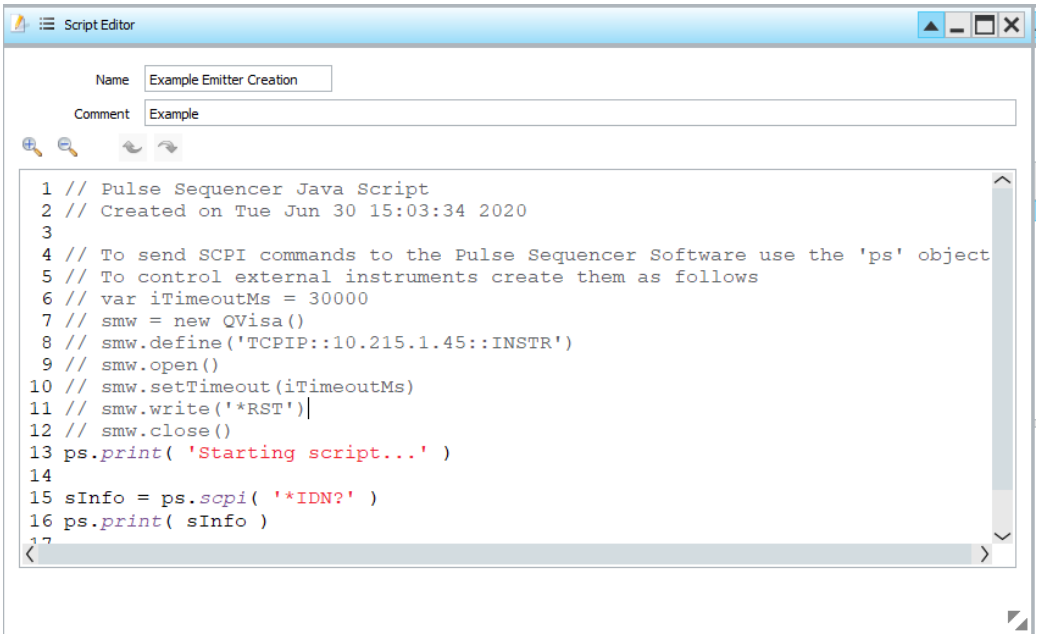


Figure 6 - Pulse Sequencer's native JavaScript editor

5 Literature

[1] Rohde & Schwarz, "Threat Simulation and Verification for Radar Warning Receiver Testing," Rohde & Schwarz.

6 Ordering Information

Designation	Type	Order No.
Pulse Sequencer Software	R&S®SMW-K300	1413.8805.02
Enhanced Pulse Sequencing	R&S®SMW-K301	1413.9776.02
Radar Platforms	R&S®SMW-K302	1413.8857.02
Moving Emitters and Receiver	R&S®SMW-K304	1413.8957.02
Multiple Emitters	R&S®SMW-K306	1413.9053.02
Direction Finding	R&S®SMW-K308	1414.1433.02
Import of 2D Maps	R&S®SMW-K309	1414.6706.02
Pulse on Pulse Simulation	R&S®SMW-K315	1414.6529.02

7 Appendix A: SCPI for Emitter Creation

The following examples can be used with the editor explained in chapter 4 or in an external script editor such as Matlab or Python to create the effects listed in their subheading.

Basic automated pulse creation

Pulses can be created as follows:

```
PULSe:CREate
PULSe:SELect "Pulse 1"
PULSe:NAME "Landbased_10us"
PULSe:TIME:RISE 1.E-07 // rise time
PULSe:TIME:FALL 1.E-07 // fall time
PULSe:TIME:WIDTh 1.E-05 //width
PULSe:TYPE:RISE RCOSINE //rise time shape
PULSe:TYPE:FALL RCOSINE
```

Basic automated PRI sequence creation

```
SEQuence:CREate //create a new sequence
SEQuence:SELect "Sequence 1" //select it by its generic name.
SEQuence:NAME "Search PRI" //rename it
SEQuence:ITEM:ADD //add a line to the sequence
SEQuence:ITEM:SELect 1 //select the line
SEQuence:ITEM:PULSe "Landbased_10us" //add a pulse
SEQuence:ITEM:REP:COUNt:FIXed 1000  //add repetitions
SEQuence:ITEM:PRI 1.E-03 //set the PRI
```

Basic antenna pattern creation

```
ANTenna:CREate
ANTenna:SELect "Antenna 1"
ANTenna:NAME "fan beam"
ANTenna:SELect "fan beam"
ANTenna:NAME "search beam"
```

Basic scan creation

```
SCAN:CREate //create a scan
SCAN:SELect "Antenna Scan 1" //select it by its generic name
SCAN:NAME "search scan” // rename it
SCAN:TYPE CIRCULAR //set the scan type to “Circular”
SCAN:CIRCular:RPM 2.E+01 //set the RPM of the scan
ANTenna:MODel:TYPE COSECANT //set the antenna model
ANTenna:MODel:COSecant:HPBW 1.E+01 //set the half power beamwidth
ANTenna:MODel:COSecant:T1 5.E+00
ANTenna:MODel:COSecant:T2 7.E+01
```

Basic emitter, mode, and beam creation

The following creates an emitter, names it, selects an antenna pattern and scan, PRI sequence (mode), and center frequency

```
EMITter:CREate
EMITter:SELect "Emitter 1"
EMITter:NAME "Search radar 1"
EMITter:SELect "Search radar 1"
EMITter:MODE:SELect 1
EMITter:MODE:NAME "Mode 1"
EMITter:MODE:ANTenna "search low"
```

```
EMITter:MODE:SCAN "search scan"

EMITter:MODE:ADD

EMITter:MODE:SELect 2

EMITter:MODE:NAME "Mode 2"

EMITter:MODE:ANTenna "search high"

EMITter:MODE:SCAN "search scan"


EMITter:MODE:BEAM:SELect 1

EMITter:MODE:BEAM:SEQuence "Search PRI low"


EMITter:MODE:BEAM:ADD

EMITter:MODE:BEAM:SELect 2

EMITter:MODE:BEAM:SEQuence "Search PRI high"

EMITter:FREQuency 3.E+09
```

Basic scenario creation, emitter positioning, and orientation

The following creates a “Localized Emitters” scenario that allows you to sequence through multiple emitters, modes, and beams.

```
SCENario:CREate

SCENario:SELect "Scenario 2"

SCENario:TYPE LOCALIZED //create a 'localized emitter' scenario

SCENario:NAME "Sequenced Emitters" //rename it

SCENario:CEMit:ADD //add an emitter from the list of emitters.  This was done
with a 'drag and drop' in the UI.

SCENario:CEMit:SELect 1

SCENario:CEMit:EMITter "Search radar 1"

SCENario:LOCalized:LOCation:EAST -2.88860625E+04  //position the emitter at 270
degrees at about 30,000 meters

EMITter:SELect "Landbased 2" //add another emitter to the scenario

SCENario:CEMit:ADD

SCENario:CEMit:SELect 2

SCENario:LOCalized:LOCation:EAST -2.491984179688E+04 //position the emitter with
the other one


//start sequencing the modes and beams of the emitter

EMITter:SELect "Landbased 2" //select

EMITter:MODE:BEAM:SELect 1 //select the first beam
```

```
//make the emitters point at the DUT
SCENario:CEMit:SELect 1 //select the first emitter
SCENario:LOCalized:DIRection:TRACk 1 //make it point at the DUT
SCENario:CEMit:SELect 2 //select the second emitter
SCENario:LOCalized:DIRection:TRACk 1 //make it point at the DUT
```

Emitter mode sequencing

```
//Query mode durations for each mode on each emitter
//keep a sum of all mode durations to know where the next mode belongs
SCENario:LOCalized:MCHG:CLEar
SCENario:LOCalized:MCHG:ADD
SCENario:CEMit:MCHG:SELect 1
SCENario:CEMit:EMITter:MODE:TRACkrec 1 //have the mode track the receiver

SCENario:LOCalized:MCHG:CLEar
SCENario:LOCalized:MCHG:ADD
SCENario:CEMit:EMITter:MODE:BEAM 2
SCENario:CEMit:MCHG:SELect 2
SCENario:CEMit:MCHG:STARt 1.014705882E+00
SCENario:CEMit:MCHG:STOP 1.553719009E+00
SCENario:CEMit:EMITter:MODE:TRACkrec 1

SCENario:LOCalized:MCHG:CLEar
SCENario:LOCalized:MCHG:ADD
SCENario:CEMit:EMITter:MODE 2
SCENario:CEMit:MCHG:STARt 1.553719009E+00
SCENario:CEMit:EMITter:MODE:TRACkrec 1

//emitter 2
SCENario:CEMit:SELect 2
SCENario:LOCalized:MCHG:CLEar
SCENario:LOCalized:MCHG:ADD
SCENario:CEMit:MCHG:SELect 1
SCENario:LOCalized:MCHG:CLEar
SCENario:LOCalized:MCHG:ADD
SCENario:CEMit:MCHG:STARt 1.877941176E+00
SCENario:CEMit:EMITter:MODE:TRACkrec 1
```

8 Appendix B: Creating Threats From User Data

8.1 ExampleScript.txt

```
// The following is a R&S® Pulse Sequencer Java Script that creates threat
// parameters from user data (ExampleScript.csv). Create a repository then run
// the script. To send SCPI commands to the Pulse Sequencer Software use the
// 'ps' object as shown. To control external instruments create them as follows
// To send SCPI commands to the Pulse Sequencer Software use the 'ps' object
// To control external instruments create them as follows

// var iTimeoutMs = 30000
// smw = new QVisa()
// smw.define('TCPIP::10.215.1.45::INSTR')
// smw.open()
// smw.setTimeout(iTimeoutMs)
// smw.write('*RST')
// smw.close()
// Template for Parsing CSV File into SCPI commands.
// NOTE: Order of elements matters.
// NAME TYPE # DESCRIPTION
// ------------------------------------------------
// THREAT text # Emitter name. e.g. Big Bird
// MODE 1 # Mode ID
// BEAM 1 # Beam ID
// WIDTH us # Pulse Width
// RISE us # Rising Edge
// RTYPE text # Type of Rising Edge
// FALL us # Falling Edge
// FTYPE text # Type of Falling Edge
// PRI 1 # Pulse Repetition Interval
// REPS 1 # Number of Repetitions
ps.print( 'Starting script...' )
sInfo = ps.scpi( '*IDN?' )
ps.print( sInfo )
```

```
// get the temporary path
dir = new QDir();
temppath = dir.tempPath();
ps.print(temppath);

// select file
fd = new QFileDialog();
filename = fd.getOpenFileName('Open a File', '', '', 0);
ps.print(filename);

var line;
var numOfColumns = 10; // Change number of columns according to the csv file

if( filename.length > 0 )
{
    file = new QFile();
    file.setFileName(filename);
    file.open(0x01); // 0x01 -- read mode
    ps.print('Reading csv File...');
    file.readLine(0); // Skip first row "Column Names"
    var prevEmitter = '';
    var prevMode = 0;
    var prevBeam = 0;
    while (!file.atEnd())
    {
        line = file.readLine(0);
        var arr = CSVToArray(line, ';');
        ps.print('\nLine:\n'+arr+'\n');
        // the translation of the parameters has be be performed by each
        // customer, since its propietary
        // Col 0 is the name
        //if emitter already exists, do not create a new one.
        if (prevEmitter != arr[0][0])
        {
            ps.scpi('EMITter:CREate "' +arr[0][0]+'"');
            ps.print('EMITter:CREate "' +arr[0][0]+'"');
        }
```

```
else
{
    // if mode exists but for different beam, create a new beam.
    if (prevMode == arr[0][1] && prevBeam != arr[0][2])
    {
        // create a new beam
        ps.scpi('EMITter:MODE:BEAM:ADD')
        ps.print('EMITter:MODE:BEAM:ADD')
    }
    // if new mode, create one.
    if (prevMode != arr[0][1])
    {
        // create new mode
        ps.scpi('EMITter:MODE:ADD')
        ps.print('EMITter:MODE:ADD')
    }
}
ps.scpi('EMITter:MODE:SELect ' +arr[0][1]);
ps.print('EMITter:MODE:SELect ' +arr[0][1]);
ps.scpi('EMITter:MODE:BEAM:SELect ' +arr[0][2]);
ps.print('EMITter:MODE:BEAM:SELect ' +arr[0][2]);

// create a pulse
ps.scpi('Pulse:CREate "' +arr[0][0] +'_M' +arr[0][1]+'B'+arr[0][2]+'"');
ps.print('Pulse:CREate "' +arr[0][0] +'_M' +arr[0][1]+'B'+arr[0][2]+'"');

ps.scpi('PULSe:TIME:WIDTh ' +arr[0][3]);
ps.print('PULSe:TIME:WIDTh ' +arr[0][3]);

ps.scpi('PULSe:TIME:RISE ' +arr[0][4]);
ps.print('PULSe:TIME:RISE ' +arr[0][4]);

ps.scpi('PULSe:TYPE:RISE ' +arr[0][5]);
ps.print('PULSe:TYPE:RISE ' +arr[0][5]);

ps.scpi('PULSe:TIME:FALL ' +arr[0][6]);
ps.print('PULSe:TIME:FALL ' +arr[0][6]);
```

```
        ps.scpi('PULSe:TYPE:FALL ' +arr[0][7]);
        ps.print('PULSe:TYPE:FALL ' +arr[0][7]);

        // put his pulse into a sequence
        ps.scpi('Sequence:CREate "' +arr[0][0] +'_Mode'
+arr[0][1]+'Beam'+arr[0][2]+'"'); //create a sequence for the threat, mode, and
beam

        ps.print('Sequence:CREate "' +arr[0][0] +'_Mode'
+arr[0][1]+'Beam'+arr[0][2]+'"');

        ps.scpi('Sequence:ITEM:ADD ' );
        ps.print('Sequence:ITEM:ADD ' );

        ps.scpi('SEQuence:ITEM:SELect 1 ' );
        ps.print('SEQuence:ITEM:SELect 1 ' );

        ps.scpi('SEQuence:ITEM:PULSe "' +arr[0][0] +'_M'
+arr[0][1]+'B'+arr[0][2]+'"');
        ps.print('SEQuence:ITEM:PULSe "' +arr[0][0] +'_M'
+arr[0][1]+'B'+arr[0][2]+'"');

        ps.scpi('SEQuence:ITEM:PRI ' +arr[0][8]);
        ps.print('SEQuence:ITEM:PRI ' +arr[0][8]);

        ps.scpi('SEQuence:ITEM:REP:COUNt:FIXed ' +arr[0][9]);
        ps.print('SEQuence:ITEM:REP:COUNt:FIXed ' +arr[0][9]);

        // assign the sequence to the selected emitter
        ps.scpi('EMITter:SELECT "' +arr[0][0]+'"');
        ps.print('EMITter:SELECT "' +arr[0][0]+'"');

        ps.scpi('EMITter:MODE:BEAM:SEQuence "' +arr[0][0]+'_Mode'
+arr[0][1]+'Beam'+arr[0][2]+'"');

        ps.print('EMITter:MODE:BEAM:SEQuence "' +arr[0][0]+'_Mode'
+arr[0][1]+'Beam'+arr[0][2]+'"');
        prevEmitter = arr[0][0];
```

```
        prevMode = arr[0][1];
        prevBeam = arr[0][2];
    }
    ps.print('\nClosing file...');
    file.close();
    ps.print('Done')
}

function CSVToArray( strData, strDelimiter ){
    // Check to see if the delimiter is defined. If not,
    // then default to comma.
    strDelimiter = (strDelimiter || ",");
    // Create a regular expression to parse the CSV values.
    var objPattern = new RegExp(
        (
            // Delimiters.
            "(\\" + strDelimiter + "|\\r?\\n|\\r|^)" +
            // Quoted fields.
            "(?:\"([^\"]*(?:\"\"[^\"]*)*)\"|" +
            // Standard fields.
            "([^\"\\" + strDelimiter + "\\r\\n]*))"
            ),
        "gi"
        );

    // Create an array to hold our data. Give the array
    // a default empty first row.
    var arrData = [[]];

    // Create an array to hold our individual pattern
    // matching groups.
    var arrMatches = null;

    // Keep looping over the regular expression matches
    // until we can no longer find a match.
    while (arrMatches = objPattern.exec( strData )){
```

```
        // Get the delimiter that was found.
        var strMatchedDelimiter = arrMatches[ 1 ];

        // Check to see if the given delimiter has a length
        // (is not the start of string) and if it matches
        // field delimiter. If id does not, then we know
        // that this delimiter is a row delimiter.
        if (strMatchedDelimiter.length && (strMatchedDelimiter != strDelimiter))
        {
            // Since we have reached a new row of data,
            // add an empty row to our data array.
            arrData.push( [] );
        }
        // Now that we have our delimiter out of the way,
        // let's check to see which kind of value we
        // captured (quoted or unquoted).
        if (arrMatches[ 2 ])
        {
            // We found a quoted value. When we capture
            // this value, unescape any double quotes.
            var strMatchedValue = arrMatches[ 2 ].replace(
                        new RegExp( "\"\"", "g" ),
                        "\""
                        );
        }
        else
        {
            // We found a non-quoted value.
            var strMatchedValue = arrMatches[ 3 ];
        }
        // Now that we have our value string, let's add
        // it to the data array.
        arrData[ arrData.length - 1 ].push( strMatchedValue );
    }
// Return the parsed data.
return( arrData );
}
```

8.2 ExampleScript.csv

```
Threat;Mode;Beam;PW;rise;type;fall;type;PRI;reps
Big Bird;1;1;1.00E-06;1.00E-07;RCOSINE;1.00E-07;RCOSINE;1.00E-04;100
Big Bird;1;2;2.00E-06;0;LINEAR;0;LINEAR;2.00E-04;50
Tomb Stone;1;1;11.00E-06;1.00E-07;RCOSINE;1.00E-07;RCOSINE;1.00E-04;100
Tomb Stone;1;2;12.00E-06;1.00E-07;RCOSINE;1.00E-07;RCOSINE;1.00E-04;100
Tomb Stone;2;1;13.00E-06;1.00E-07;RCOSINE;1.00E-07;RCOSINE;1.00E-04;100
Tomb Stone;2;2;13.00E-06;1.00E-07;RCOSINE;1.00E-07;RCOSINE;1.00E-04;100
```

Rohde & Schwarz

The Rohde & Schwarz electronics group offers innovative solutions in the following business fields: test and measurement, broadcast and media, secure communications, cybersecurity, monitoring and network testing. Founded more than 80 years ago, the independent company which is headquartered in Munich, Germany, has an extensive sales and service network with locations in more than 70 countries.

www.rohde-schwarz.com

Certified Quality Management
ISO 9001

Rohde & Schwarz training

www.training.rohde-schwarz.com

PAD-T-M: 3572.7186.00/01.00/EN

Rohde & Schwarz customer support

www.rohde-schwarz.com/support

1GP131 | Version 0E | 07.2021
Application Note | Automated Emitter Creation from User Data for Mission Data File Testing
Data without tolerance limits is not binding | Subject to change

www.rohde-schwarz.com