**RSA API Demo Apps and Utilities For RSA306, RSA306B, RSA500A and RSA600A**

**For RSA API v3.11.00xx release**

2017-09-14: Updated for API v3.11.00xx release

This document describes the RSA API Demo application programs, and some supporting utility programs for manipulating R3F and TIQ output files.

**Demo Applications**

The following RSA API apps demonstrate basic RSA API operations:

|  |  |
| --- | --- |
| **AudioOut.exe** | Play live Audio output to speaker, optionally store audio samples to file |
| **DevInfo.exe** | Display basic information about all USB-connected RSA devices |
| **DPXcapture.exe** | Acquire DPX Spectrum traces and bitmap from live input and store to file |
| **IQBLKcapture.exe** | Acquire IQ data blocks from live input and store to file |
| **IQcapture.exe** | Record live IQ signal sample data from device to .TIQ/.SIQ/.SIQH+SIQD/.CDIF/.CDIF+.DET file(s) |
| **IFcapture.exe** | Record live IF signal (AKA ADC) sample data from device to .R3F/.R3H+.R3A/.CDIF/.CDIF+.DET file(s) |
| **R3FtoIQ.exe** | Convert .R3F file to .TIQ or .SIQ/.SIQH+.SIQD file(s) |
| **SAcapture.exe** | Acquire Spectrum traces from device and store to file |
| **USBmon.exe** | Continuously monitor the USB data stream for dropped frames |

The Demo apps version numbers are **3.11.x** to associate them with the API release version. The app’s version number is displayed on the first line of console output, along with the API’s version number:



**64-bit vs 32-bit Apps**

Note the above app filenames are for the 64-bit Windows applications. 32-bit (AKA “x86”) versions are available, named the same as above but with “32” appended to the filename part. Example: *AudioOut32.exe*. Be sure to point to the appropriate API DLL library fileset when running the 64 or 32 bit applications.

**Installation**

The API Demo applications are C-language programs which use standard Windows C run-time libraries. The applications (\*.exe) are self-contained and do not requires installation. Each app program file can be copied to and run from any location, including a memory stick. It may be convenient to add the API applications folder location to the Windows path so they can be executed from any folder location but is not required.

The API demo applications **require** the RSA API software **v3.11.0047** version or laterbe installed on the computer. They should work with later API versions as it is intended that following API releases keep backward compatibility with existing apps.

It is recommended to add the API Library folder containing the API .dll files to the Windows **Path** Environment Variable. Note there are separate library filesets for 64-bit (x64) and 32-bit (x86) applications. The default RSA API library installation locations are:

64-bit: *C:\Tektronix\RSA\_API\lib\x64*

32-bit: *C:\Tektronix\RSA\_API\lib\x86*

You can add the file path permanently in the Windows Advanced System Properties dialog, or add it from the command line each time a new command window is opened, using this command:

* *set path = %path%;C:\Tektronix\RSA\_API\lib\x64*

There are example batch files for setting the system path to the RSA API libraries (see \Bats folder). There is also a batch file to set the path to the SignalVu-PC installation folders which also contain the API library files (64-bit only), so they can be used if the standalone API is not installed. This is not the recommended method to access the API files, but will work if needed. In that case, the SignalVu-PC version must be 3.11.0047 or later as well.

**Execution**

The app programs are Windows console programs and are best run from a command prompt to view the output messages. They can be run from batch files if more automated use is desired. In that case, the batch file can be executed from either a command prompt or by double-clicking in Windows Explorer.

Program execution is controlled by command line parameters. Most control parameters are optional and have default values. Generally, control parameters are text character strings of the form *<ID>=<value>*, with no spaces within the string (if necessary to include a space, enclose the entire string in quotes). Some controls are only *<ID>* switches without a value part. Each control string is separated by at least one space from other strings. An example command line invocation is shown here:

* IFcapture dev=0 rl=-10 cf=1.55e9 msec=10000 “fp=d:\data files” fn=ifdata

(Note the example of the *fp=..* string containing a space, using quotes as delimiters.)

To display a list of control parameters for each program, include “?” (without quotes) as one of the CL arguments:

* IFcapture ?

The usage listing also gives some example program usages for each application.

If no command line arguments are present, most programs by default search for attached RSA devices, display the list of devices found, and exit. They do not attempt to connect or do further operations. This provides a quick check of whether any devices are visible.

**Globally applicable command line controls:**

These command line controls apply to all demo applications, except *R3FtoIQ.exe.* See the command line usage printout for controls specific to each app. Note the *<ID>* portion of command strings is case-sensitive.

|  |  |
| --- | --- |
| *dev=<devid>* | Selects *<devid>* device (integer index value) for Connection. This argument is **required** to connect and use the device. If only one devices is physically attached, *dev=0* will connect to it. Other *devid* values allow selecting from multiple attached devices. Only one device can be connected per run. |
| *rst* | Applies a hardware reset to the selected device before connecting. If the device is failing to connect successfully, and the status LED is Red, use this control. |
| *align*  *alignp* | Runs alignment on the device after connecting.  “*alignp*” (prompted) version can be used with RSA306 which requires input signals to be turned off or disconnected during alignment. |
| *genb* | (RSA500A and RSA600A devices only)  Enables the internal GNSS Rx. |
| *gant* | (RSA500A and RSA600A devices only)  Enables the GNSS Antenna power output on the GNSS antenna connector. If not present, GNSS antenna power is disabled. Antenna power should be enabled if using the GNSS antenna provided with the RSA. |
| *gwait=<gwctl>* | (RSA500A and RSA600A devices only)  *gwctl=1*: Wait for GNSS navigation lock to be obtained.  *gwctl=2*: Wait for API RefTime system to be synchronized to GNSS timing, based on the Internal GNSS 1PPS signal and GPS time data. This provides high-precision data timestamps aligned to GNSS. The sync operation can be aborted while waiting for lock by pressing any key.  Note: *gwait=2* includes wait for GNSS nav lock (*gwait=1*) |
| *gsel=<gsatsel>* | (RSA500A and RSA600A devices only)  Selects a non-default GNSS Satellite System selection. If not present, the default GPS+Glonass system is used. See usage printout for allowed values. |
| *fref=<frctl>* | Select the device Freq Ref source.  Valid *frctl* values:  0: Internal Reference (default)  1: External Reference input. If no signal or an invalid signal is detected, the application exits.  2\*: GNSS Reference. Discipline Freq Ref based on internal GNSS timing pulse. The internal GNSS Rx must be enabled to use this selection. The application will wait until GNSS Nav lock and Freq Ref alignment are both achieved.  3\*: User Reference. Not implemented.  \* Valid only for RSA500A and RSA600A devices: |
| *extref* | **OBSOLETE**: User “fref=…” control above.  Device locks its oscillator/clock system to a valid external signal at the External Reference input. If no signal or an invalid signal is detected, the application exits. |
| *trig=<trigsel>* | Enables trigger event detection. *100*=External Trig source. *≤30*=IF Power Trigger Level source (value = power level in dBm). IQ Block captures wait for the trigger event. IF and IQ file storage wait to begin writing file(s) until the trigger event. The user can “force” a manual trigger by pressing the ‘t’ key while waiting for trigger. This can also be used as a manual “start” method by enabling triggering for a condition that will never occur, so that only manual trigger will cause the start. |
| *trigx=<trigxn>* | When triggering is enabled, selects rising, falling, or either edge transition as the trigger event. |
| *trigpct=<trigpct>* | When triggering is enabled, sets the location in the IQ Block record of the trigger event. Value in per-cent of the record size (0-100). Only applies to IQ block captures. |
| *rl=<refLevel>* | Sets the global device Reference Level (in dBm). |
| *cf=<ctrFreq>* | Sets the global device Center Frequency (in Hz). |
| *wait=<wctl>* | Wait *<wctl>* msecs after configuring, before proceeding with acquisition. A value of 0 means “infinite” wait. Pressing any key ends the wait, whether timed or infinite. |

**Examples**

Batch files demonstrating use of each program are included. Read the comments inside the batch files for more information. See:

Demo\_R3Fcapture.bat

Demo\_IQcapture.bat

Demo\_IQcapture\_benchmark.bat

Demo\_R3FtoIQ.bat

Demo\_AudioOut.bat

Demo\_SAcapture.bat

Demo\_DPXcapture.bat

Demo\_IQBLKcapture.bat

**Note on Command Line filename arguments with spaces:** CL arguments for input and output filenames can support spaces in the *<path+filename>* strings if entered properly. If there are any spaces in the *<path+filename>* string, then put quotes around the entire argument string, including the initial ID tag portion. For example to specify an input filename of **c:\My Folder\My File.r3f**, do this:

> r3ftoiq **“*fnin=c:\My Folder\My File.r3f*”** dest=2 dtyp=2 fn=d:\data\R3FtoIQ fnsfx=-2

**Note on disk speed for capturing “Live” signals with IF or IQ streaming operations:**

When storing IF (R3F, R3A or CDIF) or IQ (TIQ, SIQ, or CDIF) to files, the destination disk drive must support writing the data at the rate it is generated.

For IF capture, the data rate is always 224 MB/sec.

For IQ (TIQ /SIQ) capture, the data rate can be as high as 448 MB/sec (for 40 MHz BW and Single or Int32 data type), but decreases proportionally as bandwidth is reduced to 20 MHz, 10 MHz and lower BWs. When storing IQ at 40 MHz BW, it is recommended to use Int16 data type, which reduces the storage rate needed to 224 MB/sec. In fact, Int16 data type should be sufficient for all IQ bandwidth choices down to 5 MHz. Below 5 MHz, use of Int32 of Single types is recommended to avoid loss of dynamic range, unless only high-level (within 30 dB of Ref Level) are of interest.

The table below gives IQ streaming output data rates for various BW and data types.

**Output rates for IQ streaming:**

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | **IQ Output Data rate** | |
| **IQ BW** | **IQ Sample Rate** | **32b fixed or float** | **16b fixed** |
| 40 MHz | 56 M Sa/sec | 448 M B/sec | 224 M B/sec |
| 20 MHz | 28 M Sa/sec | 224 M B/sec | 112 M B/sec |
| 10 MHz | 14 M Sa/sec | 112 M B/sec | 56 M B/sec |
| 5 MHz | 7 M Sa/sec | 56 M B/sec | 28 M B/sec |

**RSA File Utilities**

The following utility programs are available to support RSA API file manipulation.

**r3fverify.exe**

*r3fverify.exe* reads and displays header information from an R3F file. It also verifies that the incrementing frame footer timestamps and frameIDs are valid throughout the file. A summary of footer violations is produced at the end of the run.

Alternatively, using command-line controls, *r3fverify* can export footer and/or data content of any part of the R3F to text files, or extract a subrange of the IF data to another R3F file. It can also be used to “patch” an R3F file with data discontinuity by inserting 0-fill to re-align the frame structure and timestamp/frame ID content.

This program is a “standalone” Windows console app and is best run from a command prompt. It can be run from any location without installation. It does not use the RSA API and can be run without the API installed.

To get usage:

*> r3fverify ?*

To verify R3F file content:

*> r3fverify <path+filename-of-R3F-file>.r3f <optional args>*

Note: Use quotes around the entire filename argument if there are any spaces in it.

Ex:

File: *c:\my data\rawfile.r3f*

Run: *> r3fverify "c:\my data\rawfile.r3f"*

When verifying large files, it is recommended to use the “fcap” control argument, which writes verification output to file instead of console screen.

To Export file footer or data content to text files:

* *R3fverify <filename> export=<ctl,start,num>*

The output file(s) will be automatically named *<filename>.<FTR|DAT.txt>*, with the added extension depending on the content selected.

To Extract file content to another R3F file:

* *R3fverify <filename> extract=<start,num>*

The output file will be automatically named *<filename>.EXTR-<start>-<end>.r3f*.

To Patch an R3F file:

* *R3fverify <filename> patch*

The “patched” output file will be automatically named *<filename>-PATCH.r3f*.

**r3ha2r3f.exe**

*r3ha2r3f.exe* recombines IF data saved in "raw" format file pairs (.*r3h*(header data) + .*r3a*(raw sample data without framing)) into a "formatted" .*r3f* file.

This program is a Windows console app and is run from a command prompt. It can be run from any location without installation. It does not use the RSA API and can be run without the API installed.

*r3ha2r3f.exe* copies the .*r3h* header information to the .*r3f* header location, making some minor adjustments to the data format section. It then copies the .*r3a* data to the .*r3f*, inserting frame footer blocks at the appropriate intervals. Footer timestamps and frame ID values are generated to give valid footer contents.

Note on footer timestamping:

* File version 1.1.0 and earlier (before SW release 3.10.xxxx):

Since the original footer timestamps are not recoverable, the initial footer timestamp value is set equal to the header time-reference timestamp value. The time-reference timestamp is usually set when the device Connect() is done, resulting in the file appearing to have been recorded starting at Connect time. If multiple raw file sets were recorded during a connect session, then when combined, they will all appear to have been recorded starting at the Connect time (or whenever the reference time was set).

* File version 1.2.0 and later (SW release v3.10.xxxx and after):

The actual start time of the file data is stored in the header (R3H), and is used to regenerate the correct timestamps in R3F footers.

Program input is the .*r3h* and .*r3a* files, which must have the same filename. Output is an .*r3f* with the same filename as input files.

To get usage:

*> r3ha2r3f ?*

*To convert R3H+R3A files to R3F file:*

*> r3ha2r3f <filename-of-R3H-file-without-extension>*

Note: Use quotes around the entire filename argument if there are any spaces in it.

Ex:

Have: *c:\my data\rawfile.r3h* and *c:\my data\rawfile.r3a*

Run: ***> r3ha2r3f "c:\my data\rawfile"***

Output: *c:\my data\rawfile.r3f*

**TIQutil.exe**

*TIQutil.exe* reads a TIQ file with a single DataSet record, and produces a new TIQ file with 1 or more DataSet records. This is useful for managing TIQs produced by RSA API IQ Streaming which consist of a (possibly huge) single Data Set. *TIQutil* can do the following:

1. Extract a subrange of data from TIQ file data and store it in another TIQ for more efficient transport or storage, or to reduce file size handled by post-processing application.
2. Extract all or part of TIQ file data and format into 1 or more Data Sets which allow multi-frame Replay operations in SignalVu-PC (or similar RSA application).

*TIQutil* is a Windows console app and is run from a command prompt. It can be run from any location without installation. It does not use the RSA API and can be run without the API installed. Note this app is built only for x64 platforms since it requires large buffers for copying data.

To get usage:

*> tiqutil ?*

To display file content summary (no output file written):

*> tiqutil <TIQ-filename>*

To extract subrange of TIQ data to another TIQ file (single DataSet output):

* *tiqutil <TIQ-filename> nds=1 lds=<numberOfSamplesOut> noff=<startOffset>*

To extract subrange of TIQ data to another TIQ file (multiple DataSet output):

* *tiqutil <TIQ-filename> nds=<numberOfDataSets> lds=<numberOfSamplesInDataSets> noff=<startOffset>*

Note: Use quotes around the entire filename argument if there are any spaces in it.

The output file will be automatically named *<TIQ-filename>.TIQUTIL.tiq*.