README - JANUS Tool Kit 3.0.1

Giovanni Zappa

2013-03-25

JANUS Tool-kit version 3.0.1 provides a Reference implementation of JANUS standard version 3 transmitter and a sample implementation of decoder. Latest version of the JANUS Reference implementation can be downloaded prior registration on http://www.januswiki.com on *File Galleries* > *Code repository* > *Most Recent Toolkit*.

It is coded in "C" and in Matlab. The sources of the "C" version are available in the file "janus-c-3.0.1.zip", has to be build using *CMake* and it only requires libraries *FFTW3*. Matlab code is available on the same site in the file "janus-m-3.0.1.zip".

On http://www.januswiki.com is possible to find documentation on JANUS standard.

"C" code installation

The "C" code implementation is composed of a library "libjanus.a" providing all the basic functionality, two sample command line implementation "*janus-tx*" and "*janus-rx*" respectively for generating the transmission waveform and decoding it, plus a plugin for the "reference implementation message 16" in the "*libplugin_016_00.so*".

If CMake finds a Matlab installation will be also generated "*janus_tx_mex*" and "*janus_rx_mex*" to be used with Matlab.

Compilation on Linux or other UNIX

The "C" code has been successfully compiled on different Linux version.

Once downloaded "janus-c-3.0.1.zip" form http://www.januswiki.com, unzip the file.
Use CMake for the configuration. CMake command line accepts the standard as well as custom variables.
Most useful standard variables are:

- `CMAKE_BUILD_TYPE` which can be `Release` or `Debug`.
- `CMAKE_INSTALL_PREFIX` which allow to install JANUS on a path different from the default `/usr/local`.

CMake Custom command-line options are:

- specify the Matlab installation directory if not in the default path
  eg: **-D**`MATLAB_DIR=/usr/local/matlab80`
- force using single precision floating point where possible
  -D`JANUS_REAL_SINGLE=1`
- use single precision floating point only in the FFTs operations **-D**`JANUS_FFTW_SINGLE=1`
- avoiding the use of function *alloca* (broken on some systems) **-D**`JANUS_WITHOUT_ALLOCA=1`
- to avoid compiling on 32 bits machine "position-independent code" (PIC) add **-D**`JANUS_NO_PIC=1`

A possible compiling procedure is:

```
unzip -v janus-c-3.0.1.zip
cd janus-c-3.0.1
mkdir build
cd build
cmake -DCMAKE_BUILD_TYPE=Release ..
make
```

and finally

```
make install
```

The last command might need superuser privileges if installed outside the user path.

JANUS executables need to find the pulgins describing the behaviour of the different JANUS "class id" and "application type", **it is strongly recommended to perform the installation of JANUS**.
Otherwise will be required to add the directory containing the plugins to the environment variable `LD_LIBRARY_PATH` or to system "ld.so.conf".
Those operation might be anyway needed on some broken Linux versions.

Compilation on Windows

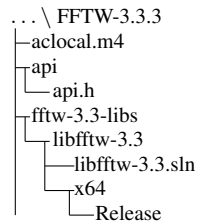
JANUS reference implementation has been installed on Windows using CMake 2.6 or newer and "Visual Studio Express 2012".

FFTW compilation under Windows

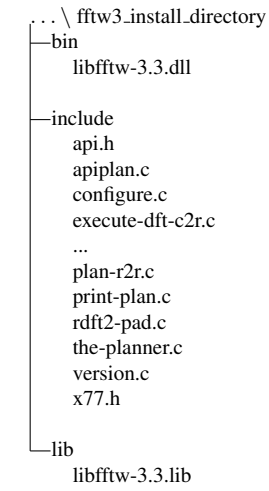
JANUS took-kit needs to link to the FFW3 library. There are some pre-compiled version or is possible to compile it.
The sources (fftw-3.3.3.tar.gz) can be downloaded from the website http://www.fftw.org/download.html.

The Visual Studio solution (fftw-3.3-libs-visual-studio-2010.zip) or the pre-compiled versions are available at http://www.fftw.org/install/windows.html

If decided to compile the FFTW3 libraries from sources, follow those steps:



- Unpack the sources `fftw-3.3.3.tar.gz` as in the figure.
- Inside the extracted folder `fftw-3.3.3` copy the content of file `fftw-3.3-libs-visual-studio-2010.zip`.
- Following the instructions contained into the Readme.txt, open with Visual studio the solution file `libfffftw-3.3.sln` in the directory `ffrw-3.3-libs`.
- Build the solution in the Release version.
- Choose an installation directory for the libfftw3 which for simplicity from now is called `fftw3_install_directory` and create inside the directories `bin`, `include` and `lib`.
- Copy inside `bin` the compiled file `fftw-3.3.3\ffrw-3.3-libs\libfftw-3.3\x64\Release\libfftw-3.3.dll`.
  In case of 32 bits system the structure will be `fftw-3.3.3\ffrw-3.3-libs\libfftw-3.3\Release\libfftw-3.3.dll`.
- Copy in `include` the contents of `fftw-3.3.3\api`
  Copy in `lib` the file from `fftw-3.3.3\ffrw-3.3-libs\libfftw-3.3\x64\Release\libfftw-3.3.lib`.
  After the installation the folder should present the following structure:



- To allow CMake and the operating system to find the FFTW3 libraries open `regedit` (just type "regedit" from search or execute of your start menu).
- Open the tree called `HKEY_CURRENT_USER` or `HKEY_CLASSES_ROOT` in case of a system wide installation which requires "administrator" privileges.
- Look for `Environment`.
- Create a new "String Value" and set Name to `FFTW_DIR` and Data to the full path of your `fftw3_install_directory`.
- Edit on regedit the entry `PATH` and prepend "`fftw3_install_directory\bin;`" to your existing PATH.

JANUS compilation

As well as for Linux the source file "janus-c-3.0.1.zip" can be downloaded form http://www.januswiki.com.

- Extract janus-c-3.0.1.zip.
- Run CMake-gui; with Browse Source find the folder where JANUS source is unpacked.
- Copy the source path on the following line (Where to build the binaries) and append \build.
- Press Configure, eventually confirm you want to create the build folder.
- Select "Visual Studio 11" (if possible Win64) or the appropriate project type for your compiler on the new appearing window and click on Finish.
- It is possible to change some of CMake internal variables from the gui.
  For example CMAKE_INSTALL_PREFIX allows to select the installation directory. For those variables refer to Compilation on Linux or other UNIX.
- Press Generate and close CMake.
- Run Visual Studio and open the solution file JANUS.sln inside the build directory specified to CMake.
- Select your Solution Configuration, Solution Platform and run Build Solution.
- To install JANUS tool-kit, select INSTALL in the solution explorer and with right mouse key build it.

Matlab code

Matlab code is stored in the "janus-m-3.0.1.zip". Unpack the zip file and move with Matlab on folder janus-m-3.0.1 or use addpath() command to be able to use JANUS functions.

Building the "C" source on a system with a Matlab installation (might be necessary to specify MATLAB_DIR=<path_to_matlab_directory>) will generate the files "*janus_tx_mex*" and "*janus_rx_mex*" which allow to run the "C" compiled code inside Matlab.

As usual the directory containing those files needs to be added with addpath() or be present in working directory.

Using JANUS Tool-Kit

JANUS tool kit interface use similar option whenever possible for the transmitter, receiver either in Matlab and in "C" version.

Typically "C" command-line options start with "--" and words are separated with "-", while on Matlab use "_" as word separator.
Eg: The "C" option --stream-driver on Matlab becomes 'stream_driver'

Command-line options

Following options control the JANUS modulation and are common between transmitter and receiver:

"C" option	Matlab option	Function	default
--pset-file	pset_file	Parameter Set File	
--pset-id	pset_id	Parameter Set Identifier	0
--pset-center-freq	pset_center_freq	Center Frequency (Hz)	0
--pset-bandwidth	pset_bandwidth	Bandwidth (Hz)	0
--chip-len-exp	chip_len_exp	Chip Length Dyadic Exponent	0
--sequence-32-chips	sequence_32_chips	Initial sequence of 32 chips (hex)	0x6BC4D788

- pset-file must point to a valid "CSV" file with standard central frequency and band.

- `pset-id` must select a valid entry of *pset-file*.

all the following options are not required:

- `pset-center-freq` override central frequency. Changing from default value may cause not only incompatibility with other devices but hardware damages on transmitters.
- `pset-bandwidth` override available bandwidth. As for previous option is strongly suggested to not use it.
- `chip-len-exp` force chip duration to multiplied by $2^{chip-len-exp}$, slowing transmission rate but probably increasing reliability of communications. Same value should be set on all the other devices.
- `sequence-32-chips` change the $0/1$ sequence transmitted in the first 32 chips acquisition signal. Different settings will cause incompatibility with other devices.

The following options allow the user to control the user interface and the waveform generation:

"C" option	**Matlab option**	**Function**	**default**
--verbose	verbose	Verbose level	0
--stream-driver	stream_driver	Stream Drv (nul/alsa/pulse/raw/wav/wmm)	wav
--stream-driver-args	stream_driver_args	Stream Driver Arguments	janus.wav
--stream-fs	stream_fs	Stream Sampling Frequency (Hz)	44100
--stream-format	stream_format	Stream Format	S16
--stream-channels	stream_channels	Stream Channels configuration	1
--stream-channel	stream_channel	Stream Active Channel	0
--stream-passband	stream_passband	Stream Passband signal	1
--config-file	config_file	Configuration file	

- `verbose` allows to get more internal informations, 0 is the minimum, 1 is consider normal verbosity and 2 more information including raw chips probability and plotting in Matlab.
- `stream-driver` select the device media for input/output stream (see the following media table).
- `stream-driver-args` media configuration (see the following media table).

Driver	**Typical option**	**Notes**	**Matlab**	**"C"**	**"MEX"**
alsa	`default` | `plughw:0,0`	ALSA Linux sound system		X	
fifo		Only useful if at library level			
mat	`janus.mat`	Save/Load Matlab file	X		
mem	`janus`	Use Matlab variable	X		
null		Fake dirver only for testing	X	X	X
pulse		PULSE sound system		X	
raw	`janus.raw`	Save/Load raw binary file	X	X	X
wav	`janus.wav`	Save/Load RIFF WAV file	X	X	X
wmm	`-1`	Windows Multimedia sound system		X	

- `stream-fs` stream sampling frequency, must be compatible with the selected media. Needs to satisfy Nyquist sampling theorem (stream-fs should be a least twice the maximum frequency).
- `stream-format` sample binary format representation (as in the following table):

Format	**Option value**
Signed 8 bit PCM	`S8`
Signed 10 bit PCM	`S10`
Signed 12 bit PCM	`S12`
Signed 14 bit PCM	`S14`
Signed 16 bit PCM	`S16`
Signed 24 bit PCM	`S24`
Signed 24 bit PCM in LSB of 32 Bit words	`S24_32`
Signed 32 bit PCM	`S32`
32 bit IEEE floating point [-1.0, 1.0]	`FLOAT`
64 bit IEEE floating point [-1.0, 1.0]	`DOUBLE`

- `stream-channels` number of channels the of the input/output stream.
- `stream-channel` which channel number should be used.
- `stream-passband` use passband signal or baseband
- `config-file` loads file with all input configurations.

Transmitter specific options are:

"C" option	Matlab option	Function	default
--pad	pad	Enable/Disable Padding of output	1
--wut	wut	Enable/Disable Wake Up Tones	0
--stream-amp	stream_amp	Stream Amplitude Factor (0.0 - 1.0]	0.95
--stream-mul	stream_mul	Stream samples multiple of given number	1
--packet-mobility	packet_mobility	Mobility	0
--packet-tx-rx	packet_tx_rx	Tx/Rx capability	1
--packet-forward	packet_forward	Forwarding capability	0
--packet-class-id	packet_class_id	Class User Identifier	16
--packet-app-type	packet_app_type	Application Type	0
--packet-reserv-time	packet_reserv_time	Reservation Time	0.0
--packet-repeat-int	packet_repeat_int	Repeat Interval	0.0
--packet-app-data	packet_app_data	Application Data	0x0000000
--packet-app-data-fields	packet_app_data_fields	Application Data Fields	
--packet-payload	packet_payload	Optional Payload	

- `pad` insert short silence at the beginning and end of signal.
- `wut` enable/disable transmission of wake-up tone. (It may create false detection with consequent loss of the following packet with sample decoder implementation).
- `stream-amp` stream amplification factor.
- `stream-mul` force generated signal to have multiple of the given integer number of samples (needed for some signal generators).
- `packet-mobility` sets mobility packet flag.
- `packet-tx-rx` sets packet flag indicating the capability of decoding JANUS signals.
- `packet-forward` sets packet flag indicating the capability of forwarding packets.
- `packet-class-id` sets class user identifier, selecting application or nation $[0 \sim 255]$. Togeither with the *application type* determine the meaning of *application data* field and payload.
- `packet-app-type` sets application type for the specific *packet-class-id* $[0 \sim 63]$.
- `packet-reserv-time` JANUS packet reserving channel for the requested number of seconds. This option is incompatible with *packet-repeat-int*.
- `packet-repeat-int` a JANUS packet will be retransmitted after the requested number of seconds. This option is incompatible with *packet-reserv-time*
- `packet-app-data` force application data field to be set with the provided bits in hexadecimal form.
- `packet-app-data-fields` comma separated list of couples $< field >=< value >$ interpreted by the plugin specified by *packet-class-id* and *application data* to fill the *app-data* field
- `packet-payload` string to encode into the following payload.

"C" option	Matlab option	Function	default
--doppler-correction	doppler_correction	Enabled/Disabled Doppler Correction	1
--doppler-max-speed	doppler_max_speed	Doppler Correction Maximum Speed [m/s]	5
--channel-spectrogram	channel_spectrogram	Enabled/Disabled channel spectrogram	1

- `doppler-correction` enable / disable Doppler correction.
- `doppler-max-speed` if Doppler correction enabled, limit speed estimation to the specified value expressed in $[m/s]$.
- `channel-spectrogram` enable / disable channel estimation around the JANUS 26 carrier frequencies and Media Access Control (MAC).

Examples

The following command line examples are provided as examples. Unfortunately different version of command line interpreters may need different quotations.

To be able to run the same examples on Windows and Linux move to the installation directory, your binary `CMAKE_INSTALL_PREFIX/bin` or default `/usr/local/bin` on Linux or to your `CMAKE_INSTALL_PREFIX/bin` if specified or the default `C:\Program Files\JANUS\bin`. Only to be able to run the following examples, on Windows is necessary to have the directory `\tmp`, while on Linux to add the path the binary directory: `export PATH="<CMAKE_INSTALL_PREFIX>/bin:$PATH"`.

"C" version

Basic examples

`> janus-tx`
To get the command line help of transmitter.

`> janus-rx`
To get the command line help of receiver.

`> janus-tx --pset-file ../share/janus/etc/parameter_sets.csv --pset-id 1`
Packet generation. This is the minimalistic command line, it relies on the default values. The result is a `'WAV'` file called `janus.wav` on the same directory of the executable. This command will fail in case the user has not write premission in installation path.

```
> janus-tx --pset-file ../share/janus/etc/parameter_sets.csv --pset-id 1
    --stream-driver-args /tmp/test_janus.wav
```
In this case the `'WAV'` file created is `/tmp/test_janus.wav` overcoming the permission problem of previous example.

```
> janus-rx --pset-file ../share/janus/etc/parameter_sets.csv --pset-id 1
    --stream-driver-args /tmp/test_janus.wav
```
To decode the previously created file `/tmp/test_janus.wav`.

Receiver commands

```
> janus-rx --pset-file ../share/janus/etc/parameter_sets.csv --pset-id 1
    --stream-driver-args /tmp/test_janus.wav --verbose 2
```
To decode the previously created file `/tmp/test_janus.wav` with increased verbosity, showing the single chip error probabilities.

```
> janus-rx --pset-file ../share/janus/etc/parameter_sets.csv --pset-id 1
    --stream-driver-args /tmp/test_janus.wav --doppler-correction 0
```
To decode the previously created file `/tmp/test_janus.wav` without Doppler correction.

```
> janus-rx --pset-file ../share/janus/etc/parameter_sets.csv --pset-id 1
    --stream-driver-args /tmp/test_janus.wav --verbose 1
    --doppler-max-speed 3
```

To decode the previously created file `/tmp/test_janus.wav` with Doppler correction estimated in the range $[-3 \sim 3]m/s$.

```
> janus-rx --pset-file ../share/janus/etc/parameter_sets.csv --pset-id 1
    --stream-driver-args /tmp/test_janus.wav --channel-spectrogram 0
```

To decode the previously created file `/tmp/test_janus.wav` without running MAC check.

Playing with different media

Connecting with a cable your audio output with audio line-in, after verifying input and output levels, is possible to transmit and decode real-time JANUS signals.

On Linux and other UNIX

```
> janus-tx --pset-file ../share/janus/etc/parameter_sets.csv --pset-id 1
    --stream-driver alsa --stream-driver-args default --stream-format S16
    --stream-fs 48000
```

To play the JANUS waveform using *alsa* sound though the `default` device using a sampling rate of 48KHz and sample format signed integer represented with 16 bits.

```
> janus-rx --pset-file ../share/janus/etc/parameter_sets.csv --pset-id 1
    --stream-driver alsa --stream-driver-args default --stream-format S32
    --stream-fs 48000
```

To decode the JANUS waveform using *alsa* sound though the `default` device using a sampling rate of 48KHz and sample format signed integer represented with 32 bits.

```
> janus-tx --pset-file ../share/janus/etc/parameter_sets.csv --pset-id 1
    --stream-driver pulse --stream-driver-args "" --stream-fs 48000
    --stream-format S16
```

To play the JANUS waveform using *pulse* sound though the `default` device using a sampling rate of 48KHz and sample format signed integer represented with 16 bits.

```
> janus-rx --pset-file ../share/janus/etc/parameter_sets.csv --pset-id 1
    --stream-driver pulse --stream-driver-args "" --stream-fs 48000
    --stream-format S32
```

To decode the JANUS waveform using *pulse* sound though the `default` device using a sampling rate of 48KHz and sample format signed integer represented with 32 bits.

On Windows

```
> janus-tx --pset-file ../share/janus/etc/parameter_sets.csv --pset-id 1
    --stream-driver wmm --stream-driver-args -1 --stream-fs 48000
    --stream-format S16
```

To play the JANUS waveform using *pulse* sound though the `default` device using a sampling rate of 48KHz and sample format signed integer represented with 16 bits.

```
> janus-rx --pset-file ../share/janus/etc/parameter_sets.csv --pset-id 1
    --stream-driver wmm --stream-driver-args -1 --stream-fs 48000
    --stream-format S32
```

To decode the JANUS waveform using *pulse* sound though the `default` device using a sampling rate of 48KHz and sample format signed integer represented with 32 bits.

Physical layer options

```
> janus-tx --pset-file ../share/janus/etc/parameter_sets.csv --pset-id 1
    --stream-driver-args /tmp/test_janus.wav --chip-len-exp 3
```

The generate waveform use the dyadic chip length option with exponent 3, chips length will be $2^3 = 8$ times longer.

```
> janus-tx --pset-file ../share/janus/etc/parameter_sets.csv --pset-id 1
    --stream-driver-args /tmp/test_janus.wav --stream-channels 2
    --stream-channel 0
```

Generates a stereo waveform modulating the signal only on first channel.

```
> janus-tx --pset-file ../share/janus/etc/parameter_sets.csv --pset-id 1
    --stream-driver-args /tmp/test_janus.wav --pad 0
```

Generates a waveform without padding at the beginning and the end.

```
> janus-tx --pset-file ../share/janus/etc/parameter_sets.csv --pset-id 1
    --stream-driver-args /tmp/test_janus.wav --stream-passband 0
```

Generates a stereo waveform modulated in baseband.

JANUS packet options

```
> janus-tx --pset-file ../share/janus/etc/parameter_sets.csv --pset-id 1
    --stream-driver-args /tmp/test_janus.wav --packet-mobility 1
    --packet-tx-rx 1
```

Sets the into the packet mobility and ability to receive JANUS signal flags.

On Linux

```
> janus-tx --pset-file ../share/janus/etc/parameter_sets.csv --pset-id 1
    --stream-driver-args /tmp/test_janus.wav --packet-class-id 16
    --packet-app-type 0 --packet-app-data-fields 'Station
    Identifier=1,Parameter Set Identifier=17'
```

Force generation of JANUS packet of *class id* 16 and *application type* 0 (those are already default values). Filling the application fields with *"Station Identifier"=1* and *"Parameter Set Identifier"=17*. It cause loading the plugin library file `libplugin_016_00.so` if loadable (either in a standard library directory or in folder contained in `LD_LIBRARY_PATH` or installed under the specified `CMAKE_INSTALL_PREFIX`). Plug-ins are not mandatory, are needed to interpret correctly the field name in *packet-app-data-fields* and especially for decoding to know the eventual payload length. The plugin name is identified by *packet class identifier* and *packet application type* in this case respectively 16 and 0.

```
> janus-tx --pset-file ../share/janus/etc/parameter_sets.csv --pset-id 1
    --stream-driver-args /tmp/test_janus.wav --packet-class-id 16
    --packet-app-type 0 --packet-app-data-fields 'Station
    Identifier=1,Parameter Set Identifier=17' --packet-payload 1234
```

Force generation of JANUS packet as in the previous example with payload text "1234"

On Windows

```
> janus-tx --pset-file ../share/janus/etc/parameter_sets.csv --pset-id 1
    --stream-driver-args /tmp/test_janus.wav --packet-class-id 16
    --packet-app-type 0 --packet-app-data-fields "Station
    Identifier=1,Parameter Set Identifier=17"
```

Force generation of JANUS packet of *class id* 16 and *application type* 0 (those are already default values). Filling the application fields with *"Station Identifier"=1* and *"Parameter Set Identifier"=17*. It cause loading the plugin library file `libplugin_016_00.dll` if loadable (either in a standard library directory or installed under the specified `CMAKE_INSTALL_PREFIX`). Plug-ins are not mandatory, are needed to interpret correctly the field name in *packet-app-data-fields* and especially for decoding to know the eventual payload length. The plugin name is identified by *packet class identifier* and *packet application type* in this case respectively 16 and 0.

```
> janus-tx --pset-file ../share/janus/etc/parameter_sets.csv --pset-id 1
    --stream-driver-args /tmp/test_janus.wav --packet-class-id 16
    --packet-app-type 0 --packet-app-data-fields "Station
    Identifier=1,Parameter Set Identifier=17" --packet-payload 1234
```

Force generation of JANUS packet as in the previous example with payload text "1234"

Again system independent

```
> janus-tx --pset-file ../share/janus/etc/parameter_sets.csv --pset-id 1
    --stream-driver-args /tmp/test_janus.wav --packet-class-id 16
    --packet-app-type 0 --packet-app-data 0x000040443 --packet-payload
    1234
```

As in the above example, where application field are directly passed to command line.

```
> janus-tx --pset-file ../share/janus/etc/parameter_sets.csv --pset-id 1
    --stream-driver-args /tmp/test_janus.wav --packet-reserv-time 1
```

JANUS packet request reservation time of 1 second.

```
> janus-tx --pset-file ../share/janus/etc/parameter_sets.csv --pset-id 1
    --stream-driver-args /tmp/test_janus.wav --packet-repeat-int 10
```

Announcing a retransmission of a JANUS packet in 10 seconds, especially useful for beacon applications.

Examples for Matlab version

Basic examples

As already mentioned between the "C" and Matlab version, option have same name and behaviour, differentiating only typographically.

To be able to run Matlab functions, move to the installed Matlab source directory or add the same to the path (`addpath`).

```
> [tx_pkt, tx_state] = simple_tx('parameter_sets.csv', 1, ...
    'stream_driver_args', '/tmp/test_janus.wav')
```

In this case the `'WAV'` file created is `/tmp/test_janus.wav`.

```
> [rx_pkt, rx_state] = simple_rx('parameter_sets.csv', 1, ...
    'stream_driver_args', '/tmp/test_janus.wav'), ...
    rx_payload_ascii = char(rx_pkt.payload)
```

To decode the previously created file `/tmp/test_janus.wav`.

Receiver commands

```
> [rx_pkt, rx_state] = simple_rx('parameter_sets.csv', 1, ...
    'stream_driver_args', '/tmp/test_janus.wav', 'verbose', 2), ...
    rx_payload_ascii = char(rx_pkt.payload)
```

To decode the previously created file `/tmp/test_janus.wav` with increased verbosity, showing the single chip error probabilities.

```
> [rx_pkt, rx_state] = simple_rx('parameter_sets.csv', 1, ...
  'stream_driver_args', '/tmp/test_janus.wav', 'verbose', 1, ...
  'doppler_correction', 0), ...
  rx_payload_ascii = char(rx_pkt.payload)
```
To decode the previously created file `/tmp/test_janus.wav` without Doppler correction.

```
> [rx_pkt, rx_state] = simple_rx('parameter_sets.csv', 1, ...
  'stream_driver_args', '/tmp/test_janus.wav', 'verbose', 1, ...
  'doppler_max_speed', 3), ...
  rx_payload_ascii = char(rx_pkt.payload)
```
To decode the previously created file `/tmp/test_janus.wav` with Doppler correction estimated in the range $[-3 \sim 3]m/s$.

Matlab variables and files

```
> [tx_pkt, tx_state] = simple_tx('parameter_sets.csv', 1, ...
  'stream_driver', 'mat', 'stream_driver_args', '/tmp/test_janus.mat')
```
Save the JANUS waveform into the Matlab file `/tmp/test_janus.mat`

```
> [rx_pkt, rx_state] = simple_rx('parameter_sets.csv', 1, ...
  'stream_driver', 'mat', 'stream_driver_args', '/tmp/test_janus.mat')
  rx_payload_ascii = char(rx_pkt.payload)
```
Decodes the file generate in previous example.

```
> global test_janus;
  [tx_pkt, tx_state] = simple_tx('parameter_sets.csv', 1, ...
  'stream_driver', 'mem', 'stream_driver_args', 'test_janus', ...
  'stream_fs', 48000)
```
Save the JANUS waveform into the Matlab file `/tmp/test_janus.mat`

```
> [rx_pkt, rx_state] = simple_rx('parameter_sets.csv', 1, ...
  'stream_driver', 'mem', 'stream_driver_args', 'test_janus', ...
  'stream_fs', 48000), ...
  rx_payload_ascii = char(rx_pkt.payload)
```
Decodes the file generate in previous example.

Physical layer options

```
> [tx_pkt, tx_state] = simple_tx('parameter_sets.csv', 1, ...
  'stream_driver_args', '/tmp/test_janus.wav', ...
  'chip_len_exp', 3)
```
The generate waveform use the dyadic chip length option with exponent 3, chips length will be $2^3 = 8$ times longer.

```
> [tx_pkt, tx_state] = simple_tx('parameter_sets.csv', 1, ...
  'stream_driver_args', '/tmp/test_janus.wav', ...
  'stream_channels', 2, 'stream_channel', 0)
```
Generates a stereo waveform modulating the signal only on first channel.

```
> [tx_pkt, tx_state] = simple_tx('parameter_sets.csv', 1, ...
  'stream_driver_args', '/tmp/test_janus.wav', ...
  'pad', 0)
```
Generates a waveform without padding at the beginning and the end.

```
> [tx_pkt, tx_state] = simple_tx('parameter_sets.csv', 1, ...
  'stream_driver_args', '/tmp/test_janus.wav', ...
  'stream_passband', 0)
```
Generates a stereo waveform modulated in baseband.

JANUS packet options

```
> [tx_pkt, tx_state] = simple_tx('parameter_sets.csv', 1, ...
  'stream_driver_args', '/tmp/test_janus.wav', ...
  'packet_mobility', 1, 'packet_tx_rx', 1)
```
Sets the into the packet mobility and ability to receive JANUS signal flags.

```
> [tx_pkt, tx_state] = simple_tx('parameter_sets.csv', 1, ...
  'stream_driver_args', '/tmp/test_janus.wav', ...
  'packet_class_id', 16, 'packet_app_type', 0, ...
  'packet_app_data_fields', ...
  'Station Identifier=1,Parameter Set Identifier=17')
```
Force generation of JANUS packet of *class id* 16 and *application type* 0 (those are already default values). Filling the application fields with *"Station Identifier"=1* and *"Parameter Set Identifier"=17*. It cause loading the plugin library file `libplugin_016_00.m` in directory `plugins`). Plug-ins are not mandatory, are needed to interpret correctly the field name in *packet-app-data-fields* and especially for decoding to know the eventual payload length.

```
> [tx_pkt, tx_state] = simple_tx('parameter_sets.csv', 1, ...
  'stream_driver_args', '/tmp/test_janus.wav', ...
  'packet_class_id', 16, 'packet_app_type', 0, ...
  'packet_app_data_fields', ...
  'Station Identifier=1,Parameter Set Identifier=17', ...
  'packet_payload', '1234')
```
Force generation of JANUS packet as in the previous example with payload text "1234"

```
> [tx_pkt, tx_state] = simple_tx('parameter_sets.csv', 1, ...
  'stream_driver_args', '/tmp/test_janus.wav', ...
  'packet_class_id', 16, 'packet_app_type', 0, ...
  'packet_app_data', '000040443', 'packet_payload', '1234')
```
As in the above example, where application field are directly passed to command line.

```
> [tx_pkt, tx_state] = simple_tx('parameter_sets.csv', 1, ...
  'stream_driver_args', '/tmp/test_janus.wav', ...
  'packet_reserv_time', 1)
```
JANUS packet request reservation time of 1 second.

```
> [tx_pkt, tx_state] = simple_tx('parameter_sets.csv', 1, ...
  'stream_driver_args', '/tmp/test_janus.wav', ...
  'packet_repeat_int', 10)
```
Announcing a retransmission of a JANUS packet in 10 seconds, especially useful for beacon applications.

Mex Matlab version

Mex files are compiled togeither with the "C" file if found a Matlab installation.

Options are the same of Matlab code, with the exception of Matlab files and variables.

The files are installed from the `CMAKE_INSTALL_PREFIX` in `share/janus/matalb/mex`. Run Matlab in the directory containing the MEX file or add that folder to Matlab path.

To run it:

```
> [tx_pkt, tx_state] = janus_tx_mex('parameter_sets.csv', 1, ...
  'stream_driver_args', '/tmp/test_janus.wav')
```
In this case the `'WAV'` file created is `/tmp/test_janus.wav`.

```
> [rx_pkt, rx_state] = janus_rx_mex('parameter_sets.csv', 1, ...
  'stream_driver_args', '/tmp/test_janus.wav'), ...
  rx_payload_ascii = char(rx_pkt.payload)
```
To decode the previously created file `/tmp/test_janus.wav`.