DopplerPSK Quick-Start Guide for v0.20

Program Description

DopplerPSK is an experimental program for transmitting Doppler-corrected PSK31 on satellite uplinks. It uses an orbital propagator to estimate the Doppler shift for your location so that it presents a constant frequency from the point of view of the satellite's transponder. It's primarily designed is for satellites that have a SSB uplink receiver and an FM downlink (which is free of Doppler shift at the baseband receiver due to the FM subcarrier).

DopplerPSK does not provide any PSK31 demodulation; you should use a separate program for that purpose. DopplerPSK does not provide any compensation for *downlink* Doppler (which you would need for fully linear transponders).

License

DopplerPSK is released on the Apache License 2.0. This is a permissive license to encourage better implementations. See the NOTICE and LICENSE files in the distribution.

Prerequisites:

- You will need the Java Runtime installed to launch the application. Java can be installed here: http://www.java.com/en/ Note that you do not need to have Java enabled in your browser.
- Windowing environment. DopplerPSK should run on any desktop OS that supports the java runtime (Mac OSX, Windows, Linux, etc.)
- 16-bit audio output device
- A way to key the transmitter (either VOX or with a manual switch—no PTT keying is provided by DopplerPSK.
- PSK31 demodulating software

Installing and Running the Program

Download the ZIP archive and decompress to an empty directory. This will create the directory structure necessary for the program to run. Before updating elements and other auxiliary files, it is best to try running the program to ensure that the system prerequisites are fulfilled.

The simplest way to run the software is to double-click DopplerPSK.jar. You can also run the software from the command line with

```
java -jar DopplerPSK.jar
```

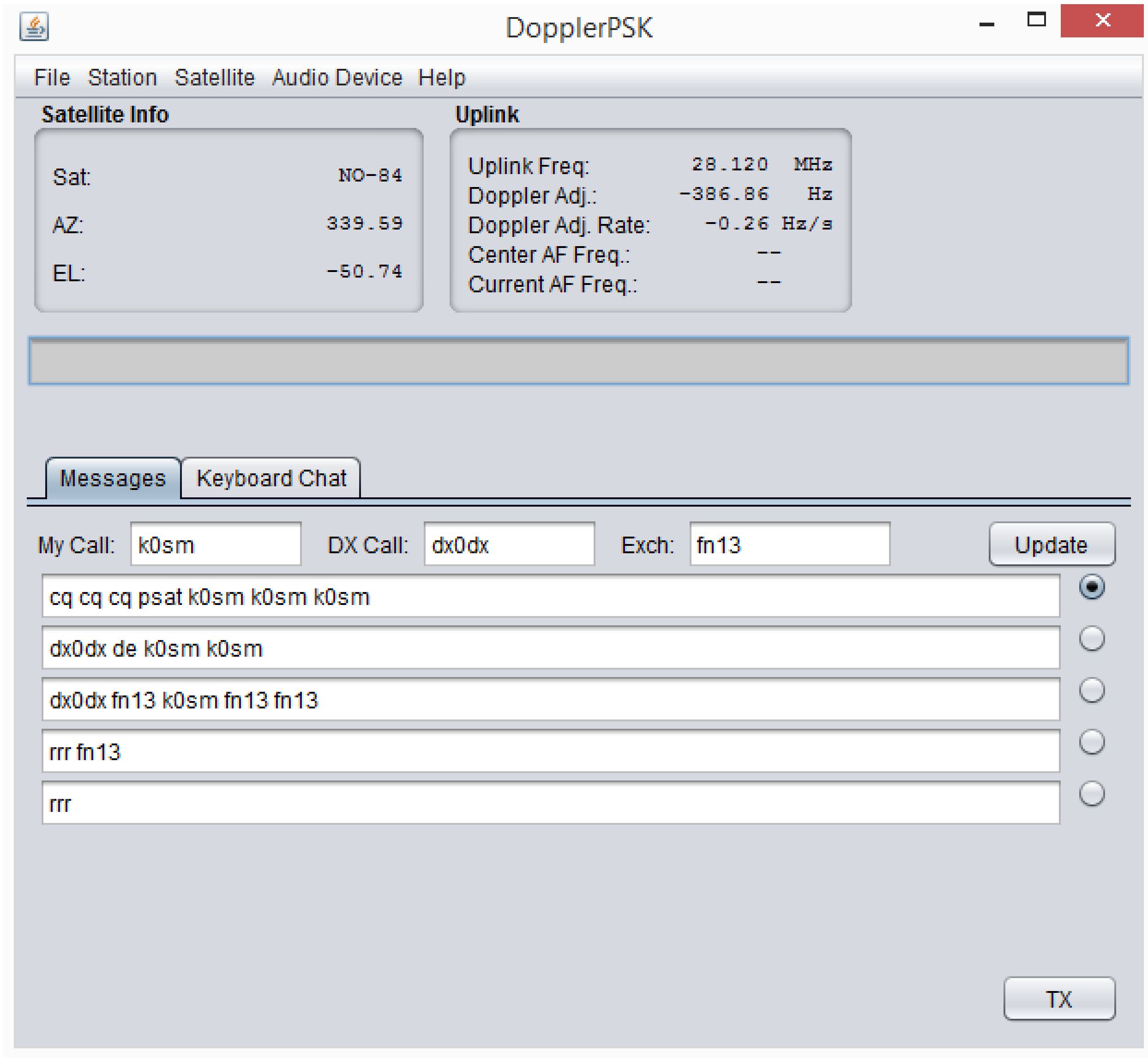
This option may be helpful for catching any error messages at startup.

If all goes well, you should see the program window tracking NO-84 with the default elements that came with the archive. Now would be a good time to set your station location from the *Station* menu. You should also be able to press the TX button to transmit a PSK31 signal from the default sound device.

The PSK31 signal is always generated at a pitch of 2000 Hz +- Doppler correction. In practice, this means a the PSK31 signal will be between 1300Hz and 2700Hz for a 28 MHz uplink on a LEO satellite like NO-84. The 2000Hz center value should pass through most transmitter IF filters, but it can be changed *in settings.ini*.

If all this works you should update your orbital elements. See the Auxiliary Files section of this guide.

User Interface



Sat: NO-84
AZ: 339.59
EL: -50.74

Uplink Freq: 28.120 MHz
Doppler Adj.: -386.86 Hz
Doppler Adj. Rate: -0.26 Hz/s
Center AF Freq.: --
Current AF Freq.: --

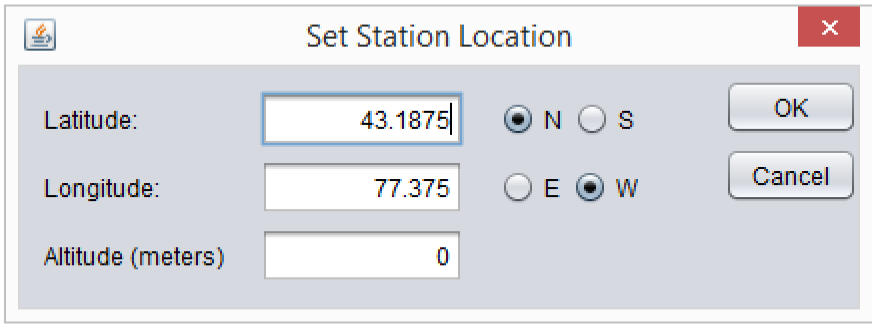
My Call: k0sm DX Call: dx0dx Exch: fn13

cq cq cq psat k0sm k0sm k0sm [●]
dx0dx de k0sm k0sm
dx0dx fn13 k0sm fn13 fn13
rrr fn13
rrr

Figure 1: DopplerPSK User Interface

File→Exit
Exits the program.

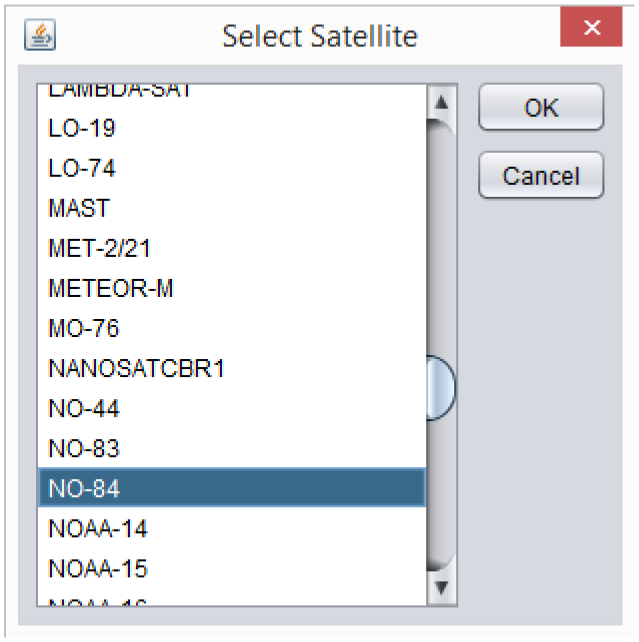
Station→Edit Station Information



Latitude: 43.1875 [✓ N] S
Longitude: 77.375 E [✓ W]
Altitude (meters) 0

Shows the interface for specifying the station location. This is needed to calculate the Doppler shift for uplink correction. Latitude and longitude is in degrees.

Satellite→Choose satellite



LAMBDA-SAT
LO-19
LO-74
MAST
MET-2/21
METEOR-M
MO-76
NANOSATCBR1
NO-44
NO-83
[✓ NO-84]
NOAA-14
NOAA-15

User interface for choosing a satellite to track. Press *OK* to select a new satellite or *Cancel* to exit without changes. The list of satellites is pulled from the headers of the two-line element sets in the tle subdirectory. See the Auxiliary Files section of this guide for editing and updating this list.

Help→About
Displays information about the program.

Satellite Info
Shows the satellite being tracked, and its current azimuth and elevation for your station location.

Uplink
This shows the transponder uplink frequency and calculated Doppler adjustments for the transmitter. Placing your cursor over the fields will display a description of the item.

Echo Window
This window echoes text as it is sent out of the transmitter in real time.

Messages Tab
Doppler PSK can provide preset messages that are sent in a loop. The messages are stored in text files in the `/macros` subdirectory. The macros are in text files in JSON format, and examples are provided. To create more macros just create a new file with the same format. The preset messages will be displayed on the message Tab in alphabetical order by filename.

TX Button
Turns on the transmitter and puts it into message mode. The selected message will be sent in a loop until the message is changed, the TX button is disengaged, or the transmitter is switched to keyboard-QSO mode.

My Call
Your callsign should go here. Pressing the update button will replace the MYCALL attribute in the message with this text.

Dx Call
The callsign of the station with whom you are communicating should be placed here. Pressing the update button will replace the DXCALL attribute in the message with this text.

Exch
The exchange you wish to send should be entered here. (Typically this will be your grid square for the exchange, but it could be an ARRL Field Day exchange or anything else). Pressing the update button will replace the EXCHANGE attribute in the message with this text.

Keyboard Tab

TX Text window
Typing in this window adds characters to the output buffer to be sent. This is for keyboard-to-keyboard QSOs instead of using the preset messages in a loop.

TX Button
Puts the transmitter in keyboard QSO mode. When the TX is on characters are read from the output buffer and send to the sound device. If the output buffer is empty, a PSK idle signal is sent. You will need a way to key the transmitter (either VOX or with a manual switch—no PTT keying is provided).

Clear Buffer Button

This button empties the transmit buffer and clears the TX window. It does not disable the transmitter. This is useful for cancelling a message.

Auxiliary Files

Updating Orbital Elements

DopplerPSK relies on a two-line element sets (TLEs) for tracking the location of the satellites. "Bare" two-line elements are the suggested format (such as the *nasabare.txt* available from AMSAT). Although DopplerPSK should handle files with headers (such as the *nasa.all* file available from AMSAT), it is not extensively tested. You should update elements frequently to maintain accurate Doppler correction.

DopplerPSK will attempt to read all files in the /tle subdirectory as TLE sets. If there are duplicate satellite names in the files, the last one read with be used (the files are read in lexical order). The names of the satellites will appear in the "Choose Satellite" menu of the user interface.

doppler.sqf

Frequency information for a particular satellite is stored in the *doppler.sqf* file, which is located in the program root directory. This format is identical to that used by SatPC32 for radio-controlled Doppler correction. This format contains the satellite name, uplink and downlink frequencies, and transponder information. A sample containing a few satellites has been provided. <u>Note that the satellite name in doppler.sqf must match the satellite name in the orbital elements exactly.</u> If *doppler.sqf* does not have an entry for the selected satellite no Doppler correction will be provided by the software.

UTC-TAI.history

This file contains the time correction between UTC and International Atomic Time (i.e., leap-second correction). It is required for the Orekit propagation library and must be in the program root directory. It has been updated for the leap second that occurred on 1 July 2015.

Creating Macros

The /macros subdirectory contains preset messages. You can create a new preset message by creating a .txt file with the following (JSON) format:

```
{
    "text": "cq cq cq psat MYCALL MYCALL MYCALL"
}
```

Lowercase text is more efficient due to the source-coding nature of the PSK31 symbols. That means that where possible, messages (including callsigns and exchanges that are entered in the UI) will have better performance if they are entered in lowercase.

There are three substitution strings that can placed in a message. These are meant to allow updating of messages during runtime using the "Update" button on the message panel:

MYCALL:	Your callsign
DXCALL:	The other station's callsign
EXCHANGE:	The exchange to be sent

The messages will be shown on the UI in alphabetical order by filename. The default filenames of "1.txt", "2.txt", etc. has been used in the distribution for this reason.

Settings.ini

Stores various program parameters.

Theory of Operation

A conventional analog BPSK transmitter mixes a baseband PSK31 signal with a carrier oscillator using mixer:

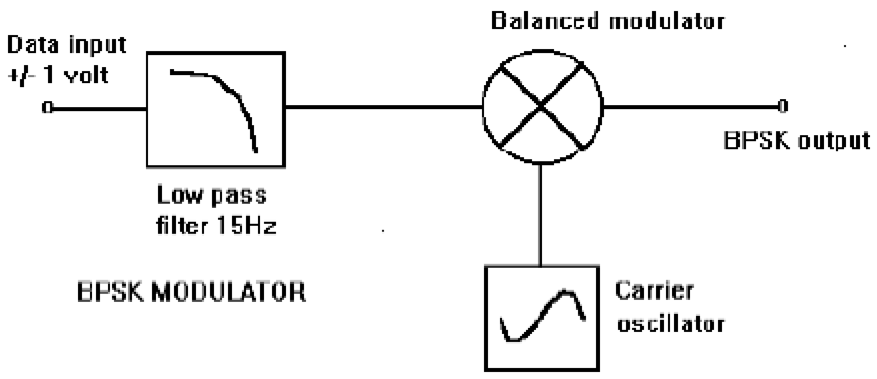


Figure 2: Analog BPSK transmitter (from Peter Martinez,G3PLX (1998). See http://det.bi.ehu.es/~jtpjatae/pdf/p31g3plx.pdf)

To provide Doppler correction, DopplerPSK turns the carrier oscillator into a VFO that tunes continuously to cancel the expected Doppler shift at any given moment. This must be done in a way that maintains frequency and phase continuity since any discontinuities will cause an instantaneous phase shift in the receiver resulting in bit errors. (Since radios often tune in discrete steps, radio control at the RF frequency often causes bit errors every time the radio updates).

Of particular importance for most PSK31 demodulators is that the frequency drift not be more than about 1.0 Hz/s, preferably a small fraction of that. Since the Doppler shift on even an 10m uplink of a LEO satellite is often more than this, some form of Doppler correction is required either on the uplink or in the AFC algorithm of the demodulator so that the demodulator can function. Correcting the uplink frequency has the advantage that every uplink signal presents a constant frequency at the satellite, ensuring that not crash into each other due to differing geometries among the stations using the transponder.

The algorithm is simply to create a piecewise linear approximation of the Doppler shift over time, and to do that we borrow a common technique for generating chirps.[1] It calculates the Doppler shift at the current time (*f_0*), and the Doppler shift at some point in the in the future *T* seconds later (*f_T*).[2] We subtract these two points to find the change in Doppler shift (or "chirp rate") *k* in Hz/s over the interval T:

$$k = \frac{f_0 - f_T}{T}$$

The order of subtraction in the numerator is so that we cancel the Doppler shift in our oscillator—we want a signal that goes up by the same amount that Doppler goes down.

Using this value, we wish to compute a continuous sweep that is linear in frequency for some amount of time. This will form one segment of a piecewise approximation of the frequency over the time period. Our frequency function is:

$$f_t = f_0 + kt$$

Ultimately we need to get a time domain function for our VFO. To get there, we note that phase φ of a time domain signal is the integral of its angular frequency, ω:

$$\omega_t = 2\pi f_t$$

We can now compute the integral substituting in the angular frequency:

$$\varphi_t = \varphi_0 + \int_0^t \omega_t dt$$

$$= \varphi_0 + \int_0^t 2\pi f_t dt$$

$$= \varphi_0 + 2\,\pi\,(\frac{k}{2}t^2 + f_0 t)$$

Finally, we get the time domain function x_t by taking the sine of the phase function:

$$x_t = \sin \varphi_t$$

Initially, φ_0, should start with a value of π/2 to ensure that the time domain signal starts with a value of zero instead of an instantaneous voltage jump which would cause a broadband "pop" in the spectrum. Additionally, f_0 should be set to whatever the absolute Doppler shift is at the start. From then on the value of k should be updated often enough that the Doppler rate is never more than a small fraction of a

[1] See "Chirp", https://en.wikipedia.org/wiki/Chirp. What follows is mostly a more verbose explanation of the derivation given there. I chose the linear chip because it is easy to understand and should be sufficient for our purposes. In theory one could use a higher-order approximation to obtain better tracking, but I suspect this would be lost in the error of the orbital elements themselves.
[2] Currently hard coded to 1 second in DopplerPSK 2.0.

Hz/s in error. When we update k we assign the current value of φ to φ_0 and recalculate the phase function for the next segment. This ensures that phase function (and by extension, x_t) remains continuous.

Since we are tracking the slope of the Doppler curve rather than the absolute frequency of the Doppler curve some error between the transmitted carrier frequency and the calculated absolute Doppler shift may accumulate over time. This numerical error shouldn't matter much in practice as it will likely be dwarfed by other system errors (errors in the orbital elements, LO drift of the transmitter and spacecraft receiver, etc.). All things equal it would manifest itself in a very slow frequency drift of the carrier in the passband.

Spacecraft position in DopplerPSK is calculated using the TLE propagation engine in the Orekit library. It uses the WGS84 one-axis ellipsoid earth model for geodetic coordinates.