The West Australian
VHF Group Bulletin
May 2003

Calendar

May	12	VHF Microwave Net
	19	Committee Meeting
	26	General Meeting
Jun	9	VHF Microwave Net
	16	Committee Meeting
	23	General Meeting
Jul	14	VHF Microwave Net
	21	Committee Meeting
	28	General Meeting
Aug	11	VHF Microwave Net
	18	Committee Meeting
	25	General Meeting
Sep	8	VHF Microwave Net
	15	Committee Meeting
	22	General Meeting
Oct	13	VHF Microwave Net
	20	Committee Meeting
	27	General Meeting

Committee

President	Alan	VK6ZWZ
Secretary	Don	VK6HK
Vice President	Terry	VK6ZLT
Treasurer	Cec	VK6AO
Activities		
Materials		
Publicity		
Librarian	Al	VK6ZAY
Museum Rep	Tom	VK6ZAF
Bulletin Editor	Ben	VK6TLA
Councillor	Luigi	VK6YEH
Councillor	Wally	VK6KZ
Councillor	Terry	VK6TRG

The official newsletter for the West Australian VHF Group (Inc), PO Box 189 Applecross. Email for the editor can be sent to vk6tla@amsat.org.

Editors Notes

Ben Rampling, VK6TLA

Welcome again to the Bulletin. Hopefully as this bulletin reaches your letter box you are tucked up cosy and warm in your shack, enjoying the spoils of the May junk sale, and not making too many ventures out to repair damaged antennas.

Terry VK6ZLT has passed on a link to a paper of interest entitled "The work of Jagadis Chandra Bose: 100 Years of MM-Wave Research". The full paper, with pictures, is available at http://www.tuc.nrao.edu/~demerson/bose/bose.html. A couple of excerpts from the paper follow:

"Just one hundred years ago, J.C. Bose described to the Royal Institution in London his research carried out in Calcutta at millimeter wavelengths. He used waveguides, horn antennas, dielectric lenses, various polarizers and even semiconductors at frequencies as high as 60 GHz; much of his original equipment is still in existence, now at the Bose Institute in Calcutta. Some concepts from his original 1897 papers have been incorporated into a new 1.3-mm multi-beam receiver now in use on the NRAO 12 Meter Telescope."

"In 1895 Bose gave his first public demonstration of electromagnetic waves, using them to ring a bell remotely and to explode some gunpowder. In 1896 the Daily Chronicle of England reported: `The inventor (J.C. Bose) has transmitted signals to a distance of nearly a mile and herein lies the first and obvious and exceedingly valuable application of this new theoretical marvel.´ Popov in Russia was doing similar experiments, but had written in December 1895 that he was still entertaining the hope of remote signalling with radio waves. The first successful wireless signalling experiment by Marconi on Salisbury Plain in England was not until May 1897. The 1895 public demonstration by Bose in Calcutta predates all these experiments."

While on the topic of interesting Internet sites, I recently found a page developed by NV3Z and NA3T to allow the generation of azimuthal equidistant projection maps. By going to http://www.wm7d.net/azproj.shtml and filling out the form, you can generate a map for your location with various VHF oriented aids such as beacons, grid squares, TV stations and more. You can then save and print the map in various bitmap and vector formats, for high quality printing.

The group is currently undertaking licencing of, with the intent to later construct, a 2.4 GHz beacon for wireless hill. The beacon could provide a useful reference signal for testing and calibrating 2.4 GHz equipment, as well as a beacon for propagation studies. The anticipated power level is 10 watts, transmitting in to an omnidirectional antenna. Installation of the VK6RST Mt Barker beacon has been postponed until August for what will be, with a bit of luck, improved weather.

In letters to the editor, we received a letter from Craig Hayhow VK6JJJ, a communications engineer currently located in Karratha:

Hi Terry, I just got your details off the WA VHF Group web site as a contact and I am just writing to you to let you know (and the other members) that I have a 144.1 MHz station set-up in Karratha beaming approximately 3Kw EIRP towards Perth. I am looking for possibilities of some tropospheric activity between Karratha and Perth, Karratha and Exmouth, Karratha and

Broom, Karratha and the eastern states and even Karratha and South Africa. If any body notices any opening between our locations or would like to organise some tests, I can be contacted to try for a contact. At the moment I am active on all modes except WSJ FSK441, but I hope to be experimenting with that mode soon also.

I know this would be a difficult path (over land) but I am willing to persist. I get a lot of near shore ducting up here and can work Exmouth and Broome without too much trouble, but there are very few hams between Perth and Darwin to experiment with. I also get interference from Indonesia at times, but have not tried any serious work in that direction. I don´t have E-mail at home, so if there was an opening and you wanted to contact me urgently, then a phone call is the way to go.

Shortly after the above letter was received, Craig wrote again letting us know that he was now on WSJ FSK441 as well, and was looking for contacts:

I'm just writing to you to see if any of your members would be interested in organising a WSJT FSK441 meteor scatter sked between Perth and my QTH in Karratha on 144.230MHz. The distance is perfect and it would be nice to give those Eastern states blokes a run for their money.

Regards,

Craig Hayhow VK6JJJ
Karratha
(08) 9185 1976
0438 897 882
craig.hayhow@woodside.com.au

Reusing WXtrack

Ben Rampling, VK6TLA

I have recently been experimenting with a number of different satellite tracking packages, from the collection of programs commonly used for AMSAT operations. All of the packages tried were found to be perfect for finding the position of a satellite at a specified time, but did not have good support for building readable pass lists for the next few weeks, or support to refine the lists by filtering out unusable passes.

The closest program to fit the bill is WXtrack, developed by David J Taylor. As the name suggests the software was developed with tracking weather satellites in mind, and will even render a preview of what a push-broom image sensor in a weather satellite will record and transmit on a given pass. It also has support for tracker hardware, as demonstrated at the April 2003 meeting, and unlike other programs is free for non-profit use and is highly functional as soon as you download it. Of course if you do use the program extensively, registration should be encouraged and is rewarded by additional advanced features and support.

Not content with the pass listings—shown in a staggered format rather than in columns and barely readable—from WXtrack, I went looking for other ways to get pass lists. I discovered that

the tracking algorithms used in WXtrack are available as a separate “software component” that can be downloaded and used in spreadsheets, macros and your own programs. The component allows you to compute satellite pass details without reference to any of the complex algorithms it contains and with out having to know about details such as the format of orbital elements.

The component contains a number of variables and functions that can be set and called. The most important variables that need to be initialised are **ObsLat**, **ObsLon**, **ObsHeight** and **KeplerPath**. The **KeplerPath** is the directory that contains the NORAD orbital elements, and if you have already have a satellite tracking program installed you can just specify the directory your satellite program uses to store elements. Once these four variables are set to sensible values, the component is ready to be used.

When reading or writing a latitude or longitude from the component, you do so in decimal degrees. Conversion to and from degrees, minutes and seconds is left up to the user. The elevation of the user is given in metres, and distances and speeds of satellites are returned in kilometres or kilometres per hour. Being an all metric bit of software, it should be safe to use on Mars exploration missions.

Once initialised you can either use the **PassListForPeriod** or **IsVisible** functions to either scan for satellite passes, or in the case of **IsVisible,** find the location, velocity and other details of a satellite at a given time. **PassListForPeriod** requires the name of a satellite, and the start date and a number of days to check for passes. You can also set the minimum time and minimum highest elevation for passes, and the function will eliminate any short or low passes. The component returns the passes in a text format for later processing. Each pass line includes the date and time, north or south direction of the pass, azimuth, maximum elevation, time of pass and the satellite name. For example, a handful of passes for UO-14 when printed look similar to:

20030421012042 S 97 15 686 UO-14,
20030421115722 N 127 29 787 UO-14,
20030421133650 N 103 27 770 UO-14,
20030421231013 S 128 26 772 UO-14,
20030422004914 S 104 31 804 UO-14

In the event that more details of the pass are required, perhaps to check the squint angles or to verify that the satellite is in sun light for the duration of the pass, the **IsVisible** function can be used.

IsVisible only requires the name, time and date (in Julian time and date) and the elevation required for a satellite to be “visible”. It then returns true or false if the satellite is visible at that time, and also returns details of the satellite location in the sky and relative to the earth. If after calling **IsVisible** you require the location of the satellite in cartesian coordinates, **SatState** can be called to return the X, Y and Z position and velocity vectors. A summary of the most useful functions and variables is shown in figure 1. The component is fairly simple, and the figure shows 90% of the functionality provided.

At first I wanted to use a computer language that can be found on almost any computer running Microsoft Windows, however I soon discovered that a “feature” of these scripting languages prevented the full use of the component. I settled on using another language, **Python**.

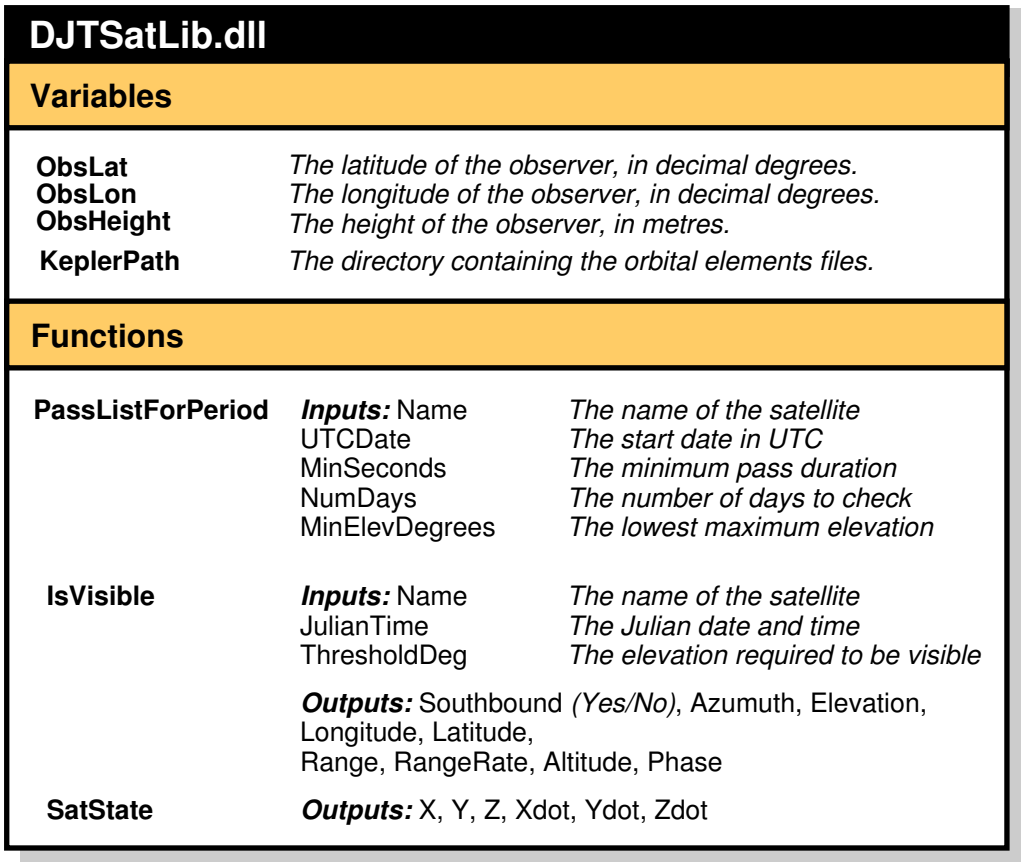

Figure 1. The contents of the SGP4 "component".

Python is very well suited for both mathematics and for beginner programmers. It also makes a fine calculator. One of the features that makes Python so pleasant to use is the way that almost all variables can have the same operators, functions and keywords applied to them. For example if *a* and *b* are Python variables containing vectors, you can add them in Python with the expression *a* + *b*. The same applies if the two variables were matrices, complex numbers or any other type where the concept of addition is valid. The other plus for numerical work is that it comes with a collection of good variable types built in, including complex numbers and large integers. Newer versions even come complete with a set type, an unusual primitive type in a general purpose language. Some examples follow, and if you download a copy of Python these can be typed directly in to the prompt:

```
>>> def vout(r1, r2):
...     # This is a comment, for a function that returns the open
...     # circuit ratio of voltage from a voltage divider.
...     return r2 / (r1 + r2)
...

>>> vout(220.0, 1000.0)
0.81967213114754101
```

```
>>> # This function can also be used with complex impedances with
>>> # out modification.
>>> vout(220.0 + 800.0j, 1000.0 - 300.0j)
(0.6155085135757018-0.49815922687528763j)

>>> 2 ** 1024
17976931348623159077293051907890247336179769789
42306572734300811577326758055009631327084773224
07536021120113879871393357658789768814416622492
84743063947412437776789342486548527630221960124
60941194530829520850057688381506823424628814739
13110540827237163350510684586298239947245938479
71630483535632962422413721 6L
```

Using the component in Python is almost as easy as if it were a built in Python component. Other languages, like your spreadsheet scripting language will have similar procedures for using these components. The following script is an example of loading the component and querying it for satellite location data:

```
from win32com.client import Dispatch
import pywintypes, string, time
from vector import *

sats = Dispatch("DJTSatLib.Satellites")

sats.KeplerPath = "E:Program FilesWXTrack"

sats.ObsLat = -31.8
sats.ObsLon = 115.25
sats.ObsHeight = 200

datetime = pywintypes.Time(time.time() - 8 * 3600)
juliantime = sats.DateTimeToJT(datetime)

visibledata = sats.IsVisible("AO-40", juliantime, 0.0)

visible, southbound, az, el, lon, lat, range, rangerate, alt, phase = visibledata
```

With a bit of extra work it is possible to calculate details not provided by the component such as squint angles, doppler correction, MA number and numerous others. The component also has support for calculating the location of the sun and moon, which when used with the satellite location—which is provided in the same coordinate system—can be used to determine when the satellite is in sun or in shadow.

The component and some documentation convering the installation and other functions in the component are available for download from http://www.david-taylor.pwp.blueyonder.co.uk/. In at least one case, the end result of tinkering with this component resulted in an entire satellite tracking program being written! The Satscape package—although a little too developed in the area of visual effects for my liking—has been built on top of the WXtrack core.