



DFS Radar Pulse Generation

Solutions for Wired and Wireless Testing

OVERVIEW:

This guide provides the instructions for using the DFS Radar Pulse Generation script to produce FCC, ETSI and Japan radar pulses.

SYSTEM REQUIREMENTS:

To produce radar pulses, the following is required:

- A PC, running Windows XP Service Pack 3 or later, or Linux
- Python 2.4.3
- One of the following VeriWave WaveBlades:
 - WBW1101
 - WBW1101P
 - WBW1104
 - WBW1104N

CAPABILITIES

Using this scripting interface, the radar pulses listed below can be generated. No equipment other than a single port of one of the WaveBlades listed above is required. The available pulse types are:

FCC

| Type | Pulse Width (us) | PRI (us) | # of Pulses | Bursts | Comments |
|------|------------------|-------------|-------------|---------|-------------------------------------|
| 1 | 1 | 1428 | 18 | 1 | |
| 2 | 1 – 5 | 150 – 230 | 23 – 29 | 1 | |
| 3 | 6 – 10 | 200 – 500 | 16 – 18 | 1 | |
| 4 | 11 – 20 | 200 – 500 | 12 – 16 | 1 | |
| 5 | 50 – 100 | 1000 – 2000 | 1 – 3 | 80 – 20 | Chirp BW is 5 – 20 MHz |
| 6 | 1 | 333 | 9 | 100 | Freq hopping range: 5250 – 5724 MHz |
| 7 | 1 | 518 – 3066 | Note 1 | 1 | |

Note 1. FCC type 7 is the weather radar now under proposal. It consists of 30 trials with PRI in range 518-3066 with each trial selecting a unique PRI. The first 14 trials are from list 518, 538, 558, 578, 598, 618, 638, 658, 678, 698, 718, 738, 758, 778, 798, 818, 838, 858, 878, 898, 918, 938, 3066. The number of pulses is calculated based on PRI selected and is defined by the following equation:

$$\text{NumPulses} = \text{ceil}((1/360) * (19E6/\text{PRI}(\text{us})))$$

ETSI

| Type | Pulse Width (us) | PRI (us) | # of Pulses | Bursts | Comments |
|------|------------------|-------------|-------------|--------|-------------------------|
| 0 | 1 | 1428 | 18 | 1 | |
| 1 | 0.8 – 5 | 1000 – 5000 | 10 | 1 | |
| 2 | 0.8 – 5 | 625 – 5000 | 15 | 1 | |
| 3 | 0.8 – 5 | 250 – 434 | 25 | 1 | |
| 4 | 20 – 30 | 250 – 5000 | 25 | 1 | Chirp BW is 5 – 20 MHz |
| 5 | 0.8 – 2 | 2500 – 3333 | 10 | 1 | Interleaved 2 – 3 PRI's |
| 6 | 0.8 – 2 | 833 – 2500 | 10 | 1 | |

Japan

| Type | Pulse Width (us) | PRI (us) | # of Pulses | Bursts | Comments |
|------|------------------|-------------|-------------|--------|---------------------------------|
| 1 | 1 | 1428 | 18 | 1 | W53 |
| 2 | 2.5 | 3846 | 18 | 1 | W53 |
| 3 | 0.5 | 1288 | 18 | 1 | W56 |
| 4 | 1 | 1428 | 18 | 1 | W56 |
| 5 | 2 | 4000 | 18 | 1 | W56 |
| 6 | 1 – 5 | 150 – 230 | 23 – 29 | 1 | W56 |
| 7 | 6 – 10 | 200 – 500 | 16 – 18 | 1 | W56 |
| 8 | 11 – 20 | 200 – 500 | 12 – 16 | 1 | W56 |
| 9 | 50 – 100 | 1000 – 2000 | 1 – 3 | 8 – 20 | W56 Chirp BW 5 – 10 MHz |
| 10 | 1 | 333 | 9 | 100 | W56 freq hopping between bursts |

COMMAND LINE INTERFACE

There are two files that need to be put into your executing directory: test_radar.py and radar.py, and duty50_radar_config.py.

To execute the script, the syntax¹ is:

- Python test_radar.py (--option)

| | |
|------------------------|---|
| -h or --help | display the command summary |
| -chassis=CHASSIS | CHASSIS is the name or IP address of the chassis |
| -slot=SLOT | SLOT is the slot number (1-9) in that chassis |
| -port=PORT | PORT is the port number (1-4) on the WaveBlade in the slot |
| -freq=FREQ | FREQ: frequency in the MHz desired. ² (default = 5250) |
| -freq_hop_min=F_H_MIN | F_H_MIN: the min freq in MHz for frequency hopping. ⁵ |
| -freq_hop_max=F_H_MAX | F_H_MAX: the max freq in MHz for frequency hopping. ⁵ |
| -radar_spec=RADAR_SPEC | RADAR_SPEC is FCC, ETSI or Japan (default = FCC) |
| -radar_type=RADAR_TYPE | RADAR_TYPE as outlined in the tables above (default = 0) |
| -power=POWER | POWER is output power (default = 0) ³ |
| -trials=TRIALS | TRIALS is the number of trials to run (default = 1) |
| -file=FILE | FILE is the name of an input configuration file ^{4,6} |
| -loop_forever | Set the script to continuously repeat (default = run once) |

Notes:

1. Note that each switch is preceded by a double dash, not a single dash.
2. For frequency hopping radar types such as FCC type 7, the frequency is set by the script and this variable is not required. See note 5.
3. For all WaveBlade types except the WBW1101P, the output power range is 0 to -50, settable in 1 dB increments. For the WBW1101P, the range is 15 to 0 in 1 dB increments.
4. The input configuration file must contain information for all the trials desired.
5. For frequency hopping radar types such as FCC type 6, the minimum and maximum of the hopping range is defaulted per the table above. The range can be set to a different minimum and maximum by setting the --freq_hop_min and --freq_hop_max variables. For example, to cause all pulses generated to land within the detection bandwidth of a given channel, the min and max variables can be set to the actual frequencies of the lower and upper detection range of that channel.
6. The --power and the --freq switches can be used with the --file switch. The value passed from the command line takes precedence over values in the configuration file for these two variables only.

Example:

```
python test_radar.py --chassis=wt-tga-00-29 --slot=2 --port=1 --freq=5120 --radar_spec=FCC
--radar_type=7 --power=-23 --trials=1
```

Referring to the radar pulse descriptions above, note that there are some selections that have a single choice for each parameter while others show a range of numbers for some parameters. For those with singular definitions for each parameter (such as FCC type 1), executing the command line repeatedly results in exactly the same radar pulse behavior being generated. For those with a range of values available (such as FCC Type 2), executing the same command line repeatedly results in a random variation from one run to the next.

Each time the command line is executed, a file with the name "radar_config.py" is created in the execution directory. This allows the user to repeat a test configuration as desired, or create a specific configuration of the parameters shown in the tables above. Each new execution of the command line will overwrite the previous instance of "radar_config.py", so it is advisable to rename the file if retention is important.

The use of the configuration file makes it simple to execute the last instance of a test. By executing the test_radar.py command with the --file=<filename> option, making the filename the configuration file saved from the run of interest, the exact same setup will be repeated.

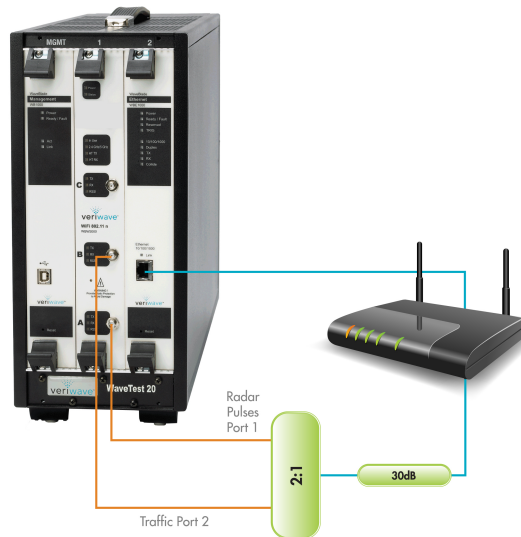
The configuration file can be directly edited to create the combination of parameters desired. However, care should be taken that there are parameter definition lines for each trial in the configuration file. An easy way to understand the relationship would be to run an instance of the `test_raday.py` script with the number of trials set to 3, then open and inspect the resulting `radar_config.py` file. Under the parameter definition lines in the last half of the file (such as Burst Period) it will be seen that there are three distinct definition lines, indexed as [0], [1], and [2], the first of which corresponds to the first trial, the second to the second trial, and so on. The `num_trials` variable in the configuration file must match the number of definition lines per attribute exactly.

Use of the configuration file also makes it possible to configure the system to produce exactly the combination of attributes desired. The recommended procedure is to run the script once for the radar type and spec of interest, then edit the resulting configuration file as needed to match the specific conditions. Any given configuration file can be run at a different power level and/or frequency by using the `--power` and/or `--freq` switches at the command line. These two variables are the only two that can be passed in from the command line that will take precedence over settings in the configuration file.

Recommended Test Configuration

The typical use of the VeriWave solution involves using an Ethernet port for generating the test traffic required by the test protocol, and a Wireless port for receiving the test traffic. A simple means of accomplishing this is to use the WaveDynamix application, and configure a downstream UDP flow of 1518 byte frames at 6 Mbits/sec intended load, after establishing clients on the port cards and connecting them. This provides the equivalent traffic to the video clip typically used by the FCC for background traffic, and eliminates the need for computers and media players to stream video. This also provides a very convenient method to alter the traffic to higher loads, or introduce upstream traffic as well, useful for stress and margin testing.

The recommended configuration is shown below. Port 1 of the WBW1104N is used to generate the Radar Pulses, while Port 2 is used for 802.11n traffic generation/reception. With this configuration, the transmit power level of Port 2 can be adjusted so that the AP is checked under the condition where the client power is relatively high, emulating the client being positioned close to the AP in an actual deployment.



Characterizing for Correct Radar Pulse Signal Level

It is important to ensure that the signal power of the radar pulses are at the prescribed level as defined in the DFS test procedures. A means to generate a 50% duty cycle output stream from the WaveBlade port so that a power meter can be used to measure actual signal power at the input to the Access Point has been provided. This is done by executing the DFS Radar script, using the configuration file named "duty50_radar_config.py" at the frequency of interest, and measuring the power level at the connection point to the Access Point. This configuration file produces a 50% duty cycle signal at 0 dBm output for 6 seconds, so the measurement observed on the power meter constitutes the total correction that should be applied to the power setting while running the script during testing.

Note: care should be taken when using the power meter. Most power meters will make the measurement assuming a 100% duty cycle by default. It is recommended that the power meter settings are configured for 50% duty cycle. If that is not possible, it is an acceptable alternative to add 3 dB to the power meter reading to obtain the proper power reading.

An example of making the measurement for use on Channel 100 (center frequency 5500 MHz) is as follows: connect the system as shown in the figure above, including the splitter and all cables. Rather than connecting to the Access Point, connect to a Power Meter. At the command line, enter

```
python test_radar.py --chassis=wt-tga-00-29 --slot=2 --port=1 --freq=5520 --file=duty50_radar_config.py
```

When the TX light illuminates on the port, the test signal is being supplied to the power meter, and the reading should stabilize over the course of 6 seconds. Record that value to the nearest integer dBm reading. If for example, the reading is -38.7 dBm, this means that the total power inaccuracy and loss in the generation system and configuration amounts to approximately 39 dB. Therefore, if the specification being tested to requires -62 dBm radar signal power at the AP during testing, the power level should be set to -23 dBm ($-62 + 39 = -23$) as shown in the command line example on page 3.

If the power meter used in this example can be configured only to measure signals with 100% duty cycles, and the measured reading on the power meter is -35.2, the reading must be corrected by adding 3 dB, resulting in the correct reading of $-35.2 + 3 = -32.2$ dBm. The total power inaccuracy and loss in this case would be approximately 32 dB. For a -62 dBm radar signal power at the AP during testing, the power level would be set to -30 dBm ($-62 + 32 = -30$).

Note: The WBW1101N and WBW1104N WaveBlades have output power adjustment capability from 0 to -50 dBm in 1 dB increments, so adjusting the power level as described above is possible. The WBW1101P has output power adjustment capability from 0 to +15dBm, so the test configuration shown above will need to be modified, inserting an additional attenuator at the WaveBlade port itself.