

# ZL1BPU Synthesiser Script Format

16 April 2020 Rev 1.2.

This format is intended as a series of offset frequency and other related instructions sent to a frequency synthesizer as a means of generating a pattern of symbols that can be read visually, for example using ARGO. It has also been demonstrated that the technique can also be used to send WSPR and mixed WSPR, CW and MT-Hell visual text messages.

Two groups of parameters are necessary for the definition of the symbols to be generated: the *fixed parameters*, and the *message script*.

## 1. Fixed Parameters

These are the absolute (or centre) carrier frequency, the nominal or base symbol frequency spacing, and the nominal or base symbol period. The message can either be repeated once completed, or repeated with a specified delay. *None of these parameters are defined in the message script.*

To some extent these fixed parameters may be synthesizer-specific.

nominal_frequency	carrier frequency in Hz at the centre of the message
step_size	carrier frequency unit step in Hz
nominal_period	symbol period in seconds

**nominal\_frequency** is that sent when script command '8' is used. There are eight incremental steps below this (7 – 0) and seven above (9 – F). All steps are the same size (step\_size or step\_size x (n+1) if the Wn command is used).

**nominal\_frequency** may be fractional, i.e. specified to a resolution smaller than 1 Hz. How it is interpreted will depend on the synthesizer.

**step\_size** may be fractional, and may also be negative. This allows the synthesizer to generate inverted patterns for use on LSB bands with a USB exciter.

**nominal\_period** is the duration of each individual tone (as defined in the script), and may also be fractional. There will be synthesizer-specific limitations to the smallest value permissible. For example with the FEI FE-56xx devices the minimum is about 0.5.

## 2. Message Script

The script defines the pattern to be sent, and consists of a file containing (typically) one line of letters and numbers. Letters and numbers other than those listed below are to be ignored. Once the end of the file is reached, the transmission has ended. Whether it starts again immediately or after a time depends on the final command.

Some letters are script commands, and may be followed by a one-letter/number parameter. Letters that can be interpreted as script commands, but do not (if required) have a correctly interpretable following parameter, will be ignored.

## 2.1. Frequency Offset

The following letters and numbers are used to define an offset from the nominal\_frequency:

- 0** Send nominal\_frequency - 8 x step\_size for the next period
- 1** ditto - 7 x
- 2** ditto - 6 x
- 3** ditto - 5 x
- 4** ditto - 4 x
- 5** ditto - 3 x
- 6** ditto - 2 x
- 7** ditto - 1 x
- 8** Send nominal frequency
- 9** Send nominal frequency + 1 x step\_size for the next period
- A** ditto + 2 x
- B** ditto + 3 x
- C** ditto + 4 x
- D** ditto + 5 x
- E** ditto + 6 x
- F** ditto + 7 x

## 2.2 Other Script Commands

The following letters and parameters are used to control synthesizer behaviour:

- Pn** Power level. n = 0 – 4. P0 is the default, full power. Each step represents half amplitude output, i.e. -6 dB. No symbol period is consumed. (The number of levels may be hardware dependent).
- Q** Quit sending (send zero frequency until timer expires, then start script again). Q (if used) is always the last command in the script. Transmission is stopped (and PTT is dropped, if fitted) when Q is reached, and is not activated again until the script restarts. One symbol period is consumed. If the end of file is reached without a Q command, the transmission continues at the start of the script.
- Sn** Period multiplier. N = 0 – 9, A – F. Nominal\_period x (n+1) is used to time the duration of symbols until the script starts again with the nominal\_period or a further Sn command is found. S0 is the default (one times nominal symbol period). No symbol period is consumed.
- T** Transmit. Causes the carrier to start on the last used frequency.
- Wn** Width, i.e. spacing of symbols. N = 0 – 9, A – F. The frequency shift value step\_size is multiplied by (n+1). The default value is W0. No symbol period is consumed. Not implemented in all compatible equipment.
- Vn** Selects the VFO to be addressed by subsequent script commands. . Not implemented in all compatible equipment. The value n is a single digit, typically 0 – 3.
- X** Send zero frequency for the next period, i.e. carrier stops. For synthesizers with PTT capability, PTT is not dropped. For Fractional-N PLL synthesizers with this capability, the carrier is turned off. For Fractional-N PLL synthesizers and others which can't send zero frequency, nominal frequency is sent and PTT is dropped. One symbol period is consumed.

## 2.3 Script Interpretation

Not all synthesizers need support all commands. The minimum command set interpreter must (in addition to frequency offset 0-9, A-F) include Q and X commands.

When Q is met in a script, any following commands will be ignored. The PTT (if fitted) will be dropped one symbol period following the last frequency command. This ensures that the last symbol isn't clipped.



Even if the compiler does not implement the full specification, it will certainly reduce the effort involved in making good-looking ARGO images. The files can always be post-tweaked manually, and probably always will.

#### 4.1. Making Scripts

Figure 2 has an example of a script which includes Sequential Multi-Tone Hell text (which can be read directly on ARGO, exactly as in Figure 1), graphical elements and CASTLE mode. It is easy enough to also (or instead) include plain carrier segments, OOK CW, FSK CW, DFCW, more complex graphics, and even WSPR.

In Figure 2 the grid represents time intervals horizontally (typically one second units), and frequency vertically (typically 1 Hz units). These values give reasonable results with Sequential Multi-Tone Hell text on ARGO in QRSS3 mode.

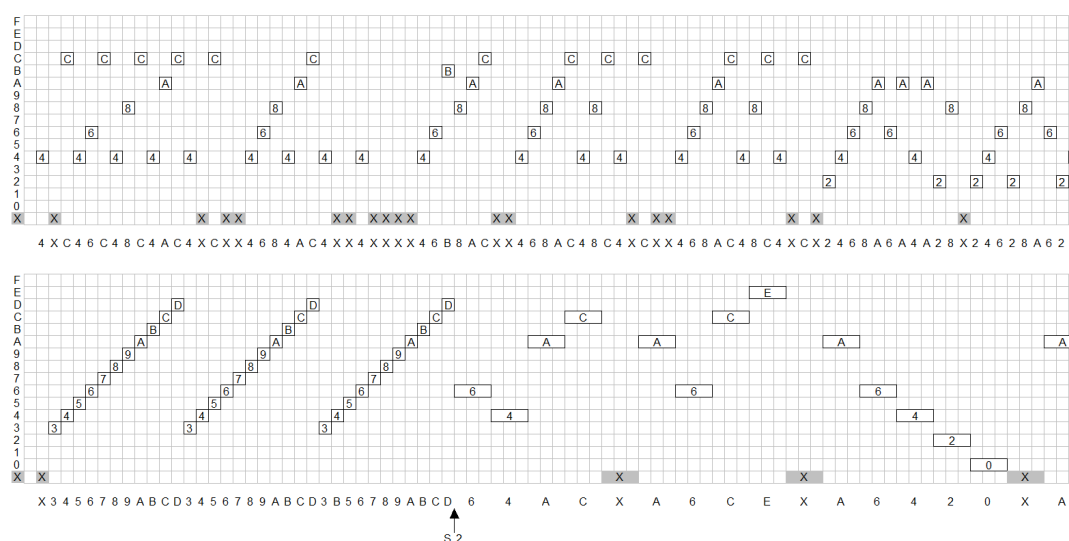


Figure 2. Script example explanatory grid

Note how to the left of the grid the rows are labelled by their frequency number, and the columns (time) have the pattern script listed under the corresponding column. The numbered squares in the grid represent the transmitted dots, and are labelled with their frequency number.

The lowest row (with grey 'X's) represents columns where there are no dots. In the corresponding script there is of course an 'X'.

#### 4.2 Sequential MT-Hell

This is the most difficult mode to represent visually. This is because in any single time slot, there can only ever be one symbol, and this considerably restricts the shape of characters. Some letters and numbers are particularly difficult to render in an appealing and easily recognisable way, for example 'Q', and often dots need to be left out, or spaces added, in order to improve the overall appearance and readability. Using a script compiler will ease this problem, but final results should always be checked with ARGO. Making good-looking MT-Hell scripts is undoubtedly an art form!

MT-Hell also has about 4dB disadvantage over most QRSS modes, because the dot length is shorter than the sample interval of ARGO. However, the results are undeniably readable!

It is also important to, as much as possible, preserve the even vertical and horizontal spacing of dots. If dots are too close, that segment of the character will look too bright, and if not spaced correctly the character shape may be difficult to recognise. The best test is how well the resulting script looks on ARGO when the signal is weak.

Look at the letter ‘Z’ in the top row of Figure 2. Here there are blank columns (‘X’) at the beginning and the end in order to preserve the horizontal spacing. If the character was rendered ‘4C46C48C4AC4C’, the bottom left and top right would look bunched up and possibly brighter.

Some characters require horizontal dots at times where there isn’t an opportunity, such as during a vertical stroke, which can consume five columns. The answer is simply to drop a horizontal dot in-between the verticals – it will hardly be noticed. There is an example of this in the letter ‘L’ and again in the ‘1’ (Figure 2). In the case of the ‘1’ a serif is dropped in, and the result is very pleasing (see Figure 1).

Some characters require ‘X’s in order to preserve proper horizontal dot spacing. Where there are no other dots to place, an ‘X’ is used. Such an example is the horizontal part of the letter ‘L’, where two blank columns are used before the final horizontal dot. In general horizontal row dot spacing should be kept at three dot intervals. The letters ‘E’ and ‘b’ don’t fit this rule. It’s all about making the best compromise. Lower case letters are especially difficult to render appropriately.

The letter ‘R’ in the example illustrates how dots can be missed and yet the eye still recognises the character. In this case there should really be a further ‘6’ about where the last ‘A’ is in the letter ‘R’. To include the dot would make the letter misshapen, and yet without the dot the character is quite recognisable. It’s all about art!

Avoid lower case letters as much as possible, as they are especially difficult to render. Also, only use characters (upper or lower case) five dots high – any higher will result in text that is stretched excessively in the horizontal direction, while fewer vertical dots will make it impossible to render characters recognisably.

Characters look best when the dots are spaced two rows apart vertically and three rows apart horizontally. This is simply a quirk of ARGO. Rather than specify 2 Hz frequency spacing for characters and send ‘6789A’, we use 1 Hz spacing and send ‘468AC’, i.e. every other row. There is an important advantage, as in-between dot rows can be used to better render characters (see the serif example in the ‘1’). This is a very important tool for rounded letters such as ‘O’ and ‘Q’. This also allows for finer detail in simple graphics, such as the diagonal lines in Figure 1.

### 4.3. CASTLE and other Morse

These modes are conventionally sent with three-second dots, as this gives best sensitivity on ARGO in QRSS3 mode. Rather than specify the dot period to be three seconds (which is perfectly OK), in a mixed-mode message such as the example in Figure 2, the S2 command is used to temporarily stretch the elements out to three seconds ( $n+1$ ). This reduces the number of characters in the script considerably, and more importantly, prevents unsightly keying transients in the middle of the elements, which can occur when using fractional-N synthesizers which do not preserve carrier phase between frequency changes.

On-Off Keyed (OOK) Morse should be avoided, as it is very difficult to read on ARGO when signals are noisy. It is better to use FSK Morse with say 5 Hz shift. FSK also works best with synthesizers you can't turn off, or amplifiers which can't tolerate loss of drive.

DFSK (also known erroneously as DFCW) is more compact but harder to read. The best Morse option is CASTLE, as illustrated in Figure 2. This is the most compact format, and while a little wider than other Morse modes, is much the fastest for a given element length, and is distinctive and arguably still quite readable.

## 4.4 WSPR

This is perhaps the most challenging mode to send using a script interpreter, but it is very effective, and can also be used in mixed-mode transmissions. The actual procedure for generating WSPR code is beyond the remit of this document, but the process is outlined here.

The first step is to set up the Windows WSPR software with the necessary message (e.g. ZL1EE RF72 20 dBmW). Then use the WSPR program command line options to store the message as a series of tone numbers. Converting these to script dots is an ugly process, and best achieved by writing a small program to extract the symbols and convert them to script values.

Always place the WSPR segment at the start of a multi-mode message, as that makes timing the start of transmission much easier (yes, you will have to time the start manually). Set the symbol spacing (step\_size) to 1.4648 Hz and the symbol speed (nominal\_period) to 0.682687 seconds. The message for other modes may need to be adapted slightly to match the unusual WSPR spacing and period.

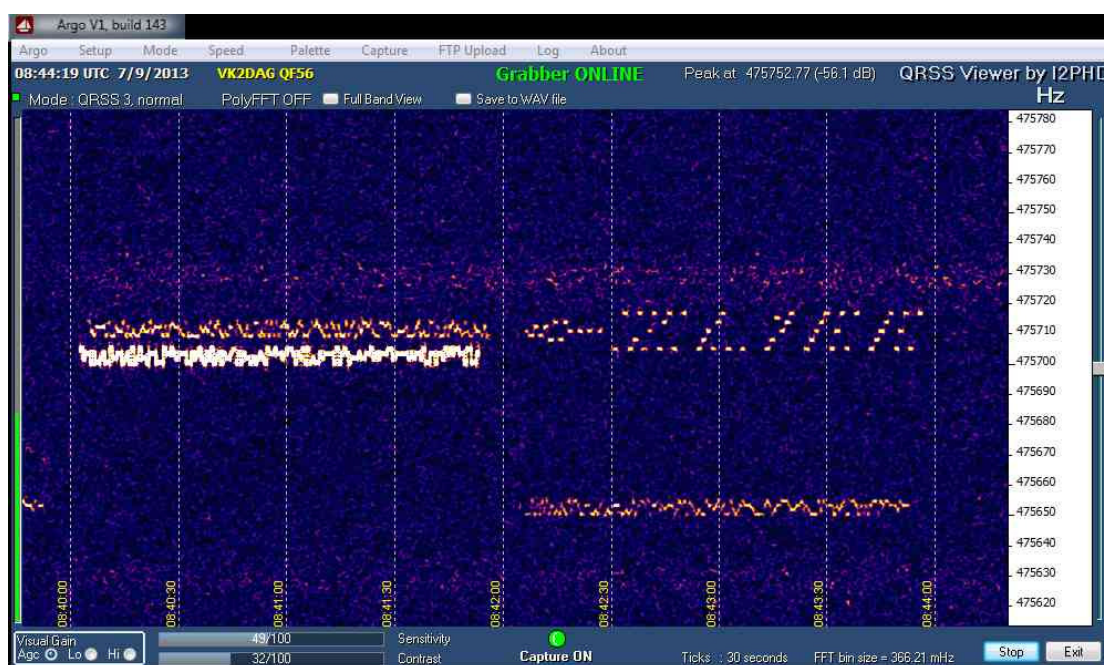


Figure 3 VK2DAG receiving WSPR/MT-Hell multi-mode from ZL1EE on 475 kHz

The example in Figure 3 was also decoded successfully with WSPR, and the text could be seen on the WSPR waterfall!

Murray Greenman  
16 Apr 2020