JT4 Data on the UKµWG Reverse DDS Module

Andy Talbot G4JNT April 2010

JT4 Overview

The JT4G data mode, part of the WSJT Suite by Joe Taylor, K1JT, [1] has proved its capability for getting through under extremely weak signal conditions, and can cope with fading, frequency drift and up to several hundred Hz of frequency scattering. Its weak signal performance alone appears to show about 6dB advantage over aural copied CW. A JT4G message has been added to three of the Bell Hill beacons, GB3SCS, GB3SCC, and GB3SCX on 2.3, 5.76 and 10GHz, and is also in use on the Central Scotland 10GHz beacon GB3CSB.

JT4 in all its variants (A-G) consists of a four tone Multi Frequency Shift Keyed (4-MFSK) waveform, with the spacing between the tones chosen depending on the frequency band and expected spreading. On microwaves the widest spacing, the G variant, has been adopted. Conversely, at HF the narrowest is used with 4.375Hz tone spacing The MFSK message consists of 207 symbols (one of four sequential tones) transmitted at a rate of 4.375Hz, the whole message therefore taking about 48 seconds to send. A rigid timing structure is in use, and the start of the transmission must coincide with the UTC minute interval. For beacon usage, the even minute has been universally chosen as the reference start time. For the decoder to work correctly, this start point must be accurately defined, being no more than a few seconds late, and no more than one second early (the protocol was originally designed for EME with its 2 seconds delay). For beacon use the entire message contains exactly 13 characters taken from an alphabet of letters, numbers and a few punctuation symbols.

The UKµWG RDDS board can be used to generate JT4 data by replacing the PIC with one containing the appropriate firmware, and addition of a GPS receiver supplying timing information. The PIC decodes the serial data from the GPS receive then at the even-minute point reads pre-stored message data and converts this to JT4 modulation by reprogramming the DDS in real-time at 4.375Hz with values corresponding to each of the four tone frequencies. The four frequencies have to be calculated beforehand and are stored in the PICs non-volatile memory, along with the message symbol data and certain other setup parameters.

A CW message is stored and is replayed immediately the JT4 transmission has completed, starting at approximately the even minute + 50 seconds mark. At a user definable point, in seconds after the odd minute, the CW message is again replayed. In typical usage this allows CW idents spaced by around 30 seconds with periods of plain carrier of at least the recommended 20 seconds. Adjusting the time of the second CW message allows one extended period of carrier every two minutes.

Connecting the GPS module.

The 20 pin expansion header provided within the RDDS unit is used for interfacing to the GPS. Three data lines are used, although only two are actually only needed for JT4 operation. The third allows GPS receiver setup information to be sent at start up of the controller if needed. In extreme cases, if reduced accuracy of the timing is acceptable only the serial data is essential.

The 7805 regulator in the RDDS module has sufficient spare capability to supply the 100 – 200mA needed by most GPS receiver modules that take a +5V input, although users may have to be aware of any additional heat sinking requirements that may be necessary.

The description that follows, as well as the PIC firmware supplied, assumes that serial data in one of two formats is available. The proprietary binary format given by the Motorola Oncore or M12 type GPS module at 9600 baud or standard NMEA text messages at 4800 baud. The polarity of the data can be selected at the time the PIC firmware is compiled. Either native 5V logic or RS232 polarity can be catered-for

At the time of writing, suitable Oncore GPS receiver modules are available from [2] Note that this family of modules can be commanded to supply data in NMEA format, but given the choice, the binary version is preferable.

Figure 1 shows the interface connections, specifically for the Oncore Module with its 10 pin header, and includes an additional red/green LED for front panel mounting.

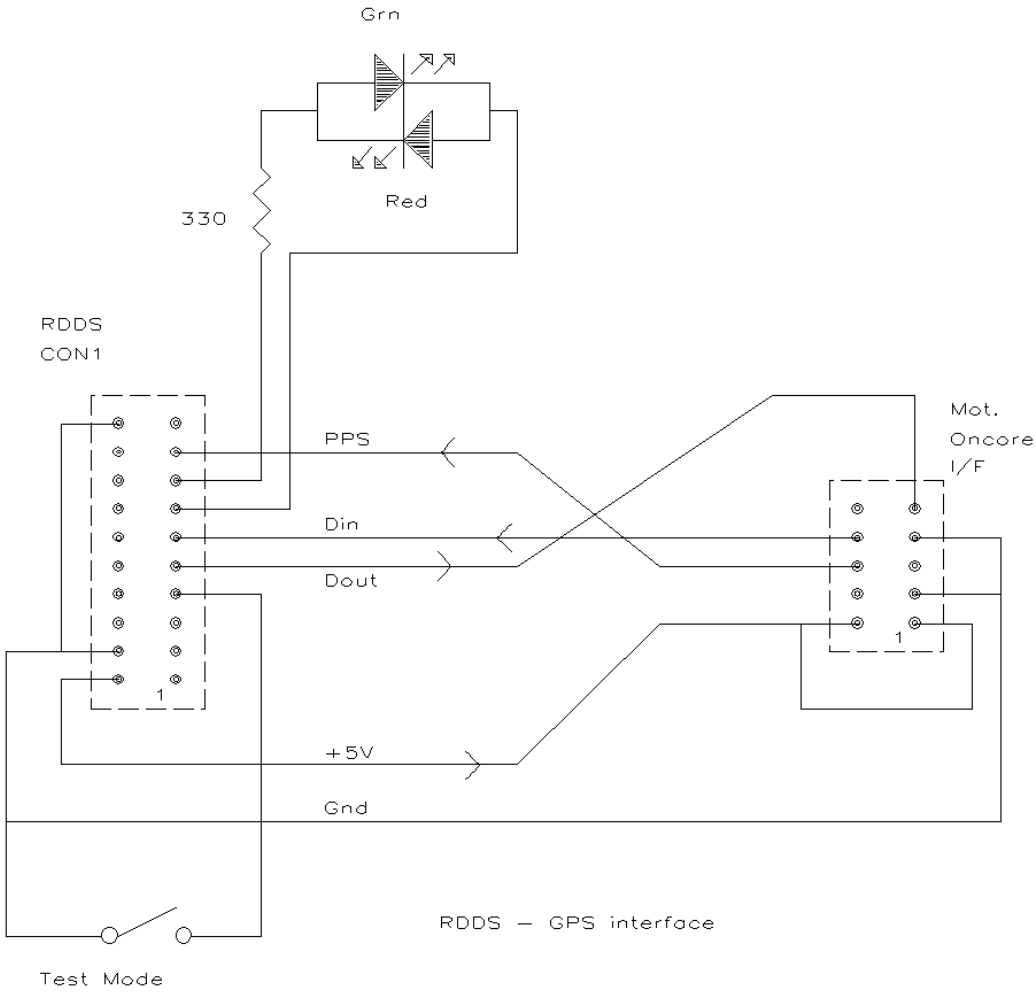

Table 1 below lists the generalised connection details for the RDDS interface for connections to other GPS receiver modules.

RDDS Header	Function	PIC I/F
2	+5V supply to GPS Module	
4 & 8	Ground	
7	Test Link to ground–carrier, leave O/C normally	PB5
9	Serial Data Out TO GPS (optional)	PB4
11	Serial Data In FROM GPS	PB3
17	One Pulse Per Second signal FROM GPS	PB0

The existing Green and Red LEDs mounted on the PCB are retained and the description that follows assumes LED1 is green, and LED2 the red one. These are on PIC I/O lines PB1 and PB2 respectively. Users may care to bring the connections to these (pins 13 and 15) out to an external anti-parallel connected red/green LED (with additional series resistor) for front panel mounting as it gives useful information about message progress and GPS lock condition.

PIC Coding Details

All information relating to the message, frequencies and setup need to be programmed into the PIC at the start. There is no facility for field updating with an RS232 interface as was provided on the original RDDS implementation and all values need to be included within the source file which is compiled to give the .HEX file for download to the PIC device. Several compile-time flags need to be specified to customise the compiled code. These relate to the data polarity, the data format and whether the 1 PPS signal is used. Also whether, for straight DDS use, the internal PLL clock multiplier is to be enabled. This PLL must NOT be enabled for reverse DDS use as there will be a danger of over clocking (and cooking) the DDS chip.

The basic PIC firmware is contained in the source file *JT4BCN_9851.ASM* which is used unmodified. The JT4 symbol information resides in an auxiliary include file *JT4SYMBS.INC* which can be generated automatically by the utility *GENJT4.EXE*. Alternatively, the symbols can be derived from the WSJT software, following Joe's instructions supplied with the software suite, formatted and entered manually into the include file. Each of the 207 symbols is formed from two bits giving a value from 0 – 3 which are packed four to a byte, most significant first to give 51 bytes in total. (As listed, they are read out in order left to right, top to bottom)

User information such as frequencies and the compiler flags described above is stored in another include file *JT4_9851.INC* whose contents are shown in the example below. Please note, formatting is important, so read the description below thoroughly before modifying the .INC files.

JT4_9851.INC

```
;Compile-time Flags

 NMEAPol      =   1  ;Polarity of data from GPS   5V Logic = 0   RS232 = 1

 GPSDataType  =   1  ;0 for Motorola Oncore 9600 baud,  1 = NMEA 4800 baud

 IgnorePPS    =   1  ;0 for normal PPS.  1 for use data only (less accurate)

;Compiler Constants

  CWSPEED     =   d'75'     ;CW Dot length, ms

  REFTONE     =   0         ;Reference Tone No. for defining tuning etc.

  KEYUPTONE   =   4         ;Tones 0 - 3 are the JT4 ones.   Set the

  KEYDWNTONE  =   0         ;  reference and FSK tones in the range 0 - 7

  CWMSG2      =   d'30'     ;Seconds count for CW message during odd minute

; EE Data

FreqData             ;Labels must start in Column 1

              ;10368.905 MHz for RDDS, *96 multiplier

       de  0x17, 0xB3, 0x9E, 0x57       ;JT4 Tone 0

       de  0x17, 0xB3, 0x9E, 0x4B       ;JT4 Tone 1

       de  0x17, 0xB3, 0x9E, 0x3F       ;JT4 Tone 2

       de  0x17, 0xB3, 0x9E, 0x33       ;JT4 Tone 3

       de  0x17, 0xB3, 0x9E, 0x45       ;User defined  Tone 4   1270Hz

       de  0x17, 0xB3, 0x9E, 0x57       ;       Tone 5   800Hz

       de  0x17, 0xB3, 0x9E, 0x57       ;       Tone 6   800Hz

       de  0x17, 0xB3, 0x9E, 0x4F       ;       Tone 7   1000Hz

CWMsg           de      "G4JNT  IO90IV58",0    ;Null Terminated

ControlPLL      de      0  ;0 for no PLL, 1 for x6 PLL (NOT for RDDS use!)
```

Values shown as *d'45'* are normal decimal values, ideally used for simple variables. Numbers in the format *0x34* are hexadecimal, generally produced from the utilities for deriving frequency and JT4 symbol data.

Any text following a semi-colon ; is a comment and ignored by the PIC compiler.

Labels such as *FreqData*, *CWMsg* and *ControlPLL* must start on the left hand side, in column 1. All other lines should be indented otherwise the compiler will throw-up a warning message

Compile-time flags.

NMEAPol defines if the polarity of the data coming from the GPS receiver is 0/5V logic level as supplied directly by most GPS modules, or RS232 polarity for direct connection to a PC. Some early Garmin modules supply this latter polarity, as do some GPS receiver systems. . Use **0** for 5V Logic level / polarity, **1** for RS232.

Please note that if true positive/negative RS232 voltage levels are encountered, an additional resistor of around 4k7 needs to be inserted in the Data In line to prevent excessive current into the PIC interface pin

GPSDataType should be set to **0** for Motorola binary format data at 9600 baud; Use **1** for NMEA ASCII format at 4800 baud

IgnorePPS should be set to **0** if the 1 PPS signal is used to define the exact edge of the UTC second. A value of **1** should be used if the timing is derived from the serial data only without connection of a 1 PPS signal. This situation may be preferred if timing data is derived from a GPS simulator such as an MSF time code utility, where the provision of a 1 PPS timing pulse will likely offer no greater accuracy than that of the serial data alone.

Compiler Constants

CWSPEED is a compiler constant and defines the dot length of the CW, in milliseconds. Use **d'100'** for 12WPM, **d'75'** for 16WPM etc.

REFTONE is the tone number (see below) in the range **0 – 7** defining which of the pre-stored frequencies is to be used for the period of plain carrier. This is usually the designated reference frequency for the beacon

KEYUPTONE is a number in the range **0 – 7** defining which of the pre-stored frequencies is used for FSK keying tone in the **KEY-UP** state

KEYDWNTONE in the range **0 – 7** defines the FSK keying tone for the **KEY-DOWN** state.

Current advised UK beacon practice for simple two frequency keying is to use the nominated frequency for key-down and for plain carrier, and the shifted one (usually 400Hz higher in frequency) for the key up state. In this case, REFTONE and KEYDWNTONE will be equal, with KEYUPTONE the alternative. The eighth tone is spare.

CWMSG2 is the number of seconds after the odd minute that the second CW message is sent. Adjust its value to ensure a sufficient period of plain carrier is available for monitoring purposes.

EE Data

FreqData is a label signifying the start of the frequency data, and MUST occur at the extreme left of the .INC file.

The next 8 lines (starting ***de*** to signify they are EEPROM data) each contain the 32 bits (4 bytes) of frequency information for the eight pre-stored frequencies, corresponding to tones 0 – 7, most significant byte first. Although shown in Hex, it is permissible to list these in decimal notation if this is preferred for calculation purposes. The first four lines are the frequencies of each of the JT4 symbols. The next four are tones that can be selected by the user to give, for example, a reference at exactly the mid point in the spectrum (tone 1.5) or a pair of precisely spaced tones for CW keying. Which tone is allocated to which function is defined in the section above, and it is not even necessary to calculate any values for frequencies 4 – 7 if all keying uses the JT4 tones. In this case programme a value of zero into the unused spaces. The programming frequency generation utility ***JT4_DDS_Freqs.EXE*** gives the option of copy-and-pasting directly into the .INClude file. Highlight the generated frequency data then copy and paste directly underneath the FreqData label, deleting old data and adding comments as appropriate.

CWMsg is a label to show that this line is EEPROM data containing the CW message inside inverted commas. It can be of arbitrary length and there is space for about 40 characters. The data must have a zero, a null terminator, after the closing inverted commas exactly as shown to indicate the end of the message. If not present, the software will crash! If the CW message is too long the compiler will generate an error message about data being overwritten.

The final line, **ControlPLL** is a flag to indicate whether the DDS clock X6 multiplier is to be used. For RDDS configuration it must be set to 0 - PLL disabled. When this PIC code used with a standalone source, the PLL may be enabled.

JT4SYMBS.INC

This include file is generated automatically in exactly the form shown as a result of running the utility ***GENJT4.EXE***. [3] It should not be necessary to alter the file in any way. As the file is regenerated and overwritten each time ***GENJT4*** is run, it is advisable to save a copy under a different name – eg, *GB3SCS_JT4SYMBS.INC*.

The WSJT software does offer the ability to generate the symbol data in a listed form, and users may want to use this route instead – for example to include a 'QSO-type' message into the beacon data instead of 13 characters of plain text. In this case, the individual symbol data in the form of 207 numbers with values 0 – 3 will have to be assembled manually into the EE data bytes, four-at-a-time starting with the most significant pair of bits. For example, if the first eight symbols generated are 3,1,2,0,2,1,3,0, the resulting first two bytes will be b'11011000' and b'10011100' or in hex 0xD8, 0xC0. Both these formats, binary or hex, (or even decimal as d'nn') are acceptable to the compiler. Read the WSJT documentation for further details of how to generate the symbol list.

```
;  JT4 Symbols  generated from GENJT4     G4JNT Jul 2009
;  Message data 'GB3SCS IO80UU'

    de  0x00, 0xD8, 0x14, 0xDA, 0xC4, 0x02, 0x8D, 0x28
    de  0xAA, 0x0A, 0xC7, 0x9C, 0xEF, 0xD6, 0x68, 0xC3
    de  0xA5, 0x74, 0x2C, 0x6A, 0x75, 0x1E, 0xB8, 0x34
    de  0xC4, 0xC6, 0xF5, 0xC4, 0x67, 0x33, 0x9D, 0xA4
    de  0x59, 0x76, 0xA9, 0x65, 0x83, 0x53, 0x73, 0x50
    de  0xC0, 0x51, 0xE9, 0x2B, 0x57, 0x63, 0xE2, 0x34
    de  0x26, 0x73, 0xD6, 0x6C
```

JT4 Software Operation

When power is first applied the DDS will be initialised to the reference frequency and the CW message will be sent once, with the red led flashing in sympathy with the CW characters. The green led will flash once and the GPS data line is monitored for a valid 'GPS Valid' code arriving . In the NMEA $GPRMC string this is the "A" sent in the third data item, in Motorola protocol it appears as a flag in the appropriate position. Until the GPS is confirmed as valid, the CW message will be sent repeatedly with a single flash of the green LED between messages.

As soon as the GPS receiver indicates valid data is now present the green LED will start to give a short flash once per second in synchronism with the serial data stream which is continuously monitored to determine the GPS lock status. Note that when NMEA data is used – the flash is very short. With the GPS-locked flag is detected, the pattern is long flash every second (for NMEA it will appear to be almost continuously on). If the GPS receiver subsequently loses lock, the PIC monitors the GPSValid flag and changes to a short green flash to show this. Since most GPS receivers will still be outputting valid data and retain their timing during such flywheel operation, no further action is taken by this software other than the short green flash to indicate loss of lock. When lock is reacquired, long green flashes resume.

The time sent from the receiver is decoded and as soon as the first second of the even minute is detected, the JT4 transmission starts on the next PPS edge. While JT4 data is being sent, the red LED flashes at half symbol rate, about 0.45 second intervals. When the JT4 message is complete, the frequency is shifted first to the reference tone for one second, then the FSK CW message is sent, with the red LED flashing in time with the key-down state of the CW. And the end of this, the green flashes resume while plain carrier at the reference frequency is being generated. When the programmed second CW time is reached, the red LED again flashes to show the CW characters, followed by green one second flashes until the even minute when the cycle repeats If the green flashing becomes of short duration, this means the GPS receiver has lost lock, possibly due to interference or jamming. The timing will usually flywheel for many hours or even days before the resulting cumulative timing error becomes unacceptable.

Test Mode

The link or switch installed on pin 7 of the interface header provides continuous carrier at the reference tone frequency (*REFTONE*) for test purposes. If activated during the CW or JT4 sequence, this is allowed to complete before Test Mode is entered. The red LED is on continuously.

References

[1] http://physics.princeton.edu/pulsar/K1JT/

[2] http://www.alan.melia.btinternet.co.uk/

[3] http://www.g4jnt.com/JTModesBcns.htm