A Position Reporting System using WSJT Weak Signal Data Modes

Andy Talbot G4JNT July 2015

Updated to include JT9 March 2016

Overview

There are a number of schemes around that use the data from a GPS receiver to report their location in real time. These all use radio links of some type, examples are the General (amateur) Packet Radio Network (GPRS) and direct Frequency shift keying, effectively RTTY, sent at 433MHz from high altitude balloons. In the commercial world, the mobile and satellite phone networks are used extensively as well as a few dedicated network based systems. I was wondering what could be done using weak signal modes and strong forward error correction (FEC) to enable automatic position reporting using low power transmitters and direct reception, ie no repeaters or networking.

The WSJT suite, widely used by amateurs for extreme weak signal communications [1] has a range of modulation types all geared to sending preformatted messages using a variety of mostly multi-frequency shift modulation variants. To save reinventing wheels, could WSJT modes be used to transmit the location information from a GPS receiver?

Message Structures

The majority of GPS receivers and modules generate an NMEA formatted serial data stream using RS232 type signalling at usually 4800 or 9600 baud. A typical output sentence containing position as Lat/Long together with time and date with a few other parameters is :

*$GPRMC,152914,A,5054.5784,N,00117.4020,W,000.0,000.0,230615,003.5,W*70*

Only the Lat/Long information needs to be transmitted as the WSJT receiving software itself uses real time supplied by the PC it is running on for logging. The precision delivered by the GPS receiver, to an accuracy of 0.0001 minutes of arc, corresponds to a distance of just 185mm when considered as a great circle angle. On longitude values it represents an even lower distance – around 100mm at UK latitudes. While this level of position accuracy is rarely going to ever be needed, it seems a pity just to throw it away in the interest of message brevity, so how could this Lat/Long information be sent using one of the WSJT weak signal data modes?

The WSJT modes under consideration are JT65 (A , B or C submodes) which use Reed Solomon encoding to transmit either a 13 character message or compressed callsign and locator or report information using one of 65 tones with a symbol rate and therefore noise bandwidth of 2.7Hz. Another is JT4 (with A – G submodes) which uses convolutional encoding at 4.4Hz symbol rate. The other WSJT modes, FSK441 and ISCAT weren't considered here as they are not specifically designed for error corrected weak signalling. Another contender is WSPR with its 1.5Hz bandwidth, and well-known capability for extreme weak signal. Unfortunately WSPR uses a rigid message structure which, ironically does include location information, but only as a four digit locator, such as IO90 with a

resolution of many tens of km. So unfortunately, WSPR had to be put to one side (for now!) But see the Appendix for ways WSPR can be used for slightly degraded location reporting.

JT65 would be the ideal choice with its 13 character message, strong FEC and narrow symbol bandwidth. But however much I have studied the mathematics of error correction, I just cannot get to grips with Reed Solomon encoding. It is NOT SIMPLE; Galois Fields, Finite arithmetic, Polynomials – I could go on. As the Lat/Long has to be encoded on the fly at the transmitter, a complete encoding process has to be done for each report. It would prove impossible, or at least would require a lot of processing power, to use one of the ready-made JT65 tools such as *JT65CODE.EXE* available from [1]. As I wanted to do the code generation in a small PIC processor JT65 was, therefore, not an option.

The encoding of JT4 consisting of four tone MFSK modulation is, however, relatively straightforward to understand [2]. I already had PIC code that would take a 13 character message, encode it to JT4 symbols, then send these to one of a variety of frequency synthesizer chips to generate the transmit waveform in real time. The DL0SHF EME Beacon on 10GHz uses just this code, with an LMX2541 Fractional-N synthesizer to send a time-stamped hash on its JT4G modulation to assist in verification of reception reports [3]. So, how to compress Lat / Long data into a 13 character message suitable for JT4 transmission?

Compressing the Lat / Long data

Consider the NMEA sentence

*$GPRMC,152914,A,5054.5784,N,00117.4020,W,000.0,000.0,230615,003.5,W*70*

This contains high resolution location information in the form DDDMM.MMMM (D = degrees, M = minutes) followed by N/S or E/W . For latitude only two digits are sent for DD.

Lat = 50° 54.5784’ N Long = 1° 17.4020’ W.

The Degrees can take on values from 0° to 179° (0° to 89° only for latitude). The minutes, ignoring the decimal point, can take on any of 600000 values from 00.0000’ to 59.9999’. One extra bit of information is needed in each case for N/S or E/W determination. By convention degrees West or South are regarded as negative values which are difficult to work with in simple processor arithmetic. A negative value can be represented by adding on, say, 180 degrees to the DDD value to act as a sign flag. Or, perhaps a bit neater, the sign bit could be encoded separately.

The total number of values needed to transmit the full longitude as supplied by the GPS receiver is therefore 180 * 600000 * 2 = 21600000 or 216 million. For the latitude, half this number is needed: 108M. The large number, X, now completely expresses the Lat/Long in units of 0.0001 minutes of arc. For the example given above:

X_{LAT} = 50° * 600000 + 545784 = 30545784
X_{LONG} = (1° + 180°) * 600000 + 174020 = 108774020

Now, how can these two large numbers be compressed into a maximum of 13 characters?

The WSJT Alphabet consists of 42 characters, made up from letters A-Z, numbers 0 – 9 and the six punctuation characters symbols *[space] + - . / ?*

One compression technique using the least number of characters is to map the value of N onto all 42 possible characters by successively dividing N by 42, and using the remainder to represent each character in turn. Ie. expressing the value MODULO(42) or BASE 42 so any character represents a value from 0 to 41. '0' = Zero, 'A' = 10, 'Z' = 35, '?' = 41 etc.

Using the WSJT alphabet "*0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ +-./?*" to represent values from 0 to 41, the latitude value, X_{LAT} can be compressed as follows :

30545784 / 42	=	727280	remainder	24	=	O (letter O)
727280 / 42	=	17316	remainder	8	=	8
17316 / 42	=	412	remainder	12	=	C
412 / 42	=	9	remainder	34	=	Y
	Most significant character				=	9

So the latitude encodes as **9YC8O**. Using the same techniques, Longitude encodes as **Y/7BQ** and the whole location information could be encoded into the 10 character string **9YC8OY/7BQ** . A compact solution but certainly not ideal. Spaces and inconvenient punctuation symbols make the result somewhat unintelligible when trying to read it, and there is the problem of the I/1 and 0/O visual ambiguity always present in alphanumerical strings. Furthermore, there are three redundant characters in the WSJT message not used and probably will now be wasted.

So is there a way of encoding the Lat/Long into a neater-looking subset of the alphabet that will still fit the answer into 13 characters? It is not possible using numbers alone, but using just the letters 'A' to 'Y' , a value of 216 million can be fitted into 6 characters using arithmetic Base 25:

30545784 / 25	=	1221831	remainder	9	= J
1221831 / 25	=	48873	remainder	6	= G
48873 / 25	=	1954	remainder	23	= X
1954 / 25	=	78	remainder	4	= E
78 / 25	=	3	remainder	3	= D
			MSB	3	= D

The MODULO(25) data for latitude now becomes **DDEXGJ**. Using six letters for Lat and Long allows a separator of '-' to be inserted between the two fields to enhance readability and deliver a final 13 character message.

One minor change allows for a neater treatment of N/S or E/W. Using an offset of 180 degrees to indicate the sign means the complete text string changes if the direction is swapped. While this doesn't affect the result, it would be nicer if the N/S or E/W flipped only one letter in the final data. Without the 180 offset value, numbers from 0 to 107999999 are generated. The most significant value MODULO 25 for this only ranges from 0 to 11, (letters 'A' to 'L') so the values 12 to 23 , or 'M' to 'X' are still available.

A neater solution to including the sign information is now possible; instead of adding an offset of 180° (X = 108000000) to indicate a negative number, use a value of $12 * 25^5 = 117187500$. This results in only the most significant character changing.

So 50° 54.5784' N encodes as **DDEXGJ** whereas
50° 54.5784' S encodes as **PDEXGJ**

In the GPS NMEA string the E/W and N/S flag appears as an individual character and it is far simpler to just read this and change the most significant character appropriately, rather than do the complete calculation of adding on 180 degrees before encoding.

Using a hyphen as the separator , the complete Lat/Long information for
50° 54.5784' N 1° 17.4020' W is now **DDEXGJ-MBYNKU**

One further advantage of using just letters 'A' – 'Y' is that 'Z' is available to act as a flag for future expansion. Any occurrence of that letter, or any number, or no hyphen in the middle position means the data has been encoded in a different way.

I have, quite unilaterally, named this format ***LocBcn***

Generator and Encoding Hardware

Beacon sources currently generating WSJT modulation already have a GPS receiver and some means of generating multi-frequency modulation so the hardware for one of these could be used directly. Figure 1 shows a PCB carrying a low cost Maestro A2200 GPS module and 16F628 PIC controller. The four pin header on the right hand side carries SPI data directly to a variety of synthesizer or DDS chips. One of these boards connected to an LMX2541 synthesizer module shown in Figure 2 forms a complete source, capable of directly generating WSJT modes at any frequency from 35 to 1296MHz. So with a change of PIC software, it could be persuaded to send compressed Lat/Long information.

The PIC code reads the NMEA data and extracts the Degrees, Minutes and direction flags. The minutes can be read directly as a five digit number simply by bypassing the decimal point. It then does the calculation X = Deg * 600000 + Mins using simple 32-bit integer arithmetic for Lat and Long values The resulting value X is then successively divided by 25, again using simple integer division routines written in raw assembler. The remainder after each stage of division is converted to a letter in the range 'A' to 'Y' and placed in the appropriate place for source encoding. The direction flags are read and the MS Character altered appropriately. The thirteen characters are encoded into the 206 JT4 symbols that are stored and sent as frequency setting commands to the synthesizer chip at the correct rate and start time.

The NMEA string also delivers real time information and a GPS data Status flag. The seconds part of the UTC time field is needed in order to start the JT4 transmission at the correct interval, at the start of every minute. The status flag is used to inhibit transmission of location data if the GPS is not giving a valid fix. In this event, the message is replaced with an identification one and GPS status bit

identifier. To comply with licensing requirements, the transmission needs to be identified at "regular intervals". As the timing data on the NMEA string is in BCD format, every ten minutes is a convenient interval to adopt for this. So every time the units of minutes = "0", instead of sending a 13 character location code, a message with a callsign and short ident stored in the PIC EEPROM data area is sent instead.

As a practical test I set the synthesizer to generate JT4D modulation on 144.372MHz and added a five Watt power amplifier. This was installed in the car to a ¼ wave whip on the roof. The base station was a crossed-dipole in the loft (yes, I know, cross-polar , but ...) feeding an IC746 on receive. Its audio was fed into a PC running the WSJT10 software. Then I went for a 50 mile round trip deliberately, circuiting the base of Butser Hill in Hampshire to give some weak signal shadowing. The listing in Table 1 shows the contents of the WSJT screen while receiving the transmission.

Table 1 Raw Data as delivered by the WSJT Software

UTC	Sync	S/N	DT	DF	W				
125200	5	-18	0.7	-46	9	*	DDCXJC-MBTCAG	1	0 A
125300	1	-21	0.6	-48	7	*			
125400	7	-15	0.6	-50	7	*	DDCPKF-MBRIIS	1	0 A
125500	5	-17	0.7	-48	9	*	DDCFFR-MBQQVD	1	0 A
125600	6	-17	0.6	-53	7	*	DDCDCO-MBPVKI	1	0 A
125700	6	-17	0.7	-48	9	*	DDCBPH-MBOYSN	1	0 A
125800	6	-16	0.6	-46	24	*	DDCIUJ-MBOHEC	1	0 B
125900	1	-22	0.7	-33	9	*			
130000	6	-17	0.7	-48	9	*	G4JNT LOCBCN.	1	0 A
130100	1	-21	0.6	-42	7	*	DDEDIP-MBNMEQ	1	0 B
130200	4	-18	0.7	-44	7	*	DDETWS-MBNJFX	1	0 A
130300	7	-16	0.6	-48	24	*	DDFLRT-MBNHXK	1	0 A
130400	6	-16	0.7	-53	7	*	DDGBMB-MBNAMO	1	0 A
130500	2	-20	0.6	-28	26	*			
130600	1	-21	0.7	-42	9	*			
130700	4	-19	0.8	-57	7	*			
130800	2	-20	0.6	-48	9	*	DDIEGP-MBLPQE	1	0 C
130900	3	-20	0.7	-35	7	*	DDIQIE-MBLPRC	1	0 C
131000	5	-18	0.6	-26	28	*	G4JNT LOCBCN.	1	0 D
131100	3	-19	0.7	-42	9	*	DDISAW-MBMNRY	1	0 D
131200	5	-17	0.8	-37	26	*	DDITRJ-MBNAVN	1	0 A
131300	0	-23	3.0	-348	7	#			
131400	3	-19	0.7	-37	9	*			
131500	1	-21	0.6	-39	11	*	DDJFWQ-MBOJFR	1	0 D
131600	5	-17	0.7	-33	11	*	DDJPGK-MBOSSF	1	0 A
131700	4	-19	0.8	-33	9	*	DDJOGN-MBPKWX	1	0 A
131800	4	-19	0.7	-35	9	*	DDJPAF-MBQENC	1	0 B
131900	6	-17	0.7	-33	9	*	DDJIXQ-MBQJFM	1	0 A
132000	7	-16	0.6	-35	9	*	G4JNT LOCBCN.	1	0 A
132100	4	-18	0.7	-31	9	*	DDIVBK-MBQTVV	1	0 A
132200	5	-18	0.8	-31	9	*	DDIRAS-MBRJAM	1	0 A
132300	4	-18	0.7	-31	9	*	DDIMGM-MBRWHP	1	0 A
132400	5	-17	0.8	-28	7	*	DDIBHK-MBRYQV	1	0 A
132500	4	-19	0.6	-33	9	*	DDHRUD-MBSIVQ	1	0 C

Note that the strong FEC applied means that any messages that are too weak to decode just give blanks rather than false values. A very important point where recovered data must be correct; better to have nothing at all than to have incorrect location information. The station identification appears every 10 minutes, these messages have a full-stop at the end. Occasionally the ident can be seen appearing out of sequence with a '?' at the end. This is where the transmitter has read from the NMEA data that the GPS data is invalid – perhaps due to GPS signal blockage or interference / jamming. The ident is then transmitted in place of the compressed Lat/Long message, with the final character changed to show the status.

Decoding the Messages

The arithmetic to convert from the 12 characters in the compressed string back to Lat/Long is a reversal of the encoding process and is trivial in a high level programming language or a spreadsheet.

Characters are taken in turn, converted to a number in the range 0 – 24 and progressively added and multiplied by 25 to get the two numbers X_{LAT} and X_{LONG} (remembering to correct the first one in each case for the N/S or E/W direction indicator flag). However, the process is tedious to do manually, even if using copy-and-paste into a spreadsheet. My *SiteCalcWin* locator conversion software [4] has been modified to recognise the 13 character *LocBcn* string in its input window and this will immediately convert to Lat/Long, IARU Locator and NGR, but still requires a manual copy and paste operation

So how can the decoding process be automated? While it is running the WSJT software generates two text files in its working directory. One '*ALL.TXT*' contains everything it has ever decoded or received, (or at lest everything since the file was cleared, using the menu item). For post processing this is useful as a convenient place to download all the received messages rather than copying them from the user screen (highlight, then [ctl]-C to copy). But they still need to be converted.

A utility called ***ReadLocBcn.exe*** will scan *ALL.TXT* and automatically recognise a properly formatted string in the correct place with its hyphen in the middle. This is then extracted and decoded. The resulting Lat/Long shown on screen is also converted to National Grid Reference [5] so the results can easily be seen on an OS map. This allowed a check of the software's operation. The output from the software with the results from the test drive can be seen in Table 2.

Table 2 Decoded LocBcn message showing Lat/Long and calculated NGR [5]

UTC	Message	Lat	Long	NGR	S/N	Freq	
1252	DDCXJC-MBTCAG	+50.85767	-1.14793	SU59970687	-18.0dB	-46Hz	1252
1254	DDCPKF-MBRIIS	+50.84938	-1.10245	SU63180599	-15.0dB	-50Hz	1254
1255	DDCFFR-MBQQVD	+50.83878	-1.08526	SU64400482	-17.0dB	-48Hz	1255
1256	DDCDCO-MBPVKI	+50.83657	-1.06397	SU65900459	-17.0dB	-53Hz	1256
1257	DDCBPH-MBOYSN	+50.83501	-1.04140	SU67500444	-17.0dB	-48Hz	1257
1258	DDCIUJ-MBOHEC	+50.84251	-1.02309	SU68770529	-16.0dB	-46Hz	1258
1301	DDEDIP-MBNMEQ	+50.88890	-1.00228	SU70171047	-21.0dB	-42Hz	1301
1302	DDETWS-MBNJFX	+50.90615	-0.99921	SU70361239	-18.0dB	-44Hz	1302
1303	DDFLRT-MBNHXK	+50.92366	-0.99785	SU70431434	-16.0dB	-48Hz	1303
1304	DDGBMB-MBNAMO	+50.93904	-0.99011	SU70951606	-16.0dB	-53Hz	1304
1308	DDIEGP-MBLPQE	+50.99403	-0.95380	SU73412221	-20.0dB	-48Hz	1308
1309	DDIQIE-MBLPRC	+51.00659	-0.95384	SU73392360	-20.0dB	-35Hz	1309
1311	DDISAW-MBMNRY	+51.00837	-0.97783	SU71702378	-19.0dB	-42Hz	1311
1312	DDITRJ-MBNAVN	+51.01010	-0.99048	SU70812396	-17.0dB	-37Hz	1312
1315	DDJFWQ-MBOJFR	+51.02178	-1.02524	SU68362522	-21.0dB	-39Hz	1315
1316	DDJPGK-MBOSSF	+51.03152	-1.03513	SU67652630	-17.0dB	-33Hz	1316
1317	DDJOGN-MBPKWX	+51.03048	-1.05304	SU66402617	-19.0dB	-33Hz	1317
1318	DDJPAF-MBQENC	+51.03126	-1.07242	SU65042624	-19.0dB	-35Hz	1318
1319	DDJIXQ-MBQJFM	+51.02494	-1.07731	SU64702553	-17.0dB	-33Hz	1319
1321	DDIVBK-MBQTVV	+51.01152	-1.08841	SU63942403	-18.0dB	-31Hz	1321
1322	DDIRAS-MBRJAM	+51.00732	-1.10315	SU62922355	-18.0dB	-31Hz	1322
1323	DDIMGM-MBRWHP	+51.00235	-1.11698	SU61952298	-18.0dB	-31Hz	1323
1324	DDIBHK-MBRYQV	+50.99093	-1.11945	SU61792171	-17.0dB	-28Hz	1324
1325	DDHRUD-MBSIVQ	+50.98209	-1.12903	SU61132072	-19.0dB	-33Hz	1325
1326	DDHOKM-MBSPVL	+50.97856	-1.13631	SU60632032	-17.0dB	-28Hz	1326

Another file generated by the WSJT software while running allows decoding of real time data as it arrives. Every new WSJT message that is decoded is stored in a small one line text file '*DECODED.TXT*' By monitoring this at regular intervals, every new message can be intercepted as soon at it appears. There can be problems in computer operating systems with two separate software packages trying to access the same file, but provided this is recognised and coped-with properly, it is permitted. The

programming language I use for this - Power Basic Command Line Complier, PBCC - has specific tags that are used when opening a file for shared access: (*OPEN "DECODED.TXT" FOR INPUT LOCK SHARED AS #1*) It effectively takes control of the file, but allows other software like WSJT to subsequently modify it while running.

The software ***RealTimeRead.exe*** , scans all incoming messages and prints Lat/Long and NGR to screen as they arrive. The output from this is identical in format to that shown in Table 2.

Both utilities including PBCC source code as well as PIC code for driving an LMX2541 synthesizer can be found at [6]

Displaying the Result in Google Earth

The Google Earth software [7] can be used with custom overlays to allow many types of user features to appear on the plot. A path defined in Long/Lat pairs can be drawn, showing the position of each decoded report and the (direct) track between them.

To be able to plot the location and path on a Google Earth display, the necessary data is first written to a KML (Keyhole Markup Language) file which Google Earth reads as an overlay onto its display. KML files share certain similarities with HTL files, but complete details are beyond the scope of this document. They are well documented on the web, and a search will throw up everything needed.

However, to turn Long/Lat pairs into a plotted line is actually quite straightforward. Most of the KML file contents are standard header information and all that is necessary is to insert list of coordinates between specified markers. The format of the coordinates is important, however. They have to appear as Long,Lat (not in the order Lat / Long more-often quoted), and in decimal degrees with no spaces. Eg.

-1.16948,+50.86348
-1.14793,+50.85767
-1.10245,+50.84938
-1.08526,+50.83878 An example .KML file appears in [6]

To generate the plot, the .KML file is then Opened Using Google Earth. If Google Earth is properly installed on your PC, this file type will be associated with the software and it should open automatically when the .KML file is double-clicked.

The Utility ***GenKML.EXE*** with its .BAS source code can be found in [6]. This software combines the capture facilities of the two utilities described, taking in either real time decodes via *DECODED.TXT* or scan the entire contents of *ALL.TXT*. In each case, every valid position is written to a .KML file in the correct format. An automatically generated file name consisting of date and time is offered as a default, or any other name can be chosen for the .KML file. In real time capture mode, the programme is terminated by pressing the [esc] key.

Figure 3 shows a typical plot using a .KML overlay generated with the ***GenKML*** software using data transmitted on 144MHz from a local drive-around.

JT9 Modulation , WSJT-X and UDP messages

A recent addition to the WSJT Suite is JT9, a nine tone, narrow spaced (1.73Hz) mode intended for LF through HF operation that is only available in the new WSJT-X package. The source coding and the first stage of the convolutional encoding structure for JT9 is identical to that of JT4, so the same compression techniques can be used for *LOCBCN* encoding. The occupied bandwidth of JT9 is around 15.6Hz, similar to that of the narrowest JT4A, but its signal bandwidth is lower at 1.73Hz instead of the 4.4Hz of JT4. Therefore, JT9 should offer an improvement of 4dB in weak signal performance.

PIC code has been written to control an AD9850 or AD9851 DDS to transmit location information. This is included within [6].

The WSJT-X suite requires a different method of accessing the decoded data. This new package broadcasts its decodes and status as UDP messages designed for intercept by any other software running on the same machine, or over a network. This considerably simplifies the techniques needed to extract the decodes, removing the need to work with files open from two pieces of software simultaneously. The LOCBCN software to intercept UDP messages is included within [6] and is transparent to whether JT4 or JT9 modulation is in use.

Conclusions and What Next?

We've seen how the strong error correction and decoding in an existing weak signal data mode can be used to send high resolution positioning information from a GPS receiver. Existing beacon generator hardware can be used unmodified apart from a change to the PIC firmware. Decoding of the messages is possible in real time with presentation of the results in an easily read form. Inserting the coordinates into a .KML file allows the track to be shown on a Google Earth plot

Other modulations included within the WSJT suite are also useable. If the Reed Solomon FEC of JT65 could be cracked, there is another perhaps 3dB of weak signal performance. FSK441 and ISCAT could be used for strong signal uses, like high altitude balloons, but the lack of FEC would necessitate some very reliable validation routines to be included within the decoder. Perhaps a better encoding scheme could be used here, with customised source error correction.

WSPR offers some possibilities with its still weaker signal performance. Appendix A suggests some ways Lat/Long information could be encoded onto a WSPR message.

JT4 has a variety of submodes – different tone shifts – that make it suitable for use on all bands from LF to SHF. JT4D was chosen for 144MHz as, with its 35Hz tone spacing, it could cope with Doppler

shift from driving at 70mph towards or away from the receiver (70mph ~ 30m/s, or 14Hz shift). But at LF to HF, with a slower moving transmitter, JT4A with its 4.4Hz tone spacing would be ideal.

References

[1] WSJT Software and documentation http://physics.princeton.edu/pulsar/K1JT/

[2] JT4 Coding process http://www.g4jnt.com/Coding/JT4_Coding_Process.pdf

[3] DL0SHF Beacon http://www.g4jnt.com/DownLoad/BeaconSource_Draft.pdf

[4] *SiteCalcWin* locator conversion and distance / bearing calculation. Now includes *LocBcn* format strings http://www.g4jnt.com/SitecalcWin.zip

[5] NGR Conversion

In the days before GPS, conversion of Lat/Long to NGR was reasonably straightforward so long as the OSGB-36 spheroid, specific to the UK, was used for positioning. GPS, however, universally uses the world-wide WGS-84 spheroid which is slightly different to OSGB-36. If the values from that are converted to NGR using the same equations as for OSGB-36, errors in the conversion can be as high as 100 – 200 metres.

This simplistic solution is used in *SitecalcWin* and the two LocBcn display utilities, so be aware that although Lat/Long will be accurate; NGR will be out. This error is conversion is visible if you plot the NGRs shown in Table 2 on a 1:50000 OS map. The entire journey was made on main roads – mainly the M27, A3(M), A272 and A32 loop - but the plotted NGRs are clearly off to one side or other of the road. There is a conversion for WGS-84 to NGR, but it is very much more complicated than before, and I have not pursued it. Others have.

[6] All software and hardware details http://www.g4jnt.com/LocBcn.zip

[7] Google Earth software https://www.google.co.uk/intl/en_uk/earth/

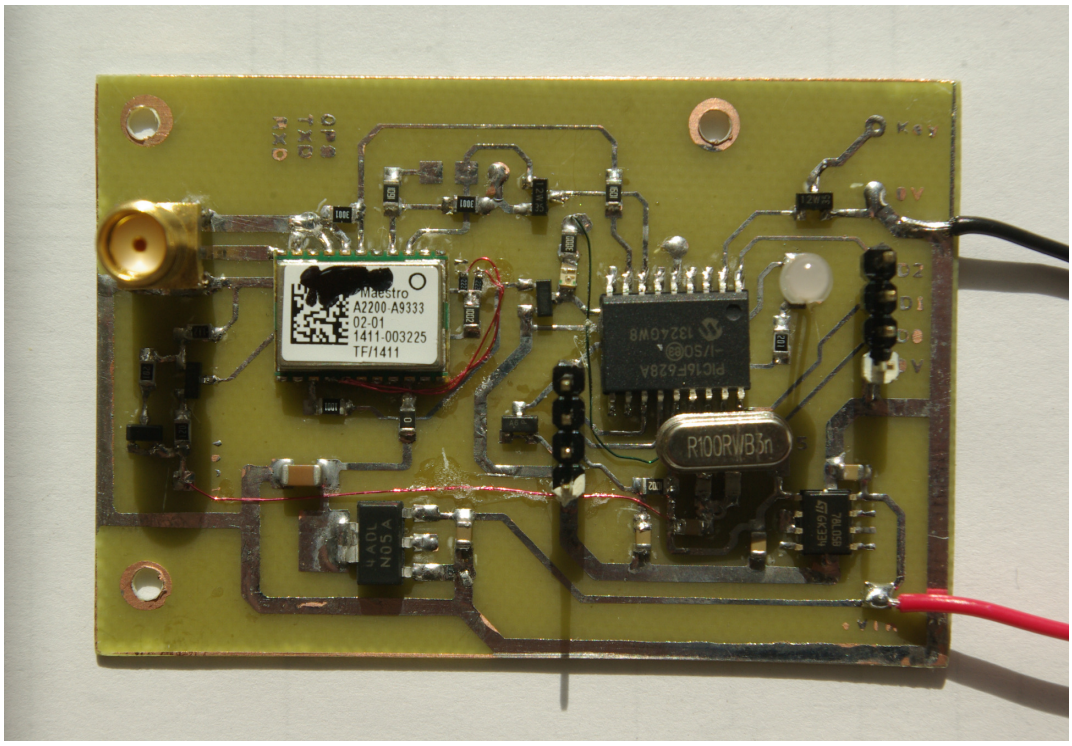

Figure 1 Maestro GPS module and PIC microcontroller. The header on the right hand side sends SPI programming data to one of many types of synthesizer chip

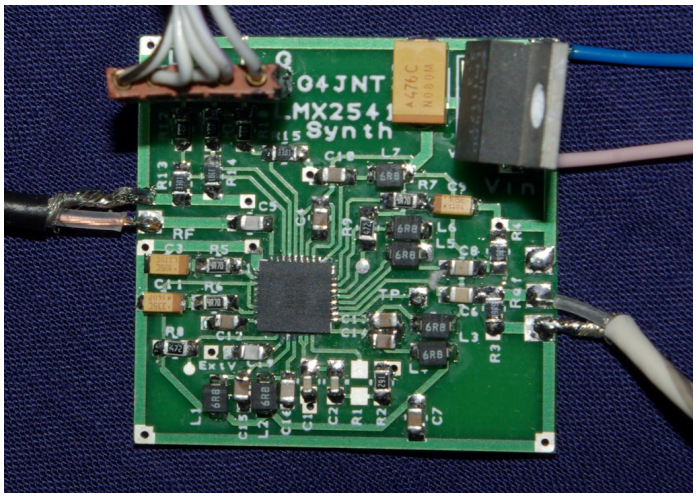

Figure 2 LMX2541 Fractional-N Synthesizer. This will generate multi-frequency MFSK data modes directly when driven from the controller shown in

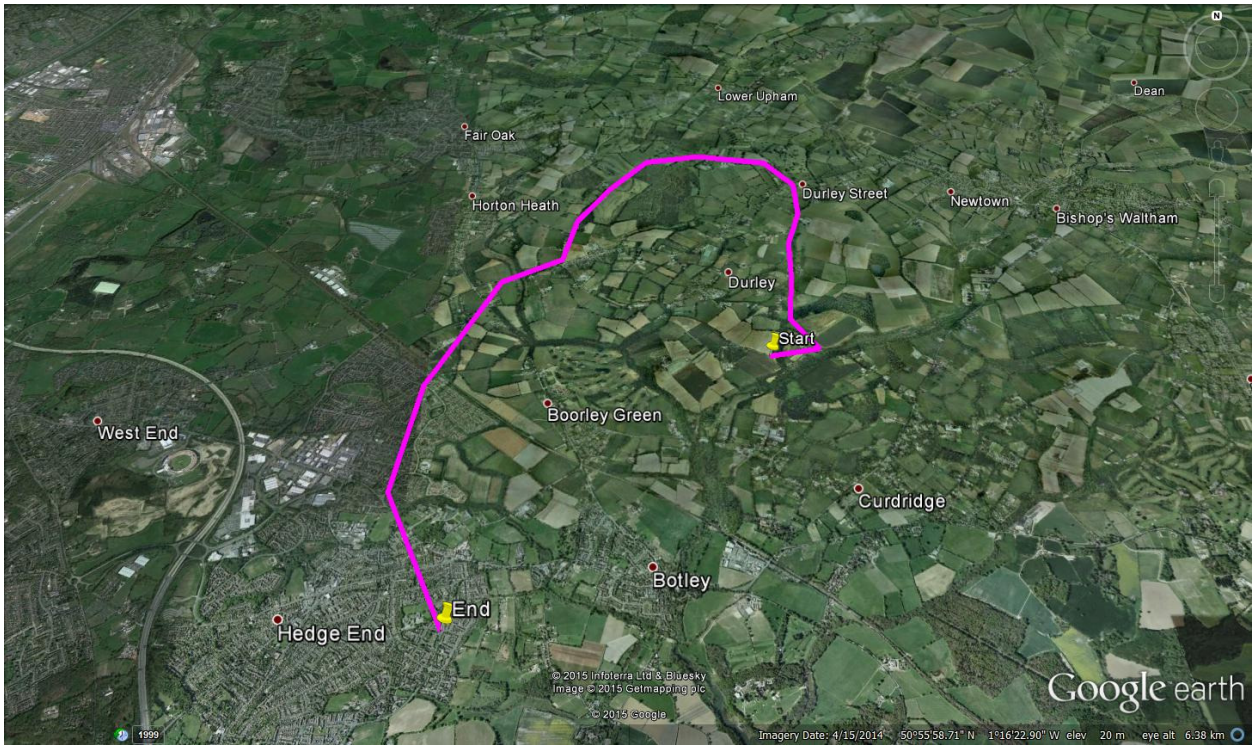

Figure 1 Resulting Plot using a .KML file generated automatically from decoded Long/Lat data

Appendix - **Using WSPR instead.**

The WSPR protocol is just asking to be pressed into service with the further weak signal advantage this mode allows. The modulation is very similar to JT4 with its convolutional encoding but the message structure is very different and customised. The WSPR message carries a compressed callsign, a four digit locator and a power level in a burst lasting a little under two minutes. The Tx power level is a value from 0 to 60dBm but unfortunately only 20 of the possible values are permitted; illegal ones flag as an error. The locator already specifies Lat / Long to a 1° x 2° 'squaroid' so all that remains is to, somehow, encode the higher resolution residual into the callsign and power fields in such as way that the WSPR decoding software will see them as valid data and not flag as an error. This is not going to be easy!

The minutes of latitude, and residual 1 degree plus the minutes of longitude generate numbers that range to 600000 and 1200000 respectively. The callsign field of WSPR is custom-coded as follows :

A maximum of six characters consisting only of A-Z, 0-9 and [space] are allowed. The third character is forced to be always a number. To cope with callsigns that start with a letter followed by a number, a space is appended to the front if necessary. So, for example, 'G4JNT' will become '[sp]G4JNT' whereas GD4JNT stays as-is. Short callsigns are then further padded out to six characters by appending spaces to the end. The 37 allowed characters are allocated values from 0 to 36 such that '0' – '9' give 0 – 9, 'A' to 'Z' give 10 to 35 and [space] is given the value 36. Further coding rules on callsigns mean that the final three characters (of the now padded out callsign) can only be letters or [sp] so can only take the values 10 – 36.

With all those characters taking on values from 0 to 36, the callsign is now compressed into a single integer N by successively building up for an absolute maximum value for N of 37 * 36 * 10 * 27 * 27 * 27 = 262177560. (The complex compression was adopted since the resulting number now fits very efficiently into 28 bits) . The value is sufficient to hold one of the Lat/Long residuals, but not both of them. Even multiplying by 20 to include all possible values of the power level field is not possible

The absolute maximum value that could be included within a WSPR message is therefore only 5243551200. This is approximately $2 * 51000^2$ and now dictates the maximum Lat / Long resolution that can be compressed into something that can be sent as a WSPR message.

The latitude residual contains 60.0000 minutes (the longitude is twice this) so if a resolution of just 0.0012 minutes were adopted (about 2 metre positioning accuracy) to give a number in the range 0 to 50000 (0 to 100000 for longitude) the result could be squeezed into the WSPR callsign and power level fields. The arithmetic is horribly messy and inelegant, but is still just multiplication and division. A project left still to do...

Pictures

Photo 1	A2200_Module.JPG
Photo 2	IMGP9366.JPG
Photo 3	LocBcnTestRun3.jpg