NEW GENERATION BEACON DRIVER

As used on the new GB3VHF 144.43MHz Beacon

Andy Talbot G4JNT 2005/11/13 Updated Appendix B and C Jan 2011

The beacon driver described here has been designed to form the basis of a new '21st century beacon'. GPS derived timing and frequency locking techniques have been combined with modern data-modes that can be received using the now ubiquitous radio plus PC/soundcard combination.

The heart of the beacon driver is an Analog Devices AD9852 Direct Digital Synthesiser, a versatile device which can directly generate frequencies up to approximately 100MHz with a setting resolution of a fraction of a micro-Hz. Amplitude and phase modulation capabilities are built into the chip and these, together with instantaneous frequency, can be altered at will, in real time, to generate complex multi-dimensional modulation schemes.

Modulation types.

Five programmable modes have been chosen for this driver design, but bear in mind that many others can be added by changing software that controls the DDS. For example, at HF and Low VHF, a beacon transmitting PSK31 messages could be made from exactly the same hardware, just by re-programming the PIC.

CW All amateur beacons are still required to give their identification in Morse for aural reception, so this is obviously the first data type that has to be included. One of up to four CW different messages can be sent at different times. The CW speed can be set to any desired value. The CW carrier is generated with a rise and fall time of approximately 1ms (fixed) so that when used with a following linear amplifier, key clicks are minimised.

RTTY Although stop-start FSK signalling can now probably be considered to be an old fashioned mode and not terribly efficient when compared with later data modes such as PSK31, it does have a number of advantages over these when used at higher frequencies. The most significant is that RTTY can be used with non-linear transmitters, particularly for high VHF through UHF uses where the DDS output has to undergo further frequency multiplication to reach the final output frequency. For decoding the message , there is plenty of RTTY decoding software around, and still has a large following of users. RTTY transmission will not be included within the GB3VHF sequence.

JT65 This is the primary data mode for the beacon driver, and JT65 is the most recent of the WSJT modes invented by Joe Taylor, K1JT. The WSJT family of data modes have been designed for different VHF/UHF propagation types (EME, Tropo, Meteor Scatter), and JT65 has been designed specifically for weak signal, EME and terrestrial propagation. Three variants of JT65 are in use, A, B and C, each having different bandwidths and tone spacing. Any one of the three can be used in this beacon controller. GB3VHF will initially adopt the middle one, JT65B, with a tone spacing of 5.38Hz and an overall transmission bandwidth of 350Hz. It is the incorporation of this transmission type that determines the timing concept for the whole beacon controller.

BPSK An experimental Binary Phase Shift Keyed transmission is included in the controller software. At a precisely timed point, 140 microseconds after the UTC one-second reference as signalled by the GPS receiver, the phase of the transmission is reversed, 28 times in total to fill up the 30 second time slot. The result is a 1 bit/second pattern of 10101010….. The BPSK mode has been incorporated to allow users to become familiar with using precise timing methods to assist in experimentation with coherent signal recovery, to measure time of flight information and propagation testing. To allow for frequency multiplication to reach the final frequency on 144MHz and higher frequency beacons, the actual phase shift generated within the DDS source can be set to any value in the PIC setup code.

Plain Carrier

Has been included as a programmable mode in case extended periods of carrier are wanted for specialist automated weak signal monitoring, eg. in propagation monitoring. For normal operation, the requirement for a period of unmodulated carrier can usually be met by the idle periods at the end of CW and/or RTTY transmissions, but the option of a dedicated 30 second slot allows for extended periods. Nb. any future automatic beacon monitoring software will probably rely on there being at least a few seconds of unmodulated carrier from any beacon to allow accurate frequency and power determination – and it is suggested that this period be made at least 10 seconds long for all VHF and Microwave beacons, 5 seconds for HF.

Message Structure

In normal usage JT65 relies on automatic timing control, making use of a one minute transmit / one minute receive sequence which has to be replicated in the receiver's computer. The timing is allowed to be in error by up to four seconds late or one second early when referred to the transmission. The rest of the beacon's timing is based on a thirty second time-slot structure which determines the data transmission pattern for the whole beacon sequence. As the JT65 transmissions must be aligned with the minute, timeslots must be set and allocated accordingly. Time information sent from the GPS receiver is used to derive a number of 30 second time slots, referenced to the hour, and any even number of up to 16 slots can be programmed with different data formats allocated to each 30s slot. Changes are made by modifying the PIC code for the beacon controller. For a typical transmission sequence, JT65 will be allocated to every fourth slot (starting at an even number), to give the one minute + one minute cycle of transmission periods. As a JT65 transmission period lasts 48 seconds it will run into the next 30s time, so this one cannot be allocated to any other mode. The rest of the slots can be allocated to CW, Carrier, BPSK or RTTY as needed. To avoid sequence breaks on the hour, the number of slots used should ideally be a submultiple of 120, so values of 2,4,6,8,10,12 are the most convenient, corresponding to 1 to 6 minute repeat cycles. The number of minutes for the cycle is specified in the setup information.

Any CW or RTTY transmission will be of arbitrary length, and the remaining time up to the end of their timeslot is automatically taken up with plain carrier. If a CW or RTTY data exceeds 30s in length, the data for the following timeslot will be ignored.

Although the second 30 second slot after a JT65 transmission is ignored, it should be defined in the sequence definition. This is because JT65 is not transmitted when the GPS receiver has lost lock. The first time slot allocated to JT65 defaults, instead, to the first CW message and the second time slot is as defined in the setup data. When GPS lock is restored, JT65 operation resumes and the second time slot definition is then ignored.

As implemented on GB3VHF the sequence programmed is :

00:00	JT65 sending the message "GB3VHF JO01DH"
00:30	CW message as below (only sent during Invalid GPS)
01:00	CW Message "GB3VHF JO01DH" (approx 13 seconds) followed by plain carrier
01:30	Phase reversals at one second intervals.

Repeated every two minutes

As far as a listener is concerned, starting on the even minute boundary, this appears as :

48 seconds of JT65 transmission
12 seconds of plain carrier
13 seconds of CW Ident
17 seconds of plain carrier
30 seconds of phase reversals
- Sequence repeats

DDS and Controller Hardware

The driver unit comprises several modules working together under the control of a single 16F628 PIC microcontroller. The DDS is clocked from a reference input, and a Phase Locked Loop on board the AD9852 multiplies this by a programmed value from 4 to 20, for a maximum allowed clock frequency of 300MHz. In this VHF-band beacon driver, a reference input of 12.8MHz is multiplied by 16 in the PLL for a 204.8MHz clock. Keeping the clock below the 300MHz maximum permitted in the AD9852 specification results in a lower power consumption and improves reliability of the DDS chip. The 12.8MHz reference oscillator is derived from a Voltage Controlled, Temperature Compensated Crystal Oscillator (VTCXO) which in turn is phase locked to the 10kHz output from a Jupiter TU-60 GPS receiver module.

The microcontroller, mounted on the same PCB as the DDS chip, communicates with the GPS receiver in order to read the time of day (UTC) and the GPS receiver / satellite status information. The 1 Pulse Per Second (1PPS) signal generated by the GPS receiver is fed to the PIC in order to generate accurate timing – the 1PPS signal is typically accurate to within 200ns of UTC. The PIC uses the data from these two signals to accurately set the start and stop times of the various modulations transmitted.

By writing suitable software for the PIC, a wide range of modulation types can be generated by reprogramming the DDS chip with new frequency, amplitude or phase information at regular intervals.

Frequency Reference

The TCXO output is divided by 1280 to give a 10kHz output. An exclusive–OR gate compares this with the 10kHz output from the GPS receiver to generate a feedback signal for the Phase Locked Loop. An analogue switch controlled from the PIC opens the feedback loop if the GPS receiver is not locked up, forcing the TCXO to run at a fixed frequency. This prevents gross frequency errors as a result of a free-running GPS receiver. The PIC continuously monitors the receiver status and controls this analogue switch when the data from the GPS receiver indicates that the status is satisfactory, ie. that it has locked up. The phase-locked loop has a time constant of 4 seconds, so there is a noticeable 'blip' as this control signal switches to its active state and the PLL begins to lock up. Frequency stability will be achieved about 30 seconds to 1 minute after the GPS receiver has locked up.

Driver Hardware and Construction

The three PCBs making up the assembly are mounted in a single diecast box. The DDS and PIC assembly is fixed above the GPS receiver and frequency reference boards to give easy access to the PIC for reprogramming. Two voltage regulators, 3.3V for the DDS and 5V for the GPS receiver and divider assembly, are mounted on the side of the box with an additional heatsink on the outside. (The PIC on the DDS board has its own low power 5V regulator). The DDS chip consumes around 650mA during operation and runs quite hot. Its main heatsinking mechanism normally consists of conducting heat away from the base of the chip through the PCB, but here no air flow is possible inside the diecast box so additional chip cooling has to be provided. A sprung copper strip conducts heat away from the top of the DDS chip to the side of the diecast box, the strip presses onto the top of the AD9852, with thermal grease added to ensure heat conductivity. At a room temperature of 20C, the voltage regulator heatsink sits at around 34C, the black painted surface of the diecast box at 28C, and the chip is estimated to sit at around 40C - without the copper strip, it was closer to 55C. When the diecast box is mounted on a metal baseplate giving further conduction cooling, all the temperatures drop proportionately.

Interfacing and Connections

Five signal / power connections are provided on feedthrough capacitors:

DC POWER — Requires a voltage greater than 8V and internal linear regulators take this down to 5 / 3.3V. Current consumption is approximately 0.8 to 0.9A. To avoid excessive heat generation, avoid using a supply voltage much greater than 12V – use an external dropper resistor if necessary

NO GPS — Goes high when the GPS receiver has not acquired lock; it is pulled to +5V via a 2k2 resistor and can be used to directly drive an LED. A high brightness type is advisable due to the limited current available from this pin.

RED LED + GRN LED — These are intended to drive a bi-directional (antiphase connected) two-colour LED *'CONTROLLER STATUS'*. The pattern of flashing / colours on this is determined in software and used to indicate the status of the DDS controller software. Each pin is driven from a 5V source/sink via a 150 ohm resistor. The positive side of each colour goes to the pin labelled.

TEST MODE — is an input pin which is left open-circuit for normal operation. When shorted to ground, the controller enters carrier-only mode as soon as the current transmission interval has completed and stays there until the short is removed. Its function is to allow for frequency setting / measurement, and for testing the additional RF hardware.

Bi-Colour LED Flashing Codes

As implemented, the LED flash codes are :

When power is first applied to the driver, the LED flashes three times Red/Green alternately each second.

Before the GPS receiver has locked up, or if the GPS receiver loses signal:

The Green LED gives a short flash once per second while carrier-only is being transmitted.
The Red LED flashes in sympathy with the CW characters when CW is being sent
Alternate Red/Green flashes for the BPSK mode, once per second corresponding to each phase state

When lockup has been achieved and the beacon is operating normally :

The Green LED gives a long flash once per second while carrier-only is being transmitted.
The Red LED flashes in sympathy with the CW characters when CW is being transmitted.
Alternate Red/Green flashes for the BPSK mode, once per second corresponding to each phase state
For the JT65 transmission, the LED flashes Red at the JT65 symbol rate of 2.7Hz

---

Frequency Doubler and Output Buffer.

At the 204.8MHz clock frequency of the DDS chip, the maximum frequency that can be generated by the AD9852 chip is around 80MHz. A 5th order elliptic filter on the PIC output, mounted on the DDS board, reduces alias and harmonic products to at least 50dBc A frequency doubler built in a separate module is used to generate the final 144MHz output,

The output from the DDS board is amplified to a level of approximately 11dBm and feeds a push-pull diode doubler circuit. This provides 40dB rejection of the fundamental and is followed by a two section bandpass filter that ensures all spurii are at a level of at least 55 dBc. Two stages of amplification raise the filter output to a level of 12.5dBm with the final modamp running at 1 - 2dB of gain compression. The buffer has no harmonic filtering on its output so if the doubler is to be followed by broadband amplification, extra harmonic filtering will be needed.

All amplifier stages consist of MAR-3 and MAR-4 modamps supplied from constant current regulators to give immunity from supply voltage and temperature variations. Any supply voltage from 9 to 13V is acceptable. The doubler / buffer assembly is constructed in a screened tinplate box with SMC connectors for the RF, and a feedthrough capacitor for the supply voltage.

User Programming

Frequency information (the codes that need to be sent to the DDS chip) together with the text of the various messages are stored in the EEProm memory area of the PIC. The boundaries of the various messages need to be known by the assembler, so it is not possible to reprogramme the EEProm contents alone if the length of any of the messages have been changed. In general it is safer to re-assemble the whole file and re-programme the entire PIC device – this should take less than one minute for most PIC programmers. Any change to the message sequence, or time slot allocation, must involve a complete re-programming of the PIC.

Frequency and message contents are contained in a separate include file (*VHFBCNE2.INC*) which can be edited on its own. A total of 128 bytes of storage is available, and of this the first 30 bytes are used for frequency data, leaving 98 bytes for storage of all the characters needed for all the messages.

The frequency information (DDS codes) can conveniently be generated from an accessory programme called *VHFBCNCALCS.EXE*. See Annex A for an example of this utility. The programme generates a file which includes lines of PIC code that can be directly cut and pasted into *VHFBCNE2.INC*. See the example below. The include file also has other constants used within the programme, such as *RPTTime* - the number of minutes (pairs of time slots) for the transmission sequence; *CWSPEED* (the CW dot length in milliseconds) for the CW messages; and the RTTY baud rate defined in units of 500µs. It is convenient to let the assembler calculate this value for you, by dividing 2000 by a specified baud rate. The constant *NMEAPol*, shown in the example, is not applicable to this software, but is needed in other PIC software that makes use of the same include file.

Note that the DDS controller software here does not make use of the carrier frequency itself. Instead, the following values are stored in EEProm to produce the required frequencies from the specified clock:

The absolute JT65 Sync frequency	Generates a tone of 11025 / 1024 * 118 ≈ 1270.5Hz in an SSB receiver tuned to the specified carrier frequency.
The increment between tones	11025/4086 * M where M is 1 for JT65A, 2 for JT65B and 4 for JT65C. For JT65B the increment is approximately 5.38Hz The PIC software then calculates the frequency data to be sent to the DDS by multiplying this value of increment by a number between 2 and 65

	depending on the character to be sent, then adding the result to the value for the sync tone.
The Centre frequency for CW and carrier	Corresponds to a tone frequency as shown - an 1500Hz value is shown in *VHFBCNE2.INC* below).
RTTY Mark	1275 Hz Tone
RTTY Space	1445 Hz Tone

If the DDS output is subsequently frequency-multiplied, all the values are modified by the multiplication factor.
Frequency codes are stored in EEPROM in the order high byte to low byte so the full numerical value can more easily be seen from the individual data in a listing.

As an example, take the sync tone which has a hexadecimal value of 0x5A44C365E354
This value can be converted back to the actual frequencies, as follows:

Convert hex number to decimal	N	=	99251382510420 $_{(10)}$
Divide by 2^{48}	$N / 2^{48}$	=	0.352611744
Multiply by the DDS clock frequency	$N / 2^{48}$ * 204.8MHz	=	72.21488523 MHz
and finally by the RF multiplication	… * 2	=	144.42977046 MHz
ie. 1270.46Hz above the suppressed carrier at 144.4285MHz			

All message contents are stored in the 128 bytes of user EEprom. Text messages must be terminated with a null character to indicate their end, but as the pre-coded data for the JT65 transmission take up exactly 63 spaces, it is therefore self terminating. The various labels of the form `JTMSGOFFSET = $ - 0x2100` are an essential part of the software and must be correctly placed immediately before the start of each character string. If EEProm storage becomes full, it is allowable to reuse strings between different messages. In this event, either the main body of the assembler code needs to be changed so the data-sending subroutines point to the appropriate data label, or more conveniently, both labels can be placed at the same point.

The assembler include file, *VHFBCNG2.INC* , contains the allocation of message type to timeslot. Up to 16 GOTO statements cause the PIC code to jump to the appropriate section of code for each time slot. Although only sufficient GOTOs need to be included to cope with the value of *RPTTime* defined in *VHFBCNE2.INC*, it is safer to include sufficient jumps to cope with the maximum number of slots that could conceivably be allocated – setting an incorrect value for RPTTime will cause havoc with the PIC code if a jump is not in the correct place! Any mistakes or typing errors made when changing this file will usually (hopefully !) show as errors when assembled. Error messages will not necessarily be generated if incorrect values appear in the ..E2 file, as the assembler cannot identify many numerical and memory allocation errors.

Typical contents of VHFBCNE2.INC

```
    NMEAPol     =   0               ;1 for RS232 levels,   0 for Logic
    RPTTime     =   2               ;Sequence repeat interval, minutes
    CWSPEED     =   d'70'           ;CW Dot length, ms
    BAUDCONST   =   d'2000'/d'50'   ;RTTY baud interval,  Units of 0.5ms
    JT65MODE    =   1               ;A=0, B = 1, C = 2
    PHASE180    =   0x1000          ;AD9852 Phase reg for 180 deg shift,
                                    ;  to allow for frequency multiplication
    CONTROLPLL  =   0x50            ;Part of control register to set PLL
                                    ;  clock multiplication
;   __config    0x3F61      ;LV Programming off , OSC XT Mode
    __config    0x3F63      ;LV Programming off , OSC EC Mode (RA6 I/O)

    org 0x2100
;AD9852 DDS register data for VHF/UHF beacons running WSJT Modes
    ;Frequency data
    ;Fcarrier = 144.4285 , RF Mult = 2 , Fo(carrier) =  72.21425
    ;Ref in   = 12.8 , PLL Mult= 16 , Clock freq  =  204.8

    de  0x5A,0x44,0xC3,0x65,0xE3,0x54    ;JT65 Sync tone
    de  0,   0,   0x00,0x38,0x72,0xB0    ;JT65 Tone Interval
    de  0x5A,0x44,0xCC,0xCC,0xCC,0xCD    ;1500Hz Tone
    de  0x5A,0x44,0xC3,0x95,0x81,0x06    ;RTTY Mark  1275 Hz
    de  0x5A,0x44,0xCA,0x8C,0x15,0x4D    ;RTTY Space 1445 Hz

        JTDATAOFFSET = $ - 0x2100
;    GB3VHF JO01DH
    de   d'26', d'2' , d'61', d'34', d'15', d'15', d'35', d'50'
    de   d'17', d'50', d'29', d'54', d'47', d'37', d'1' , d'16'
    de   d'19', d'14', d'37', d'43', d'47', d'18', d'41', d'40'
    de   d'39', d'20', d'17', d'63', d'43', d'59', d'8' , d'50'
    de   d'57', d'50', d'12', d'45', d'7' , d'45', d'12', d'18'
    de   d'2' , d'40', d'9' , d'19', d'12', d'33', d'11', d'2'
    de   d'49', d'10', d'40', d'48', d'13', d'54', d'24', d'20'
    de   d'46', d'25', d'24', d'49', d'60', d'21', d'28'

        CWMSG1OFFSET = $ - 0x2100
        CWMSG2OFFSET = $ - 0x2100
        de      "GB3VHF JO01DH",0
        RTTYMSGOFFSET = $ - 0x2100
        de "GB3VHF JO01DH", 0

;maximum total EEPROM data length 128 bytes
```

Typical contents of VHFBCNG2.INC (nb. RptTime = 2 defined in VHFBCNE2.INC)

```
;Setup the beacon sequence format
;GOTO segment of code for handling each 30 second ttransmission interval
; note, JT65 needs 47 seconds so interval afterwards is ignored
;Options are CarrierOnly, SendJT65, SendRTTY, SendBPSK, SendCWx (x = 1,2,3,4)
;Ensure null-terminated message data is present in EE for all messages used
;Ensure all the labels used are in their correct places

        goto    SendJT65        ;  0:00 - 0:30
        goto    SendCW1         ;  This line is ignored, except at boot-up
        goto    SendCW1         ;      and for loss of GPS signal
        goto    SendBPSK        ;

        goto    SendCW1         ;
        goto    SendCW1         ;
        goto    SendCW1         ;
        goto    SendCW1         ;

        goto    SendCW1         ;
        goto    SendCW1         ;
        goto    SendCW1         ;
        goto    SendCW1         ;

        goto    SendCW1         ;
        goto    SendCW1         ;
        goto    SendCW1         ;
        goto    SendCW1         ;
```

Explanation of PIC assembler codes:

Any text on a line after a semicolon is a comment and plays no part in the code itself.

GOTO statements cause the programme flow to jump to a specified location. This is nearly always defined by a label elsewhere in the assembly code which automatically allocates a numerical address to it. In the sequence definition table, *VHFBCNG2.INC*, labels such as SendCW1, SendJT65 have been used to make clear the functions to be performed. These label locations and the routines are defined in the main body of the code, and in turn call up the data defined in *VHFBCNE2.INC* by the labels CWMSG1OFFSET and JTMSGOFFSET

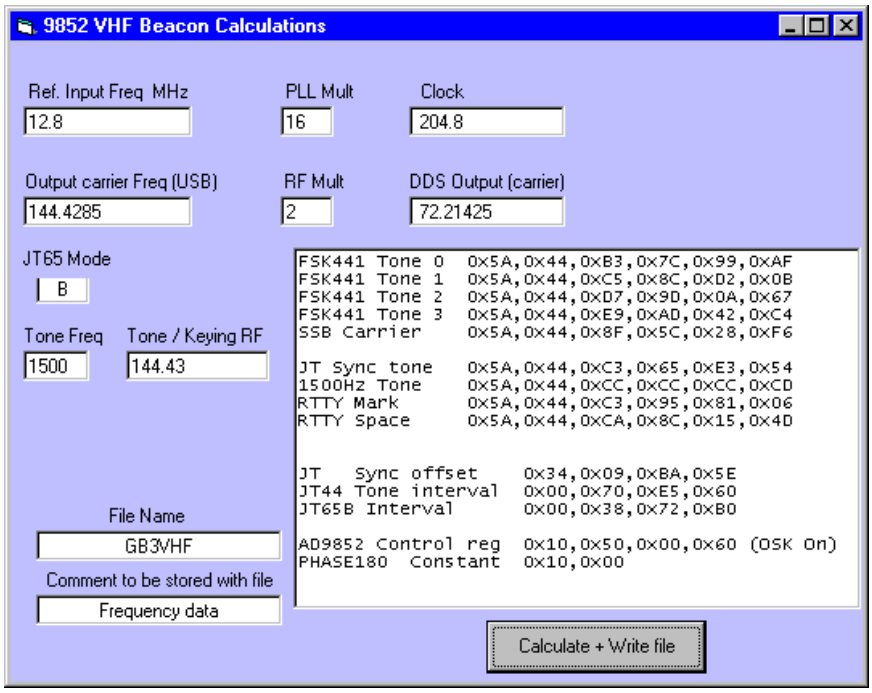
Statements beginning **de** are defining data that is stored internally. This can be in Hexadecimal (shown as 0x3F, for example), decimal eg d'99' or as text characters between pairs of inverted commas, stored in successive memory locations.

Annex A VHFBcnCalcs.exe

Calculation of frequency codes.

The Windows software *VHFBcnCalcs.exe* allows a user to enter the DDS reference input frequency, PLL clock-multiplier, the RF carrier frequency, a tone for the centre (CW) keying and the RF multiplication after the DDS output. The software uses this data to calculate the frequency codes needed for the beacon driver; the user window is shown below. All codes are given in hexadecimal, preceded with 0x. If a file name is specified, the data will be written to this file in a format suitable for cutting and pasting the lines of data directly into the PIC assembler code. The contents of a such an output file is shown below; compare the second part of this with the contents of *VHFBCNE2.INC* shown earlier.

As well as the JT65 frequency codes (specifically, the sync tone and increment value) the software gives the values for the CW / continuous carrier frequency, and the RTTY mark and space frequencies corresponding to the standard 1275/1445Hz tone shift. Although not implemented in this beacon driver, the four frequency codes corresponding to each of the equivalent FSK441 tones is also calculated for use in other beacon controller software (such as *FSK441BC.ASM*) as well as the SSB carrier value itself. The values for the DDS control register are derived from the PLL multiplication factor and the clock frequency; this information is needed in the DDS setup code. Finally, the data needed to be programmed into the DDS phase register to obtain a 180 degree phase shift is presented, taking takes into account the final RF multiplication factor.



CONTENTS OF 'GB3VHF' FROM THE '*VHFBcnCalcs' utility.*

```
;AD9852 DDS register data for VHF/UHF beacons running WSJT Modes
; Cut and paste into PIC assembly file

    ;Frequency data
    ;Fcarrier = 144.4285 , RF Mult = 2 , Fo(carrier) =  72.21425
    ;Ref in   = 12.8 , PLL Mult= 16 , Clock freq  =  204.8

    de  0x5A,0x44,0xB3,0x7C,0x99,0xAF    ; FSK441 Tone 0
    de  0x5A,0x44,0xC5,0x8C,0xD2,0x0B    ; FSK441 Tone 1
    de  0x5A,0x44,0xD7,0x9D,0x0A,0x67    ; FSK441 Tone 2
    de  0x5A,0x44,0xE9,0xAD,0x42,0xC4    ; FSK441 Tone 3 Hz
    de  0x5A,0x44,0x8F,0x5C,0x28,0xF6    ; Carrier, Fo

    de  0x5A,0x44,0xC3,0x65,0xE3,0x54    ;JT   Sync tone
    de  0,   0,   0x00,0x70,0xE5,0x60    ;JT44 Tone Interval
    de  0,   0,   0x00,0x38,0x72,0xB0    ;JT65B Tone Interval

    de  0x5A,0x44,0xCC,0xCC,0xCC,0xCD    ;1500Hz Tone
    de  0x5A,0x44,0xC3,0x95,0x81,0x06    ;RTTY Mark  1275 Hz
    de  0x5A,0x44,0xCA,0x8C,0x15,0x4D    ;RTTY Space 1445 Hz

        ;AD9852 Control reg  0x10,0x50,0x00,0x60 (OSK On)
        ;PHASE180  Constant  0x10,0x00
```

Annex B **Generating the data for the JT65 transmission**

JT65CODE.EXE , J65TOPIC.EXE and GENJT65.EXE

Generation of the 63 coded symbols for any JT65 message is far too complex a procedure to be incorporated within the beacon controller itself – involving message compression, interleaving and Reed-Solomon coding to turn the 13 character text into 63 codes corresponding to the transmitted tones. A utility called *JT65CODE.EXE* can be downloaded from the WSJT web site (http://pulsar.princeton.edu/~joe/K1JT/Download.htm), this is needed to generate the symbols that are to be stored in EEPROM. The full source code for the utility can also be found here, and is invaluable for understanding the inner workings of the JT65 coding.

The 63 code words are generated by typing JT65CODE "GB3VHF JO01DH" from a DOS prompt. Replace the text between the inverted commas with your 13 character message of choice. This instruction sends the result to the screen. To get the result into the PIC listing, the data can either be captured from screen with a text capture utility, or the programme output can be sent to a file by the DOS redirection utility (which still works, even under Windows NT) by typing :

JT65CODE "GB3VHF JO01DH" > GB3VHFJT.TXT

The result is sent to the file named. An example of the output from *JT65CODE* is shown below.

```
Message:   gb3vhf jo01dh
Plain text.
Packed message, 6-bit symbols:  24  8 29  9 25 30 11 30 54  8 36
23
Channel symbols, including FEC:
   26  2 61 34 15 15 35 50 17 50 29 54 47 37  1 16 19 14 37 43 47
   18 41 40 39 20 17 63 43 59  8 50 57 50 12 45  7 45 12 18  2 40
    9 19 12 33 11  2 49 10 40 48 13 54 24 20 46 25 24 49 60 21 28
Decoded message: GB3VHF JO01DH
```

The symbols can now either be typed into the PIC assembler listing manually, or a second programme *JT6TOPIC.EXE* (available from G4JNT) can be used to convert the data into a form for direct pasting into the assembler file. To use this utility (assuming the filename as generated above) type :

J65TOPIC GB3VHFJT.TXT

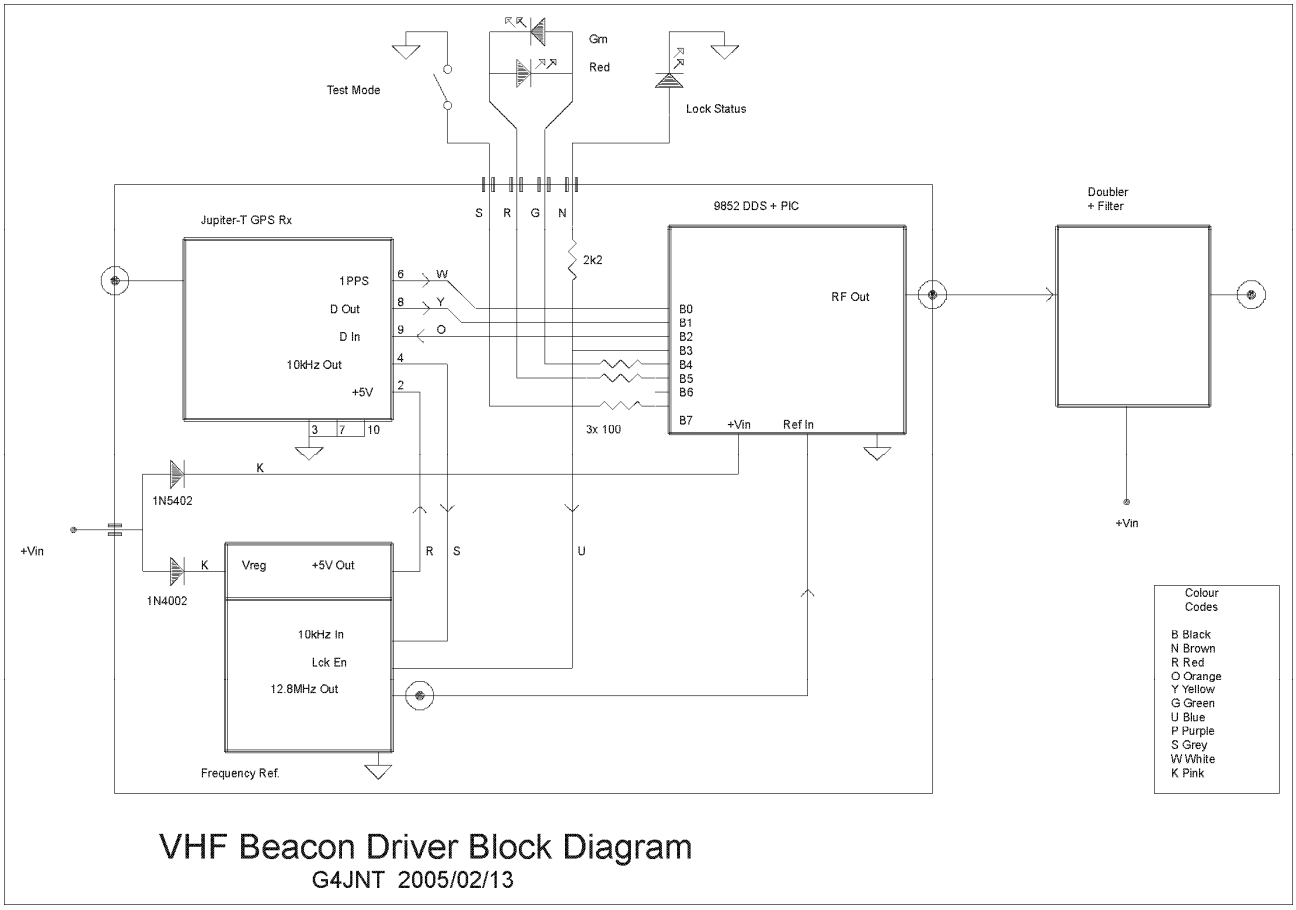
You will be prompted for a destination file name, which will be generated containing the data in a form suitable for pasting into *VHFBCNE2.INC*

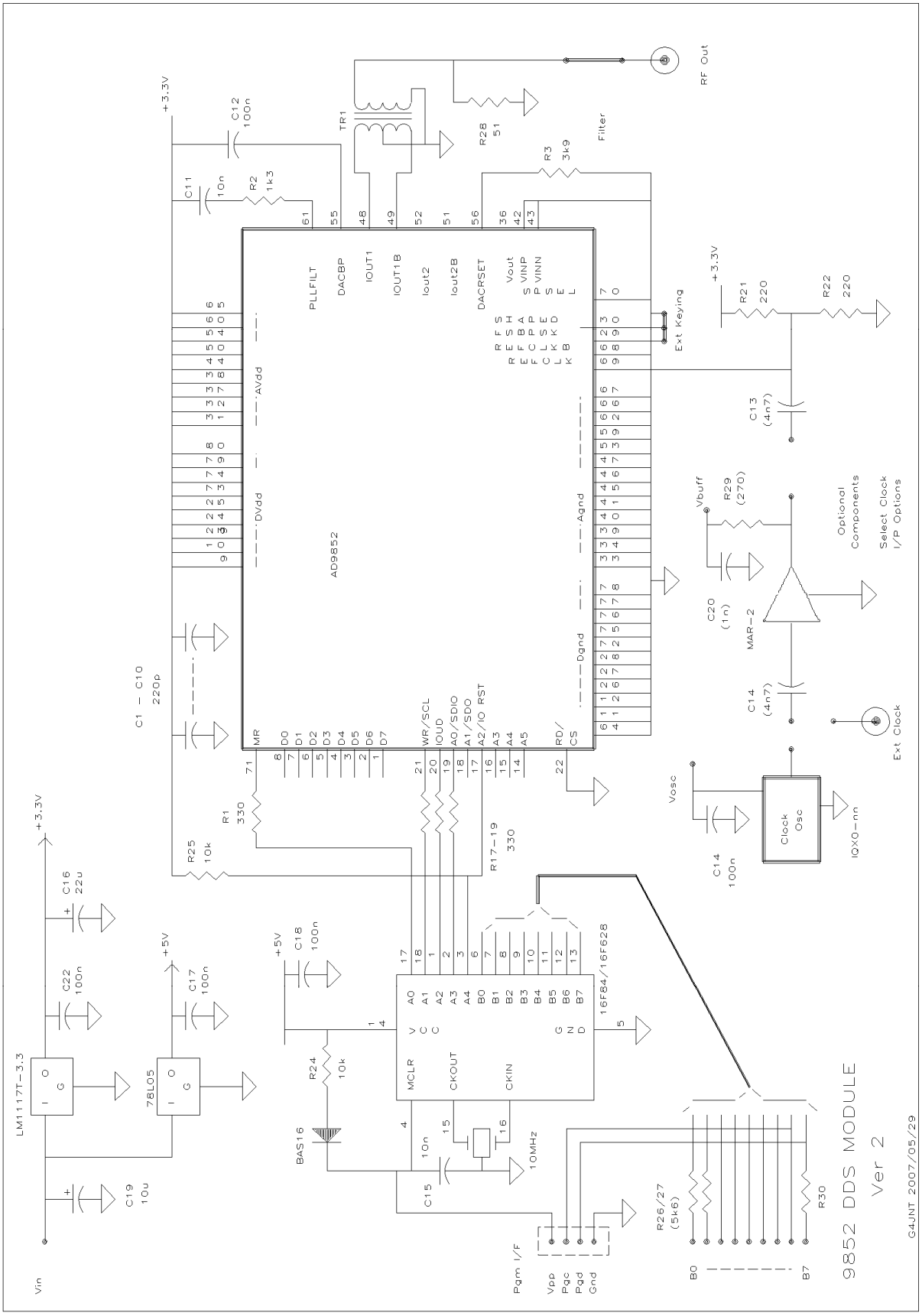
Typical output from *J65TOPIC.EXE*, as stored in the destination file suitable for directly pasting into *VHFBCNE2.INC*

```
    de   d'26', d'2',  d'61', d'34', d'15', d'15', d'35', d'50'
    de   d'17', d'50', d'29', d'54', d'47', d'37', d'1',  d'16'
    de   d'19', d'14', d'37', d'43', d'47', d'18', d'41', d'40'
    de   d'39', d'20', d'17', d'63', d'43', d'59', d'8',  d'50'
    de   d'57', d'50', d'12', d'45', d'7',  d'45', d'12', d'18'
    de   d'2' , d'40', d'9',  d'19', d'12', d'33', d'11', d'2'
    de   d'49', d'10', d'40', d'48', d'13', d'54', d'24', d'20'
    de   d'46', d'25', d'24', d'49', d'60', d'21', d'28'
    ;    GB3VHF JO01DH
```

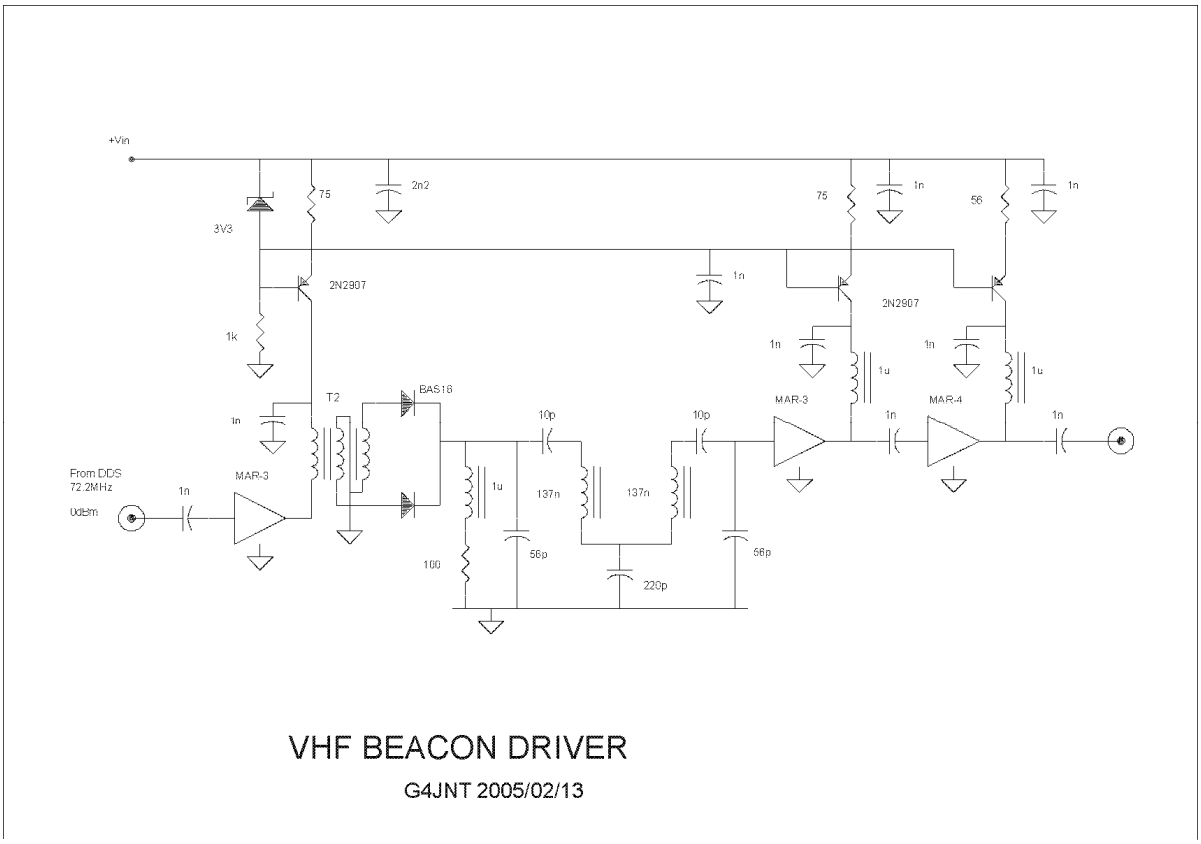
Update Jan 2011
The programme **GENJT65.EXE** combines the two stages by making the formatted call to Joe's JT65CODE and generating the .INC file. Run *GENJT65* and follow the instructions. Please note that you still need JT65CODE.EXE to be present in the same directory as GENJT65.EXE

Annex C Circuit Diagrams





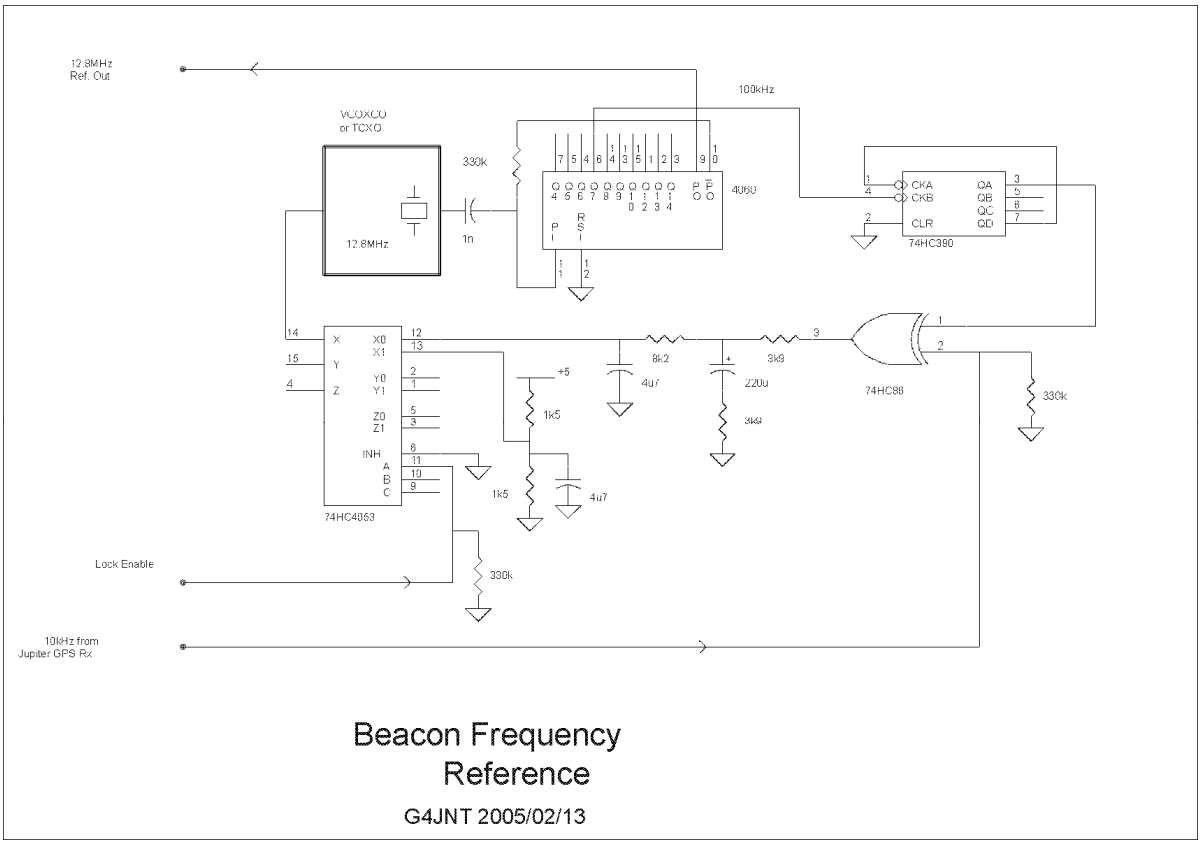
DDS Module Circuit Diagram **(updated Jan 2011)**



VHF BEACON DRIVER

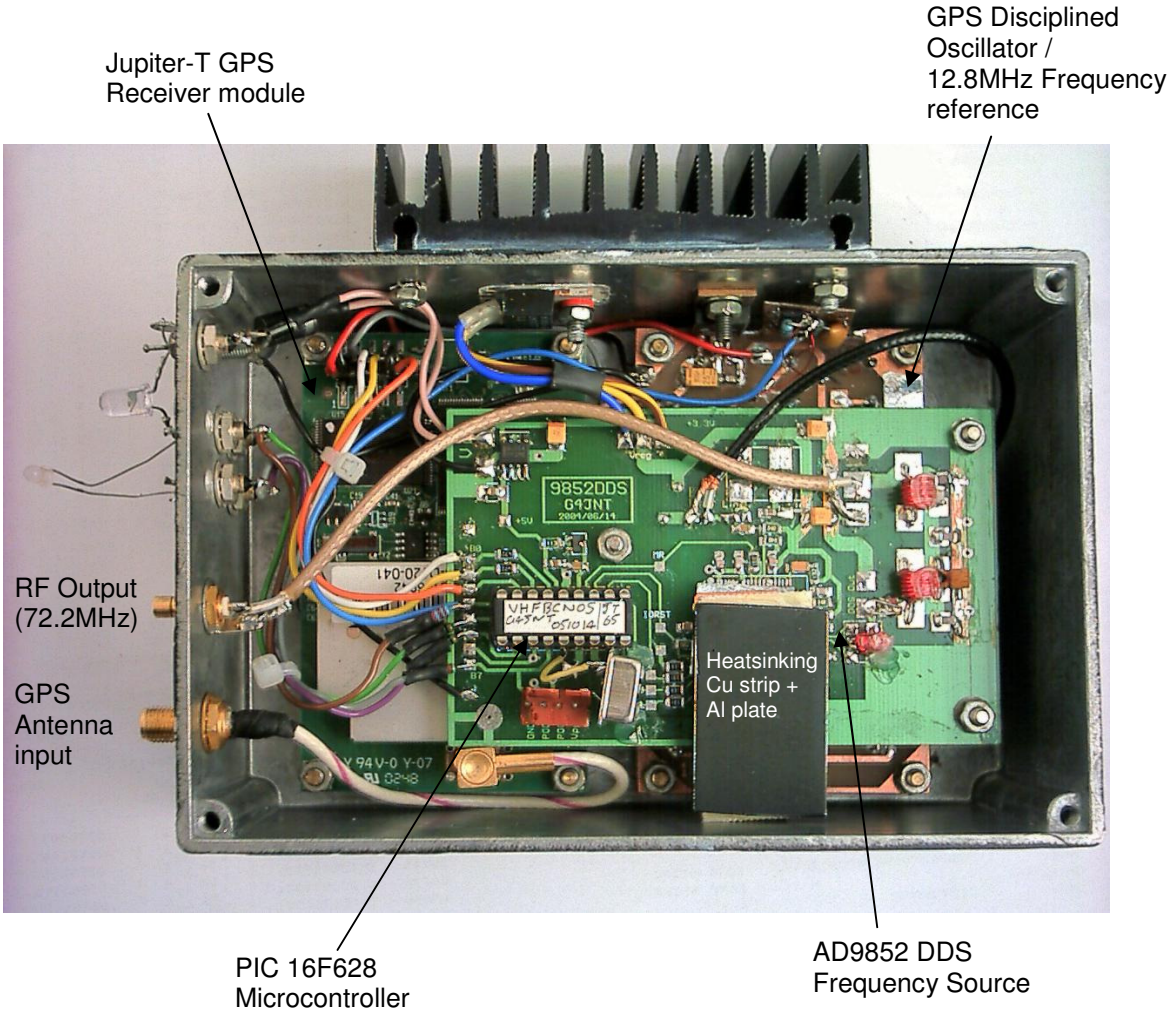
G4JNT 2005/02/13

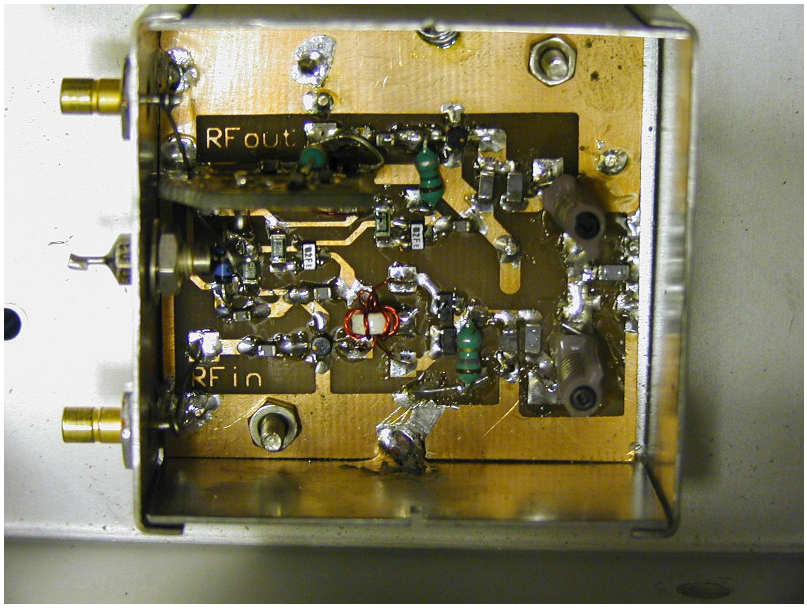
72 to 144MHz Frequency Doubler



Beacon Frequency
Reference

G4JNT 2005/02/13





72 to 144MHz Frequency Doubler