Audio Keyer For Microwavers

Andy Talbot G4JNT May 2023

PIC firmware ***KEYER5***

Having used an FT817 and now an Elad FDM-Duo for microwave operations, there is one big annoyance that can cause problems in the heat of the moment when trying to set up a contact; switching from CW mode to SSB. It is often normal practice to send a carrier, or pulses, or a CW message on a loop when setting up a contact, then revert to SSB for the actual QSO exchange. This often leads to several issues and things that can go wrong.

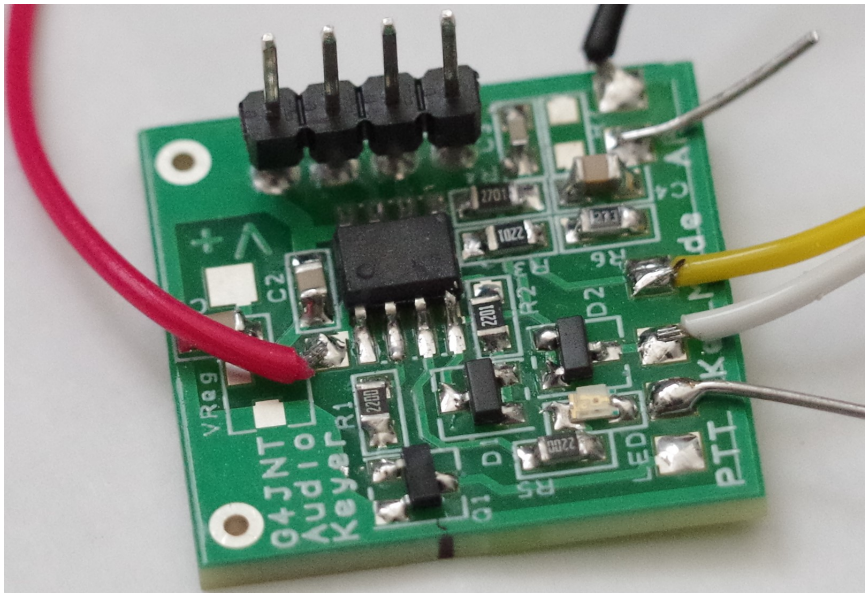
The tuning point for CW and SSB rarely agree, so the receiving station often has to retweak the tuning dial, missing parts of the message in the process.

On many of the smaller transceivers, typically those used for microwave transverter driving, mode change involves cycling through a menu, repeatedly pressing a button while watching the display to get the right one. When the talkback message arrives saying "I can hear you", it is all too easy to forget to change to SSB mode and just start shouting into a microphone with no RF output – and not noticing. The solution is not to switch to CW mode on the radio at all, to stay in SSB at all times and generate CW using an audio tone fed into the microphone socket in parallel with the microphone. Generating the test signal using an audio source gives other possibilities for sending a unique recognisable waveform, such as a chirp.

Module Functionality

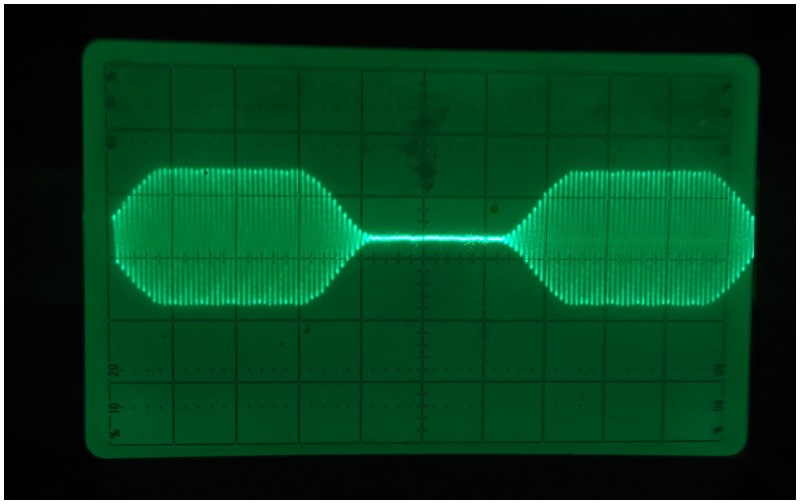
This PIC based module can do three different tasks:

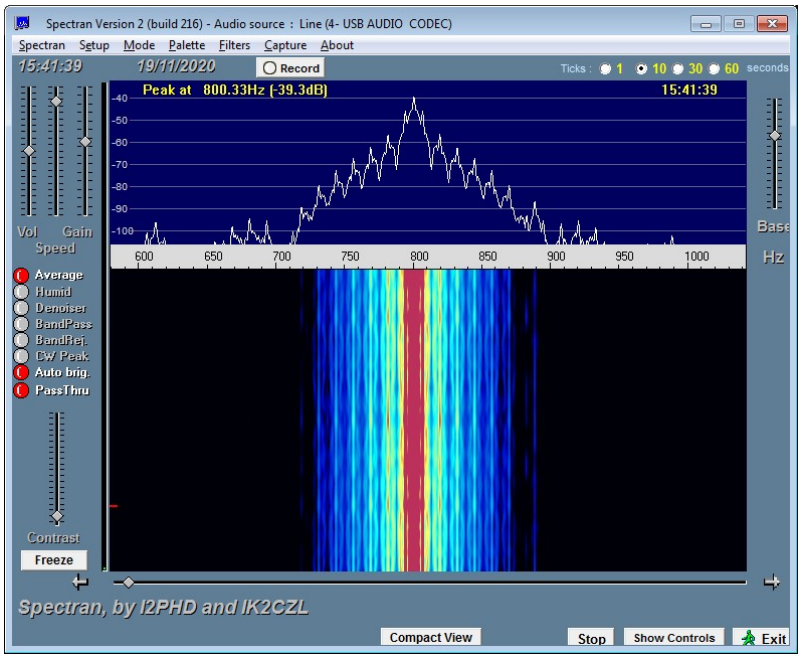
Manual CW Keying	Take a manual key input and generate a clean sinusoidal audio tone to feed to the transceiver input. Control of the PTT line is enabled, with a Tx/Rx delay before dropping back. The CW keying has a shaped rise and fall giving a particularly smooth-sounding CW signal. One of eight tone frequencies can be chosen.
Chirp Test Signal	Continually transmit a rapid chirp, giving a distinctive signal for the far end to recognise and to peak up on.
CW Beacon Keyer	Transmit a prestored CW message. The same beacon keyer capability with user programming from a serial COM port with nearly identical commands and interface as for the original G4JNT/G0IAY beacon keyer [1]. Both polarities of serial interface are now allowed. If a blank message is stored, the keyer instead plays an alternative distinctive sequence made up of a sequence of three tones with a 'musical' spacing.



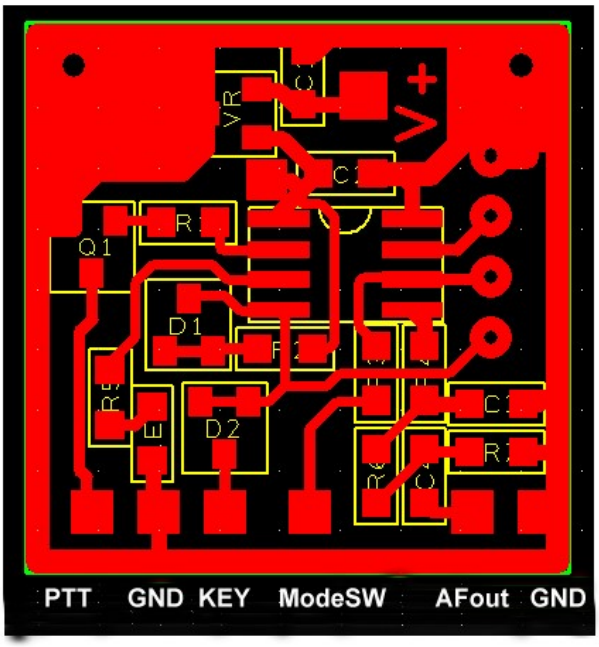
Hardware

A circuit diagram of the controller can be seen in Figure 1 and is built onto a single sided PCB approximately 25mm square. Photo 1. Shows a PCB without the LM3480 SOT23 voltage regulator installed.





The 12F617 PIC microcontroller includes an on-chip pulse width modulation source which is used to generate the sinewave output. The tone is generated using a Numerically Controlled Oscillator sampled at 7812.5Hz +/- 1% uncertainty due to the PIC's factory calibrated RC clock oscillator. The PWM output is sampled at four times this, 31.25kHz. The CW pulse edges are amplitude shaped over a 16ms interval using a 32 point raised cosine waveform, shown in Figure 3 along with its close-in frequency spectrum. At a sampling rate of 7.8kHz and only basic RC filtering on the module's output, some of the 7.8kHz waveform will be present on the audio output. Any commercial transceiver will filter this out so there is no need for additional audio filter components.



The output from the module can be taken via a resistor whose value needs to be in excess of 10k to avoid degrading the low pass filtering. This is connected in parallel with the microphone input. Choose a resistor value that attenuates the 5V Pk-pk waveform down to the few tens of mV needed for maximum audio drive. A value in the region of 100k – 1MΩ will be typical. Alternatively, install the shunt resistor on the output for further attenuation. This component position on the PCB can alternatively be used to add another filter capacitor.

The CW key (***CWkey***) and a mode control pushbutton (***ModeSW***) control the unit's operation. An LED indicates operating state.

CW tone frequency can take on eight user-programmable values from 432Hz to 1760Hz. The chirp goes from 800 to 1600Hz with a repetition rate of 2.5 chirps per second. These values can be changed by device reprogramming, but are not user adjustable.

The pre-stored CW message is held in non-volatile memory and can be changed in user-programming mode, via a serial link at 1200 Baud (either RS232 polarity or 5V TTL from such as an FTDI-Chip device) using standard ASCII terminal software such as *Putty* [2]. The entered message data can contain embedded tokens that allow CW speed to be altered during the message, tokens are also used to give delays inserted into the message. The delay time can have a range of lengths from 1 to 90 seconds, and each separate delay can be individually specified with PTT on or off, and tone on/off. Delays can be cascaded to give more precise control over any delay duration. Thus the module can be used as a standalone beacon keyer with audio tone output.

User programming mode is entered by powering-up the module while holding down both ***CWkey*** and ***ModeSW***. More details are given in the User Programming section below.

Power Supply

A on board regulator supplies +5V from any input from 6.5 up to +20V. However, if a supply in the range 2.5V to 5.5V (absolute maximum) is available the voltage regulator can be dispensed with. A pad on the PCB adjacent to pin 1 can be used for this supply input, or a link can be added across the pads for the voltage regulator

Functionality

When it is first turned on the unit is in manual keying mode. A tone is generated whenever the key is pressed, with rise and fall times smoothed by the raised-cosine pulse ramp shaping. The PTT line goes active as soon as the key is pressed and remains active for a period of 0.8s after the key is released. The LED comes on with PTT operation

Chirp mode is entered by pressing ***ModeSW*** briefly. The PTT line goes active as soon as the chirp starts. Chirp transmission stops when ***CWkey*** is pressed, whereupon the unit reverts to manual mode releasing the PTT after the Tx/Rx delay period. The LED is on continuously while the chirp is being generated

CW beacon mode, replaying the pre-stored message, is started by pressing ***ModeSW*** and ***CWkey*** together. Transmission commences as soon as the key is released. Beacon mode is stopped by again pressing ***CWkey*** which needs to be held down until the current symbol, a dit or dah, is completed. The LED follows the CW keying pattern and during the programmed delay periods it indicates the transmit state.

Tone Frequency

Although any receiving station will adjust the SSB tuning to set the received tone to their own personal preference, the ideal situation would be that very little further tuning change is needed when switching to voice. This is accomplished by making the transmitted tone equal to a favourite listening frequency. General opinion is that this usually lies in the region 500 – 700Hz, and sometimes lower. Therefore a user-programming option has been added allowing one of eight tone frequencies to be selected and stored in memory. The choice can be taken from 432, 528, 645, 789, 964, 1178, 1440 and 1760Hz.

User Programming

First connect a serial, or COM port interface to the four-way header as shown in Figure 2. The keyer has been designed to detect both serial interface polarity options: RS232 polarity signalling (a negative voltage that pulses positive when data is transferred) and the alternative TTL level format from devices such as the FTDI-Chip family. The latter sits in a high state at either 3.3V or 5V and pulses low when data is transferring. The 4k7 resistor shown in the programming lead in the circuit diagram is to prevent excessive current into the PIC input pins from the typical +/-12V RS232 levels. When 5V TTL levels are used, a resistor in this position is not essential, but does help towards preventing an annoying situation that can arise. A TTL source driving the Serial Input is in its quiescent state sitting at 5V. With no power applied to the keyer and the supply line open circuit, the serial input connection can supply enough power which passes via protection diodes in the PIC, to power up the whole module. A resistor in the serial input line reduces the chance of this happening.

The beacon keyer module is reprogrammed using this link. On a PC run *Putty* or some equivalent terminal emulator programme to drive the serial COM port. Set the operating parameters to 1200 Baud, No parity, 1 Stop bit and all handshaking off; half duplex operation with no local echoing of characters.

Power-up the module while holding down both ***CWkey*** and ***ModeSW***. The LED will fast-flash while the buttons are held down and for 1 second after they are released, then will illuminate fully. The keyer will then respond with a message on the terminal :

```
G4JNT Beacon Keyer

[D]isplay  [E]nter  [S]end  Fre[Q]

?
```

If you get this message, the module has correctly entered programming mode.

With the terminal software running on the PC the keyer module is then programmed as follows:

Press D and the module will respond with its current stored message which may look something like:

```
<WF>G4JNT UW KEYER
```

(or anything else that happens to be in memory). This is interpreted as :

Words per minute rate F (approx 20WPM) for one message sequence, then repeat.

To change the message, press E and the keyer will respond with

```
Token Codes
Delay seconds A-1 B-5 C-10 D-15 E-20 F-30 G-60 H-90
WPM           A-6 B-8 C-10 D-12 E-15 F-20 G-24 H-30
?
```

Type in the wanted message. Speed and delay tokens are entered as characters surrounded by angle brackets <…> . Details of the makeup of these are given below.

Carriage Return terminates message entry.

Pressing D again should display the new stored message.

The tone frequency can be changed using the Q command.

Finally, press S to start sending the message and exit programming mode. Check that the LED flashes correctly and the audio tone is as wanted, then remove the serial COM port link. The new message has now been stored and will be sent when beacon mode is selected. Note that when the message is first replayed after the 'S' command, PTT will not be activated unless a programmed delay with PTT-active has been stored, such as <DTDA> == Delay, Transmit, key Down, delay code A, in which case PTT will then be activated as soon as it is encountered. Normal PTT operation occurs when the Beacon message is called directly from (***ModeSW*** + ***CWkey***).

Tokens

Apart from text characters, 'tokens' may be stored within the message. These consist of certain characters enclosed between angle brackets that define speed and delays.

Tokens can :

1) Change the speed of the keying over a range of values.

2) Include a delay, with PTT On or Off and with the tone on or off.

Details of making up the tokens can be found in the **Programming Summary** below.

References

[1] Beacon Keyer http://g4jnt.com/JNT_BeaconKeyer.pdf

[2] Putty terminal software
https://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html

PCBs and a programmed PIC are available.

To obtain these, or to download the PIC firmware, go to
http://www.g4jnt.com/BcnKeyer.htm

Programming Summary

Terminal (eg. *Putty*) set to 1200 baud, N81, full duplex.

Connect the serial COM port lead, hold down both ***ModeSW*** and ***CWkey*** .and power-up the keyer. The LED will flash while the switches are held, for second after they are released then will fully illuminate.

The unit responds with an introductory message and menu –

```
 [D]isplay / [E]nter / [S]end  Fre[Q]
```

Press D To see the current message stored.
E To enter and store a new message.
S To go to normal sending (with out immediate PTT operation) and exit programming mode.
Q To change the tone frequency.

Press E and the prompt for the CW Message will appear:

```
Token Codes

Delay seconds A-1 B-5 C-10 D-15 E-20 F-30 G-60 H-90

WPM           A-6 B-8 C-10 D-12 E-15 F-20 G-24 H-30

?
```

Enter the CW message data and any programmed delays, [rtn] completes the message. The LED will flicker slightly as data is entered.

Press D To see the new message.

CW Speed and delays are set by entering a special code (a token) in the message entry string, placed between angle brackets.

<Wx> Where x = A to H, sets the CW Speed according to the table below.
<Dxyz> Sets a programmable delay
x = R / T sets PTT line to Receive or Transmit
y = D / U sets key Down or Up
z = A – H Delays for a duration according to the table below.

eg. `<WC>G4JNT <WE>G4JNT IO90IV58 <DTDC>` sends callsign at 10WPM with a space, then again at 15WPM followed by the locator, then a delay in transmit mode with the key down for 20 seconds. If no WPM code is included, a default speed of 15 WPM is used.

Approximately 55 locations are available for message storage. Each token takes up one location. A warning is issued when entry overflows the storage capability.

If a mistake is made, press [rtn] to go back to the main menu, then start again with the [E]nter option.

Delay times and CW speed are taken from this set of values.

Letter Code	A	B	C	D	E	F	G	H	
CW Speeds	6	8	10	12	15	20	24	30	Approx WPM
Delays	1	5	10	15	20	30	60	90	seconds

While sending, the LED flashes with the CW characters, and is on when PTT is active with Key Down during delays. The LED is on during programming mode and flickers very slightly during RS232 data entry.

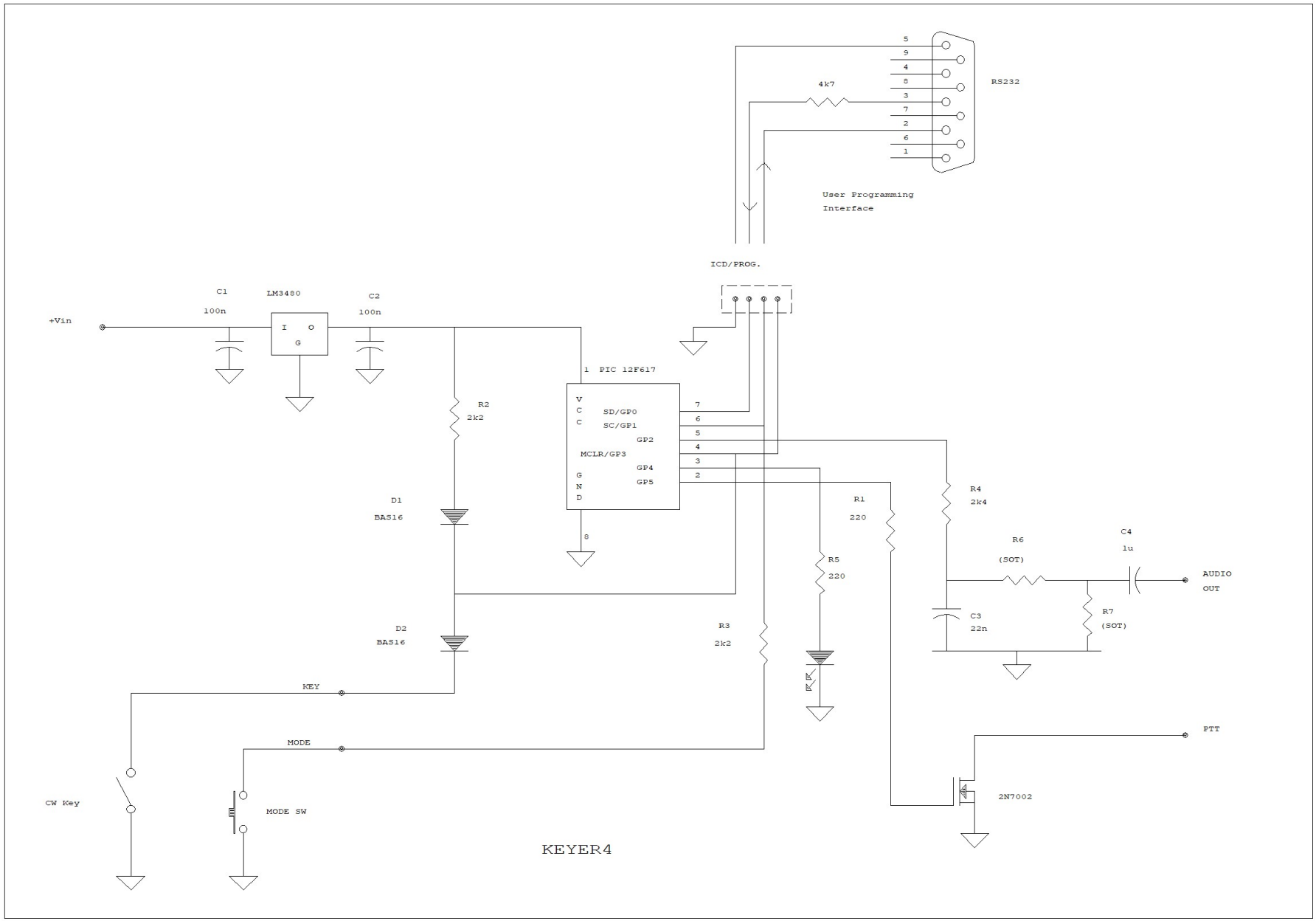
If the CW memory is programmed as blank, ie. [rtn] pressed at the E prompt without entering any other characters, then instead of any CW message a special transmission sequence of three tones at frequencies of 494, 523 and 587Hz is sent repeatedly at roughly 1/3 second intervals. These tone frequencies correspond roughly to musical notes B, C, D and when correctly tuned on an SSB receiver should “sound right”. Tones and interval spacing as well as the chirp limits and rate can be changed by PIC reprogramming.

To change the tone frequency, at the main menu Press Q and the prompt for tone frequency appears:

```
Tone Code  A-432 B-528 C-645  D-789  E-964 F-1178 G-1440 H-1760

?
```

Enter a letter A – H and the keyer returns to the main menu with the new tone frequency stored.

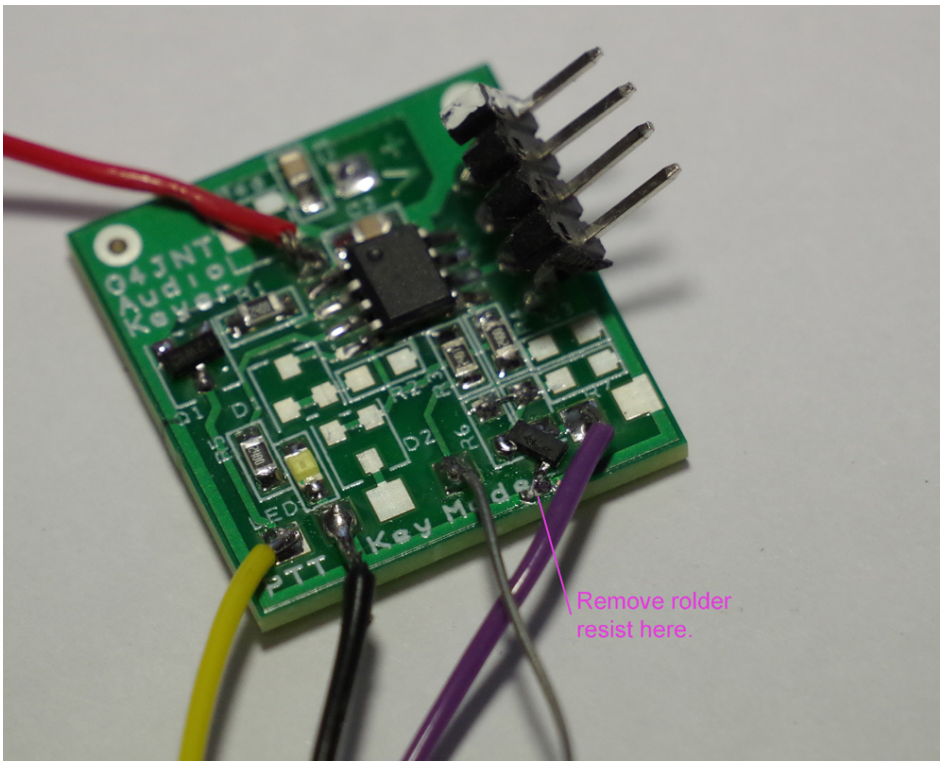

5
9
4
8
3
7
2
6
1
RS232
4k7
User Programming
Interface
ICD/PROG.
C1
100n
LM3480
I
O
G
C2
100n
+Vin
1 PIC 12F617
V
C
C
SD/GP0
SC/GP1
GP2
MCLR/GP3
GP4
GP5
G
N
D
7
6
5
4
3
2
8
R2
2k2
D1
BAS16
D2
BAS16
R1
220
R4
2k4
R5
220
R6
(SOT)
C4
1u
AUDIO
OUT
C3
22n
R7
(SOT)
R3
2k2
KEY
MODE
CW Key
MODE SW
PTT
2N7002
KEYER4

Appendix A **Direct Standard Keyer Option**

PIC Firmware ***DirectKeyer***

The same PCB can be used, with a bit of micro-surgery, as a direct beacon keyer module with on/off key output. As such it replaces the originally G4JNT Beacon Keyer [1]

The key input connection is not used and components R2, D1 and D2 can be left out. Key output comes from Pin 5, the original PWM output. For direct TTL output, take a link from the pad for R4 adjacent to pin 5 to the output pad, or insert any combination of resistors and capacitors that may be required to filter the output. As an alternative, an additional MOSFET can be added to the board for the conventional 'switch-to-ground' keyer output. To install an SOT23 type FET on the PCB requires a bit of micro-surgery on the board as shown in Photo 2. Scrape away resist on the earth track where shown, and an SOT23 device will just fit across the gap. A resistor of a few hundred ohms should be used in series with the gate drive; this can conveniently be in the R4 position.



Transmitter control via the T or R options in the Delay token is via the PTT connection, the same as for the tone output version.

Programming mode is entered by holding down MODESW while powering up the module. Programming is then exactly the same as for the tone output version except that the 'Q' command is no longer available..