On-site reprogrammable beacon keyer – Includes Analogue Version

Andy Talbot G4JNT/G8IMR

March 2011 - New QRSS version. See Annex 1

Overview

The beacon keyer is a small module that generates pre-stored CW messages and controls a Tx / Rx line. As well as CW messages, delays of different lengths can be embedded within the message, and CW speed can be changed at any point within the message using embedded commands.
The module is programmable on-site for new messages and formats using ASCII commands on a serial interface. Messages are stored in non-volatile memory.

A new version is now available with a single channel of analogue input. An internal A/D converter reads the voltage on this within a range of 0 – 5V and converts the reading to CW data. An embedded code within the stored message reads the A/D at the appropriate time and inserts the formatted value into the CW stream. A variety of different formats and scaling factors for sending the analogue data can be set up on the programming interface.

Hardware

The standard keyer module is based around a 16F629 PIC microcontroller; a 16F675 device is used for the analogue version. The circuit diagram is shown in Figure 1 below. Two active low outputs are provided; a key output carries the CW information, and the Tx output allows a transmitter to be keyed from data stored within the message. A small postage stamp-sized PCB contains the chip and peripheral components, with power and output connections made via small pads. An LED on the standard version allows the CW message and certain items of programming status to be observed. On the analogue version, the LED is not present, its allocated pin on the PIC being used for the analogue input. The series resistor is kept and this, in conjunction with an additional 10nF filter capacitor, provides a modicum of protection and filtering for the analogue input port.

A four-way header serves a dual purpose. It allows connection of the serial programming interface which connects directly to the RS232 port of a computer running a terminal programme. One current limiting resistor whose value is uncritical is needed in the TXD line from the PC. It is unlikely any damage will occur if this resistor is omitted, but its inclusion is good practice to limit input current into or out of the PIC pin. It also serves as the in-circuit programming interface for the PIC

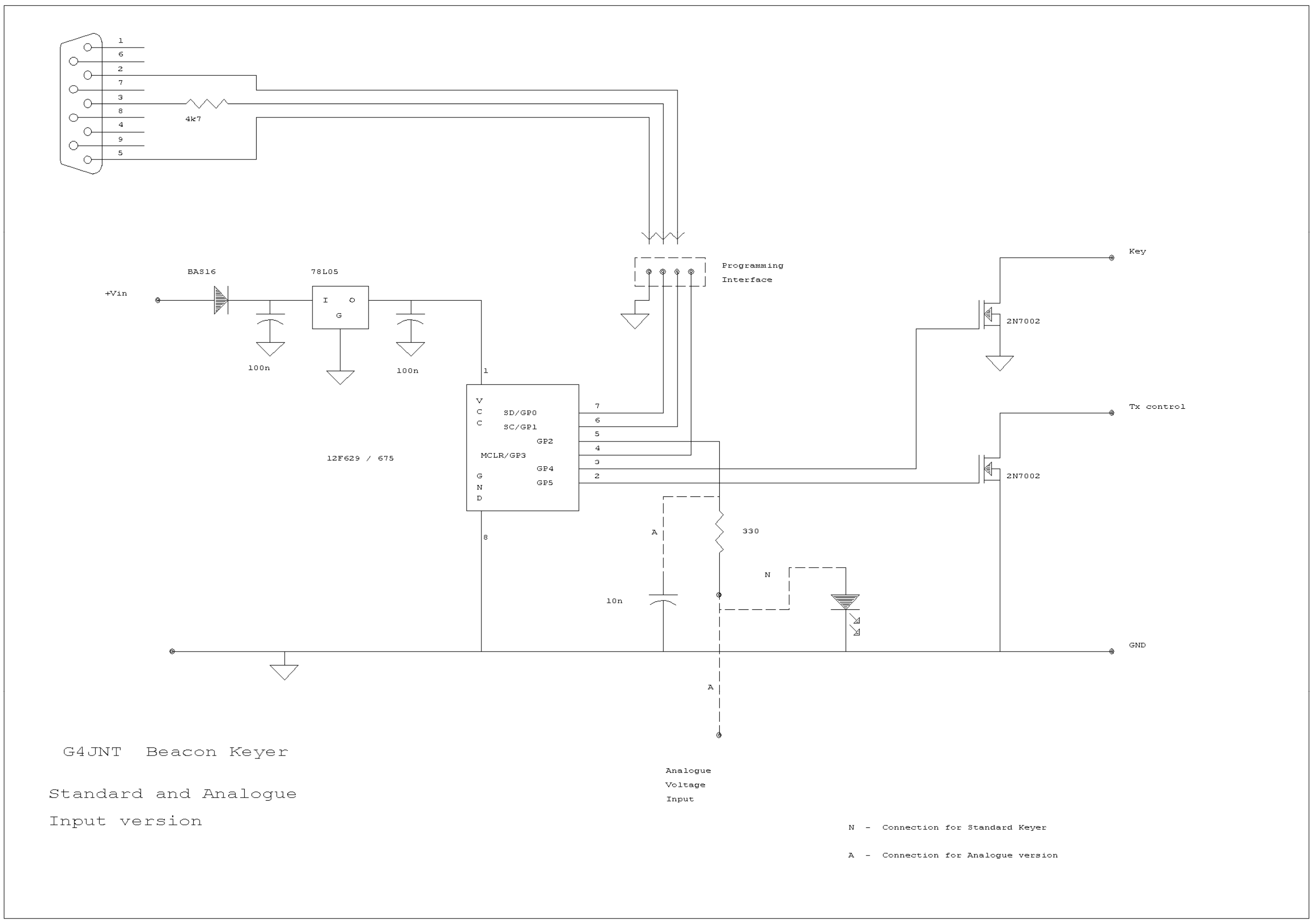

Figure 1 Circuit Diagram

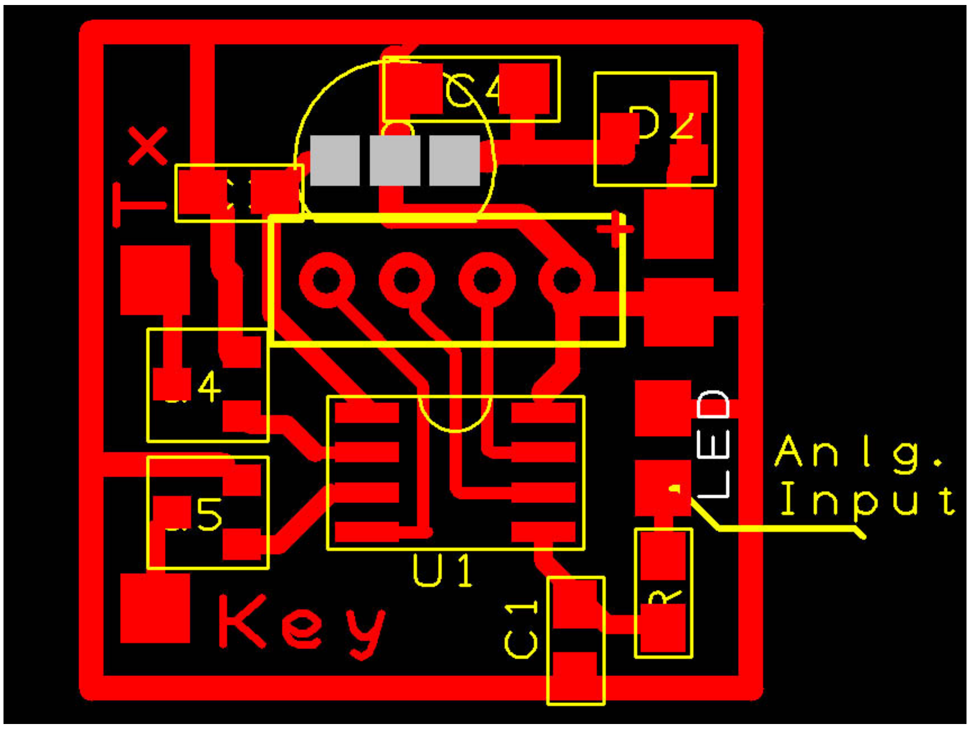

Figure 2 PCB Layout

Programming

Programming details that follow are common to both the standard and analogue keyer modules. The additional commands for the Analogue version are given at the end. When programming the analogue version, as this has no integral LED, all references to the LED should be ignored.

The beacon keyer module can be reprogrammed on site using an RS232 link. On a laptop run Hyperlink 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 typed.
With Hypertrm (etc.) running on the laptop the keyer module is then programmed as follows :

The keyer module enters programming mode when the RS232 link is connected, and power is applied or the PIC is reset. So switch off the modules main supply, then switch it back on again with the interface plugged in. The keyer should respond with :

> *G4JNT Beacon Keyer*
> *Display / Enter / Send.*

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

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

> <WC>GB3SCX <WH> GB3SCX IO80UU59 <DTDA>
> (or anything else that happens to be in memory)

Which is interpreted as :
Words per minute rate C (approx 10WPM) for one callsign sequence, then WPM rate H (approx 20WPM) for the rest of the text. Followed by a Delay (The D) in Transmit mode (T) with key Down (D) for a time A (the shortest delay allowed). After which the sequence repeats continuously..

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

> *Enter CW & Delay Data*

Type in the wanted message, with speed and delay tokens surrounded by <…>
Carriage Return terminates message entry

Pressing D again should display the new stored message.

Finally, press S to start sending the message and check that the LED flashes correctly, then remove the
RS232 link - the new message has now been stored and will start automatically next time power is applied without the RS232 interface connected.

Tokens

Apart from text characters, 'tokens' may be stored within the message. These consists of certain

characters enclosed between angle brackets that define speed, delays etc.
Tokens can :

1) Change the speed of the keying over a range of values.
2) Add a delay, with key up or down, and with the Tx line set or reset.

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

Programming Summary

Terminal (eg. Hypertrm) set to 1200 baud, N81, full duplex.
Connect RS232 lead, and switch on or reset the beacon keyer which responds with intro message and menu - "Display / Enter / Send"
The LED will illuminate.

Press	D	To see current message stored in EEPROM
	S	To go to normal sending and leave programming mode
	E	To enter and store new message :

The prompt for the CW Message appears.
Enter the CW and delay message data, [rtn] completes the message
The LED will flicker slightly as data is entered

CW Speed and delays are set by entering a special code in the CW string:

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

eg. <WC>G4JNT <WE>G4JNT IO90IV58 <DTDC> sends callsign at 10WPM + space,
then at 15WPM followed by locator, then a delay in transmit mode with the key down for 20 seconds.
If no WPM code is included, a default speed of 12 WPM is used.

The backspace key works for normal character entry. Complete token entries can be deleted by pressing [BkSp] only after the closing **>**
Up to 128 memory location are available message storage.
A warning is issued when entry overflows the storage capability.
If a mistake is made, press [rtn] (+[rtn]) to go back to D / E / S menu,
then start again with the [E]nter option

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

When sending, the LED shows CW characters, and is on when Tx with Key Down during delays,
LED is on during programming mode and flickers very slightly during RS232 data entry.
There is an extended off period during Token entry.

Analogue Keyer

The analogue version of the beacon keyer module has an identical command and message structure to the standard version, with the addition of a single channel of A/D conversion.

This measures the voltage within the range 0 – 5V present on the pad that was allocated to the LED on the standard module and reports the value in the CW massage.

A voltage measurement is taken by inserting a <Vx> token into the message string. The values of ‘x’ takes a letter from A – H and controls how the measured voltage is reported within the CW output message. The speed of the resulting CW is that set by any previous <Ws> command. Apart from the special BCD case listed, the actual A/D reading is triggered immediately it appears in the message, so the reading is near-enough real time.

Several formats are available, trading precision and message complexity / length.

Token	Output Data	Description
<VA>	‘0’ to ‘9’	Single number , Linear 10 steps, 0.5V per step
<VB/C/D>	‘000’ to ‘255’	Full 8 bit value, see below for details
<VE>	‘A’ to ‘Y’	Single letter, Linear 25 steps, 0.2V per step
<VF>	E, I, S, H, 5	Linear, 5 steps, 1V / step sent as a dot-count
<VG>	E,I,S,H,5	Expanded 5 steps, thresholds at 0 - 0.4 - 0.8 - 1.6 - 3.1– 5V
<VH>	[sp]	*Not yet allocated – available for expansion*

B,C,D are a special case and must be used in a group. Only B (the full token is <VB>) triggers an A/D conversion, whereupon the full 8-bit value in the range 0 – 255 from the /AD conversion is available for reporting. <VB> on its own sends the ‘hundreds’ value of the measured voltage level in the range 0 - 255, so can only ever take on the characters 0, 1 or 2 . <VC> reports the tens digit of the conversion, but does not itself trigger another A/D conversion. It uses the value obtained from the last one. <VD> similarly does not trigger an A/D reading, but just reports the units digit of the previous value. Therefore, for a full 8 bit report taking three digits to send, the message string has to contain the following tokens: <VB><VC><VD> (which should be easy to remember...digits...BCD ...).
Alternatively, <VB><VC> alone will give a roughly 25 step report that may be easier to interpret than the single letter A – Y given by the <VD> format.

Some examples of complete beacon messages containing voltage measurements follow :

<WH>P OUT <WD><VA>
Sends text “P OUT” at 24 WPM followed by the voltage quantised to 10 * 0.5 Volt steps as a digit from 0 to 9

VOLTAGE<VB><VC><VD>
Sends the text “voltage” followed by the measured input between 0 – 5V as a number from ‘000’ to ‘255’ at the default speed

GB3SCX <VE> <DTDA>
Send the callsign followed by 1 to 5 dots corresponding to 1,2,3,4 or 5Volts on the measurement pin, followed by a Tx one second delay with key down.

A complete Beacon identifier, with the beacon sending its own measured output power might take a form similar to :

<WC>GB3SCX <WF>GB3SCX IO80UU59 <DTDD><VF><DTDA>
Which is interpreted as :

Callsign at 10 WPM followed by callsign and locator at 20WPM. A delay on transmit with key down for 15 seconds, followed by a 5 level logarithmic quantised measure of the input voltage as a count of 1 to 5 dots. Then followed by a further 1s delay before the sequence repeats.

Accuracy

The PIC's internal A/D converter uses the regulated (nominal) +5V line as its reference, and for critical applications, please note that this could be in error by a few percent. The internal A/D conversion is actually to 10 bit resolution, but only the highest 8 bits are used for determining the value to be sent.

.

Annex 1 QRSS Version

PIC Code ***KEYERQRSS*** (.asm and .hex) contains a version of the standard (non-analogue) keyer targeted at QRSS (SlowCW) operation. Functionality, programming etc are identical to those for the standard keyer; except that CW speed and the programmed delays are as shown in the table below.

Letter Code	A	B	C	D	E	F	G	H	
CW Dot	1	2	3	5	10	20	30	100	seconds
Delays	10	20	40	60	80	120	240	360	seconds