PIC Based 'Opera' Mode Beacon Source .

Andy Talbot G4JNT 2011-01-08

Ver 2 Feb 2023 to support QSO mode Opera (Ver 1.4) and others using data copied to the clipboard

Here is a PIC based solution for driving a transmitter source with the on-off coded symbols for the 'Opera' datamode. Options allow for the six transmission speeds from 1 to 31 minutes currently used at Ver 1.1.2 of the Opera software, with two remaining speeds for future upgrade. At the moment these offer two high rates that can be used for testing. Eight beacon durations can be selected but unlike the main Opera software, here they are defined as a duty cycle and are set by three links. The values allowed are 100%, 50%, 33%, 25%, 20%, 17%, 14%, 12.5%. The actual Tx delay will therefore be a function of the duty cycle and transmission speed.

Figure 1 shows the circuit diagram of the hardware (a higher resolution is available as *operabcn.gif*). The transmitter on-off is keyed from the active low Key Line output. A second output allows the transmitter to be enabled, and can be used when running a low duty cycle to save standby power consumption between Tx periods. A LED flashes at half symbol rate to show the operational status and runs continuously, even during standby periods. The Test strobe on port A3 generates a pulse at the interrupt rate of 1.024ms. It is there for convenience when initially testing and can be ignored for all practical purposes. A 4MHz crystal is used for the PIC, as this conveniently allows a divided down rate for the interrupt using just the internal 8 bit timer overflow and the prescalar..

The link settings can be changed during a Tx period. The new speed is updated immediately after the current symbol ends – so may give unpredictable results if the settings are changed while actually transmitting the code.

The new duty cycle takes effect as soon as the current Tx period (whether in standby or Tx mode) ends. It is usually advisable to reset the module after changing settings.

B2 B1 B0	Speed	B5 B4 B3	Duty Cycle
0 0 0	256ms / OP1	0 0 0	100%
0 0 1	512ms / OP2	0 0 1	50%
0 1 0	4.096s / OP4	0 1 0	33%
0 1 1	8.192s / OP8	0 1 1	25%
1 0 0	4.096s / OP15	1 0 0	20%
1 0 1	8.192s / OP31	1 0 1	16.7% (1 in 6)
1 1 0	5ms (test mode)	1 1 0	14.3% (1 in 7)
1 1 1	20ms " " " "	1 1 1	12.5%

The links are configured such that a link in place, to ground represents a logic 1.

Generating the new PIC Code

As the internals of the Opera coding scheme have not been made available to us, the symbols to be transmitted have to be generated from the ***Opera.exe*** software using the PIC option in the top menu Ver 1.4.1, used for 'QSO mode' copies the bit pattern to the clipboard using a 'copy' button.

Copy the bit pattern that results to the clipboard.
The text now has to be converted to a format for the PIC assembler to read. The utility ***GENOPERA2.EXE*** will do this for you. Just run *GENOPERA2"* from a command prompt or by clicking on an icon etc. If the clipboard contains valid data (ie. nothing but '0'and '1' characters), the software will then ask for a comment to be included in the PIC listing. Use something like the callsign plus any other notes – or nothing at all. When that is done, the software will generate a new file **OPSYMBS.INC** which should look not totally unlike that shown in the box:

```
;callsign  G4JNT
        de   b'11011010'
        de   b'10101010'
        de   b'10101001'
        de   b'01100110'
        de   b'10010110'
        de   b'01100101'
        de   b'01101001'
        de   b'10011010'
        de   b'01011010'
        de   b'10011001'
        de   b'01011010'
        de   b'01011010'
        de   b'01010101'
        de   b'01101001'
        de   b'10100110'
        de   b'10011010'
        de   b'01101001'
        de   b'01101010'
        de   b'01101001'
        de   b'10100101'
        de   b'01010110'
        de   b'01101010'
        de   b'01010101'
        de   b'10101010'
        de   b'10010110'
        de   b'01100101'
        de   b'10010101'
        de   b'01101010'
        de   b'10011001'
        de   b'01011010'
```

Then, using your favourite PIC assembler (such as MPASM from www.Microchip.com) , assemble the PIC source code **OperaBcn.asm** . This will read the custom data set and generate the resulting .HEX code to be programmed into the 16F628A controller.

A four pin header is shown on the circuit diagram and is intended for in-circuit programming. The connections to this are the established (and non-ideal) 'JNT standard' for PIC programming and are the same as all my published PIC designs, including all available circuit boards. A lead will need to be made up for interfacing to the PIC programmer. I recommend the PicKit 2 (or 3 if you want) .

Once programming is complete, the module should start sending immediately after power up or reset, with the format sent depending on the links in place.

Different versions of the Opera software have used various numbers of symbols per transmitted frame. At Ver 1.1.2, there are 239 symbols per frame, but previous versions have used 183 and 225. It is quite reasonable to expect this may change in future versions. However many symbols there are, the ***GENOPERA2*** utility will read these then pad out with zeros to a multiple of 8 and generate a correctly formatted **OPSYMBS.INC** file.

Supplied Files

The archive contains the PIC source code , ***OperaBcn.asm*** with an include file to get you started. The code in this will generate the transmitted callsign G9BOF – so modify before using on air.

The Data conversion utility ***GENOPERA2.EXE*** is included – copy into the same subdirectory as all the other files. It was written in super-ancient 16 bit Powerbasic and the source code can be seen in ***GENOPERA2.BAS***. The ***OperaBcn.hex*** file compiled using this callsign is included to get the hardware operational; change the PIC code to your own callsign once you know the hardware works.

Please note, some of us are quite happy to provide (at least for non commercial purposes) all software and code to assist home constructors, and for the general self training specified as part of the Am. Rad. licence.

Wouldn't it be nice if *GENOPERA* could have accepted a callsign then directly generated the bits in the same way as my utilities *GENWSPR* and *GENJT4* do. Instead of having to go though the complicated cut and paste routine.

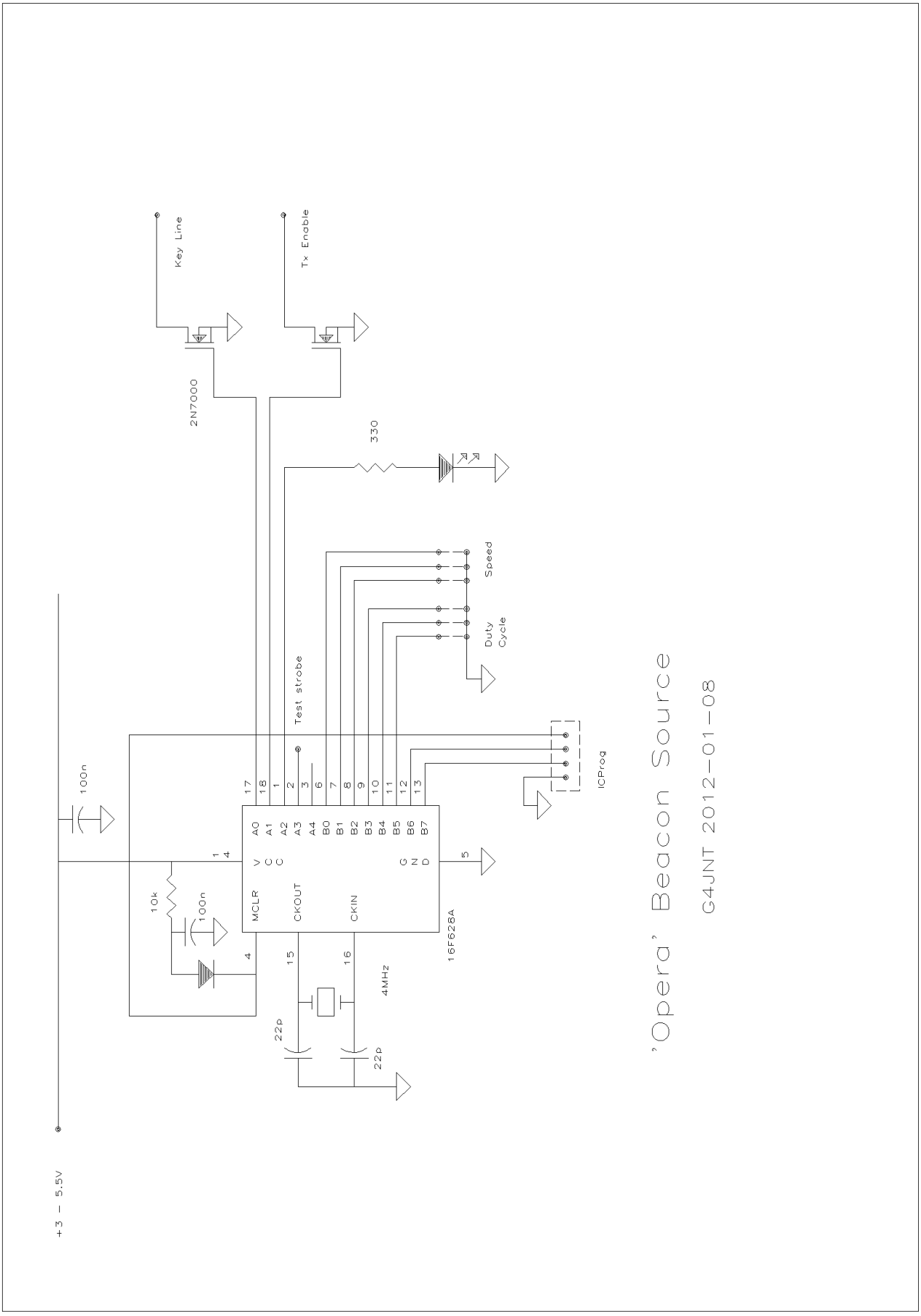

Figure 1 - Circuit Diagram