September 2015 • RadCom
Jonathan Hare, G1EXG • j.p.hare@sussex.ac.uk

PIC push button control of your FT-817

…or other compatible radios

INTRODUCTION. The FT-817 is an amazing little radio that covers the amateur radio bands from 1.8 to 430MHz. Although it's been around for almost 15 years there is still no other low power transceiver of similar size and spec available so it is still a very popular radio. However it's so compact that a lot of the functionality of the radio requires you to go through numerous menu button presses, which can be tedious. This project uses a PIC chip and the FT-817's CAT computer control facility to quickly go to your favourite modes and frequencies by a simple press of a button without having to tackle the menus. The device should also work with the FT-857 and FT-897 transceivers.

G4JNT has described a PIC project to directly enter a frequency into the FT-817 via a small telephone type keypad [1]. Rather than fiddling around with the function keys you can get to where you want to go very easily just by typing it in. I made up a version of the project and learnt a lot in the process [2].

The circuit I describe here does not use a keypad: instead, I have set up push buttons to control the transceiver. The circuit will not only take you directly to a chosen frequency but it can also set the mode as well as other features. For example if you want to go to the 2m SSB calling frequency all you have to do is simply press a button and the unit will take you to 144.300MHz and set the mode to upper side band (USB). There is a '20 up' button to take you away from the calling frequency to 144.320MHz. Another of the push switches is a 'club net' button. The Worthing and District Amateur Radio Club (WADARC) meets at 7.30pm on Monday nights on 145.425MHz FM and one button push sets everything up.

The full list of functions on my version of the controller is:

1. Push to talk (PTT) ON
2. PTT OFF
3. Toggle between VFO A & VFO B
4. 144.300MHz USB SSB calling frequency
5. '20 up' from SSB calling frequency, ie 144.320MHz
6. 2m WADARC Monday night club net, 145.425MHz FM.
7. 80m PSK frequency, 3.580MHz, USB

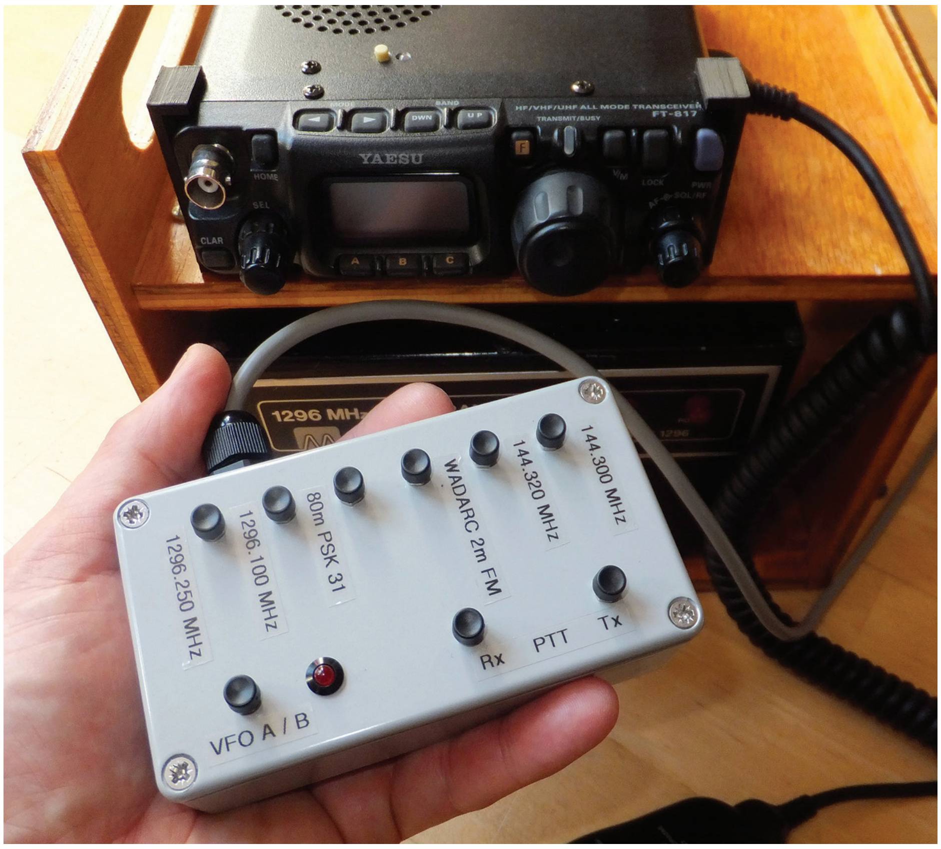

PHOTO 1: The completed controller provides an easy way to set the FT-817 to commonly-used frequencies and modes.

8. 23cm transverter frequency 1296.100MHz (FT-817 to 146.100MHz, USB)
9. 23cm transverter QSY 1296.250MHz (FT-817 to 146.250MHz, USB).

I have also included push to talk (PTT) ON and OFF buttons. If you are adjusting a circuit that needs the FT-817 to transmit (eg setting up a linear amplifier into a dummy load), instead of trying to hold the microphone switch ON while making adjustments, you can simply press the PTT ON button to transmit and then OFF to go back to receive. You could use this like a latching base microphone if you want to experiment with your own desk mic.

I assigned two buttons for a 2m to 23cm transverter. If I want to go to 1296.100MHz I need my FT-817 to tune to 146.100MHz, which I can do with a press of a button. I have also included 146.250 (150kHz up) so that I can QSY (note: a re-setting of the FT-817 frequency coverage [3] is required to transmit here).

In normal use you can change from VFO A to VFO B using the three selection keys on the front of the FT-817. However I have often found that when you portable the placement of these keys and the way you have to push them is not always very convenient. This little circuit makes this much easier as a single button push will toggle between them.

Of course the idea here is that you can modify the code described here to incorporate your own favourite settings and functions to custom design your own controller.

NUMBER OF SWITCHES / PRE-SETS. I used a 16F688 PIC microchip [4] set to use its own built-in oscillator so we don't need to add an extra crystal (this frees up two of the PIC pins, which is good as it's only a 14 pin chip). The PIC has two 6 pin

Jonathan Hare, G1EXG • j.p.hare@sussex.ac.uk

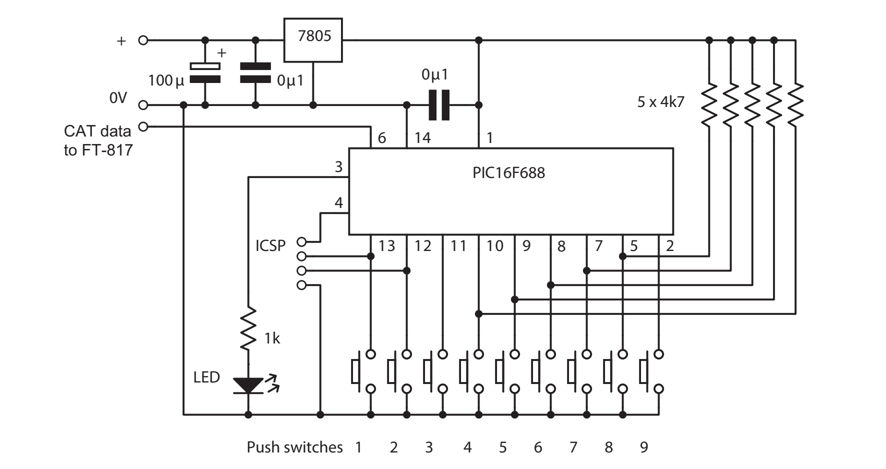

FIGURE 1: The circuit diagram of the FT-817 controller.

ports (PORT A and PORT C) so we have a total of 12 free input / output pins to play with. One pin on Port C (C4 on pin 6 of the PIC) is used for the serial data transmission (PIC to radio), one on Port A (A4 on pin 3 of the PIC) is for an LED and another for the MCLR pin that is part of the in circuit programmer (ICSP): more on this later. This leaves 9 port pins to go to the front panel switches (listed in **Table 1**). A matrix arrangement could be used to get more switches for the available input pins but I have opted for simplicity in this prototype and simply used one switch on one input.

When a switch is pressed it pulls the PIC input pin to ground. PORT A has internal pull-up resistors that can be selected in the software to allow this to work but PORT C does not have this facility. So we have to include five resistors, one for each of these inputs.

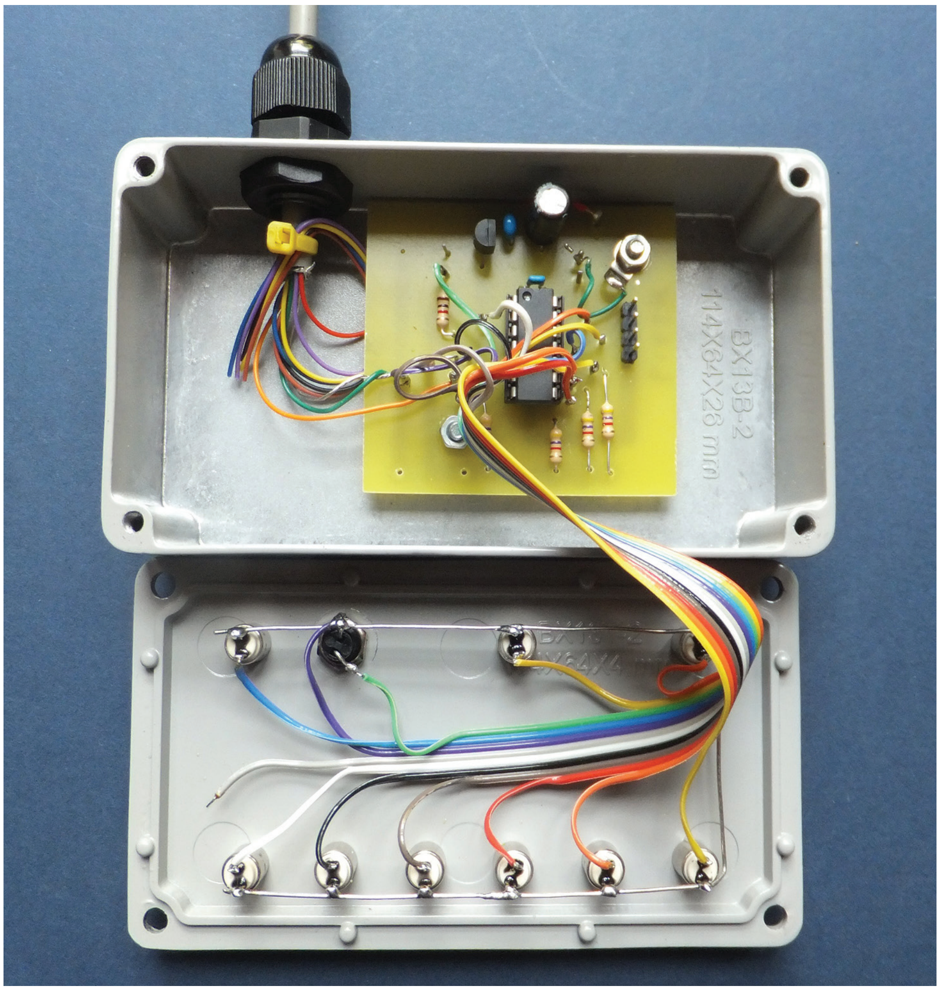

PHOTO 2: Inside the box the layout is quite straightforward. I used a PCB but the circuit is so simple that Veroboard or even dead-bug construction would be fine.

CAT COMMS BETWEEN PIC AND FT-817.

The PIC communicates to the transceiver via a serial data link, which Yaesu calls CAT – computer assisted transceiver. It is based on a TTL level version of RS232. The serial port configuration is set up at the start of the code. For this project we only need to be able to send data to the FT-817 so we don't need to receive anything back to the PIC. Therefore I have only used the Tx port, which is on pin 6 of the 16F688. The CAT protocol needs 1 start bit, 8 data bits and 2 stop bits. I followed G4JNT and used a baud rate of 9600 (you need to set the FT-817 to this rate via the menu settings, as the default is 4800 baud **[5]**).

Note: the 5V signal levels created by the PICs are correct to go directly to the radio; it does not need to be inverted to higher voltages of a typical RS232 line (eg ±15V).

To get the radio to respond to our PIC data we need to send the radio five groups, or sets of numbers: the first four consist of data, while the last is the command telling the radio what to do with the data, eg [data1] [data 2] [data 3] [data 4] [command]

To tell the FT-817 which frequency to go to you need to break up the frequency into data parts and a command. There are four data parts, which consist of the frequency digits split into pairs. The command completes the sequence and tells the radio to use this data to set the frequency (the command is '01' in this case). For example to set the radio to 145.425MHz the CAT system needs to send '14', '54', '25', '00' and then '01'.

To change the mode the first group sent is the code for the particular mode (eg '00' = LSB, '01' = USB, more on which later) followed by three 'null' groups (which can be any value) and the fifth is the command to order the mode change, which is '07'. So for example to change the mode to USB we need to send: '01', 'xx', 'xx', 'xx', '07'. Or, more generally, 'AA', 'xx', 'xx', 'xx', '07', where xx can be any data and AA is 00 = LSB, 01 = USB, 02 = CW, 03 = CWR, 04 = AM, 08 = FM, 0A = DIG, 0C = PKT

Other FT-817 hex commands [5] include 01 = set frequency, 07 = set operating mode, 08 = set PTT ON, 88 = set PTT OFF, 81 = toggle the VFO.

To change the frequency and the mode when one of the buttons is pressed, the code needs first to send the five groups of data for the frequency change then the five for the mode change. To set up for a local repeater we would need to send four different sets of parameters one after the other: frequency, mode, repeater offset and the subaudible tone frequency (CTCSS).

Changing from the front BNC to the rear SO239 antenna socket would be useful, as would the ability to set the radio to low power for my 23cm transverter, but unfortunately the commands for these do not seem to be provided. It is possible to re-programme the internal EEPROM of the radio but I didn't want to go down that road with the current project as you can wipe the radio setting if you do this incorrectly.

THE CODE. You can download the original code from my website [2] and I have annotated the PIC ASM code with comments so you can see how it works. You will want to change the data groups for your particular set of club frequencies and modes etc. In operation the code scans the 9 press buttons and if one is pressed (ie if the input pin is low) the program diverts to the relevant subroutine. Each subroutine then sends out the CAT signals for that particular frequency change, mode change etc then the program returns to scanning the press buttons. For simplicity the code only detects if one button is pressed but I suppose one switch could be used as a 'shift' if more functionality was required for the remaining eight switches (although simplicity is really the key to this device).

One of the Port A pins (A4 on pin 3) is used as an output to drive an LED. Once the device is plugged into the radio the first thing the code does is to flash this LED a couple of times to show you that all is OK with the power and the controller device. The LED also flashes when data is sent to the radio, giving a visual indication from the controller that something is happening when you press a button. I have written the code so the LED flashes with the number of separate instructions that are sent to the radio eg change in frequency = 1 flash, change in frequency and change in mode =2 flashes etc.

The PIC responds much faster than human reaction times so I have built delays into the code. For example if you press switch 4 it will change the frequency and mode. Pressing it again, or not letting go of the button quickly enough will not be a problem as it just repeats the same set up and you don't notice any problem. However, if you keep the 'toggle VFO' switch pressed too long (switch 3) the circuit will switch very quickly between the VFO A and VFO B, producing an erratic response. I therefore built in a short delay after sending the data to give you time to let go of the button.

Once you have changed the code to incorporate your own setting you can assemble the code and transfer the hex file to the PIC. You can do this by putting the chip into a suitable programmer and flashing the code into the PIC. I used the MPLAB IDE software and a PICSTART programmer to do this but once the PIC was fitted in the PCB circuit I used an in-circuit programmer (ICSP, MPLAB PM3). There are four wires on the ICSP and you need to check the connections on your programming cable. Note that you need to provide power to the PCB while programming with the ICSP and then you need to disconnect the ICSP for the circuit to work.

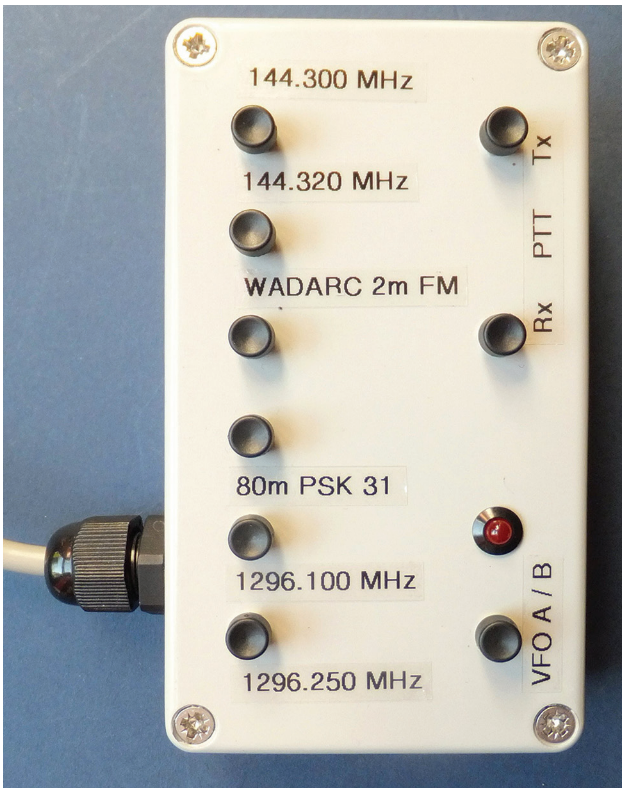

PHOTO 3: Close-up of the controller front panel. You can select your own operating frequencies by altering the PIC code (see text).

CONSTRUCTION. I built the device into a small painted dicast box with the CAT cable going out from the back. I used a PCB, but the circuit is so simple that it could easily be made on Vero or perf board or even birds-nest style on top of a copper clad PCB. For this reason, no PCB is being published here. I covered the back of the box in a thin layer of rubber matting to avoid slipping.

A brief description of the CAT cable specifications and pin connections at the back of the transceiver can be found in the FT-817 manual. Note the socket is the ACC socket, not the DATA socket. Instead of buying a cable and the 8 pin mini DIN plug I brought a pre-wired 8 pin mini DIN to 8 pin mini DIN lead from eBay as they are only a couple of pounds each. The lead I used seemed to be a cross-over lead so if you cut the lead in half, to wire it into the PIC circuit, the colour of the wires to the 8 pins on one plug do not match the colours on the other. So you need to check the connections (I have some info on my website [2]).

Note: don't be tempted to buy a generic 8 pin mini DIN to D type CAT lead, even though they look like they would be ideal, as they don't seem to have enough wires connected up for use with the FT-817.

POWER. The ACC socket provides power for our project so you don't need an extra battery to power the controller but a small 78L05-type regulator is required to drop the supply to 5V. One issue using the ACC socket to provide power is that it does not shut off when you turn the radio off. So if you leave the device plugged in it will drain the battery. I had thought of including an on / off switch but it seemed to me that one would be as likely to forget to turn this off as remember to unplug the cable.

COST. I used a quality painted dicast box and mini push switches from RS Components. You could make the project more economically if you use cheaper components such as found on eBay etc. The PIC is only a few pounds and provided you have a suitable programmer you could make a budget version of this project for less than £15.

TABLE 1: Summary of PIC16F688 port connections.

Switch number	PIC Port	PIC pin number
1	PORT A0	13
2	PORT A1	12
3	PORT A2	11
4	PORT C0	10
5	PORT C1	9
6	PORT C2	8
7	PORT C3	7
8	PORT C5	5
9	PORT A5	2

WEBSEARCH
[1] G4JNT page: www.g4jnt.com
[2] my PIC web page on my website: www.creative-science.org.uk/pic.html
[3] search for 'MSCOMM' and 'WidebanderV4' to extend the FT-817 frequency range.
[4] see the Microchip website www.microchip.com for the PIC16F688 data sheet
[5] see the Yaesu FT-817 operating manual for more details of the CAT commands