Extending VKCL software for easy logging of microwave contacts

= Or -

A TOP16 Compatible Module: The VK4ADC Method.

Feb/March 2016

Mike VK3AVV advised a while ago that VKCL V4 will have proper inbuilt support for controlling or sensing transverter selection through the use of a TOP16 USB I/O module. The concept is based on the availability of two 8 bit ports on the TOP-16 device, one specifically for input and the other for output, the latter capable of driving relays of some kind. The TOP16 device is connected to the computer via a USB port but the story doesn't quite finish there.

Firstly, VKCL needs to have the TOP16.DLL library in the same folder as the VKCL4 executable (i.e. VKCL4.EXE) for the for the transverter support to function at all. The absence of the TOP16.DLL is reported and it simply doesn't provide the functionality to "Save and Activate" the settings.

Next, the TOP16 will report a value from the input port back to VKCL4 and the Transverter configuration page has to be set up so that these values mean something to the software in terms of which band is currently selected. The Local Oscillator (LO) frequency needs to be set up for each transverter too so that VKCL can do the maths and the correct frequency is logged.

Finally, the selection of the transverter can be automatic as set up by switching relays on the 8 bit output port, or in my case it is a manual selection by virtue of a rotary switch which does three things: (1) switches the IF from the microwave IF transceiver to the relevant transverter; (2) switches the transceiver's PTT across to the same transverter; and (3) switches +12V to different output pins/terminals. It is this last function that I need to use to extend the use of VKCL for my microwave operations.

My microwave selection switch is labelled (1) 1296, (2) 2403, (3) 3400 [or 3398 eventually], (4) 5760 and (5) 10368 plus one unused position that might get used at some future time. By extending the +12V signals for each transverter frequency to the TOP16, I can get VKCL to automatically display the correct frequency and log it with minimal interaction but more importantly, no band selection errors which have plagued me in the past when I have forgotten to push a VKCL Band radio button..

The question that arises is "what if I don't have a TOP16 module ?" or the money to buy one ($100+ at this point in time). VKCL4 only recognises the commercial TOP16 devices - but would a compatible device work ? Where might you find a compatible unit you can purchase? My answer, nowhere!

Being a bit of an enterprising type, I researched as much as I could about the TOP16 and downloaded descriptive documents and source code in C (not that I program in C) and worked out basically what the TOP16 did but not necessarily how it did it. I also needed to find out more about the FTDI FT232RL chip too because it was the connection medium to the USB port. Visitors to my web site will probably have noticed that I use PICAXE chips in a lot of projects and since these are programmable microcomputer chips with multiple port pins, why not try to build a functionally compatible unit based on a PICAXE in lieu ?

I 'drew' up a PCB layout using ExpressPCB and based on a FTDI FT232RL SOIC-style device, a 28 pin SOIC PICAXE 28X2 plus a Micro-USB type B USB socket and all of the normal SMD resistors and capacitors typically in 0805 format. The artwork was then printed on single-sided photosensitive positive PCB material, developed, etched and then built up. The hardest part of the assembly was getting the FT232RL into place on the SOIC pads simply because it was a very tight fit. I subsequently altered the PCB layout so that there was more 'meat' extending these pads a little and that will make it easier if I build up further boards.

The 28X2 board was built up with a USB port sharing a high speed serial function on the PICAXE chip, a single main 8 bit port connector, an I2C port plus 4 pins of A/D / Input / Output, a serial programming port. I have to state that the designation of the 8 bit port functions was completely unknown at the beginning and only became clearer the further I advanced the project. The input pins were eventually fitted with a series 15K resistor then a shunt 5.1V Zener so that it would work with input voltages up beyond +15V without damaging the PICAXE input pin. The output pins (7 & 8) had 330 ohm resistors fitted in lieu of the 15K so a LED could be connected directly to either of them.

Ok, now I had a project built physically but no software to run on it - but I had a few insights as to what I needed to do to create some. Firstly I had to find out more about what the TOP16 software sent to their module so I set up some code in the PICAXE that would echo the USB-derived serial data back to the PICAXE programming port, ran terminal software on that

and then ran the TOP16 Manager software. No, it didn't see my project board. Damn. Fortunately I had downloaded an FTDI utility in Delphi (my preferred programming language for Windows applications) so tried that. It would see the FT232RL so I knew that part was working and I could send a text file to the PICAXE and after working on the baud rates for a while, finally was able to read that and echo it back to the programming serial port.

I thought I was at a project dead end until I realised that there must be a fundamental reason why the FTDI software was seeing the FT232RL chip but the TOP16 Manager wasn't. That left how the FT232RL presents to Windows and a wonderful utility from FTDI called USBView started to help out to show me how Windows saw the USB port device. So what were the differences that the TOP16 module presented? There was nothing in the way of that type of info on the web itself so I posted a question on the VKLogger Forum asking for anyone who had a TOP16 to run USBView and do a screen print of the results. A few days later I received a graphic by email: a Top16 port view. Once again, FTDI came to the rescue with their Programming package which allows you to alter some of the characteristics of the FT232RL and what it presents to Windows. My first reprogramming wasn't quite there, my USBView screen didn't quite correspond with the graphic so I tried it again with a few more changes. Yep, it matched. I unplugged my module from the USB port and then put it back in just so it reset itself to Windows. I ran the TOP16 Manager and there was the module showing and it was only at that point that I started to think positive about it all once again.

After working on the baud rates and port settings for a while, I started to show actual data exchanges from the Top16 Manager software at 115.2KB… I could see the data was formatted into a series of commands polled to the module. One was #SSSS<cr>, another #0000<cr> and a further one #FF00<cr> – but wait, by changing an output pin setting in TOP16 Manager, the data was altering too. Now it was #FF01 or #FF02…

It was time to go deeper into the supplied C source code and it revealed that the TOP16 response is always in the format >XXXX<cr> where XXXX represents the reply data and the whole reply is always 6 characters long.

The PICAXE code was improved so that the incoming stream was parsed for the TOP16 command and a suitable 6 character reply was generated. Initially it was dummy info sent back but the TOP16 Manager software started to show the settings I had hard-coded into the PICAXE's data for the input port commands. The other functionalities built into the original module were not really of too much interest to me except when I recalled that I wanted to use two pins of the 8 bit port as outputs, the remaining 6 as inputs.

The code was improved more so that it read the 6 inputs and placed real values (0 to 32) into the replies from my device, the two outputs were settable with values of 64 and 128, on port pins 7 and 8 respectively.

The final step was to try it with VKCL. I used VKCL3.13 and it showed that it could find my board so it loaded the Top16.DLL and showed my board's serial number. Then I hit a full stop. The transverter support finished at that point in that version.

I contacted Mike VK3AVV (the author of VKCL) and advised him as to my progress on this project and he subsequently provided a beta copy of VKCL4 for me to try out the transverter support. I duly installed the code, copied across TOP16.DLL and ran the code. Initially I thought there was no change but then I hit the Transverter setup screen and it all *happened*. It said the DLL was loaded and the Save and Activate button was not greyed out. I put in the values 1, 2, 4, 8, 16 in the first (Input) column, 128,128,128,128,128 in the second (Output) column, selected Radio1 then put the relevant LO frequencies in the next column (1151.000, etc..) and finally 23, 13, 9, 6 and 3cm in the last column. Just to be careful, I used Save first and then the Save and Activate. The main screen now showed a Transverters tick box !

For info, an input value of 1 means Pin1 is high, 2 is Pin2 high, 4 is Pin3 high, 8 is Pin4 high, 16 is Pin5 high etc. The output 128 values are set so that a LED indicator on Pin8 and mounted on the microwave switching panel will indicate whether the interface is active or not.

The radio was set to 145.150 and Omnirig was set running so I ticked the box and nothing happened/changed. Hmmm. I used a clip lead on my module on Pin1 to +5V and the frequency changed from 145150 to 1296150. Shifting it to Pin2 and it changed again to 2403150. Shifting it to Pin3 and it changed yet again to 3400150. Shifting it to Pin4 and it changed once more to 5760150. Finally, shifting it to Pin5 and it changed to 10368150. WOW ! This was going to work out.

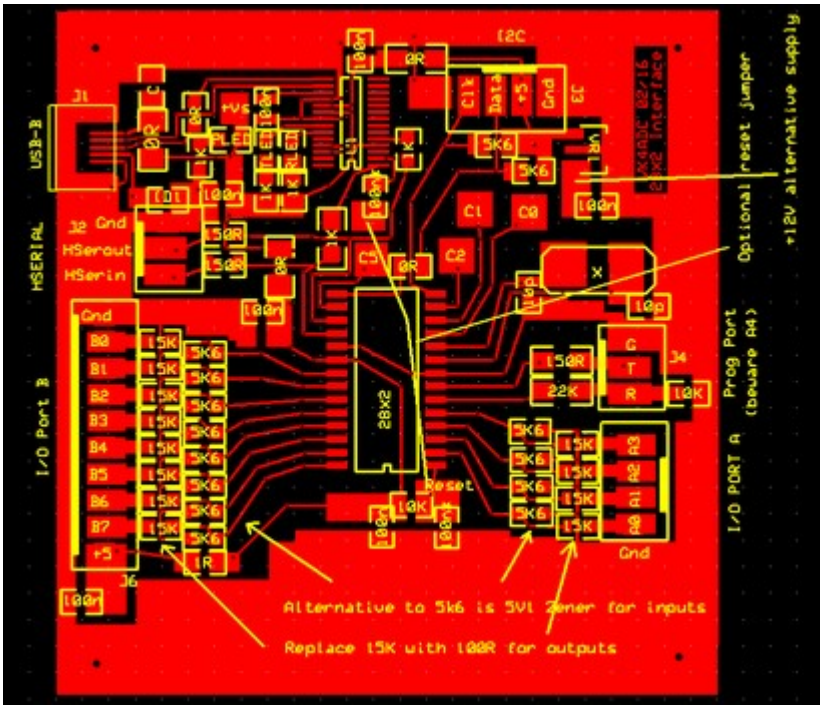
My final steps were to wire the switched +12V connections from the rotary band switch to my module via a multiway connector, plug the module into the powered USB hub and cross-checking VKCL for correct display.

Summary:

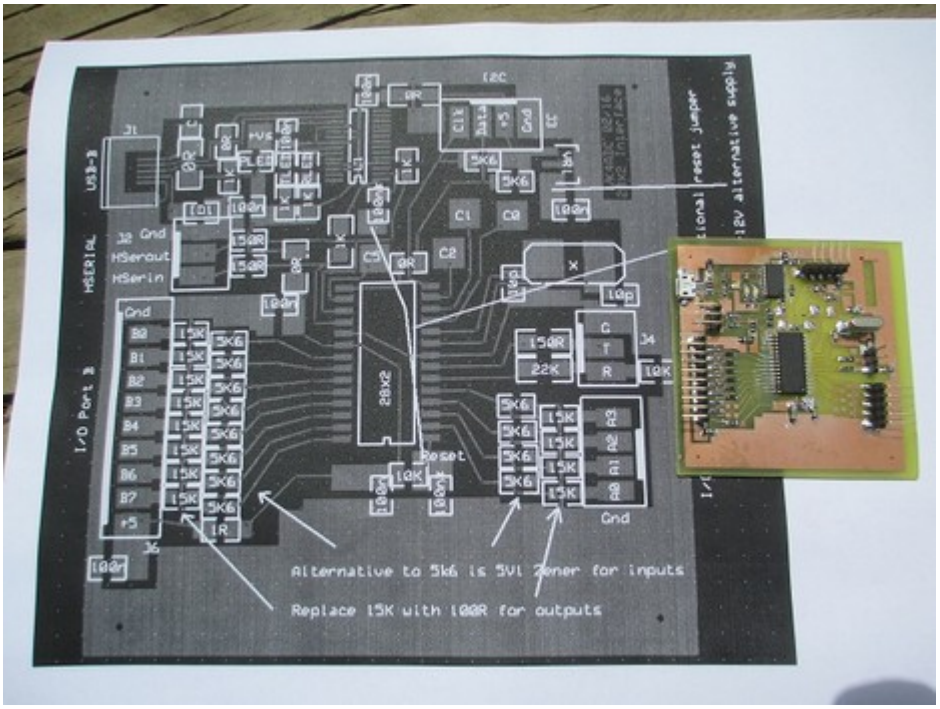
Just looking at the PCB itself, the cost of the bits on it total probably $30-$35 based on a good SMD parts box for the typical R and C parts, a real FT232RL at about $6.60 (RS Components), a PICAXE 28X2 SMD at $15 (Wiltronics), a Micro-USB Type B USB socket at $3 (RS again), the photosensitized PCB material about $6 (RS again). No labour in that costing but well over an hour in build time was involved, particularly since it was a prototype with nothing to copy from except the PCB parts layout diagram.
I have also done the PCB layout for a PICAXE 40X2 SMD (44TFQP package, $16 at Wiltronics) as well and that gives more port pins, certainly separate 8 bit in and out ports plus the I2C port etc.. and will use virtually the same overall PICAXE coding. The TFQP chip will be harder to solder to the PCB but has the advantage of the extra 8bit port, the

output port being buffered by a ULN2803 8-port device.

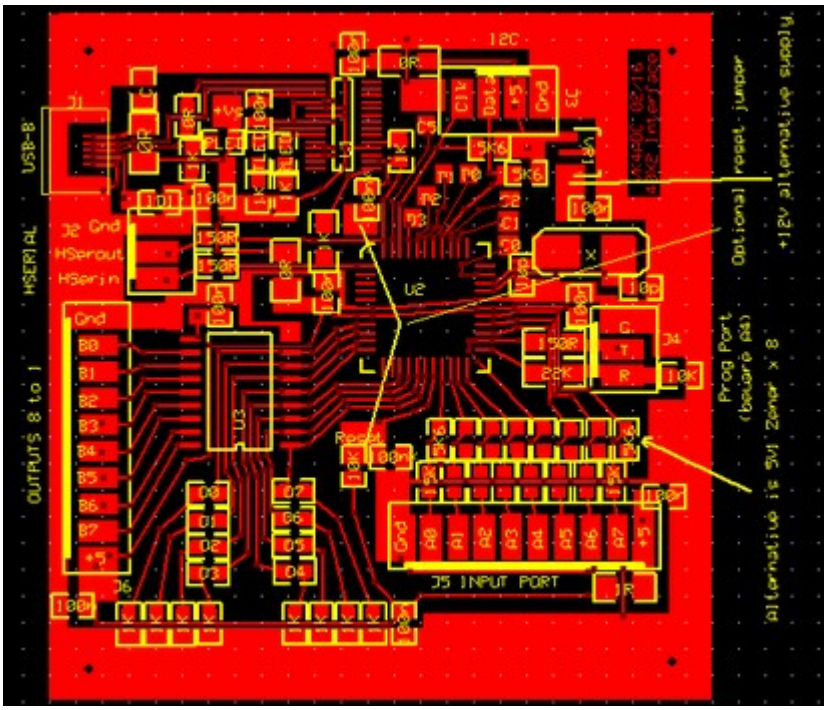
Am I making and selling these boards ? No, I don't plan to. The aim of this article is to show that it can be done.



The 28X2 PCB layout...



The 28X2 board against the layout (A4-sized sheet)



The 40X2 board layout, not built at this stage (Feb 2016). Note the availability of the extra input port at bottom RHS and the use of an octal driver to the output pins @ centre left. The board size is identical to the 28X2 layout.