

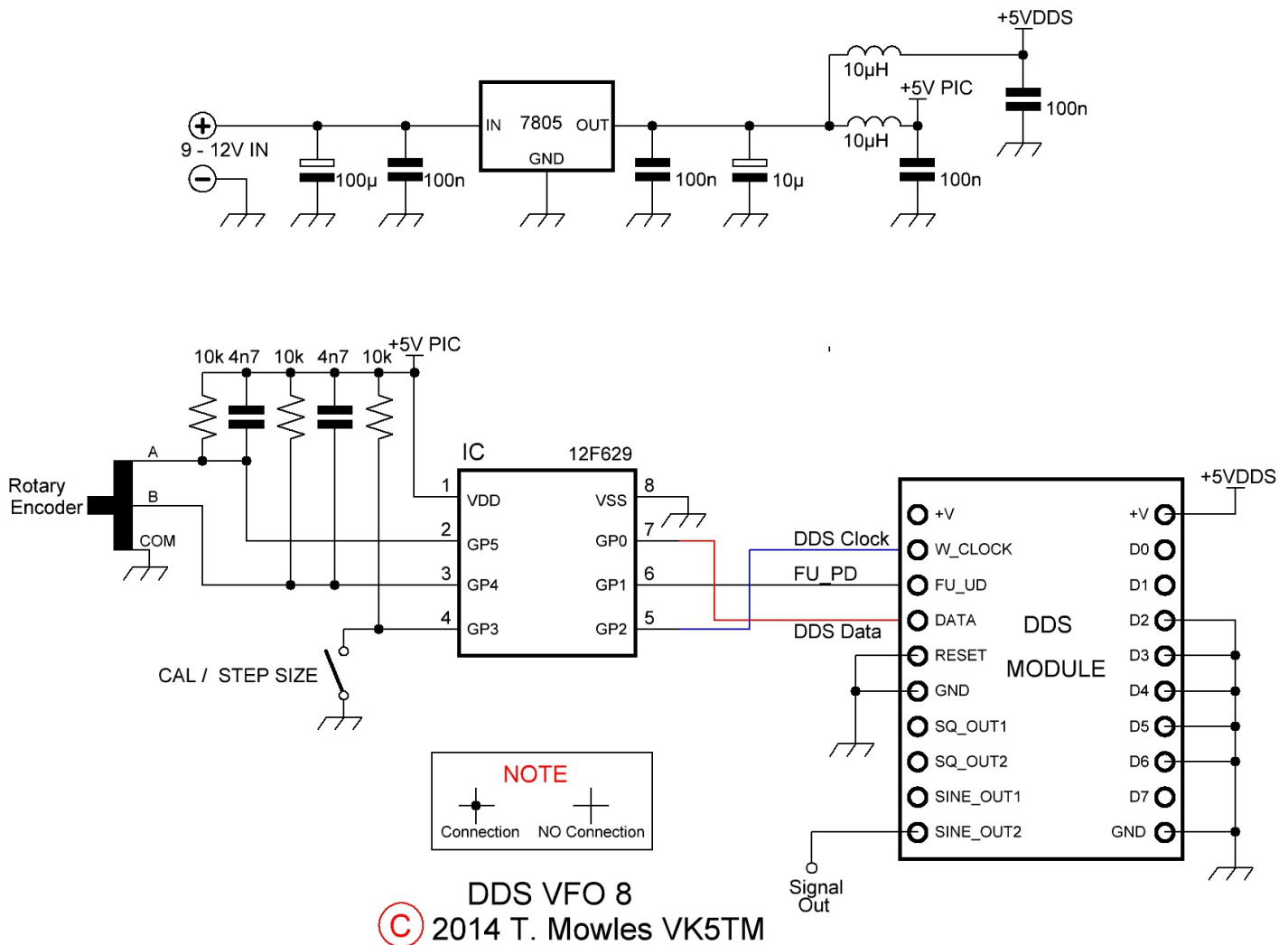
# VK5TM Simple DDS VFO

This a copy of the original webpage for the VK5TM Simple DDS VFO.

## A Simple DDS based VFO using an 8-pin PIC

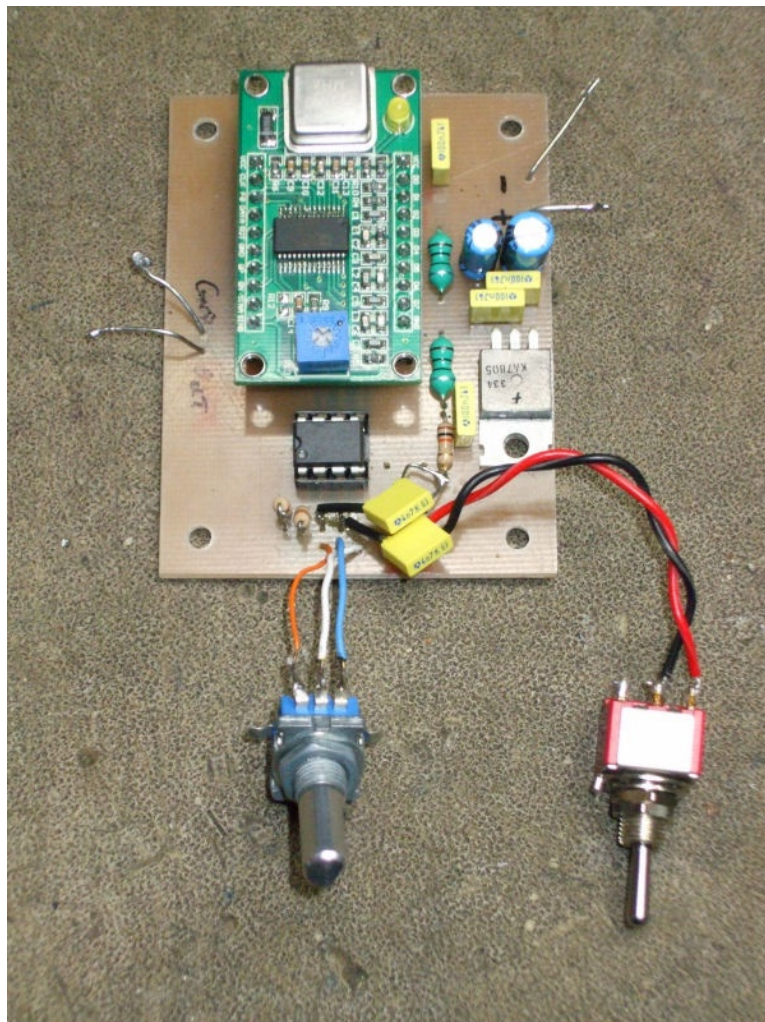
I have updated the software again (June 6th 2014). See the Software update notes near the end of the page. File at the bottom - DDS\_VFO\_8c.asm. November 26th 2016 - Version 8d has been added - it uses the encoder pushbutton to change steps - see notes with the download at the bottom of the page.

A visitor to my website mentioned that he was looking for a simple DDS based VFO. He didn't need the LCD display or any fancy goings on in software and this is what I came up with.

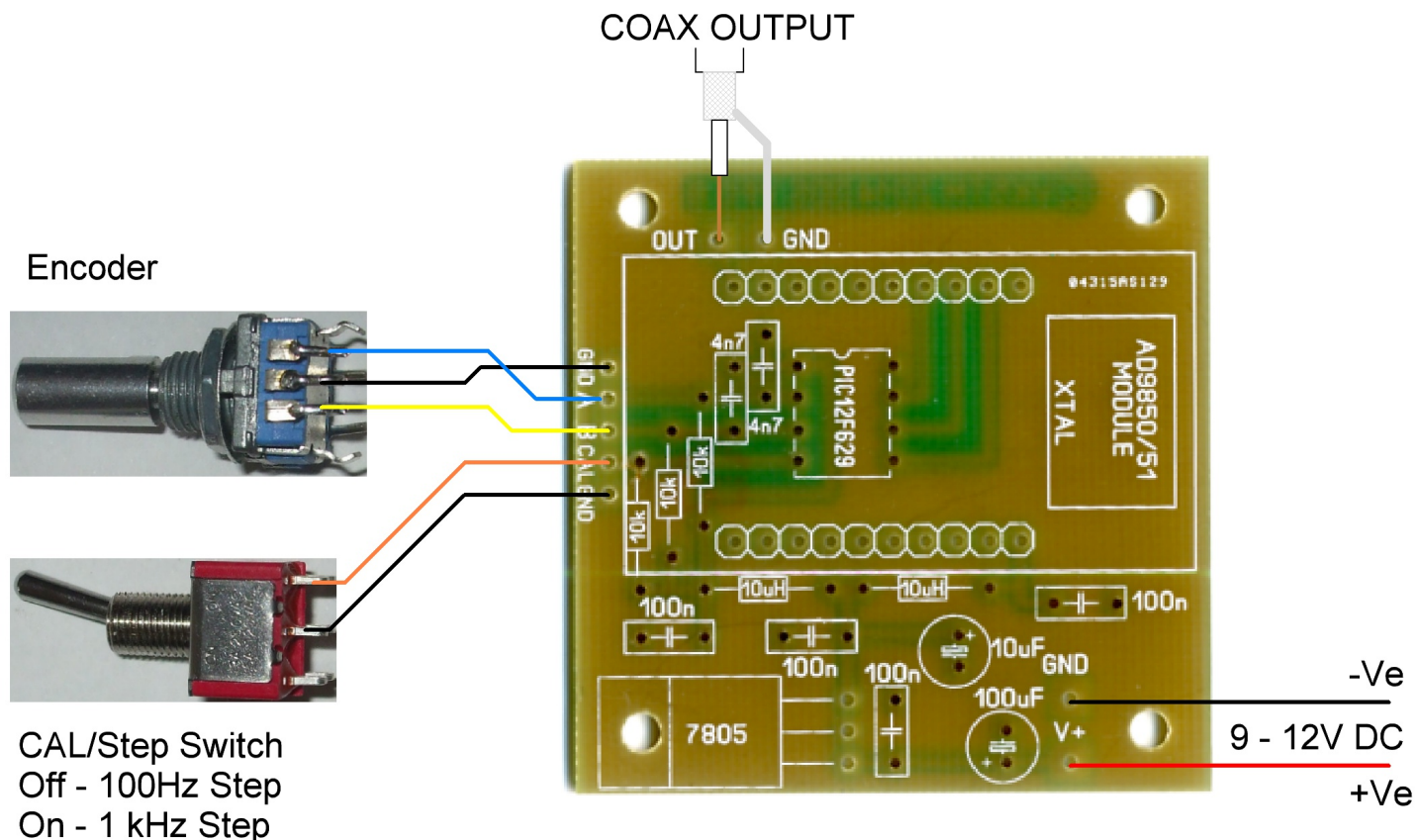


If you have one of the 3.3v modules, all that needs to be done is to replace the 7805 with a 3.3v regulator. The PIC will happily run at 3.3v and it doesn't control anything but the DDS module, so no other interfacing problems.

It is a VFO that can be used just as an ordinary transistor/fet VFO would be. This has been kept as simple as, any buffering, amplification or filtering is left up to you to build to suit your purposes. The pcb is only 2 inches (50.8mm) square. You can use either the AD9850 Module or the same format AD9851 module.



I have been asked for a hook-up diagram, so here it is.



## Software

A few words about the software.

You will need to input the values of  $\alpha$  and recompile the software for your required VFO frequency:

Upper limit - the upper limit frequency you want the VFO to reach.

Lower Limit - the lower limit frequency you want the VFO to go down to.

The default start frequency - this will be overwritten during operation by your last used frequency, it is just the first time start up default (it must be within the VFO range above).

The Calibration frequency - This is the frequency the unit needs to be adjusted to during the calibration procedure. I suggest making it in the middle of your VFO range.

If you use MPLab to compile the software, the above are between lines 120 to 149 in the asm file. (If these do get put out as a kit, I will program your frequencies in if you cannot do it, but calibration is up to you).

In the file available to download below, the frequencies are 5.5MHz, 5.0MHz, 5.0MHz & 5.250MHz in order as listed above. There is no limit to the frequency range of the VFO other than that of the AD9850 or AD9851, so your limits could be 1Hz and 40MHz if that is what you wanted, but it will take a very long time to get from one end to the other in 100Hz/1000Hz steps.

One point to note: Until the calibration is done, you need to make sure that the 'CAL' input is not held low when first turned on. Otherwise the unit will enter 'calibration' mode.



Once the calibration has been done, this restriction does not apply.

If you do not want to do a calibration, all you need to do is switch the Simple VFO into CAL mode and back out again without making any changes (see below).

The software 'should' work with either mechanical or optical encoders (I don't have an optical encoder to test with. Some kind soul may like to donate one). You need to remove the pullup resistors and associated caps on the encoder input lines to use an optical encoder (unless your encoder data sheet says otherwise).

## Calibration

Once you have built the unit, it really should be calibrated before you fit it into anything. Calibration requires the use of an accurate frequency counter with 1Hz resolution and is used to adjust the "OSC" values in software to correct for off frequency Xtals on the DDS modules.

Connect the frequency counter to the output of the unit and short the pads marked "CAL". The short must not be removed until calibration is complete. I would suggest you wire an on/off switch across the pads.

With the pads shorted (or the switch "ON"), apply power. The frequency counter should show a frequency somewhere in the vicinity of the calibration value put into the software.

Go away and have a cup of coffee, maybe a biscuit or 2 or what ever your choice. Once you have left the unit to run for a minimum of fifteen minutes, then continue. Or in other words, give it time to warm up before doing any adjustments.

Turn the encoder in which ever direction is required to change the frequency so that it becomes your calibration value. Note that the oscillator calibration steps are small, so you may have to turn the encoder a fair number of turns to reach your calibration frequency.

Without turning the encoder further, remove the short (or turn the switch "OFF") and then turn the encoder again (direction does not matter). This will save the new computed "OSC" values into EEPROM and the unit will now be functional.

At this point, you can remove power and fit it into what ever it is you are going to use it in.

## Calculating and entering different frequencies

I have had an email asking for some extra information regarding this project. While I haven't had a reply as to what extra info is wanted yet, I figure it may help some as to how to calculate and insert "the numbers" for different frequencies. For those with Windows systems, it is easier to use the upgraded scientific calculator available for download (in "Programmer" mode). Users of other operating system (or those that don't want to download the calculator), will need a decimal to hex converter program of some sort (unless you are lucky enough to be able to do it in your head).

In the DDS\_VFO\_8 asm file, starting at line 120, are several 'EQU' statements for upper frequency limit, lower frequency limit, default start frequency and calibrate frequency. I will use the values currently in the file to show you how to get the numbers. The following sequence applies to all four of the 'EQU' sections (with different numbers of course).

Firstly, enter the dec number 5500000 into your dec to hex calculator/program (I'm not going to tell you how to work your calculator/program, you need to work that out).

The HEX number will be 53EC60

Now break this number down into groups of 2, starting from the right hand side:- 53 EC 60  
The right hand number is the least significant byte, so you enter 0x60 into the line that says, funnily enough, least significant byte. Enter 0xEC in the next line up and 0x53 into the line above that. The '0x' tells the MPLab compiler that the number is in HEX format.

So, where is the most significant byte? Well, in this case, it is 00, because 5500000 only converts to the three bytes just shown. So, if your calculation only gives you three nice even groups of two numbers, the most significant byte will be 0x00.

Lets do another one that doesn't give you three nice even groups of two numbers.  
Enter 17000000 (seventeen million) into your calculator/program.

It should come back with the HEX number of 1036640.

Breaking that down into groups of 2 as before gives you: 1 03 66 40. Not so nice even groups of two numbers plus an odd one.

To fix that, you put a '0' in front of the 1 to get 01 03 66 40. Because nought is nought (or zero is zero), it has no effect on the calculated value, so you can enter the numbers as before, but the most significant byte will now be 01. Don't forget to put 0x in front of your numbers, otherwise, strange things will happen.

Using the AD9850 or '51 modules, the most significant byte will not be any more than 03 hex (which puts you upwards of 50MHz).

## Software update

I have cleaned and updated the software.  
The following two features have been added:

1: Once the calibration has been done, there is no need to make sure the 100 Hz / 1000 Hz switch is in the 100 Hz position at power-up (Note: Until calibration is done, you still need to make sure the switch is in the 100 Hz position). The only downside to this is, if you want to do another calibration, you need to reprogram the PIC. I am looking at ways of changing that.

2: If the frequency doesn't change for ~2 seconds, the PIC will go to sleep. While it won't change much in regards power consumption, it may help reduce spurious noise from the PIC. The save last frequency function is still there, the PIC goes to sleep after performing this function.

As a result of the clean-up, I have removed the original files. If you really want them, contact me and I will send the to you.

## Downloads

These files are provided free for personal use ONLY. I retain all copyright on all works published on this website. They may NOT be used in any commercial or profit making

enterprise of any kind without the express WRITTEN permission of the copyright holder.

I am aware of at least one outfit making copies of my pcb's without even the decency of asking permission!!!!

## RE-PROGRAMMING PROBLEMS:

I have had a couple of reports of people not being able to re-program their PIC's. This is a programmer problem. Some programmers do not explicitly follow the programming protocols of these chips and apply both Vdd (5v supply) and Vpp (programming voltage) at the same time. When these voltages are applied in this fashion, the PIC starts to run the program and the programmer is unable to capture the pins necessary to perform the programming. (Also, some programmers, in "Erase" mode will wipe out the "OSCCAL" value in the chip which will also cause problems. Do an internet search on Osscal calibration and read up on fixing this problem)

There are several solutions to the re-programming problem:

1/ Look for a menu option in your programming software to see if there is an "Assert Vpp first" or similar worded option.

2/ If that option is not available, you may be able to modify your programmer by fitting a 47 - 100 ohm resistor in series with the 5v line to the PIC and adding a 47uf cap to ground at the 5v pin of the PIC. DO NOT try this if you are not sure or don't have the requisite skills to do it. Note that I have not tried this, but it is mentioned as an option on several forums dealing with PIC's and programming.

3/ Borrow or buy a programmer that is capable of asserting Vpp first. The PicKit 2 & 3 are two I know of that can do this and I have confirmed that they do work to rescue the PIC.

[DDS\\_VFO\\_8c.asm](#) The 12F629 version ASM file for the VFO. It is set to AD9850 by default. Instructions in the file to change it, including to reduce step size to 10Hz. Default VFO range 5 - 5.5MHz.

AD9850 hex file: [9850\\_DDS\\_VFO\\_8c.hex](#)

AD9851 hex file: [9851\\_DDS\\_VFO\\_8c.hex](#)

**Version 8d of the software** - this came about at the request for a version that used the pushbutton on the encoder to switch between steps. As in version 8c, the frequency range is 5 - 5.5 MHz, but there are now three step sizes - 10kHz, 1kHz and 10Hz. Just wire the encoder pushbutton in place of the toggle switch. At start-up, the default step size is 10kHz. Push the encoder button once and the 1kHz step size is selected, the next push selects 10Hz step and another push will return it to the 10kHz step size. The new section of code is marked in the asm file (two places) if you want to change the step sizes.

Currently, the sleep function is disabled, as it interferes with the button function. When I have worked out why, it will be reinstated.

You will need to add a 100n cap across the encoder switch contacts. This can be done either directly on the encoder or under the pcb between the input pad and ground.

Also, you will still need to temporarily use an on/off toggle switch to do the calibration.

[DDS\\_VFO\\_8d.asm](#) The 12F629 version ASM file for the VFO. It is set to AD9850 by default. Instructions in the file to change it. Default VFO range 5 - 5.5MHz.

AD9850 hex file: [9850\\_DDS\\_VFO\\_8d.hex](#)

[12F675\\_DDS\\_VFO\\_8c.asm](#) The 12F675 version ASM file for the VFO. It is set to AD9850 by default. Instructions in the file to change it. Default VFO range 5 - 5.5MHz.

AD9850 hex file: [12F675\\_9850\\_DDS\\_VFO\\_8c.hex](#)

AD9851 hex file: [12F675\\_9851\\_DDS\\_VFO\\_8c.hex](#)

The PCB layout. See the PCB Info page for information on using this file.:-

[DDS\\_VFO\\_8.lay6](#) The Sprint Layout 6 pcb file.

A zipped copy of the above files can be downloaded from the current Simple DDS VFO webpage [http://www.vk5tm.com/homebrew/dds\\_vfo\\_8/dds\\_8.php](http://www.vk5tm.com/homebrew/dds_vfo_8/dds_8.php) under the Old Files heading.

© 2017 T Mowles VK5TM