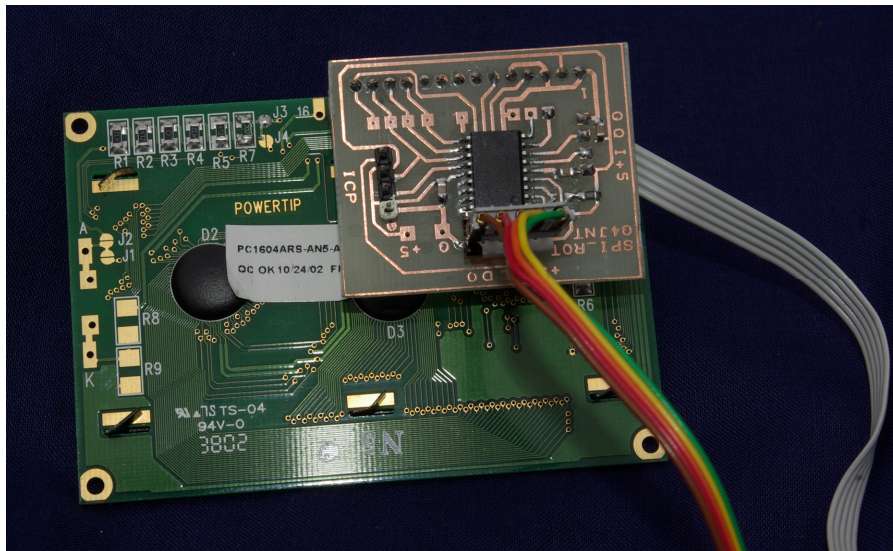


Rotary Encoder / LCD Control for LMX2541 and LMX2470 Synthesizer Modules

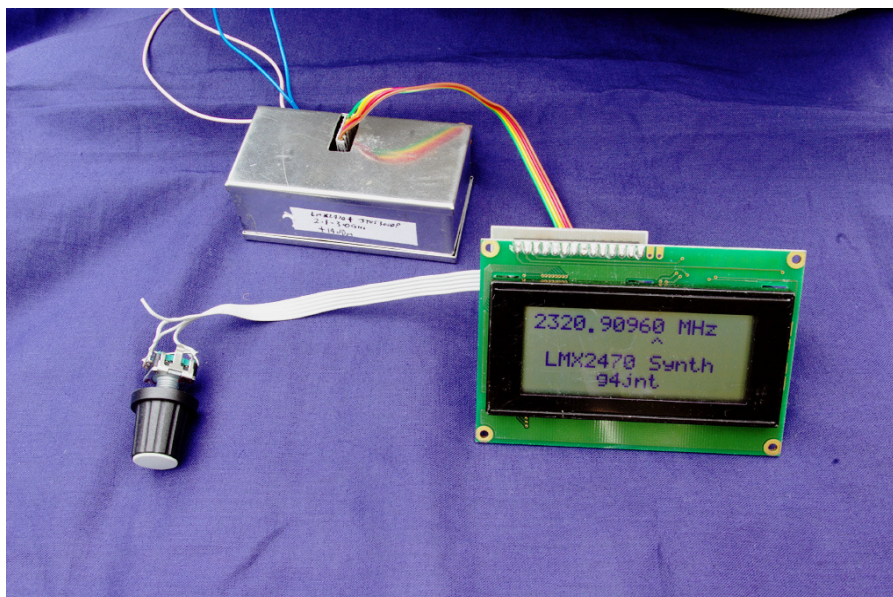
Andy Talbot G4JNT May 2013

Both of these two FractN synthesizer designs originally started out with a 12F629 PIC based operating system that allows users to set individual registers with ASCII based commands on an RS232 interface. PC software allowed register values to be calculated and sent to the controller over this interface, with the option of storing these in EE memory for immediate loading at switch on. On the LMX2470 this PIC resides on the PCB with the RS232 interface going to a four-way header. On the LMX2541 board, the



controller has to be installed externally on a daughter-board. The header on the main PCB carries just the SPI signals direct to the synth chip.

This design uses a rotary encoder and LCD module to allow the synthesizer to be set to any arbitrary value using just a quadrature tuning wheel and pushbutton. The hardware for each is identical, with a 16F628 PIC on a PCB designed to mount directly onto the 14 pins of a standard alphanumeric LCD module as shown in the photograph. The PCB is identical to that used for the AD9850 DDS controller described at Ref [1].





Only the registers on the synth chip that are directly applicable to frequency setting are programmed. All others are stored in non volatile EE memory that have to be defined and blown in using a PIC programmer. Therefore this controller cannot be supplied as a “plug and play” unit. Each will need to be customised to your own purposes.

Frequency Setting Algorithm

Both versions use a similar technique to set the FractN registers. The synthesizer is set to tune in steps of 10Hz and this is the basic resolution of the system. The output frequency of a FractN synthesizer is equal to :

$$F_{OUT} = F_{REF} / R * (N + F/D) / O_{DIV} \quad (O_{DIV} \text{ is the output divider, only applicable to the LMX2541})$$

Assuming $F_{REF} = 10\text{MHz}$ and $R=1$ for a 10MHz comparison frequency, then to get 10Hz tuning steps at the basic synthesizer level means the value of D has to be 1000000.

There are two ways of converting between displayed frequency (F_{OUT}) and the values of N and F that have to be sent to the synthesizer chip. Both are derived by first rearranging the above equation:

$$F_{OUT} = F_{REF} / R / D * (N.D + F)$$

Noting that $F_{REF} / R / D$ is the tuning step, this simplifies to :

$$F_{DISP} = N.D + F \quad \text{with the display in units of 10Hz.}$$

Showing the correct frequency is then just a matter of placing the decimal point appropriately in the display. Step sizes other than 1Hz, 10Hz, 100Hz etc. would need an additional multiplication or division.

The first method of calculation involves having a counter containing the value of F_{DISP} which is incremented or decremented by adding or subtracting values corresponding to the tuning step each time the rotary encoder is operated. A 32 bit counter allows a count up to about 4.3 billion which with 10Hz steps means frequencies up to 43GHz can be catered-for.

F and N are then calculated from : $N = \text{INT}(F_{\text{DISP}} / D)$ and $F = \text{REMAINDER}(F_{\text{DISP}} / D)$

For the LMX2541 F_{DISP} has to be replaced by $[F_{\text{DISP}} * O_{\text{DIV}}]$ in the above equation

The second method is to maintain N and F counters, with F incremented or decremented as before. However, F counts modulo D, so when it reaches a value of 999999, the next step resets F to zero and increments N. The displayed frequency is then calculated from $F_{\text{DISP}} = N.D + F$. For the LMX2541, a further division by O_{DIV} is then needed to show the correct final frequency value

Both methods are equally valid and make use of 32 bit integer maths. The second method uses multiplication, with a simple 8 bit division for the LMX2541 output divider only. The first method uses 32 bit division. As I didn't have a complete and tested 32 bit division routine in the library, but did have a suitable multiplication one, the second method was used here. It is also a bit simpler, and therefore faster in operation.

Each step of the rotary encoder increments or decrements the F / N count. The step size (the value by which the count is incremented /decremented) can be selected in decimal increments by repeatedly pressing the pushbutton. The digit being varied is shown by a caret under the associated position in the LCD (Photo 2, above).

When the output divider $[O_{\text{DIV}}]$ is used in the LMX2541, the value used to alter the F/N count has to be modified so the tuning step is maintained at the final output frequency. Therefore, the step value is multiplied by the value of O_{DIV} before adding or subtracting it.

LMX2470 Controller

The source code for this is contained in LMX2470_ROT.ASM which is commented sufficiently to be able to customise it to your own requirements. All the EEPROM contents for the fixed register settings are at the beginning of the file. For a standard unit with median loop bandwidth and FractN settings, no changes should be needed to the default values. (The default frequency of 2320.9045MHz will be changed as soon as you store your own setting)

Select the digit to be set (tuning rate) by repeatedly pressing the pushbutton momentarily. Use the rotary encoder to adjust at this tuning rate. The current frequency can be stored to non-volatile memory, to be recalled next time the unit is turned on, by pressing and holding the pushbutton for two seconds. The display momentarily shows “-SAVED-”

There are no upper or lower limits programmed into the controller – it up to you to make sure you stay within the lock range of your specific synthesizer system!

LMX2541 Controller

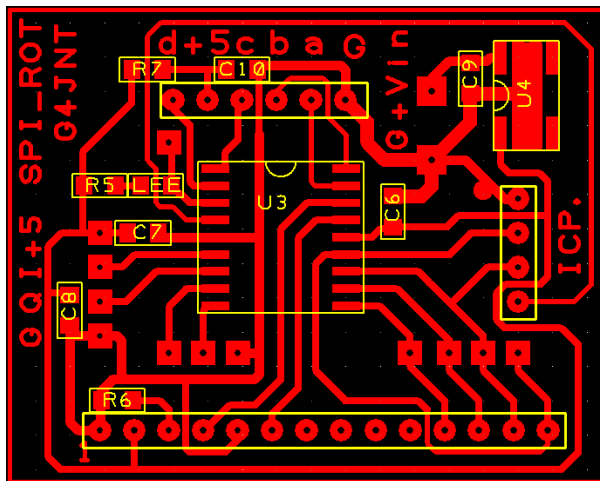
This is contained in LMX2541_ROT.ASM. This operates similarly to the LMX2470 version, but allows for the output divider to be set. To change the value of this, press and hold the pushbutton whilst turning on the unit. The display shows “Change and Store O/P Divider = /XX” where XX is the current value of O_{DIV} . Use the rotary encoder to cycle the divider value from 1 to 64. When the correct value appears in the display, press the button briefly to save this to non-volatile memory and return to normal operation.

At the time of writing, there is an annoying issue with high values of O_{DIV} and large tuning steps. A problem comes when the new increment /decrement value *after* it has been multiplied by O_{DIV} ends up

greater than D. The ideal solution would be to correct this by some complicated division and maths, but a simpler, more pragmatic solution was chosen. For values of O_{DIV} that could give values greater than D at certain step sizes, these steps are simply not allowed to happen. It is then only possible to cycle through valid step values with repeated pushbutton presses. To illustrate this, set a value of O_{DIV} greater than 9, then try selecting different step sizes. Notice how only values from 10Hz to 100kHz can now be selected. With $O_{DIV} = 20$, for example, a 100kHz step needs $20 * 10000 = 200000$. A 1MHz step would need a value of 2000000, which is greater than D. With O_{DIV} values from 1 to 9 tuning steps up to 1MHz are permitted. (Another boundary at $O_{DIV} = 40$ is also in place, limiting to 10kHz step maximum. Although not essential here, it was included because in the future this one might become necessary; and better to have the code in place than forget about it later!)

This solution is not ideal, and later versions of the software may change completely how the register and display values are calculated. One version under consideration is to allow seamless tuning from VHF to maximum and allow O_{DIV} to be determined automatically according to a lookup table based on the chip version in use, while testing-for and prohibiting illegal frequencies. This will require a major code re-write – but the same hardware will be used.

Hardware and Construction



PCB layout for SOIC packaged 16F628A PIC devices. The 14 pin header plugs directly onto the LCD modules available from [3], with all other connections made via pads or headers.

The finished unit is shown in the photograph above, with the circuit diagram in Figure 1. This is identical to that for the AD9850 DDS controller at Ref [1]. The PIC code in .ASM and .HEX formats for both LMX2470 and LMX2541 controllers can be found at Ref [2]. This archive contains the file ***SPI_ROT.pdf*** which is a 1:1 mirror-imaged monochrome plot of the PCB for direct laser printing to acetate or iron-on transfer for homebrew PCB construction. Suitable four line LCDs and a limited number of PCBs can be obtained from G3AAF at ref [3]. Many builders have reported that a two line LCD works satisfactorily

The PIC can be programmed through the four way header labelled 'ICP'. The connections are Gnd,

Data, Clock, Pgm. This programming interface is identical to all other PIC products from the 'JNT stable but, unfortunately, no one else's. So you'll have to make up a programming lead!

The LED serves no useful purpose in this design and can be left out. It is a hangover from an earlier project that used this PCB. Additional pin connections on the SPI connector and other unused pads are also a legacy of backwards compatibility.

The four connections from the controller header, (G, a, b, c) pass directly to the synthesizer PCB on a four-way ribbon cable. The connections for the synth module programming are defined as:

a = Data b = Clock c = LoadEnable G = Ground.

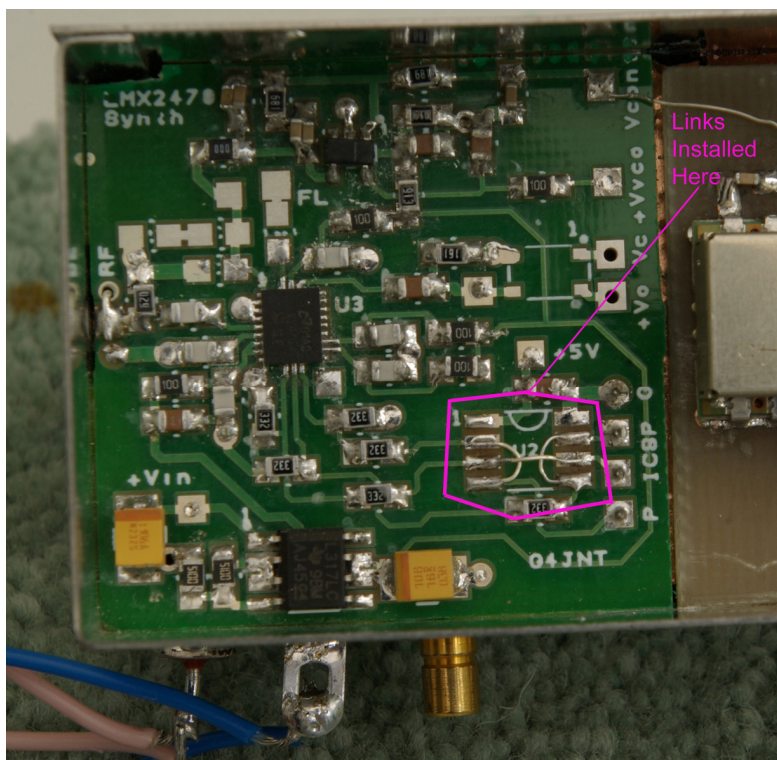
Both synthesizer PCBs contain level shifting resistors permitting 5V SPI control signals to be delivered to the chip at the right voltage levels.

If a +5V rail is used to supply the synthesizer's own voltage regulator input, this 5V may be picked up on the 5th pin, labelled '+5', and used to supply the PIC. In this case, leave out the 78L05 voltage regulator.

Rotary Encoder

The circuit diagram shows an optical or otherwise powered encoder. A simple mechanical switch type works equally well, ignoring any connection to +5V. There is no need for pull-up resistors, the PIC already has them enabled. However, to avoid contact bounce, two capacitors across the switch contacts will probably be needed. Use a value in the region 10 – 100nF.

Bypass the LMX2470 on-board controller.



The PIC mounted on the LMX2470 needs to be removed and the pins jumpered as shown, giving direct access to the SPI serial control.

Pads linked are: 2 – 4, 3 – 6, 5 – 7

The LMX2541 PCB already has direct connection for one-to-one linking.

References

- | | | |
|-----|---|-----------------------------------|
| [1] | http://www.g4jnt.com/AD9850-Controller.pdf | Controller for AD9850 DDS Modules |
| [2] | http://www.g4jnt.com/LMX2470_2541-Controller.zip | Full constructional archive |
| [3] | Kevin@avery03.fsnet.co.uk | 4-Line LCD modules |

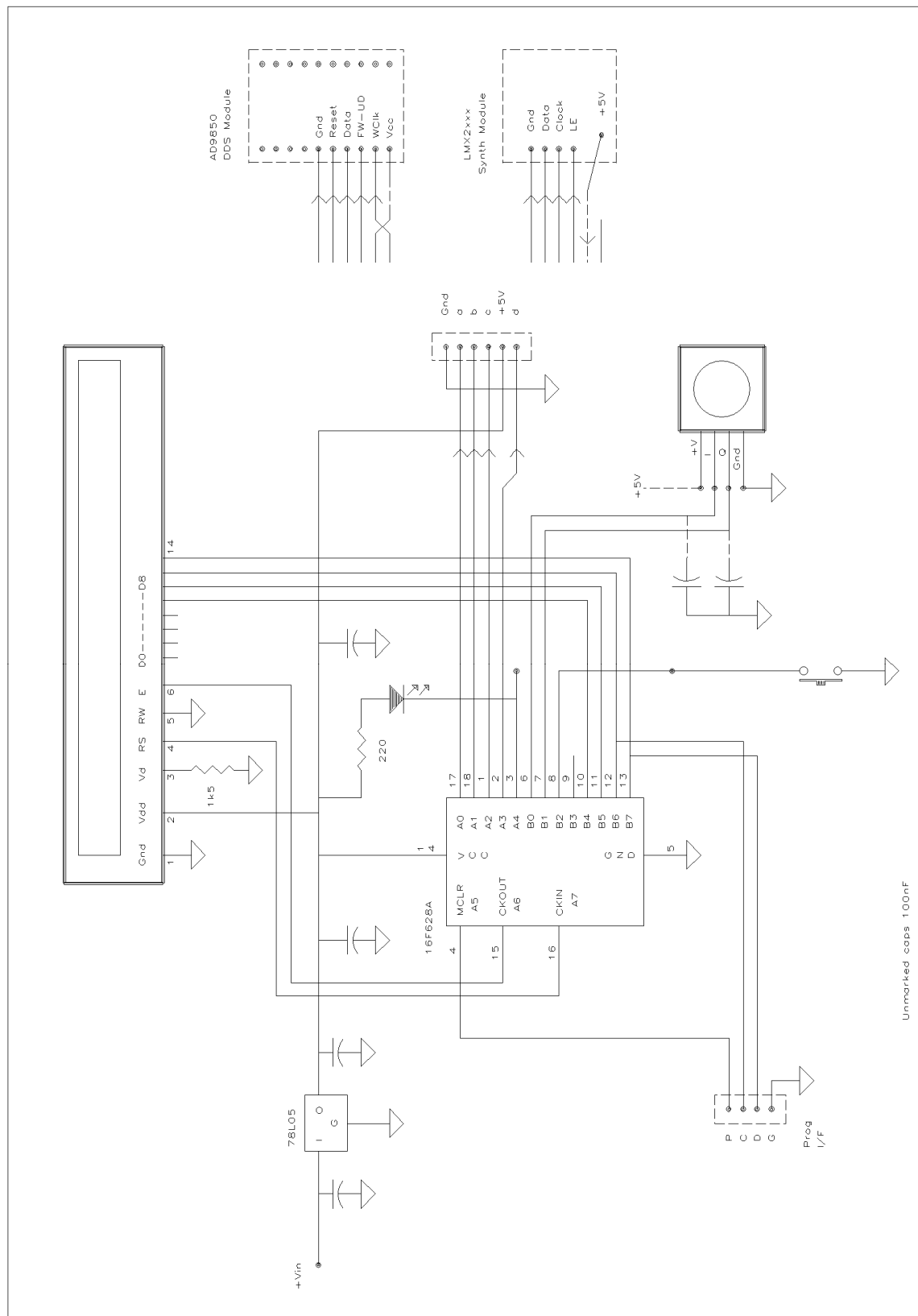


Figure 1, Circuit diagram of the PIC controller. Connections to the LMX2xxx synthesizer module are :
a = Data b = Clock c = LoadEnable G = Ground