

# Frequency Synthesizer Project

## ECE145B Spring 2008

The goal of this last project is to develop a frequency synthesized local oscillator for the 144 MHz amateur radio band using your VCO from Lab 2. The VCO will be locked to a stable crystal reference frequency (10.000 MHz) by a frequency synthesizer chip.

The frequency synthesizer board is provided. You will begin by interfacing your VCO with the synthesizer chip. Beef up the buffer amplifier to provide 0 dBm – needed for the synthesizer chip.

Fully evaluate the synthesizer performance. Here are the requirements.

### Synthesizer specs:

Frequency step size	20 KHz
Frequency range	133.3 to 137.3 MHz
Varactor tuning voltage range	2 to 12 volts
Overshoot	< 30%
Reference Spurs	Better than 40 dBc
Settling time	10 ms to 1%
Output Power	0 dBm
Crystal reference frequency	10.00 MHz

### PART 1. Frequency Synthesizer

Implement the synthesizer shown in Fig. 1 and evaluate its tuning and noise characteristics. You will use the MC145170-D2 CMOS frequency synthesizer chip, a 10.00 MHz reference crystal, and an LMC6482 CMOS dual rail-to-rail op amp. A type 2 third-order loop should be used. (read the lecture notes and Vaucher<sup>1</sup> if more explanation is needed). Refer to data sheets for details on use of the chips. A PC board is provided for this exercise.

The VCO and synthesizer are on separate boards for convenience of testing, but be sure you use coax or twisted pair wiring to interconnect the VCO control voltage input to the loop filter output when interconnecting the boards to avoid noise pickup which would modulate the VCO. Also, note that your VCO tuning voltage input port RC time constant will add an extra pole to the PLL. *You need to make sure it doesn't interfere with stability.*

\*\*\*\*\*

\* Note that CMOS chips are easily destroyed by static charge. This has been a problem in the past. **Do NOT handle the CMOS chip unless you are wearing a grounded wrist strap.** Once the chip is installed, you should handle your board only while wearing this strap as well.

\*\*\*\*\*

<sup>1</sup> C. Vaucher, "An adaptive PLL tuning system architecture combining high spectral purity and fast settling time," IEEE J. Solid State Cir, Vol. 35, #4, pp. 490 – 502, April 2000. (on course web page)



types as described in the lecture notes. In either case, design for worst case using the actual  $K_O$  measured at 133.3 and 137.3 MHz along with the corresponding  $N$  values.

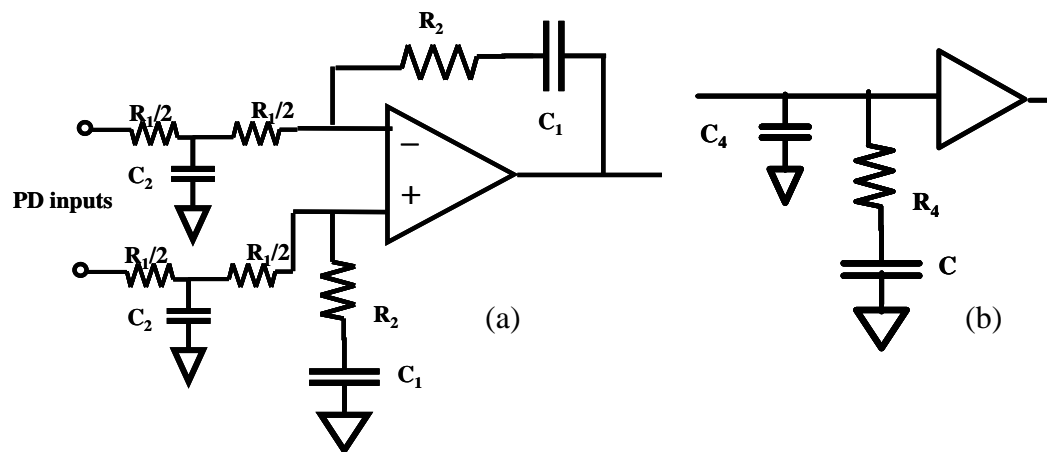
The loop filter shown below in (a) can be used with the phase/frequency detector outputs. The input low pass filter network is required because the phase detector produces a strong 20 kHz rep rate pulse stream when the synthesizer is changing frequency. This filter will produce a third-order system.

- Use the design procedure described in the class notes and ref [1].
- Verify your design using ADS.

You may need to experiment with these to determine the best performance tradeoffs.

Watch out for any additional poles added due to low pass filtering networks at the input of your varactor tuning port on the VCO.  $R_1$  should be no less than 15 k $\Omega$  in order to satisfy the maximum current output limit (0.36 mA) of the phase detector.

The charge pump filter (b) is another alternative for the loop filter. It also includes a higher frequency pole created by  $C_4$ . This makes the loop a third-order loop, so the pole frequency must be selected as described in class. Use ADS PLL Design Guide to verify your design. The current pulse amplitude coming from the phase detector is + or - 0.36 mA. The peak voltage of the pulse is 5V, so the opamp buffer should have enough gain to provide the 12V needed for your VCO. Include opamp gain as part of  $K_V$  in the ADS simulation. Again, beware of extra poles in your VCO tuning port.



**Figure 2. Loop filters for phase/frequency detector (a) and charge pump phase detector (b).**

## **PART 2. Make the following measurements on your synthesizer:**

1. Tuning range. Vary the N counter moduli to find the maximum tuning range of your synthesizer. You can use the tuning knob or the “radio mode” to set this frequency. What limits the tuning range?
2. Tuning rate. Use the toggle mode to jump between two frequencies. Test at 133.3, and 137.3 MHz. Use  $\Delta N = 10$  to avoid losing lock during the transition. Measure settling time and overshoot by observing the VCO tuning voltage on the oscilloscope and note the time constants. Modify your loop filter design if necessary to obtain a smoothly damped response. Report the shape and time constants observed. If time permits, try both types of loop filter/phase detector.
3. Noise spectrum. Observe the output on the spectrum analyzer. Zoom in on the fundamental signal and report and comment on the observed noise spectrum (evidence of phase noise? Reference sidebands?)
4. Compare measurements with simulations. Explain differences observed.

ECE218B Frequency Synthesizer lab      parts list:      Spring 2008

- 1      MC145170D2   CMOS Frequency Synthesizer chip
- 1      LMC6482AIN   CMOS Opamp
- 1      10.000 MHz crystal (HC-49 package)   Digikey X443-ND   ECS-100-18-4
- 2      10 uF tantalum
- 1      0.1 uF leaded ceramic capacitor
- 1      1 Meg leaded resistor
- 1      board mounted SMA female connector
- 2      twisted pair of insulated wire (#24 or 26) 18 inches long
- 1      PC board (will be provided)
- 1      8 pin header
- 4      standoff posts + screws

5/12/08

## **Microchip PIC Based Microcontroller Board**

### **Kyle Wilson, Computer Engineering, 2003**

#### **How it works.**

The Microchip PIC18F252 is the heart of the board. Powered by a 10Mhz external crystal, it contains an internal PLL which multiplies the external clock by 4 to get an internal clock of 40Mhz. A digital encoder is used as an input to the system, while a 2 line, 8 character LCD display is used as an output along with an RS232 serial interface.

The digital encoder has two outputs which are pulled high by 10K resistors. These outputs are in quadrature format, which allows the system to determine which direction the encoder is being turned. If a pulse from one pin is high at any instance, and the other pin is low, the system can determine which direction the knob is being turned. By counting these pulses, the amount the knob is rotated can also be determined. In order to handle the input from this encoder in an efficient way, each encoder output is connected to an interrupt on change pin of the microcontroller. When an input signal changes state on this type of input pin on the microcontroller, an interrupt is generated. As a result, an interrupt is roughly generated each time the knob is turned slightly. In software, each time this event occurs, an internal counter is either increased or decreased, depending on which direction the knob was turned. With this counter value, the system can use it to determine how much the knob has been turned and add this offset to the N value it is controlling on the PLL circuit.

The LCD display is connected to the microcontroller over a parallel 4-bit bus with two control signals: E and RS. All data to be displayed and special LCD commands are sent over the 4-bit bus, while the E input signal determines if the LCD is enabled to accept data, and the RS signal determines whether the input data is data to be displayed or is a command word. Since everything is quantified in bytes, two transfers have to occur to transfer a full byte over the 4-bit bus. When the system is first turned on, the LCD must be initialized and setup using special commands to determine character size and enable/disable certain features. After this process has been completed, data can be displayed on the LCD. Certain timings must be met to ensure proper initialization and data transfer to the LCD.

The RS232 serial interface with the microcontroller is done through the USART. This module allows data to be sent and received at different baud rates. When this capability is combined with a level shifter, like the MAX232, the microcontroller can interface with the RS232 serial interface on a personal computer and data can be sent and received using any terminal program. For this project, a baud rate of 19.2kbps was used and is compatible with HyperTerminal, which is found on most Microsoft Windows installations.

#### **Configuring HyperTerminal for Serial Communications.**

Most user interaction with this project is done through the serial port. The HyperTerminal program that comes with Windows can be used to communicate with the board over a standard serial cable (NOT a NULL serial cable). When HyperTerminal is started, a new connection can be created in which you select which COM port the board is connected to and lastly the baud rate (Bits per second) at which to communicate with

the board. The baud rate that should be used is **19200** and all other settings can be left at default. These settings can be saved in a preferences file so that a new connection does not have to be made each time. One is provided on the CD.

### **How to use the User Interface.**

When the board is first powered on or reset, the menu is displayed. This menu describes all the appropriate actions you can take where each command consists of one letter. Commands can only be entered when a prompt '>' is displayed on the last line of the terminal. To view the menu at any time when a prompt is available, simply hit the 'm' key and the menu will be displayed.

1. Configuration Mode allows you to set the C, N, and R values on the PLL circuit. All values are decimal and the default values are shown in brackets [ ]. To simply keep a default value, do not enter anything in the terminal and just hit enter. These values are kept, except for the N value, until they are changed again in this mode. The N value can be modified by both the Toggle Mode and Radio Mode.
2. Toggle Mode allows you to toggle between two N values at a specified time interval. When you enter this mode, it will ask for a *Delta N* value. This value will be added onto the current N value and the board will cycle between the current N value and the current N value plus the delta N value. After the delta N value is configured, the *Interval* time must be specified in  $\mu\text{s}$ . This time is roughly the amount of time between each time the N value is toggled. Typically, values below  $400\mu\text{s}$  will not be stable. After the interval time has been specified, the two values will be toggled with the specified interval until the 'q' key is hit to stop the toggle mode and return to the prompt.
3. Radio mode allows you to fine tune the N value at a specific interval as set by the R value. In this mode, the encoder knob is used to fine tune the output frequency and this frequency is displayed on both the LCD and the serial terminal. The C, N, and R values used in this mode are adopted from the values set in the Configuration Mode. The frequency interval at which the overall frequency changes is determined through dividing 10 Mhz by R. This mode can be exited at any time by pressing the 'q' key.
4. At the prompt '>', the current configuration can be viewed by pressing the 'd' key. This will display the current values for C, N, R, and Fif. To change these values, simply enter the Configuration Mode.