Frequency Synthesizer Project

ECE145B Winter 2011

The goal of this last project is to develop a frequency synthesized local oscillator 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. Be sure the buffer amplifier provides at least -7 dBm – needed for the synthesizer chip.

Fully evaluate the synthesizer performance. Here are the requirements.

Synthesizer specs:

Frequency step size	20 KHz
Frequency tuning range	134 to 137 MHz (foVCO +/- 10MHz)
Varactor tuning voltage range	1 to 5 volts
Overshoot	< 30%
Reference Spurs	Better than 40 dBc
Settling time	10 ms to 1%
Crystal reference frequency	10.00 MHz

PART 1. Frequency Synthesizer

Build the synthesizer shown in Fig. 1 and evaluate its tuning and noise characteristics. You will use the ADF4002 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[1] for more explanation). Refer to data sheets for details on use of the chips. A PC board is provided for this exercise. The ADF4002 will already be soldered to the board.

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 board with theCMOS chip unless you are wearing a grounded wrist strap.**

****

[1] 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)

Signal levels:-

A	Pulses. 0 to 5 V. Current sink/source setting: CP13,12,11=(011). Current value is controlled by Rset. See Fig. 18 of data sheet.
B	Control voltage. 2 to +5V (to VCO). Series R at output is needed to stabilize OpAmp
C	AC coupled. > 282 mV peak-peak at RFin when output is terminated in 50 ohms
D	Din, CLK, and ENbar from microcontroller board. Isolated with 330Ω resistors in series with each line.
E	Regulated 3.3V for AV_{DD} and DV_{DD}. 5V for CP V_{DD} (V_P). Note that each power supply pin must be bypassed with a pair of chip caps: 0.1uF and 10pF (not shown).

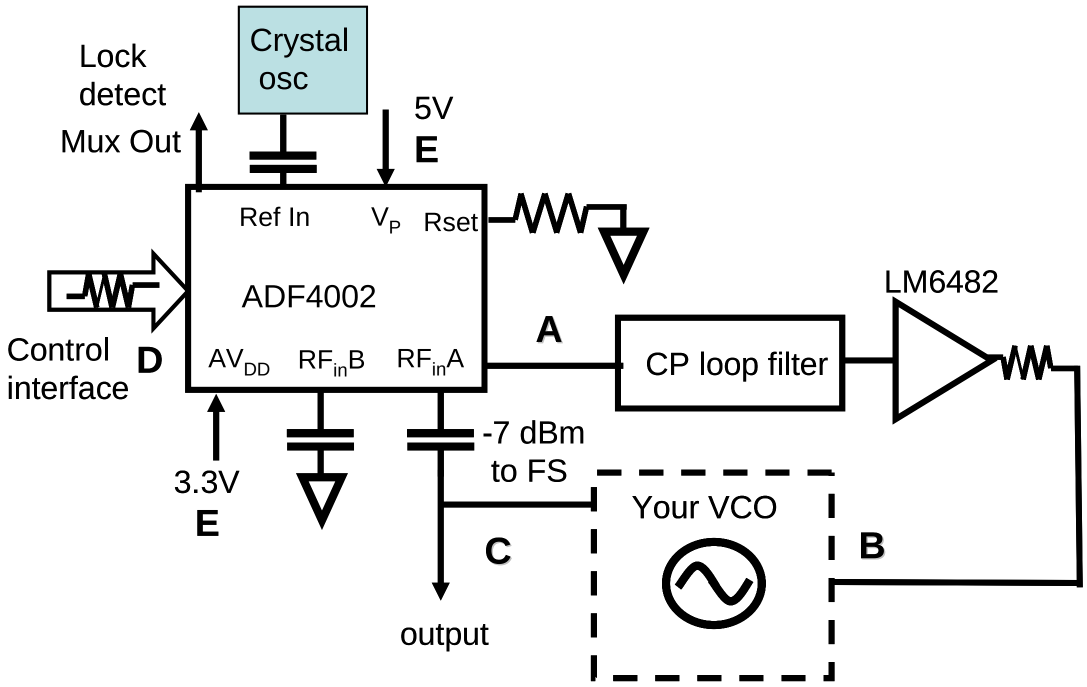


Figure 1. Block diagram of frequency synthesizer test board interfaced to VCO.

Power supply: +5V is the input to the board and is required for the opamp. This could come from your VCO board which has the 5V regulator chip on it, then only one power supply input (9V) would be needed. +3.3V is needed for ADF4002. A LP38691 regulator chip (National Semiconductor) is provided for that purpose. See data sheet. 10uF tantalum caps are required on input and output for stability. Note that a jumper wire must be attached to the board between the 5V input pin and the supply pin of the opamp. As always, good bypassing at both low and high frequencies is essential for all components.

Reference Oscillator: The first step is to design the reference crystal oscillator. Unlike many PLL synthesizer chips, there is no internal circuitry for this. Figure 2 shows a design compatible with the PCB provided.

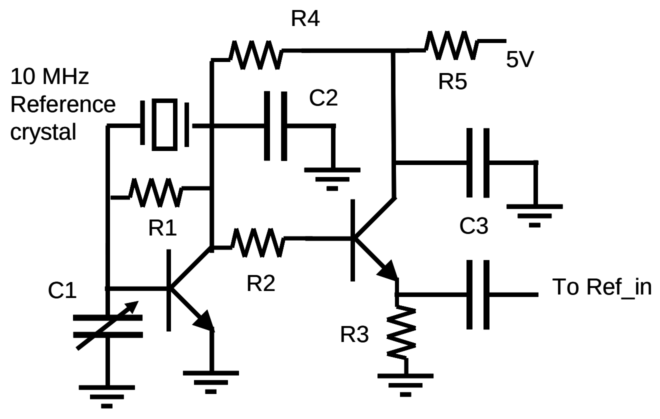


Figure 2. Reference crystal oscillator circuit.

C1 and C2 are capacitors needed to obtain the correct oscillation frequency. All parallel resonant fundamental mode crystals are specified for a certain load capacitance CL.
To obtain the correct frequency:

$$C_L = \frac{C_1 C_2}{C_1 + C_2} + C_a + C_o + \frac{C_{in} C_{out}}{C_{in} + C_{out}}$$

C1 is a variable capacitor (5-30 pF) so that the frequency can be adjusted precisely to 10.0000 MHz. Ca and Co are amplifier feedback (Cbc) and package (shunt) capacitances. Cin and Cout are amplifier input/output capacitances.

Crystal: ECS – 100 – 18 - 4	10.000 MHz
Co	7 pF
CL	18 pF
Max series resistance	60 ohms
Max drive power	0.5 mW

Bias the oscillator transistor by selecting R1 for the desired base current. Note that the collector current should be rather small (try 0.25 mA) so as to not overdrive the crystal. The transistors are 2N3904. Output amplitude of greater than 0.8V is required for the reference input to the ADF4002 to meet the slew rate requirement. Choose R4 and R2 to obtain sufficient amplitude at the output (but in no case more than 3.3V peak-peak). The emitter follower provides isolation so the reference input capacitance does not load the crystal.

An SMA connector is provided on the board for testing the reference oscillator. Temporarily connect the output to the SMA connector and use the frequency counter to set the frequency precisely to 10.0000 MHz by adjusting C1.

After making this adjustment, you need to jumper the output of the oscillator to the reference input of the synthesizer. Use a short piece of RG174 coax to make this connection under the board.

Frequency control. There are 4 serially loaded registers on the FS chip that are set by the PIC controller card. This controller contains some firmware that can be addressed through the PC serial interface to drive the 3 control inputs (Clock, Data, and Enable_bar). A short description of the user interface for the controller is at the end of this lab assignment.

The Initialization and Function latches are for initial configuration. See pp. 11-13 of the data sheet. This allows selection of the MUX output, default is lock detector, the charge pump current, and the fastlock and timer settings. There are default settings that set current in the 011 state and disable fastlock. The Reference Counter latch sets the divide ratio for the reference frequency, 20 kHz, determined by the required frequency step size. Choose R accordingly. The N Counter register sets the divide ratio for the feedback path divider. The output frequency from the VCO is:

$$f_{out} = f_{ref} \frac{N}{R}$$

Phase Detector and Charge Pump: The PLL contains a phase/frequency (up/down) phase detector that drives the charge pump. Charge pump current is controlled by external resistor Rset and the Function Latch bits 17,16,15 (preset to 011). For example, if Rset = 5.1k, I_{CP} = 2.5 mA; if 11k, 1.176mA. You will need to choose a current in order to design your loop filter.

Loop Filter. Design a third-order loop filter (Fig 3) for the worst case using the actual VCO K_O measured at the minimum and maximum frequencies along with their corresponding N values. Your design must meet the specs given for overshoot, settling time, and reference spur ratio at both ends of the tuning range. The peak charge pump pulse amplitude is 5V, so the opamp buffer will not need more than unity gain.[2]

- Use the design procedure described in the class notes and ref [1].
- Verify your design using ADS.
- Additional leaded components can be obtained from the ECE shop

[2] If the VCO can't meet the 3 MHz frequency range with the upper limit of 5V, try decreasing the minimum voltage below 1V. If that still doesn't work then the opamp must have a higher supply voltage and some gain is needed. Include opamp gain as part of K_V in the ADS simulation.

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.*

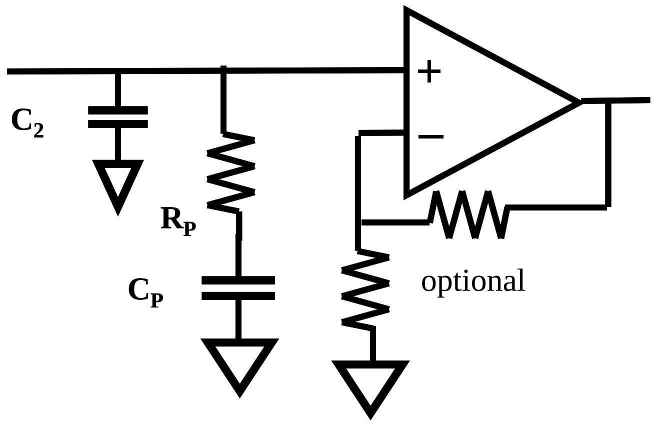


Figure 3. Loop filter for third-order charge pump PLL.

PART 2. Testing: Make the following measurements on your synthesizer:
Connect the synthesizer to the controller board and your VCO. Be sure to get pin 1 of the ribbon cable correctly positioned at both ends. Use the spectrum analyzer and oscilloscope to make the measurements below:

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 minimum and maximum frequency extremes. Use $\Delta N = 2$ 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.
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?) Measure the amplitude of the reference sidebands and compare with specs.
4. Compare measurements with simulations. Explain differences observed.

Design projects (Labs 2 & 3) must be demonstrated to the TA or instructor to verify that all specifications are met. You will need to make an appointment with the TA when you are ready to checkout.

ECE145B Frequency Synthesizer lab parts list: Winter 2011

1	ADF4002 CMOS Frequency Synthesizer chip mounted on PC board
1	LMC6482AIN CMOS Opamp
1	LP38691 3.3V regulator chip
2	2N3904
1	10.000 MHz crystal (HC-49 package) Digikey X443-ND ECS-100-18-4
2	10 uF tantalum
1	5 – 30pF ceramic trimmer capacitor (yellow)
2	board mounted SMA female connectors
1	twisted pair of insulated wire (#24 or 26) 18 inches long
12”	insulated wire for 5V jumper on board
12”	RG174 coax jumper to connect reference oscillator
1	header strip with 12 pins. 8 pins for control input; 2 for Mux output and 2 for CP output.
2	mating socket pins for connecting to the VCO tuning port wires.
4	standoff posts + screws

2/8/11

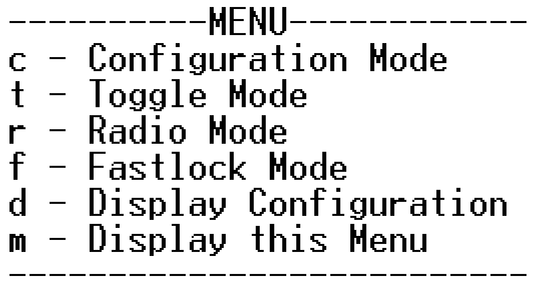
Microchip PIC Based Microcontroller Board
Kyle Wilson, Computer Engineering, 2003

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.



```
----------MENU------------
c - Configuration Mode
t - Toggle Mode
r - Radio Mode
f - Fastlock Mode
d - Display Configuration
m - Display this Menu
--------------------------
```

1. <u>Configuration</u> Mode allows you to set the N (feedback divider), and R (reference divider) values for the PLL. All values are decimal integers 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 retained, 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.
   - Fif affects only the LCD display in radio mode– offsetting the VCO frequency by an IF frequency offset. Defaults to 0.
   - Mux (0 – 7) sets the state of the mux output. See Fig 18 in the data sheet. Defaults to 1 (lock detect: 0V if unlocked; 3.3V when locked).

2. <u>Radio</u> 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 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.

3. 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 µs. This time is roughly the amount of time between each time the N value is toggled. This value must be larger than the settling time of the PLL. 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.

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.

5. Fastlock is an extra feature of this synthesizer chip. Current is increased to current setting 2 for a selectable number of reference cycles. See p. 15 of the data sheet. We will not be using this feature for this lab.

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.