Task 6

Digital Function Synthesizer

"Get the habit of analysis – analysis will, in time, enable synthesis to beome your habit of mind."
– Frank Lloyd Wright (1867-1959)

In this task, we will start add a couple of I^2C devices, a digital function synthesizer via a SPI DAC, and a soft-menu driven user interface to the ECE4723/6723 development target.

6.1 D/A converter

Continue updating your development target errata list and manufacturing instructions as you populate the digital-to-analog converter to the development target. Confirm all connections and supporting discretes before you solder. Once populated, test the functionality of the D/A converter via a quick ESOS testbench app. (You will want to review the textbook chapters on SPI and visit the ESOS documentation on the provided SPI services.)

Also, connect the Maxim DS1631 temperature sensor and Microchip-Atmel AT24C02D to the I^2C bus. Configure the three I^2C address lines on these devices to 0.

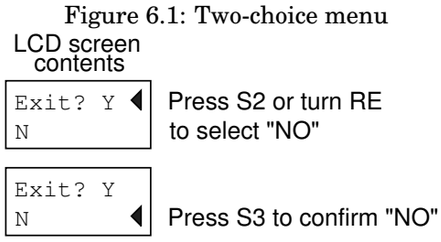
6.2 Soft-menu examples

The rich user-interface devices – switches, rotary encoder (RE), and the LCD module – allow us to create potentially complex menu system for user input and feedback. Some foundation soft-menu code has been provided for you.

6.2.1 Single-value, two-choice

Single-value, two-choice editing can be using the RE and the switches as shown in the figure below.

Figure 6.1: Two-choice menu

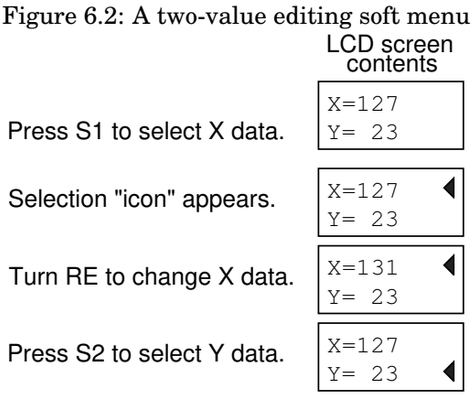


Two choice alternatives can be selected with S1 and S2. The user's choice can also be made by turning the RE.

6.2.2 Single-value or double-value numerics

One or two values can have their numerical value edited via the switches and the RE. For example, data for a tuple could be changed by selecting the appropriate value with S1 and S2, and turning the RE making changes to the data values. Of course, you will want to take advantage of the "fast" turns of the RE to enable rapid, large value changes to the numeric. Your numeric editing routines will also need to accept minimum and maximum allowed values from the calling code to keep numerics enrange. The figure below shows a two-value editing soft menu scenario. Single-value editing would simply entail only one of the display lines in the example below.

Figure 6.2: A two-value editing soft menu



6.2.3 Single-value data selection

Using the code from single-value numeric editing as a base, create a single-value item selection menu item. In this menu item, a single-value is changed to one of a finite number of programatically defined choices.

6.2.4 Static information display

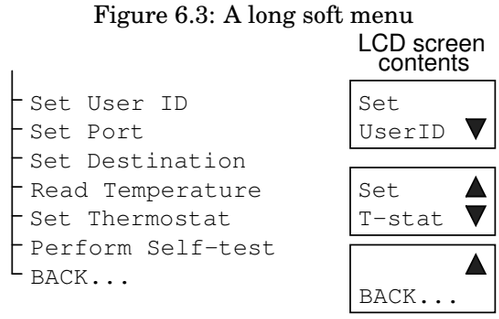
Create a menu item that will display "static" information on the screen. The information will display until the user pressed any key SW1-SW3. The static display is very likely to be longer

than what our screen can display at one time, so allow the user to "scroll" through the information with the RE. You may consider adding an option that allows the information to be dynamic – changes while the user is viewing it.

6.2.5 Menu items and menus

A long selection menu can be implemented with the RE and the switch S3. With our limited screen size, only a single menu items is displayed at a time. The RE allows for the "menu viewing window" to be scrolled "up" and down". S3 allows for the "current menu item" to be selected. Upon pressing S3, the screen with can switch to one of the available menu system items – a two-choice, single-value numeric data entry, or two-value numeric entry – shown above. The figure belows shows an example three menu item examples for a long menu.

Figure 6.3: A long soft menu



6.2.6 Development

Using the ESOS softmenu demontration app in the repo as an example, create the necessary routines to interface with the RE monitoring task and the LCD module service to implement the soft-menu system examples above. Clearly document your code. Properly factor your effort so that the menu system can grow/evolve and the assisting services for UI and LCD can change beneath your menu system with minimal to no impact.

6.3 ESOS Function Synthesizer

Write an ESOS user application that will allow the user to enter the desired parameter to control the analog function synthesizer output from the D/A converter. The required user menu is given in the figure below.

Figure 6.4: Function synthesizer soft menu

	NOTES:
Set wvform	choices: tri, sine, square, user1, ...
Set freq	single data entry (freq=64-2047)
Set ampltd	single data entry (amp=0-3.3 V)
Set duty	single data entry (duty cycle=0-100)
Read LM60	static display screen (clear w/ S3)
Read 1631	static display screen (clear w/ S3)
Set LEDs	single data entry (LEDs=0-7)
About...	static display screen (clear w/ S3)

Your function generator will be capapble of at least four waveforems: triangle, square, sinusoidal, and at least one user-defined waveform. [1] You are welcome to create additional user waveforms. The waveform samples and a maximum five-character name of the waveform are to be contained in the AT24C02D. The oranization of the EEPROM is up to you, but it should be logical, well-documented, and expandable.[2]

The frequency is given in Hertz. The user should be able to select the frequency value in 1 Hz increments.

The output amplitude is given in Volts. The amplitude display screen could represent the voltage as a “floating point” with tenths, or a thermometer type display may be used, or both.....

The duty cycle can be selected for the square wave only, i.e. if the selected waveform is “sine” or “tri”, the duty cycle menu item is not shown. The default duty cycle for the square wave is 50%, and can take on integer values 0-100.

Temperatures should be shown in Celsius with the thermometer display on the right. (Optionally, the RE or S1/S2 may be used to also display the temperature in Fahrenheit.)

The LEDs set screen with display the number 0-7 in binary on the development board LEDs (LED1-LED3) with LED1 being the MSb.

The “About...” screen can be used to give credit to the developers and provide a simple information statement.

6.4 Check Off

Demonstrate the following things to the TA:

1. Your ESOS function synthesis user application. You must devise a demonstration/test that will convince the TA that your service is working fully and properly. Be sure to demonstrate proper I^2C operations concurrently to function generator output.

[1] Recall that floating point numbers are not allowed in your application. There are many tried-and-true methods by which sinusoids can be “created” without floats. The simpliest of which is a quarter-waveform look-up table. There are others. Research and identify a suitable method.

[2] I may specify additional or different waveforms in later labs. You will want modification to be quick and easy

6.5 Submission

1. Commit your project to your repository to your "trunk" folder in accordance with the procedure prescribed by the TA. Your repository should all files required to produce your "build", including the appropriate build file(s).
2. A "tag" release in your repository called `task6` that can be recalled at any time to build this task's deliverables.

Each team should submit to their team repository the following files[3]:

- Any created or expanded files `esos_menu.h` and `esos_menu.c` that contain code for your soft menu systems
- The ESOS function systhesis user application `fcn_synth.c`
- any unit test or test bench files used to verify your code created for this task

[3] Files should be possess the same licensing header as the other ESOS source code files and be fully commented.