Cadyn Maddocks Embedded Systems Final Project Write up 3/16/2022

Abstract:

The Goal of this project was to design and construct a portable EKG monitor that would be capable of displaying the BPM, outputting the waveform through the digital to analog converted, saving the input for a selectable number of data points before or after the push of a button, as well as be able to load the saved waveform to the DAC at a latter time. Some of the more notable features included in this device are, the ability to store multiple waveforms, the running average of BPM calculations, and a simple yet effective menu system. Everything in the project was completed except for the Store preceding waveform, which isn't properly storing data. Likely the problem is that the code is failing to iterate somewhere in the process. The rest of the system, including the EKG mode, store following waveform, select sample rate, select data points, and load from memory, are all functioning as intended.

Description:

The device uses a MKL25Z microcontroller on a freedom I/o board with an attached small LCD display, 5 switches, 4 LED's, a 12-bit ADC, a 16-bit DAC, and a number of available ports for GPIO usage. The clock speed chosen for operation is 48MHz with a bus speed of 24Mhz. The power supply voltage is 3.3V.

User Guide:

To turn on/reset the device press the black button between the two micro-usb ports. EKG mode (mode 1) will take several seconds to arrive at a correct value. Load from Memory (mode 6) will override the ADC input and output whatever is stored in the selected memory to the DAC in a loop.

Mode Navigation – Press switch 5 to navigate forward through the available modes, Press switch 1 to navigate backwards through the available modes.

Setting Selection – In modes 2,3, and 6 pressing switches 2 or 4 will change which memory is currently selected to be saved or stored. (Options are DACMem1 or DACMem2).

In mode 4, pressing switch 2 will cycle forward through possible ADC sample rates, while pressing switch 4 will cycle backwards through them.

In mode 5, pressing switch 2 will cycle forward through possible numbers of data points for saving and loading. Pressing switch 4 will cycle backwards through them.

Save – In mode 3, pressing switch 3 will start saving the following data points of values into the currently selected memory.

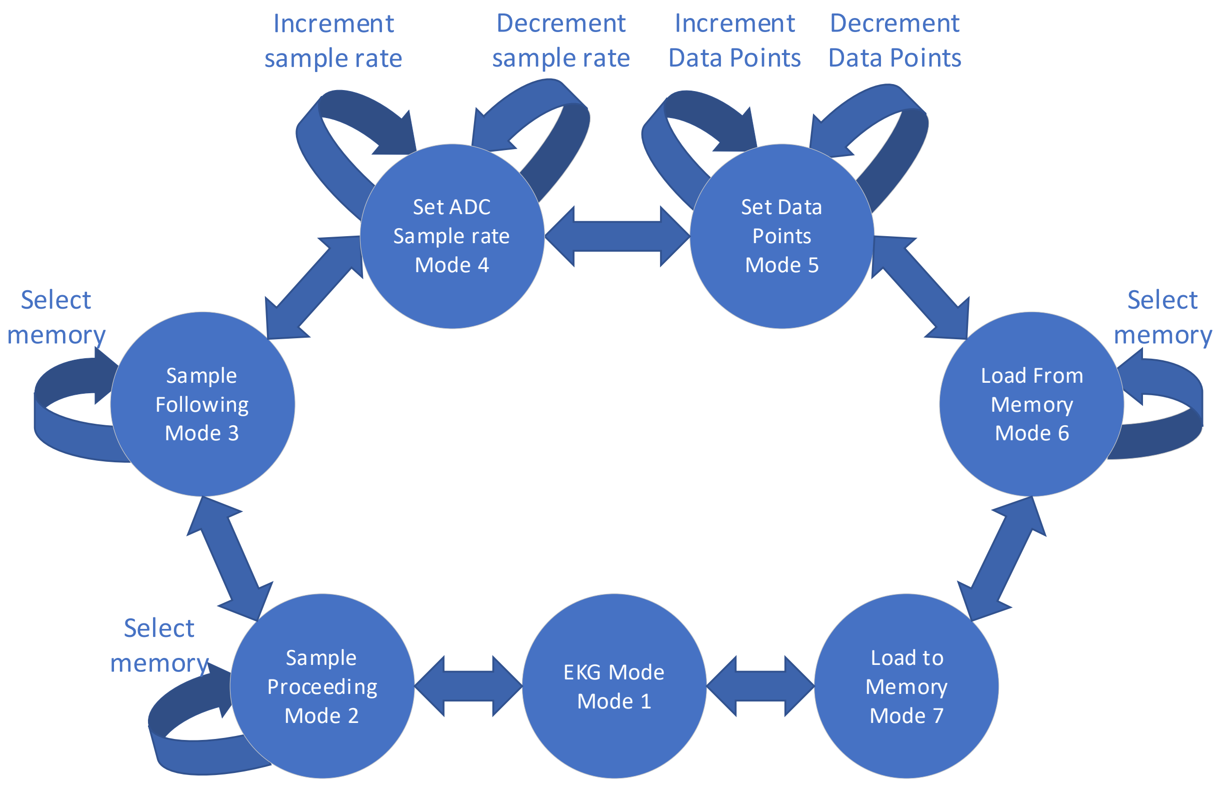
If mode 2 was operational, pressing switch 3 would save the previous data points of values into the currently selected memory.

Hardware Design:

The only thing we did in terms of hardware design was for the input amplifier, which uses an instrumentation amplifier, in conjunction with a band-pass filter, to give an appropriate input to the ADC, with low noise and in the correct voltage range.

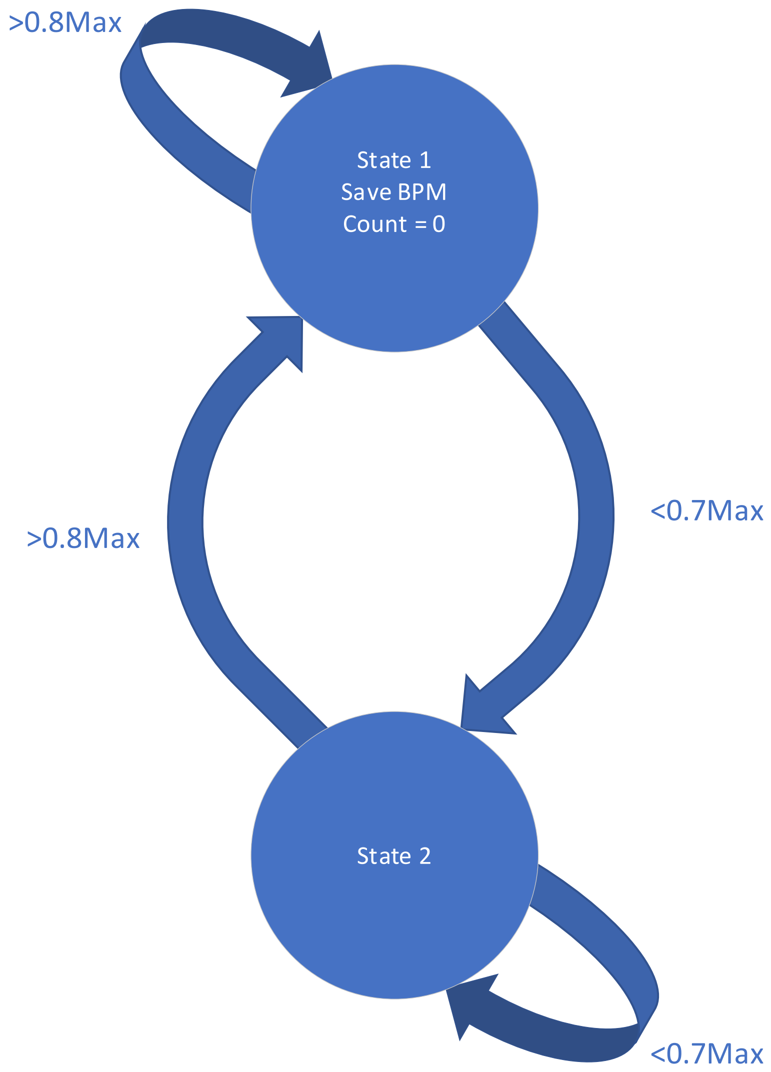
Software Design:

The design for the menu simply iterates through the below diagram using one switch to go forward and another to go backwards. If in a specific mode you can Increment of Decrement that modes specified setting by pressing the associated buttons. In the case of select memory it only changes the one selected since there are only two available. I made a design choice to change the PIT timer when I change sample rate, which means I had to do a conversion of sample rate*BPM/1000 to get the appropriate BPM to display. To change the sample rate, I first stop the pit, then initialize it with 24million/sample rate, and then start it back up again. This means that I don't have to implement code in the ADC interrupt to skip values and can instead use every value that the interrupt occurs on. Another design choice I made for the menu was not to clear screen. All this means is that every menu needs to take the full 16 characters on the screen. To make this possible I made a custom display_decimal function that would display any 4-digit integer value in exactly 4 characters. To carry over all the information into my main from my switches code, I needed a variable for mode to cycle between modes, a variable for ADCsamplerate to increment/decrement between sample rates, a variable for dataPoints to increment/decrement between possible numbers of data points, a variable to indicate when I'm saving, a variable to indicate which memory is currently selected, and a variable to tell when I've changed the ADCsamplerate so that I only change the PIT when I change the sample rate and not all the time its in mode 4.



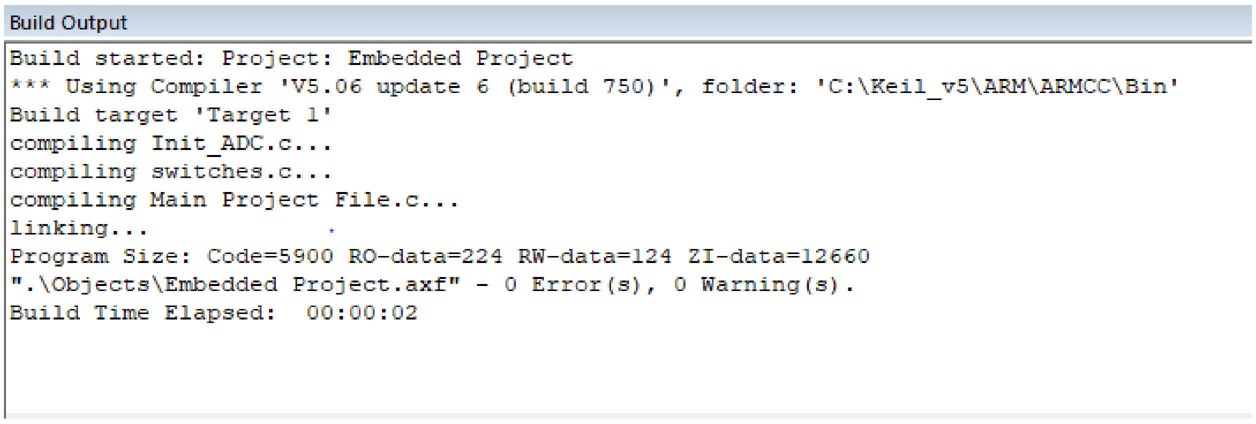
The BPM calculation is done, by finding the period (countTime) by using a finite state machine to only save BPM whenever you have reached the same point again on the heartbeat. The BPM calculation is

60000 (ms/minute) divided by countTime (ms/Beat) to get BPM. This is then stored in an array of 5 values (original idea btw) and then averaged to be displayed in the main.



The maximum voltage value is found by replacing the maximum with the new ADC input whenever the ADC input is greater than the maximum voltage. This allows us to dynamically determine the proper values to trigger at since we have no idea what the actual amplitude will be prior to hooking it up.

Saving the input is done temporarily to an array meant for temporary storage for save preceding. Save following is accomplished just be saving up to datapoints value in the array specified by dacmem. When an array is saved the current number of datapoints is also saved along with it so that outputting the array will always yield the exact input. Save preceding isn't functional, but it was designed to be accomplished by saving datapoints value of points in the temporary array, starting from data points before the current place in the temporary array. The method of output is the same for both of these functions. So long as its in mode 6, the code will iterate through the currently selected memory and display it starting over whenever it finishes.



```
Build started: Project: Embedded Project
*** Using Compiler 'V5.06 update 6 (build 750)', folder: 'C:\Keil_v5\ARM\ARMCC\Bin'
Build target 'Target 1'
compiling Init_ADC.c...
compiling switches.c...
compiling Main Project File.c...
linking...
Program Size: Code=5900 RO-data=224 RW-data=124 ZI-data=12660
".\Objects\Embedded Project.axf" - 0 Error(s), 0 Warning(s).
Build Time Elapsed:  00:00:02
```

Discussion of Results:

Everything worked except for the save preceding, I thought what I have should work, but for some reason it only saves one value, so I’m pretty sure its somehow not iterating through something correctly. Besides that, everything works as intended.