EKG Project

Eric Walsh

ENGR 355

Table of Contents

Abstract 3

EKG Project 4

User Guide 4

Hardware Design 5

Software Design 6

Final Results and Comments 8

Appendix 9

Appendix A: PCB casing dimensions 9

Appendix B: Hardware Schematic 10

Abstract

The goal of the project was to create an EKG using a MKL25Z128VLH4 microcontroller, a 2x8 LCD, 4 push buttons, ADC input, DAC output, and 2 LEDs. The 2x8 display was to be used to display the calculated heartbeats per minute, sample rate, and current file. The user was to use the 4 buttons to select the sample rate and the current file. The device was to be able to record incoming wave data from the ADC and save it, then later send it to the DAC. It also was to be able to use the ADC to calculate the number of heat beats in a minute and flash one of the LEDs to signify that a heartbeat was detected. There was to be two menus: a main to display the BPM, sample rate, and current file, a file menu that showed to files and allowed the used to select which file to save recorded wave forms to. In the end, the device was able to properly display the menus (with some small, but manageable, glitches), record incoming wave files, send saved files to the DAC, and flash LEDs for every detected heartbeat. However, it was unable to properly calculate the number of beats per minute.

EKG Project

The device made for this project is a handheld portable device that analyzes a periodic voltage wave input (from 0 to 3.3V), in this case an EKG, and calculates its rate. It also has the ability to save the incoming wave and output it at a later time. The user interface of the device includes a 2x8 display, 4 push buttons, and 2 LEDs. The 2x8 display is an alphanumeric display that displays the calculated rate, the current sample rate, and the currently selected save file. The buttons are used to change the sample rate and the save file that recordings are saved to. The first LED flashes whenever a heartbeat is detected and the second LED lights up for the duration of the recording. The user can use the buttons to select the sample rate for the analog to digital converter with a range of 50 to 10,000 samples per second. The user can connect to three ports: the ADC input, the DAC output, and the PC comm port. The device also has an expanded memory from an external EEPROM memory component. The PCB is held in a custom plastic housing[1] that also holds a pair of AA batteries to provide power to the device.

User Guide

When starting the device, wait for the LCD to properly show the menu display before pressing any buttons as shown in figure 1-2. This should take up to 20 seconds. When the device finishes the startup sequence, the BPM will update with the calculated BPM if there is a wave form connected to the input. If not, there will be three dashes next to the BPM similar to figure 1-1.

Figure 1-1: The device while still loading

The main menu is as shown in figure 1-2. One the top line, there is the calculated beats per minute. On the second line, the sample rate is shown next to the “S:” and the file number “F1” is shown.

Figure 1-0-1 the device after startup sequence

[1] See appendix 1 for dimensions

In the menu state, the user can set the sample rate, start recording an incoming wave, or switch to the file menu. Refer to figure 1-3 for the button layout and labels. In the main menu, to increase the sample rate, press “A,” to decrease the sample rate, press “B,” to start recording the incoming wave file, press “D,” and to enter the file menu, press “C.” The selectable sample rate goes from 50 to 10,000 samples per second. Values from 1000 to 9500 are labeled as 10h, where h represents the SI prefix hecto. Hecto represents 10^2. Thus 10h is $10*10^2$ samples per second. At 10000, the value is labeled as 10k, where k is kilo. Kilo represents 10^3. Thus, 10k is $10*10^3$ samples per second.

Figure 1-3 Button Layout

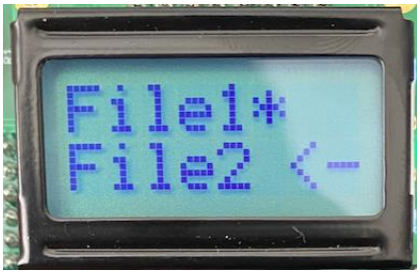


Figure 1-4 The file menu

Within the file menu, as seen in figure 1-4, the “*” symbol represents the currently selected save file. The “<-“ symbol represents the cursor. When in the save file, the user can move the cursor, select a file and exit, and send a file to the output. To move the cursor up in the file menu, press “A.” To move the cursor down in the menu, press “B.” To send the file that the cursor is hovering over to the output, press “D.” To select the file that the cursor is hovering over as the new save file and exit back to the main menu, press “C.” Other things to note: The BPM updates every 10 seconds, an LED will flash every time a heartbeat is detected, an LED will light up when a wave is being recorded, and if the display gets stuck with the display half in the main menu and half in the file menu, pressing “A” or “B” will place the display back in either the main or file menu.

Hardware Design

The hardware for this project includes a programming connector, four buttons, a crystal, a ADC connected to an op amp with negative feedback, a serial I/O, a 2x8 LCD, 2 LEDs, and a

DAC output. The schematic for the hardware can be found in the appendix. The hardware for this project doesn't work due to bridges across the microcontroller.

Software Design

The source code for this project includes 24 files. 11 C files and 13 header files. The files that directly control hardware are Pit_Functions.c, ADC_Functions.c, LEDs.c, LCD_Functions.c, DAC_Functions.c, and switches.c. The functions that process the information gathered from, or going to, the hardware are Calculations.c, Interface_Functions.c, Recordings.c, and StartUp_Functions.c. The user interface for this followed a state diagram for inputs as seen in figure 3-1. In state 1, the starting state, the menu shows the BPM, the initial sampling rate, and the current file. Buttons A and B controls the sampling rate, button D controls the recording of the voltage input, and c changes the menu to state 2. In state 2, the file menu, the use can see 2 files, a symbol, "*," to show current file, and a cursor, "<-." Buttons "A" and "B" are used to move the cursor., button "D" is used to send a file to the DAC, and button "C" is used to select a file and return to state 1.

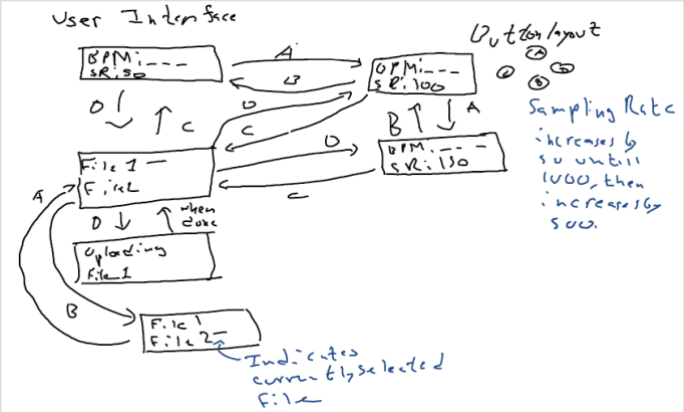


Figure 3-1 initial UI state diagram

The logic for the process to calculate the number of beats per minute centered on counting the number of beats detected in 10 seconds then multiplying that count by 6. To detect the beats, the ADC operates in 3 stages. First, it will collect 1000 values, it also does this every time the sample rate is changed. In the second stage, it will then calculate the magnitude of the collected values (maximum value – minimum value). The magnitude is used to set a high and low threshold of 80% of the sum of the magnitude and the minimum value of the first 1000 value

and 70% of the sum respectively. In the third stage, it will compare the incoming values to the thresholds. If the value is higher than the threshold, and it is the first since the last peak (the flag that signals a rising value is not set), it will set the rising flag, increase the count of beats by one, and turn on an LED until the values fall below the low threshold where the rising flag is set to zero and the LED is turned off. The PIT timer on channel 1 will cause an interrupt every 10 seconds, collect then reset the number of beats, and have the number sent to be converted to ASCII hexadecimal to be displayed on the LCD. The process is reset by setting the variable that counts the number of reference values to zero.

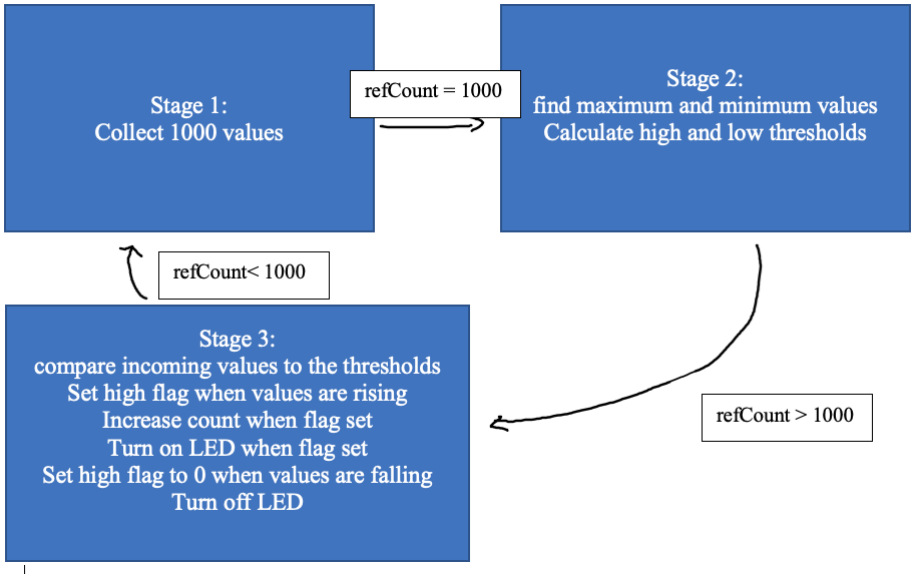


Figure 3-2 The flow process for counting the number of beats

The process for recording the wave file is simple. There are two arrays of length 1000 which the saved values to go to. A flag is set to indicate which file the recorded values to go to. When the recording is started, a record flag is set, a LED turns on, and a counter starts that counts to 1000. When the counter gets to 1000, the recording stops and all the values are stored, the flag is set to 0 and the LED gets turned off. When the recording is to be sent to the DAC, a flag is set and the selected array is sent to a function that loops through the array 1000 times to ensure that the output wave is read. The other simple process is the changing of the sampling rate. When a button is pressed to change the sampling rate, the PIT on channel 0 and channel 1 is

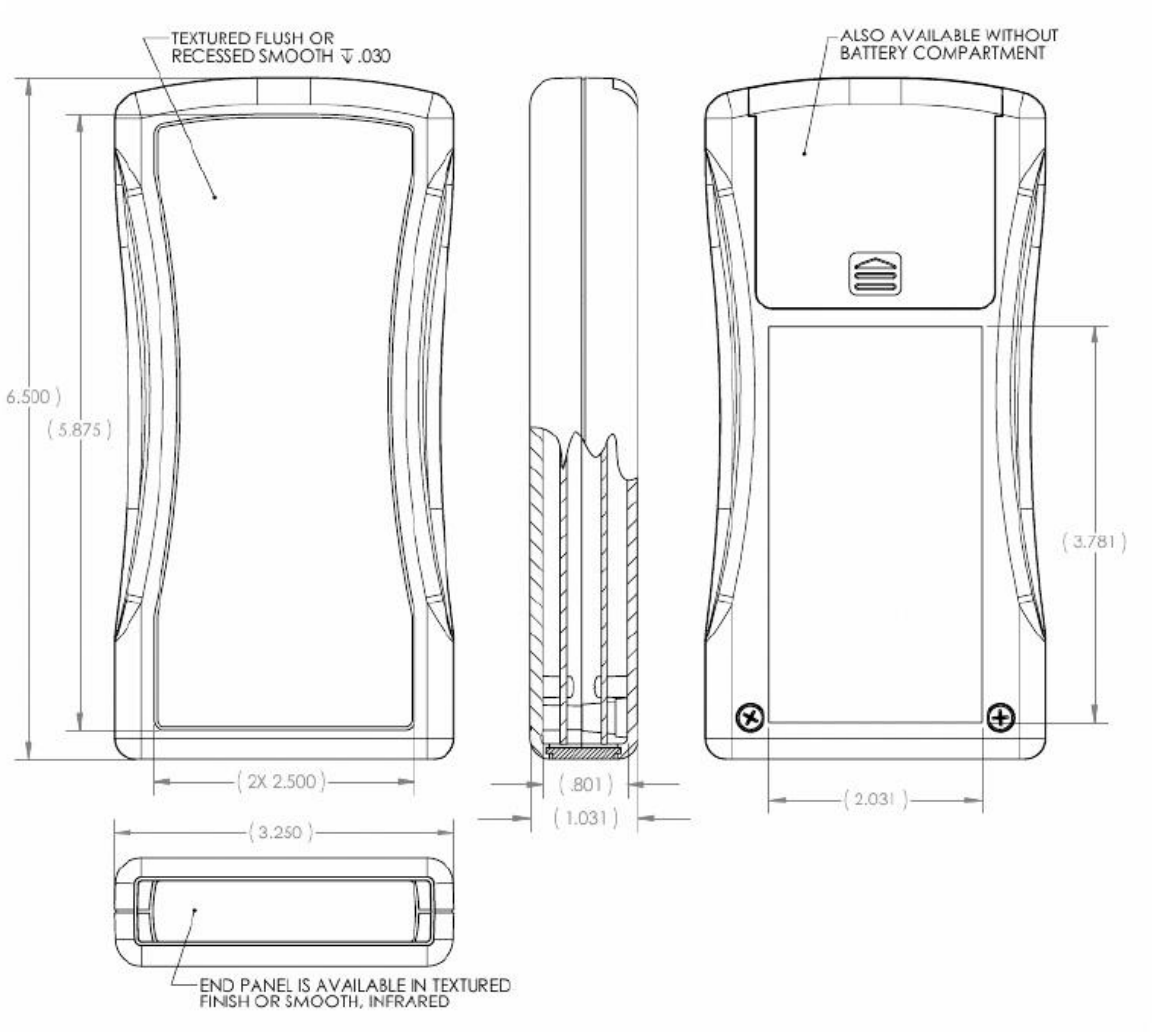
stopped. The rate is taken from an array that holds the list sampling rates and used to reinitialize the PIT on channel zero, then PIT on channels 0 and 1 is restarted.

Final Results and Comments

Overall, the project works. There are only two areas of the project that aren't working as intended. The first part is the LCD. When switching between menu states, sometimes the display will show half of the main menu and half of the file menu. This isn't a major issue as it can be fixed by pressing either "A" or "B" to put it firmly into one or the other. The second area that doesn't work is the beat detection. For whatever reason, the program either detects too many beats or doesn't detect any beats at all. I suspect the issue to be due to some bad math on my part. There were also hardware issues in the project. During the process of putting components on the custom PCB, bridges formed between the pins of the microcontroller and the microcontroller was orientated the wrong way. I would need to redo the placement of the microcontroller to fix the circuit. Other than those issues, the project ideally. The user inputs do what I want them to do, the incoming wave gets recorded properly, the recorded wave gets output properly, and the LEDs get activate properly.

Appendix

Appendix A: PCB casing dimensions



Appendix B: Hardware Schematic