LAB REPORT 7: CONDITIONING, CONVERSION, CALCULATION AND COMMUNICATION

Written by: [redacted]
Class: [redacted]
Instructor: [redacted]
Teaching Assistant: [redacted]
Deadline: May 3rd, 2024 at 11:59 PM

INTRODUCTION

The Electrocardiogram (ECG) is a necessary tool in every medical care facility. In this lab assignment, the task is to prototype an ECG heart rate monitor using the MSP430FR2433. The design will need to utilize all the skills learned in previous labs. This includes amplification, filtering, interrupt service routines (ISR), timer modulation, and serial communication. The resulting prototype would ideally exhibit an accurate and reliable measure of the heart rate on RealTerm.

MATERIALS AND METHODS

PRE-PROCESSING

To start this lab project, the heart rate signal that is received needs to be amplified and filtered. The signal itself is 50 mVpp large and fires at about 1-2 Hz. However, there is a signal bias that needs to be filtered. The bias is 0.1 Hz and 2.5 Vpp. To remove this bias signal, it was necessary to develop a Cascading 3rd Order High Pass Filter (HPF) with a steep attenuation curve. This was done by cascading a 1st order HPF and a 2nd order Sallen-Key HPF as shown in Figure 1.

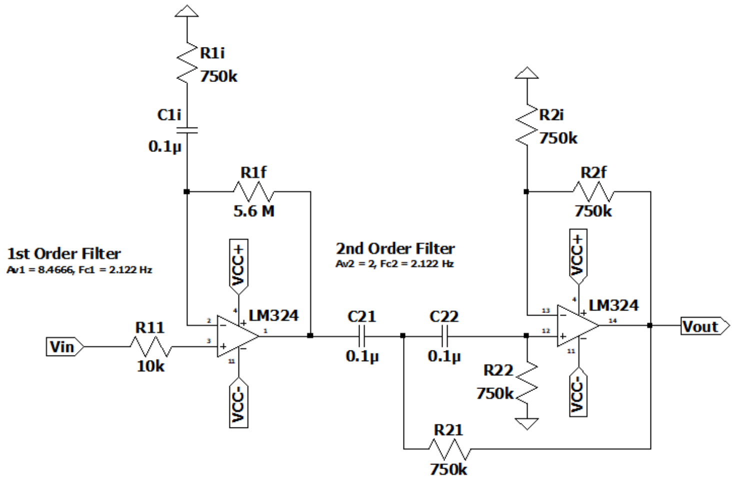


***Figure 1**: 3rd Order Cascading HPF*

The reason the filter was designed in this manner was because at the time, the goal was to have the signal hit around 0.9 Vpp. The reason for this was because a 0.9 Vpp was 60% of the 1.5 Vpp was set as the maximum for the ADC converter module in the example programs provided. Hence, to go from 50 mVpp to around 0.9 Vpp, the gain needed was around 18. In addition to this, with the signal being at around 1-2 Hz and the noise being at 0.1 Hz. It was decided that a cutoff frequency of around 2 Hz would be appropriate. While this would slightly attenuate the signal, it was fine because the signal would receive gain from the amplifier. However, it would be needed to completely attenuate the 2.5 Vpp noise.

For the first order HPF, the equations were the following:

$$f_{c1} = \frac{1}{2\,\pi\,\mathrm{R}_{1\mathrm{i}}\,\mathrm{C}_{1\mathrm{i}}} \;\; ; \; A_{v1} = \frac{R_{1f}}{R_{1i}} + 1$$

To calculate the components for this system, the capacitor values were pre-selected to be 0.1 µF. Once this was done, the resistor values were quickly calculated using the following math:

$$\mathrm{R}_{1\mathrm{i}} = \frac{1}{2\,\pi\,f_{c1}\,\mathrm{C}_{1\mathrm{i}}} = \frac{1}{2\pi(2Hz)(0.1uF)} = \boxed{796\;k\Omega = \mathrm{R}_{1\mathrm{i}}}$$

750 kΩ were used simply because they were the closest resistor option. This ultimately gave a true cutoff frequency of 2.122 Hz.

The feedback resistor was calculated by having a goal gain of 9 for the first order filter. The math went in the following manner:

$$R_{1f} = (A_{v1} - 1)R_{1i} = (9-1)750k\Omega = \boxed{6M\Omega = R_{1f}}$$

The closest resistor option available was a 5.6 MΩ resistor. This ultimately gave a gain of 8.4666.

For the 2nd order HPF, the equations were the following:

$$f_{c2} = \frac{1}{2\,\pi\,\sqrt{\mathrm{R}_{21}\,\mathrm{R}_{22}\,\mathrm{C}_{21}\;\mathrm{C}_{22}}} \;\; ; \; A_{v2} = \frac{R_{2f}}{R_{2i}} + 1$$

For the cutoff frequency calculation, the following values were set equal to one another $\mathrm{R}_{21} = \mathrm{R}_{22}$, $\mathrm{C}_{21} = \mathrm{C}_{22}$. This made it simpler to calculate the cutoff frequency:

$$f_{c2} = \frac{1}{2\,\pi\,\mathrm{R}_2\,\mathrm{C}_2}$$

The capacitance for this was preselected to be 0.1 µF. With this, the calculation of the resistor is the same as the one for the 1st order HPF. This meant that the cutoff frequency was also the same. For the other resistors, they were set to equal the same value because the Sallen-Key filter would otherwise malfunction. Hence, the gain for this filter was set to 2.

Together, the composite gain was 16.932 (through multiplication of the two gains), and the composite cutoff frequency was 2.122 Hz since both cutoff frequencies were the same. The resulting circuit can be seen built in Figure 2. The result of the filter can also be seen in Figure 2. In Figure 2, the input is the yellow line and the output is the green line.

Finally, this filter was characterized, and the resulting data and bode plot can be found in the results and data section of the lab report.

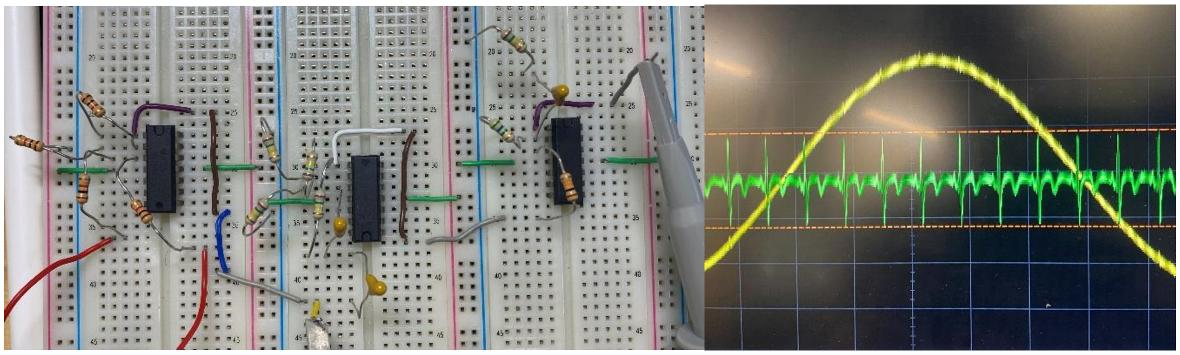


Figure 2: *The 3rd Order Cascading HPF circuit and resulting oscilloscope output.*

MSP430 PROGRAMMING AND ALGORITHM

<u>When deciding on which MSP430 example code to use</u>, the decision was obvious. The example code 'msp430fr243x_adc10_21.asm' was the ideal example code to start with. This is because this example code already had a threshold system set up, hence, slightly less work needed to be done to get the right output.

```
ADHI        ; A1 > 1V
            cmp     #bufferMax, beatBuffer
            jge     beatInc
            inc     beatBuffer
            jmp     skipADHI
beatInc
            bit.b   #BIT0, &P1OUT
            jnz     skipADHI
            bis.b   #BIT0, &P1OUT
            inc     beatCount
            clr     beatBuffer
skipADHI
            bic.w   #ADCINIFG, &ADCIFG
            reti

ADLO        ; A1 < 1V
            bic.b   #BIT0,&P1OUT
            bic.w   #ADCINIFG,&ADCIFG
            reti
```

Figure 3*: ADC logic.*

First, it was necessary to get a beat counter to start working, the major changes made to the programming to do this were that the ADHI and ADLO interrupt actions were adjusted. Moreover, the ADIN function was completely removed and the interrupt vector that pointed to ADIN was reprogrammed to point to ADLO. All ADLO did was turn off the P1.0 LED. However, for ADHI, the programming was a little more work. ADHI needed a debouncing system so that the beat counter would not increment every clock tick that the voltage was above the threshold. Hence, two debouncing mechanisms were added.

<u>First, a check was added to see whether the P1.0 LED was set or reset.</u> If it was set, meaning that the voltage had not yet dropped below the threshold, then the ADHI function would be entirely skipped, and no beat would be counted. However, there came another. The noise of the waveform would occasionally spike in and out of the threshold as the voltage through the QRS wave. Because of this issue, a small standby timer was added so that no counts could be made unless the standby timer had counted to a certain number. This number was set as an adjustable constant in the parameters section of the program for later troubleshooting. By utilizing these two methods, the beat counter was starting to count the number of heartbeats accurately.

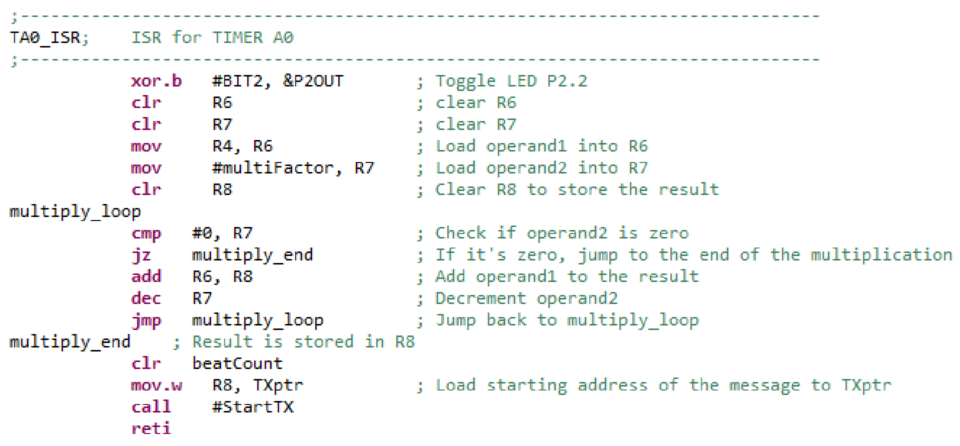
However, looking back, it may have been easier and better to make use of the ADIN and ADLO threshold. Perhaps, by having the LED reset in the ADLO instead of the ADIN, the heartbeat counter would have been developed a lot easier.

<u>The next major change was the interrupt for timer A0.</u> Luckily, timer A0 was already set to interrupt every 2 seconds, so there was not much need to change the initialization of this timer. However, what was needed was a way to see whether the timer was working. Hence, the timer was set to toggle the P2.2 port that was connected to an LED via a wire. This would allow for easier troubleshooting later in the building process. The next item needed was a multiplication algorithm. <u>Since the buffer was 2 seconds long, to have the result number in beats per minute, the number had to be multiplied by 30.</u> To do this a simple multiplication algorithm was added to interrupt service routine.

This algorithm works through the following instruction:

1. Check if operand 2 = 0.
2. If operand 2 = 0, then end the multiplication loop.
3. Add operand 1 to the results register.
4. Decrement operand 2.
5. Loop back to instruction 1.

Finally, the initialization, interrupt service routine, and subroutine code from the serial monitoring lab was added to this program. All that was seemingly needed to output the number to the serial port was to place the number in the TXptr location and call the StartTX subroutine.



```
;-------------------------------------------------------------------------------
TA0_ISR;    ISR for TIMER A0
;-------------------------------------------------------------------------------
            xor.b   #BIT2, &P2OUT        ; Toggle LED P2.2
            clr     R6                   ; clear R6
            clr     R7                   ; clear R7
            mov     R4, R6               ; Load operand1 into R6
            mov     #multiFactor, R7     ; Load operand2 into R7
            clr     R8                   ; Clear R8 to store the result
multiply_loop
            cmp   #0, R7                 ; Check if operand2 is zero
            jz    multiply_end           ; If it's zero, jump to the end of the multiplication
            add   R6, R8                 ; Add operand1 to the result
            dec   R7                     ; Decrement operand2
            jmp   multiply_loop          ; Jump back to multiply_loop
multiply_end    ; Result is stored in R8
            clr   beatCount
            mov.w   R8, TXptr            ; Load starting address of the message to TXptr
            call    #StartTX
            reti
```

***Figure 4**: Timer A0 ISR logic.*

However, this ultimately did not work as the decimal number was not converted to ASCII. Unfortunately, the conversion to ASCII was complicated and due to the time restrictions of the lab, it was not possible to finish. This became a major limitation of the ECG prototype.

Ultimately, it was decided that this would be a good stopping point for an amateur ECG device, and a video of the prototype was taken. Lastly, the materials were cleaned up and put away.

RESULTS AND DATA

3rd Order Cascaded High Pass Filter (2 Hz)				
Frequency (Hz)	**Vin (Vpp)**	**Vout (Vpp)**	**Gain**	**Gain (dB)**
0.25	0.500	0.025	0.050	-26.021
0.5	0.500	0.075	0.150	-16.478
0.75	0.500	0.150	0.300	-10.458
1	0.500	0.350	0.700	-3.098
1.25	0.500	0.550	1.100	0.828
1.5	0.500	0.900	1.800	5.105
1.75	0.500	1.130	2.260	7.082
2	0.500	1.390	2.780	8.881
2.25	0.500	1.650	3.300	10.370
2.5	0.500	1.870	3.740	11.457
2.75	0.500	2.070	4.140	12.340
3	0.500	2.270	4.540	13.141
5	0.500	3.300	6.600	16.391
10	0.500	4.240	8.480	18.568
20	0.500	4.540	9.080	19.162

***Table 1**: 3rd order cascading HPF gain data.*

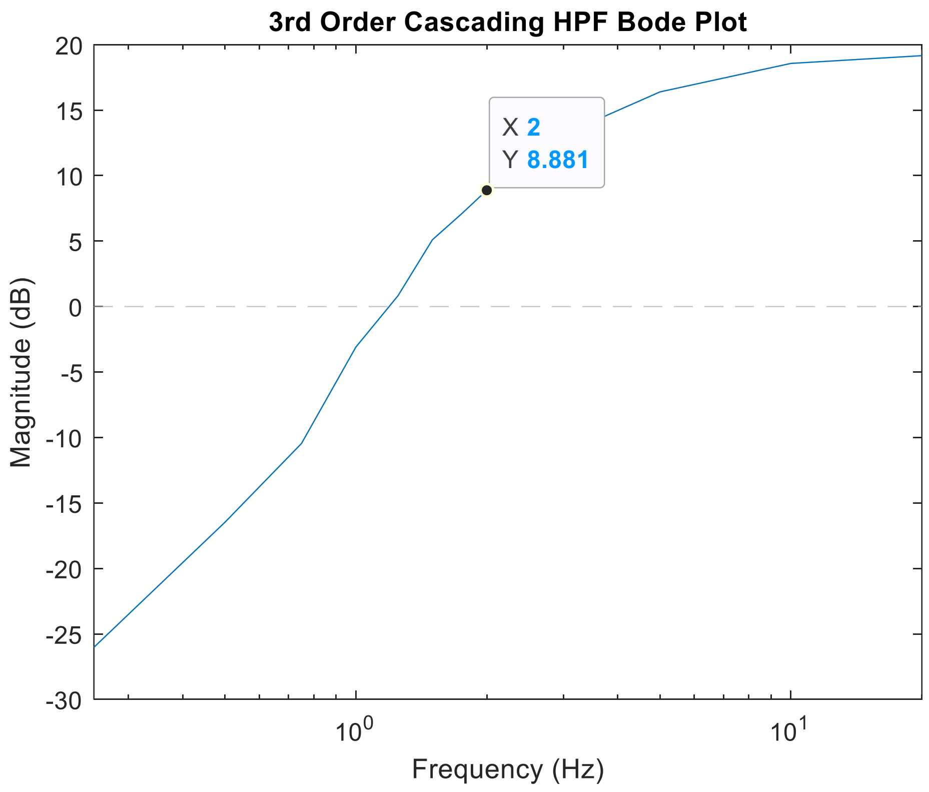


***Figure 5:** 3rd order cascading HPF bode plot.*

DISCUSSION AND CONCLUSION

To summarize, this project provided valuable insights into the practical application of various electronic engineering concepts, including amplification, filtering, interrupt service routines, timer modulation, and serial communication, within the realm of biomedical instrumentation. However, despite the comprehensive implementation of these techniques, the heart rate monitoring algorithm showed significant limitation. The observed volatility in bpm readings, moving between 60 bpm to 30 bpm, displayed the algorithm's inadequacy.

Moving forward, improvements to the algorithm are needed for enhancing the device's reliability and accuracy. One potential strategy involves implementing a rolling average mechanism, using the past 5-6 measurements to mitigate for the erratic fluctuations and reduce the error margins. Another alternative would be extending the buffer window from the current 2 seconds to a longer duration, perhaps spanning 5-10 seconds. This would be a much easier approach.

By addressing this critical limitation, future versions of the ECG prototype can aspire to deliver more consistent and dependable results. Despite the challenges encountered, the project serves as a valuable learning experience, offering invaluable insights into the intricacies of biomedical instrumentation design and implementation.