ENGR-355
Software strategy for the class project

The class project definition describes a device that will digitize an EKG waveform (from an analog input), determine the time between beats, and display the rate. Because we have a processor, A/D converter, D/A converter and other I/O hardware in the microcontroller it is possible to implement other desired features. While conceptually additional features may seem simple it still takes time to implement and debug them. Thus I recommend that you design your software with a structure that allows basic functionality to be achieved first and then add additional features as time allows.

Modularity is a key concept to keep in mind. Note that we started programming on our embedded system by creating functions (we can call these modules) for writing information to the LCD display, enabling and using the PIT, etc. I suggest that you continue to use a modular approach.

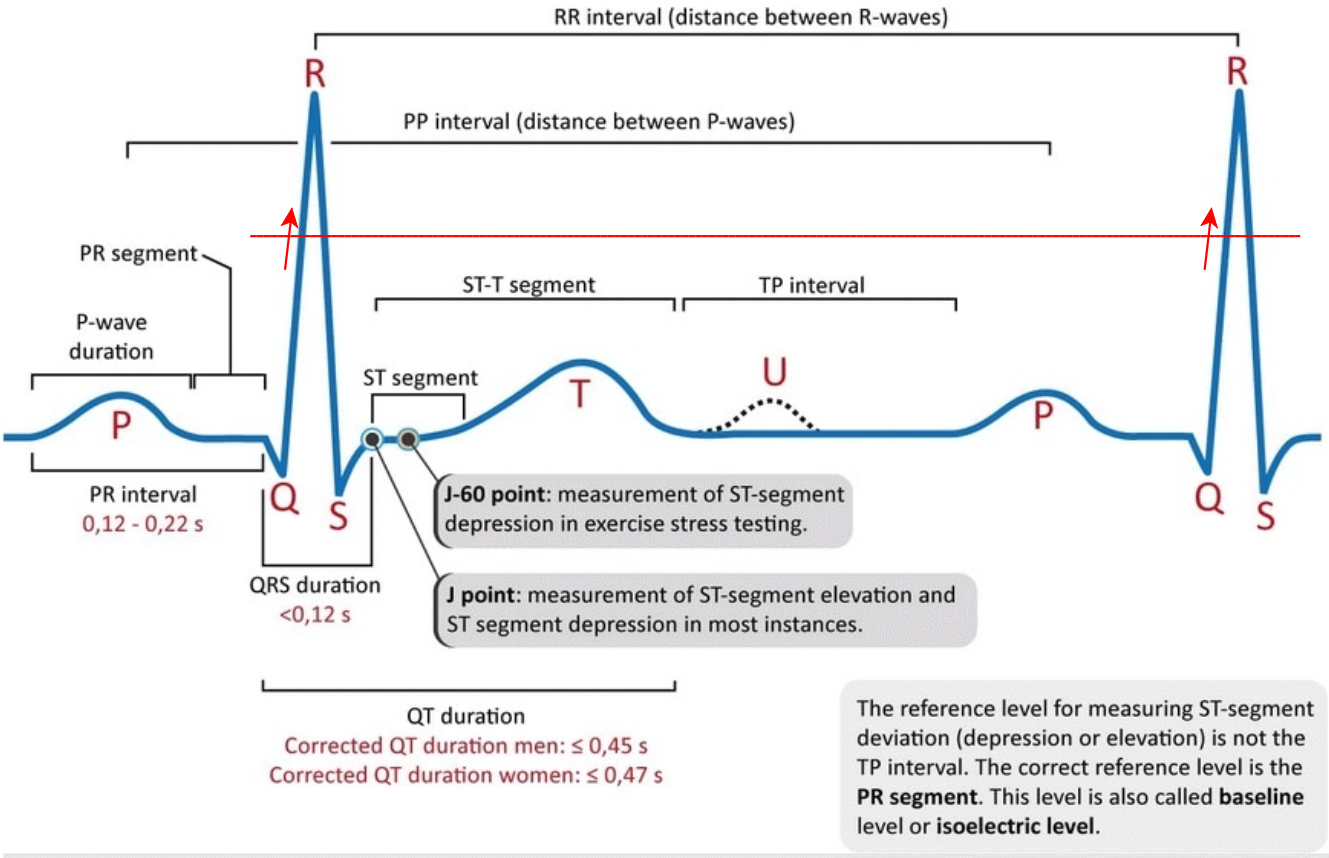
Regarding the overall structure of your program, recall the concepts from chapter 3. Create task modules that can be called from a scheduler. The scheduler can be structured as a state machine which should make it reasonable to understand, implement, and debug.

EKG waveform details

As you likely know, the voltage of an EKG signal obtained using electrodes attached to a person is small, approximately one millivolt peak-to-peak. Since our ADC has an input range of about 0 to 3volts an amplifier with a gain of 1000 or more would be needed to scale the EKG signal to the range of our converter. However, our circuit does not have a high gain amplifier. Rather, we assume that the EKG signal has been amplified before connecting it to our circuit. We will use a simulated EKG signal for testing our system and confirming operation. The Agilent 33250 signal generators have a button labeled Arb (Arbitrary) which means this generator can read a set of data from memory and create most any shape waveform that can be created with a finite number of data points. Built in is an EKG waveform. We will use that to test our project.

NOTE: Before connecting the signal generator to your project set the peak-to-peek voltage and the offset voltage so the waveform voltage is always positive and never negative. Check the waveform with the oscilloscope before connecting to the microcontroller A/D input.

The figure on the next page shows a typical EKG waveform annotated with letters like QRS that refer to particular parts of the waveform. Note that the R wave is the dominant feature of the waveform and thus should be the easiest to identify and use to determine heart rate in beats per minute (BPM). While we only need to be concerned with finding the RR interval to determine BPM the figure shows other intervals and timings of interest to a clinician who is evaluating heart function.



(https://ecgwaves.com/topic/ecg-normal-p-wave-qrs-complex-st-segment-t-wave-j-point/)

As implied in the figure, to find the RR interval one could look for an R-wave top and then for the next R wave top. However, that may not work well. I suggest that your algorithm might be to determine the maximum value of the waveform (R wave maximum) and minimum of the waveform (S wave minimum) to find the peak-to-peak magnitude and then set a target trigger level at say 70-80% of peak (illustrated by the red dashed line in the figure) and look for a rising voltage that crosses the target level (the red arrow). That starts a time interval (reset a count) and a search begins for the next rising voltage that crosses the target level. When this second crossing is found use the count value. If you just look for a peak level to find the first R wave and then expect to find the exact same level for a second R wave you could miss the second R wave because the magnitude of the R wave can shift a little from one beat to the next.

The rate at which the waveform is sampled needs to be fast enough so that if data points are plotted it would graphically reproduce the waveform (the Nyquist sample rate is not the metric to use here). A one millisecond interval between A/D samples is suggested. Use the PIT to measure time and start A/D conversions. Configure the ADC to create an interrupt when conversion is complete. You might not need data that often and if not just count and skip a sample or a few. To determine if the waveform is rising or falling you will need to keep a running list of the current value as well one or more prior values.

Do not send data to the LCD display from within an interrupt handler.

A strategy to detect the R waves of the EKG waveform and measure time between them might go something like this. See figure below.

1) For some length of time, say 5 seconds which is nominally 5 beat times, search for the maximum and minimum values coming from the ADC. Waveform magnitude = (max-min). Set a positive going threshold at about 80% of waveform magnitude. Set a negative going threshold at 70% of waveform magnitude.
2) Find when positive going on the waveform (current data point greater than previous might work if there isn't too much noise, otherwise looking back more a couple data points might be better)
3) If positive going and magnitude changes from below the "going up threshold" to above the threshold then reset a variable to zero that will count the number of time ticks to the next R wave. Set a flag to denote crossing the threshold. Time ticks are the interval between A/D conversions, i.e. the period of time that the PIT is set to.
4) Then to avoid being tricked by small up/down ADC data changes due to noise don't allow another rising edge detection until waveform magnitude falls below the going down threshold by use of the flag variable. Rest the flag when going below the down threshold. This is what we call hysteresis.

5) Don't write to the LCD from within the IRQ handler.