# Notes on Technical Challenge:

* I’ll start with a simple Visual Studio console application to flesh out the program, then I’ll add in the platform-dependent code and build system afterwards.
* The max count for a 16-bit timer is 65535, so this timer will overflow back to 0 right after it triggers the LED.
* Since the timer is a 21.6kHz clock, it has 216 ticks after it reaches 65000 to toggle the LED. Luckily, this will occur before the timer overflows, so I don’t need to account for the possibility that the timer has overflowed but the LED could still be toggled.
* The initial part is easy: check the incoming signal to see if it’s 0xFDE8 to see if the LED should be triggered.
* Here’s how I’ll deal with the 30ms requirement:
  + Instead of immediately starting the next logic, the function should set an “enable” flag and queue a function callback to execute the next part of the logic.
    - On a real-time system, this callback would be added to the scheduler’s queue. For the sake of simplicity, my code will just call the callback immediately.
  + Each time the timer tick increments after the enable flag has been set, the logic will check to see if the timer count ends in “216” or 0xD8. If it does, then the enable flag will be turned off.
    - Once again, on a real board this logic would either occur in it’s own thread or be implemented in VHDL, but for simplicity it will happen serially.
  + The first thing the function callback will do is check to see if the enable flag is still set. If it is, then the next step will commence. If not, then the window was missed and nothing should happen (in real life, some sort of error or warning would be saved off).
* In a real time system, to actually toggle the 2nd LED off and on, each off/on action would be it’s own function callback. For simplicity I’ll comment where the callback would be located but then call the function directly.
* I’ve been avoiding it, but since the 16-bit timer will overflow during the time that the 2nd LED is flashing, I’ll have to increment a variable with the timer to count how long the 2nd LED should stay off and on.
  + I’ve been avoiding it because storing the exact clock tick on every tick can be extremely expensive, depending on the system.
* Since the 2nd LED needs to be turned on and then off 5 times, it needs to be toggled 10 times. It will be easier to keep track of how many times it’s been toggled than to keep track of how many times it’s been off or on.
  + This also means that, though LED2 is flashing 5 times a second, we need to toggle it once every 100ms.
* This code works, so I’ll add in a makefile for build automation and upload everything to GibHub!

## Assumptions:

* Based on the Heartbeat picture, the timer reaches 65000 about once every 3 seconds, so I’ll assume the timer is a 21.6kHz timer.
* I’m assuming that the count needs to reach exactly 65000 (no more and no less) to begin to toggle the LED. I won’t be accounting for a missed correct input signal.
* The processor is fast enough to execute all required logic between timer ticks.
* The time that the 2nd LED needs to “on” is the same time that the LED needs to be “off” to be viewed as flashing