Christopher Downey

Korg Berlin – SW code challenge

Demonstration and notes

I use this document to briefly describe some points regarding the implementation and testing of the SW code challenge.

1 Assumptions

Given that the hold time was described as being one second and defined in the header as 6000, I concluded that the update frequency of the timer should be 6kHz.

To set the timer frequency I therefore set the timer clock to 24MHz and the period to 3999.

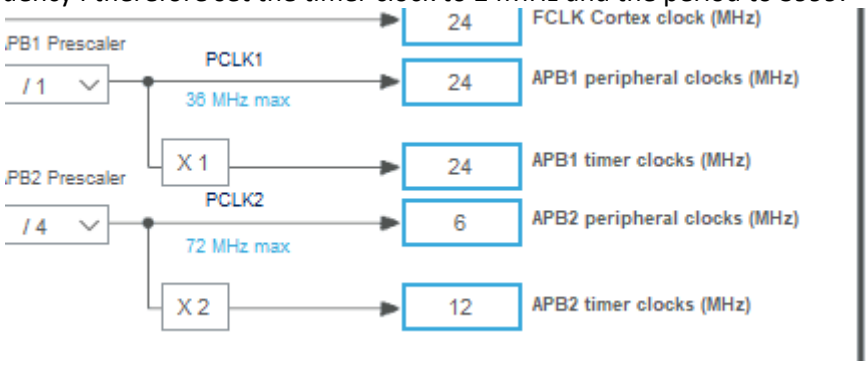


Figure 1: Timer Clock

Since the breathe period was defined as 200, giving a frequency of 30Hz (video update rate), I concluded that this was the period of a single step in the breathe animation and not, for instance, the period of the entire animation. The breathe animation would therefore be 64 breathe periods where each period was a single PWM cycle whose duty cycle determined the brightness.

The breathe period also served as a suitable period for the dim functions PWM signal.

The blink period was defined as 1500. I interpreted that to mean that an entire blink animation (LED off, then on, then off) would need to fit in this period. I therefore dedicated 500 update cycles to each of these stages of the animation.

There were defines in the header file for some offset. I couldn't determine what their purpose could be for, perhaps some offset from the time of function execution. In the end I chose not to use them, however if they indeed were intended as a time offset I would say their inclusion wouldn't be much additional effort.

2 Testing

I carried out some intermittent testing during implementation to confirm timing. This section will show some of the more important testing results.

The following figure is the output from a USB oscilloscope. In the top left corner can be seen the results of measurements between the cursors. Initially I briefly activated an LED on each timer update to be able to visualise the update frequency, confirming it to be 6kHz (1/dT).

Christopher Downey

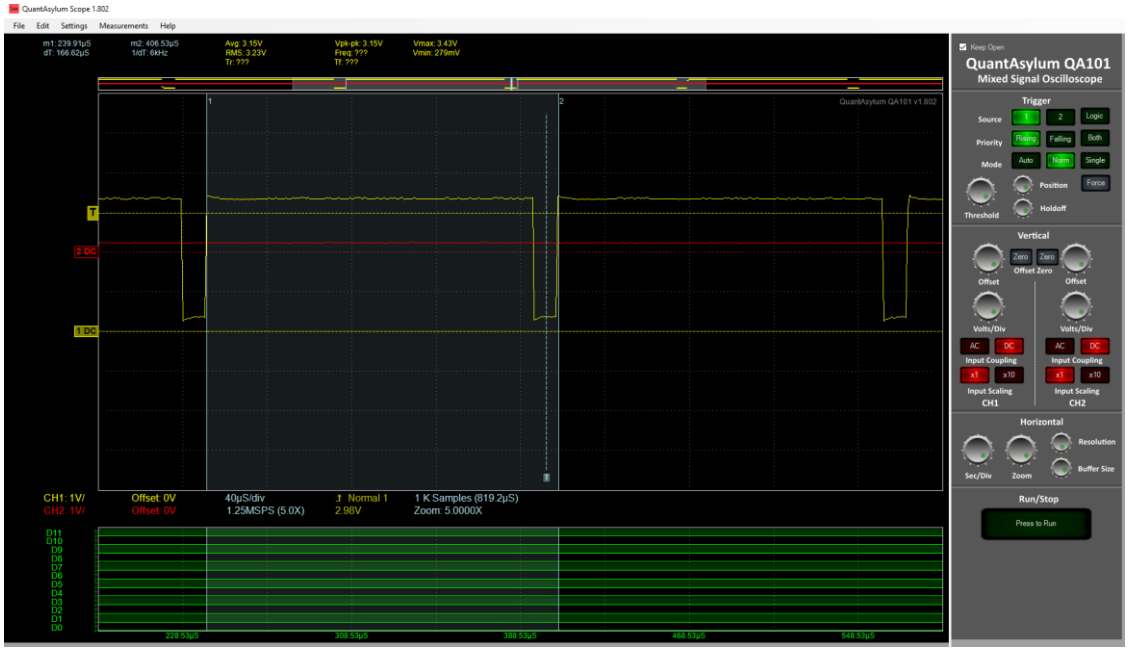


Figure 2: Update Frequency

I probed the PWM signal of the dim function to confirm it's frequency and to test the conversion from the input of the function to the duty cycle.

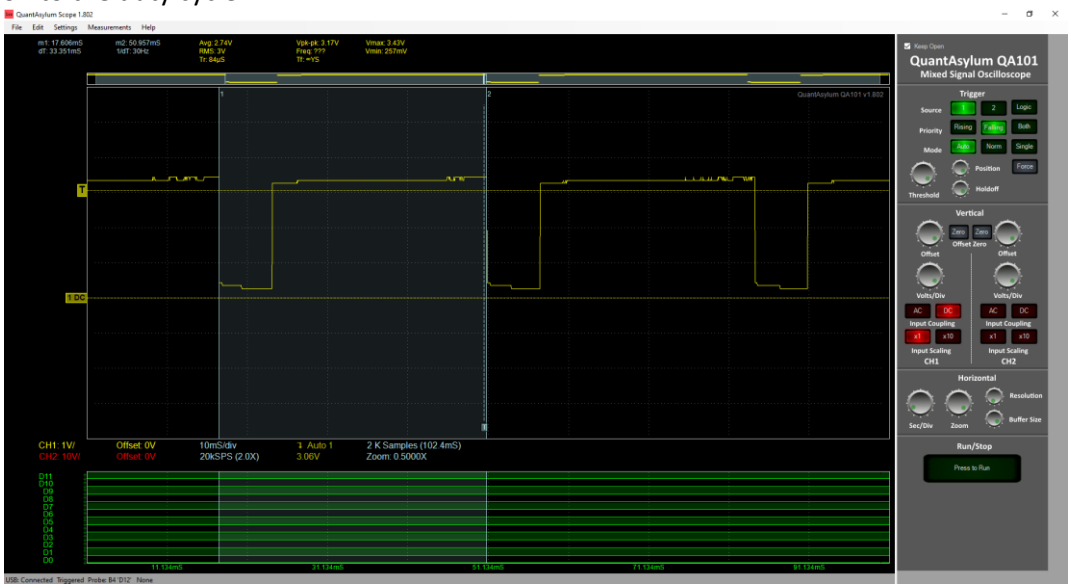


Figure 3: Dim Function Period

After implementing the blink function, I checked the frequency of the blinks to confirm it they were at 4Hz and also that it returns the LED to it's previous state after blinking. I set up an LED in a dimming mode and triggered an execution of blink using a button press, which is probed in yellow.

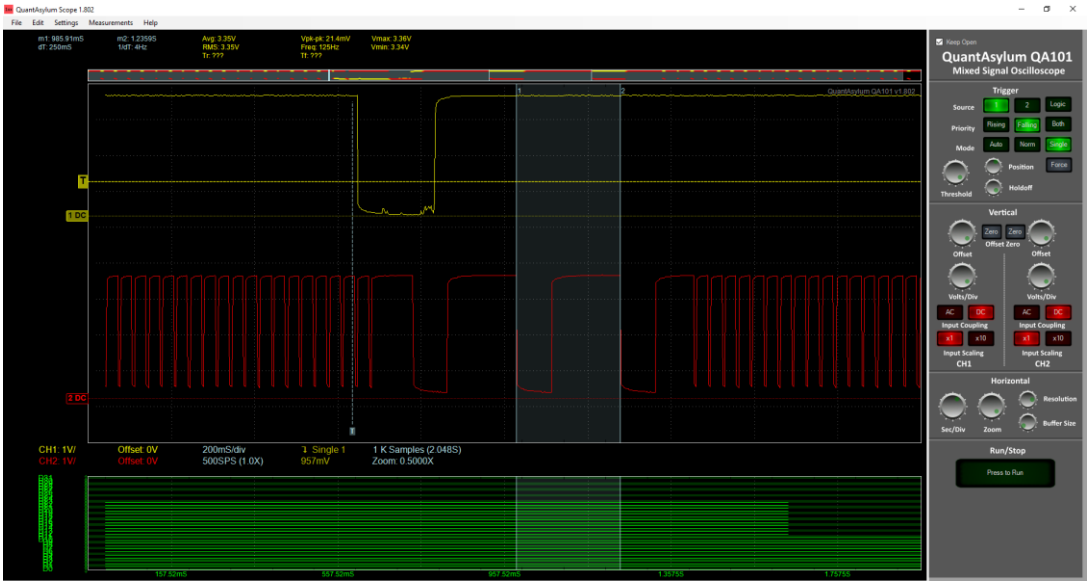


Figure 4: Blink Frequency

3 Demonstration

In section I will demonstrate the functions of the main loop as described in the README.

Firstly the LED_PWR output is probed after set in breathe mode. The duty cycle of the PWM signal linearly cycles between 0 and 100%. The resolution of my USB scope struggled with a buffer large enough to see the entire animation, as such a couple of cycles can't be see in the figure below when the duty cycle gets close to 0 and 100%, but it was visible when zoomed in.

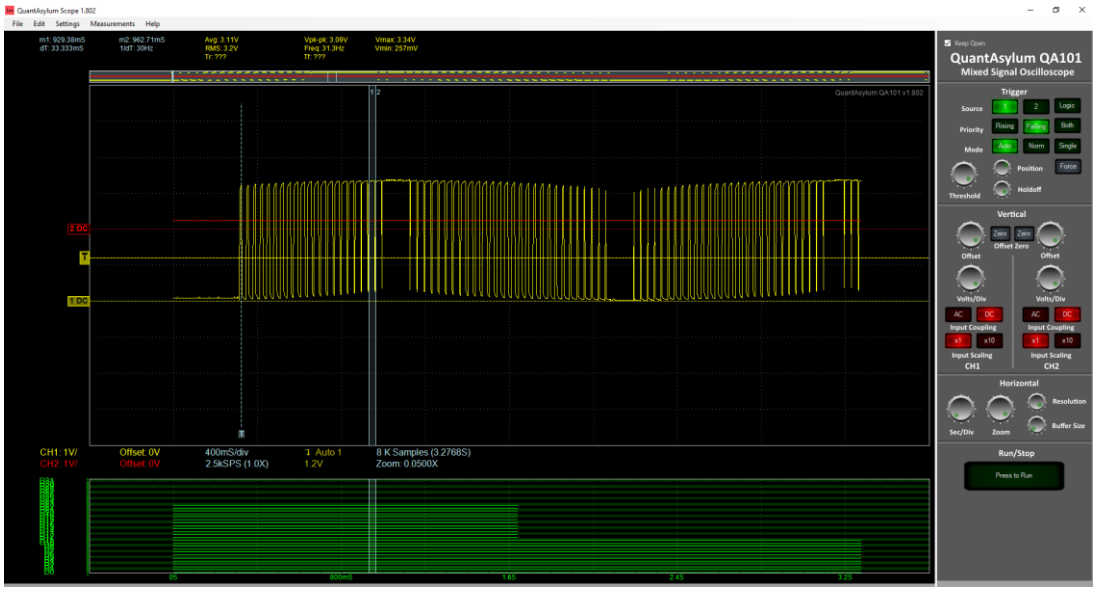


Figure 3: Breathe Animation

Next LED_CLIP is set to 20% brightness. Taking the period of the signal from Figure 3 and the period of the "low time" in the figure below it confirms a brightness close enough to 20%, with some small error tolerated due to the resolution of 32 brightness levels.

Christopher Downey

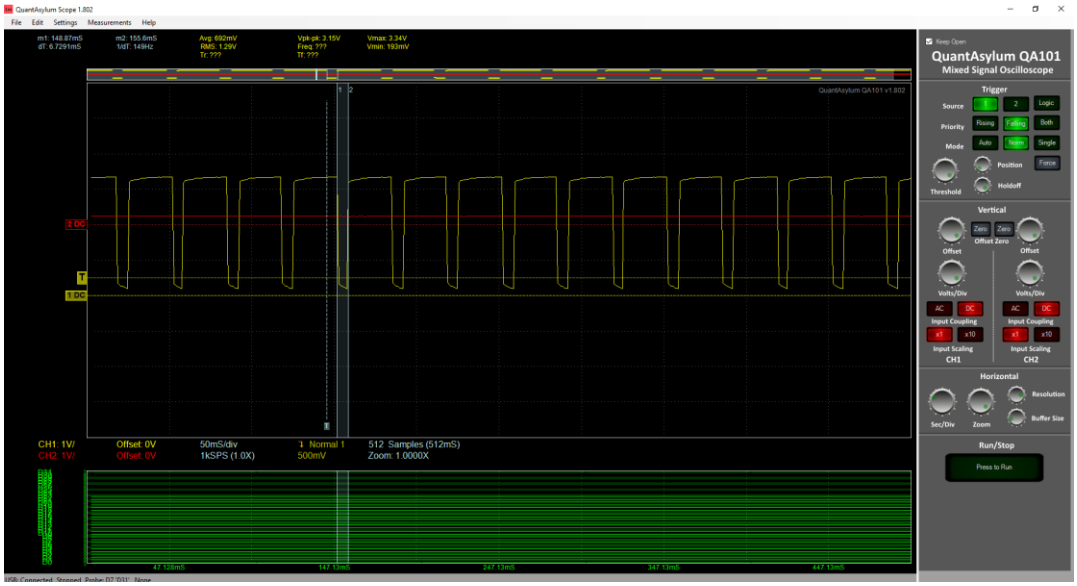


Figure 4: Dim Duty Cycle

The next two figures show LED_SW1 (red) toggling it's state on the press of a button (yellow).

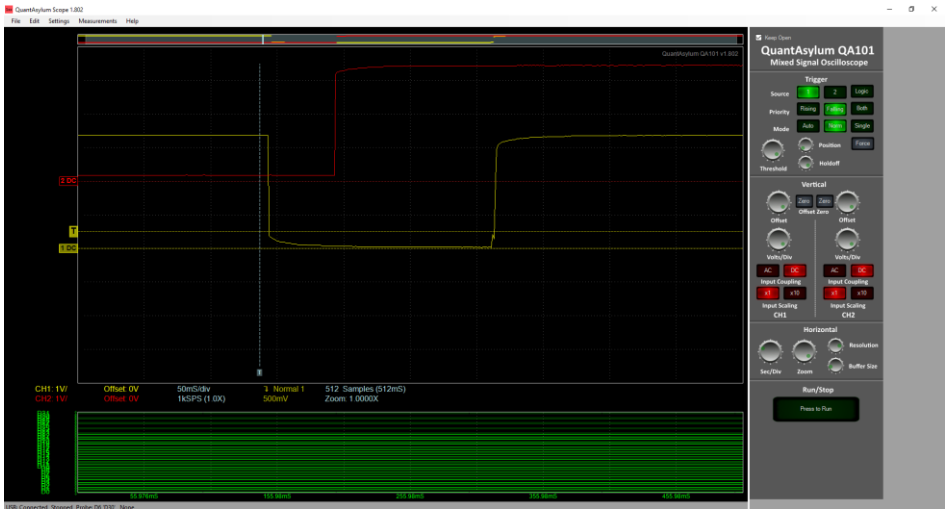


Figure 5: Toggle On to Off

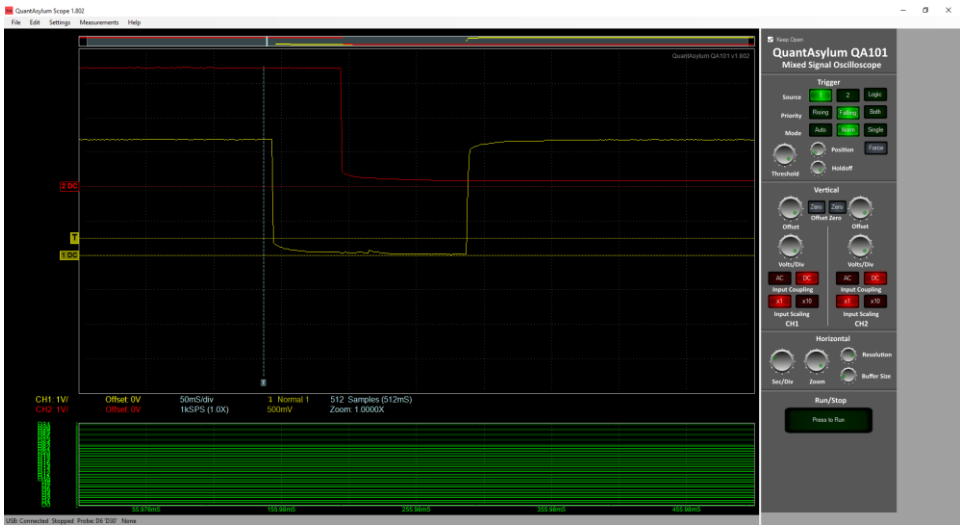
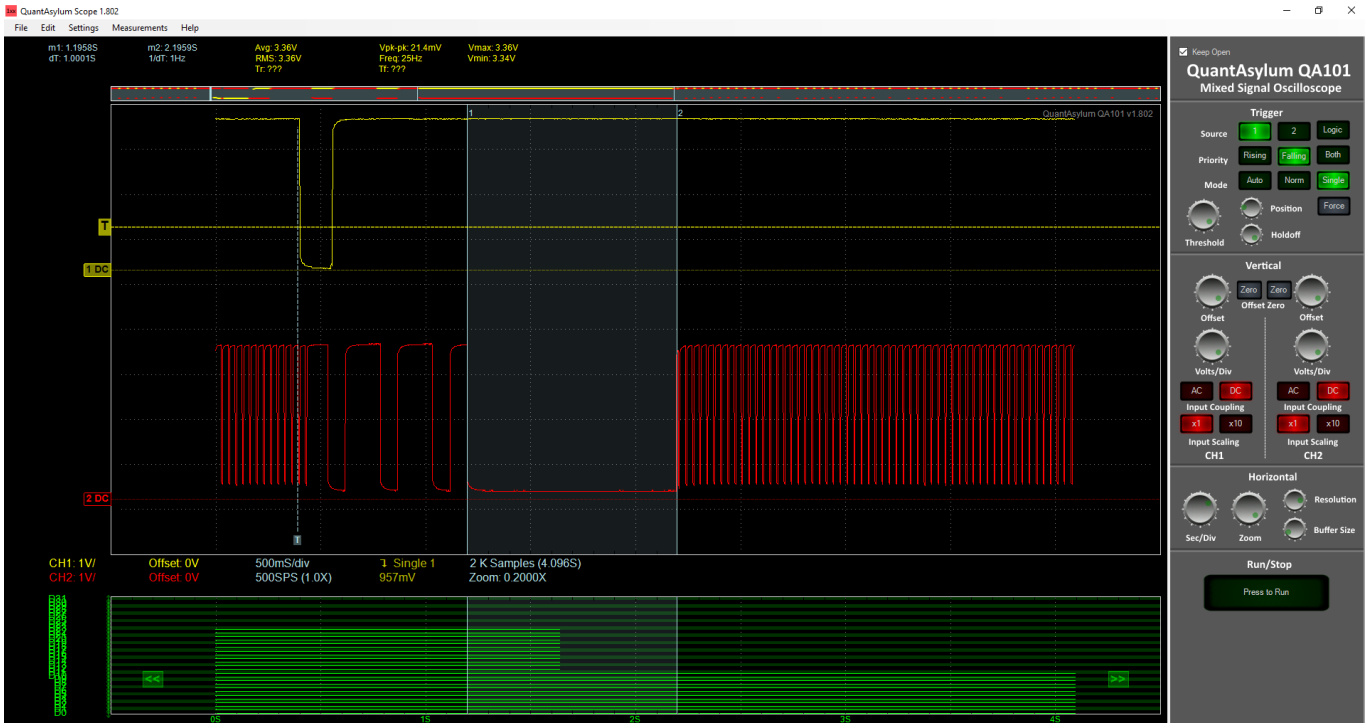


Figure 6: Toggle Off To On

Finally the following figure plots the output of LED_CLIP (red), which was on 20% brightness, as it “blinks and holds on” on a button press (yellow) and then returns to its previous state. The period of the hold is also confirmed here.



4 Conclusions

Ultimately the implementation fulfils the functions as described in the README document and conforms to the given header file. The implementation is quite conditional in my opinion and ideally I would have explored some other implementations.

A more hardware based approach for instance could have been quite elegant, with the update timer perhaps directly triggering other timers with PWM channel outputs (connected to the LEDs) whose period could be controlled to implement the dimming, breathing, blinking etc. However this approach would have diverged too much from the class implementation provided in the header file.

State Pattern Design could also have been a viable approach and would have removed the need for many of the conditionals used. I would need some more time to be comfortable trying such an approach and ultimately I think the necessity of creating multiple more classes may have meant too much altering of the header file.