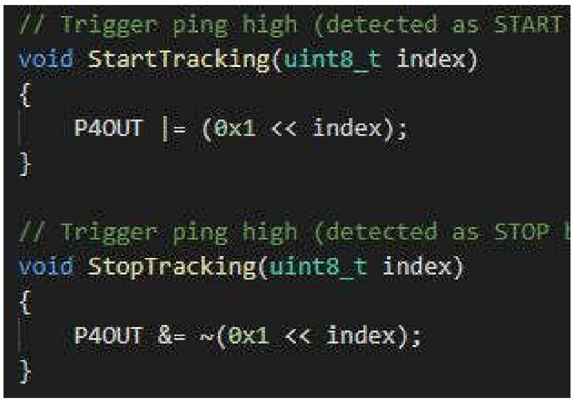
Appendix A - Challenge 1 (Timing script)

This appendix introduces the **TCL timing script** for GTKWave and measure the high-time of a **signal indicating overhead**. This signal is trigger by the TimeTrack code (startTracking, stopTracking), but because we don't use PrintResult and we **do** use output pins, the result is immediately visible.

Code adjustment:

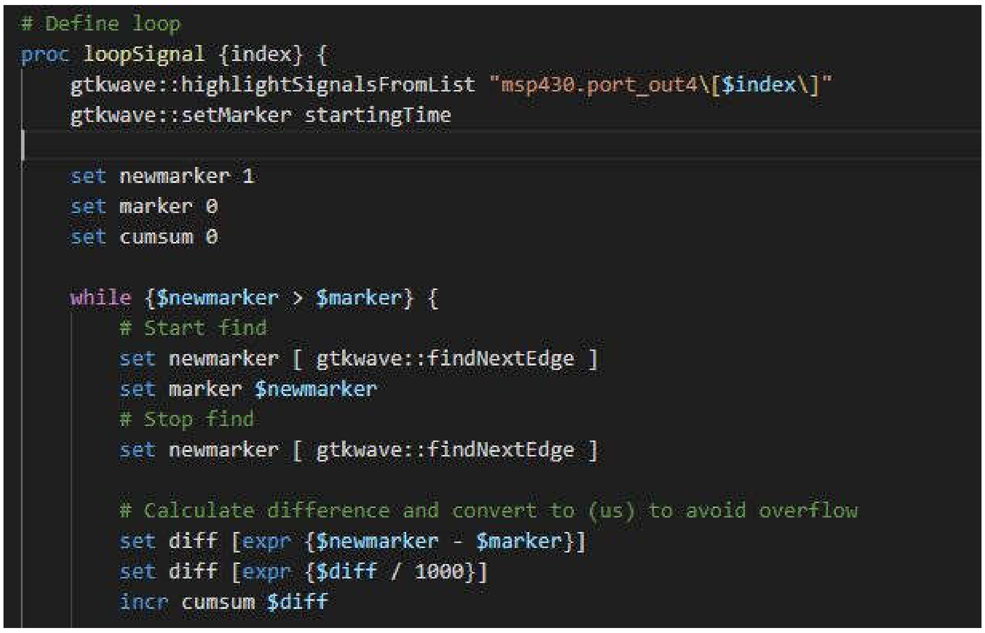
- Use P4out pins
- Define pins as constants
- Write to pin on pin bus with bit mask and the constant
- (Our implementation) stop triggering signal after 1 hyperperiod.

Script output: micro-second accuracy high-time of a signal for one hyperperiod **on each P4 pin.**



```c
// Trigger ping high (detected as START
void StartTracking(uint8_t index)
{
    P4OUT |= (0x1 << index);
}

// Trigger ping high (detected as STOP
void StopTracking(uint8_t index)
{
    P4OUT &= ~(0x1 << index);
}
```

Setting pin on bus with bit-mask.



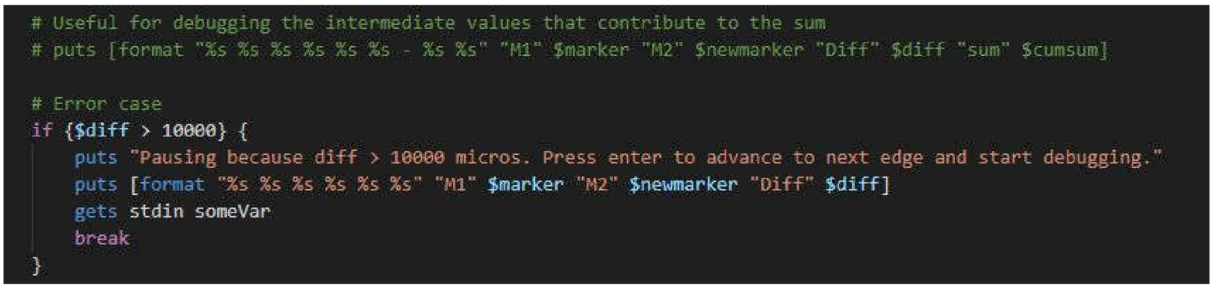
```tcl
# Define loop
proc loopSignal {index} {
    gtkwave::highlightSignalsFromList "msp430.port_out4\[$index\]"
    gtkwave::setMarker startingTime

    set newmarker 1
    set marker 0
    set cumsum 0

    while {$newmarker > $marker} {
        # Start find
        set newmarker [ gtkwave::findNextEdge ]
        set marker $newmarker
        # Stop find
        set newmarker [ gtkwave::findNextEdge ]

        # Calculate difference and convert to (us) to avoid overflow
        set diff [expr {$newmarker - $marker}]
        set diff [expr {$diff / 1000}]
        incr cumsum $diff
```

The TCL code to highlight the correct trace by pin $index, to find the new edge. Do that twice and you'll stick with signal-is-1-time.

The code needs to start after the first reset edge, that's why the script starts from the startingTime (now **1ms**) (which also allows skipping other setup time events).

The **loopSignal** procedure can be called with any index, making it very convenient to call it for a specific index and aggregate the results from the outputs.
Our implementation uses pin 1 and 0 on the P4 bus and calculates the difference between the markers placed on the positive and adjacent negative edge. This difference in nano-seconds is converted to micro-seconds to prevent overflow in the script.

The cumulative sum is incremented with the time difference between the rising and falling edge to count the total. After no new edge is found (same edge returned by GTKWave, the marker hasn't increased), the total value is returned. The script below **loopSignal** prints the value in a convenient way.



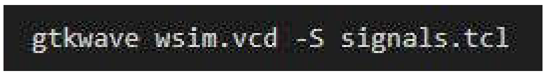
```
# Useful for debugging the intermediate values that contribute to the sum
# puts [format "%s %s %s %s %s %s - %s %s" "M1" $marker "M2" $newmarker "Diff" $diff "sum" $cumsum]

# Error case
if {$diff > 10000} {
    puts "Pausing because diff > 10000 micros. Press enter to advance to next edge and start debugging."
    puts [format "%s %s %s %s %s %s" "M1" $marker "M2" $newmarker "Diff" $diff]
    gets stdin someVar
    break
}
```

The error detection and handy output in loopSignal.

Final tips

- Dont let the signal spike the whole simulation, because the script is not instantly done. It has to detect a lot of edges, so limit the signal measurement to f.e. 1 hyperperiod. In that case the script takes less than ~2 seconds.
- Dont use all pins, again the edge detection isn't quite fast because of the GUI updating in between. You'd soon find yourself waiting 5 seconds per run.
- Stop tracking as fast as possible to prevent overflow and unclarity. Measurements beyond 10ms will be questioned and paused by the error detection.
- Use `gets stdin someVar` to pause and wait for input. This allows breaking and debugging the script or looping through the edges of the trace.
- The TCL file will sum all signals until the end of the simulation. In order to detect only 1 hyperperiod while simulating for longer, use conditional statements to only use startTracking during this 1st hyperperiod

Take a look at the way to call the .tcl script:



```
gtkwave wsim.vcd -S signals.tcl
```

Find the gist for **signals.tcl**:
https://gist.github.com/davidzwa/ef1eafc6cd23e613af612e27eddb054b