The University of Western Australia
Electrical, Electronic and Computer Engineering
A/Prof. Thomas Bräunl

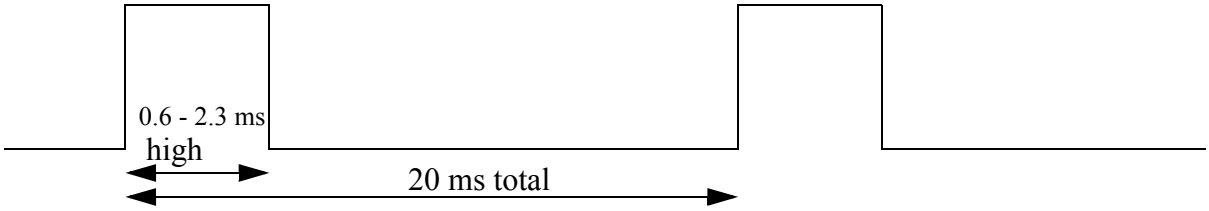
Embedded Systems
ENGT2303

Lab Assignment 4 week 6

Equipment used: EyeBot boxes and interface boxes

EXPERIMENT 1 Motor Signal Measurement

Servos are DC motors with built-in logic. They are e.g. used in model cars, airplanes, ships. Servos assume a position (often in the range of 360 degrees) according to an input rectangle signal.



Connect a servo via the TPU box (timing processor unit) to the EyeBot. Then use the buttons: Hrd / HDT / Servo / Tst to move the servo to its end positions.
Connect an oscilloscope to the servo's signal pin and analyze the signal.
Write down measurements for signal length and total signal period.

EXPERIMENT 2 Read Analog Input

Write a C program that reads an analog input value from the EyeBot box using C function:

```
int OSGetAD(int channel);
     Input:      (channel) desired AD-channel range: 0..15
     Output:     (returncode) 10 bit sampled value
     Semantics:      Captures one single 10bit value from specified
                     AD-channel.  The return value is stored in the
                     least significant bits of the 32 bit return value.
```

On the interface box, connect the potentiometer with an analog input of the EyeBot.
In an endless loop, read the analog value and print it on the LCD.
If the values are increasing over time, print ("up") in addition to the value.
If the values are decreasing over time, print ("down") in addition to the value.

EXPERIMENT 3 Motor Signal Control via Output Latches in C

Extend the C program from Experiment 2 to control a servo, setting it to a position proportional to the analog input value. The servo should assume a position between its minimum and maximum position according to the analog input signal (potentiometer).
Your program still needs to print the analog input value plus "up/down".

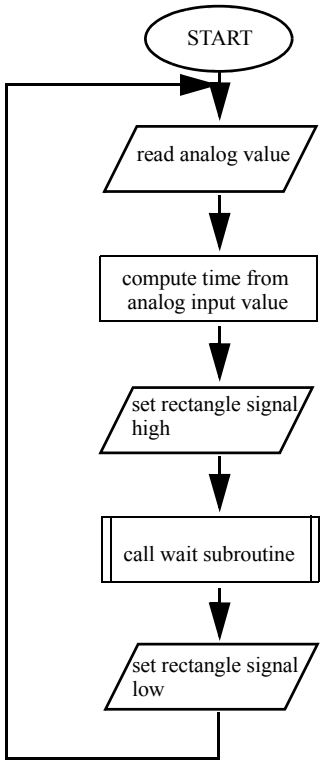
System function you may need for this are:

```
BYTE OSWriteOutLatch(int latchnr, BYTE mask, BYTE value);
        Input:          (latchnr) number of desired Outlatch (range: 0..3)
                        (mask)    and-bitmask of pins which should be cleared
                        (inverse!)
                        (value)   or-bitmask of pins which should be set
        Output:         previous state of this outlatch
        Semantics:      modifies an outlatch and keeps global state consistent
                        example: OSWriteOutLatch(0, 0xF7, 0x08); sets bit4
                        example: OSWriteOutLatch(0, 0xF7, 0x00); clears bit4

int OSWait (int n);
        Input:          (n) time to wait
        Output:         NONE
        Semantics:      Busy loop for n*1/100 seconds.
```

Note: - You can use OSWait(2) to wait for 20ms downtime, but you need to write your own idle routine (e.g. busy wait) to wait for the 0.6 .. 2.3 ms uptime
- Use 0 as latch number.
- Use the oscilloscope to verify your generated signal **before** you connect the servo!

Program structure:



EXPERIMENT 4 Motor Signal Control in C + Assembly

Write a main program in C that links with an assembly subroutine for setting the output latch to achieve the same result driving a servo as the previous experiment.
Your program still needs to print the analog input value plus "up/down".

You may re-use your main program from the previous experiment, but you have to replace the calls to `OSWriteOutLatch` with the call of an assembly subroutine.

The assembly addresses for the output latch is: `OutBase`