EECS 373 - Homework #2 - Solutions

Name: ______________________________ **unique name:** ________________

Due at 12pm (noon) on 9 February via Gradescope. We'll release solutions immediately after the deadline for use in midterm review, so late assignments cannot be accepted. Please use the answer boxes provided. Submit a PDF of your completed assignment to Gradescope. Typed answers and neat handwritten answers are both acceptable.

Question 1

Short answer questions: **[15 points, 3 each]**

A) When the APB Bridge wants to read data from a peripheral which signal does it use and what is the state of that signal?

PWRITE, the ABP Bridge will drive it low to read (0=Read)

B) What is the memory address of input data register for GPIO Port C (GPIOC_IDR)?

0x48000810

C) What does the GPIO Mode Register (MODER) control?

The GPIO Mode Register controls what mode the individual pins on a given port are in.
Options include: input, output, alternative function and analog mode

D) Why would an APB bus peripheral need to use wait states?

It is not ready to respond to the ABP bridge

E) Does there need to be a physical memory or register module for each valid memory address on an ARM Cortex-M processor, justify your answer.

No, a memory address can be represented by a state machine that acts like one or more memory address

Question 2

A) In reference to the GNU toolchain for ARM process explain in a straightforward manner what the linker does and what it is used for. **[10 points]**

A linker is a computer program that takes one or more object files generated by a compiler and combines them into one, executable program. The linker combines these separate files into a single, unified program, resolving the symbolic references as it goes along and allocating code to specific address.

B) Consider the linker script at http://www.bravegnu.org/gnu-eprog/lds.html. Briefly explain why the label “eoa” is at a higher address than the label “stop” even though the label “eoa” appears first. (Though largely irrelevant to the question, be aware this appears to be ARM not Thumb assembly). **[10 points]**

The linker script specified that the data section come after the text section. Since eoa is in the data section (which starts at location 0x400), it is at a higher (larger) address than any label in the text section (which starts at location 0x000), including stop.

Question 3

Write rewrite the function “A” in UAL assembly which does the same thing as the following C code. You should assume “print” is some ABI compliant function which takes a single integer argument and does _something_ with a (probably prints it somewhere eh?) and B is also an ABI compliant function. (Note, one point of the ABI is to be able to mix C and assembly like this, the linker will make it all work!) **[15 points]**

```
void main()
{
  int a=4,b=3;

  b=A(a,b);
  print(b);

}
int A(int x, int y)
{
  int a;
  a=B(x+y);
  print(a);
  return(a+x-y);
}
```

```
@ There are a number of ways to do this, here is one.

@ params: r0 = x, r1 = y
A:
    push {lr, r4, r5, r6}
    mov r4, r0
    mov r5, r1
    add r0, r0, r1    @ r0 = x + y
    bl B              @ a = B(r0)
    mov r6, r0
    bl print          @ print(r0)
    add r0, r6, r4    @ r0 = a + x
    sub r0, r0, r5    @ r0 -= y
    pop {lr, r4, r5, r6}
    bx lr
```

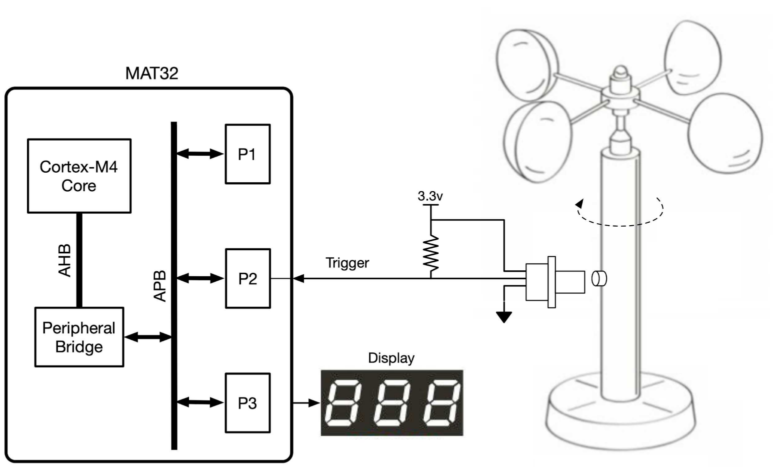
Question 4: Anemometer Design Problem

Your task is to design the measurement system for a hand-held anemometer. The device measures wind speed with a traditional rotating cup system. Wind speed is determined by measuring the rotation period, scaling and displaying accordingly. Your task is to design the peripheral modules that are connected to the APB bus and function properly with the core. Then write a C function that interfaces with each module and calculates the wind speed.

You will likely find it helpful to read the rest of the problem before solving any of the subparts.

Interface to the APB bus

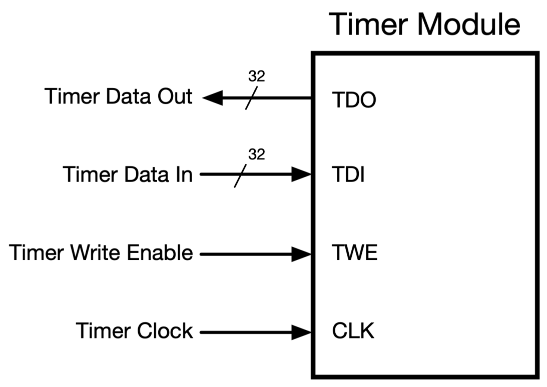
The MAT32F20 is a new ARM Cortex-M4 microcontroller designed by Matt Smith and is identical in nearly every way to the STM32L4 microcontroller. Matt wants your help in designing two custom APB peripheral modules for his MCU, an input for the hall effect sensor and a custom 32 bit timer module. In order to measure the rotational speed of the rotating cups a magnet has been placed on the rotating shaft of the anemometer. When the magnet passes in front of the hall effect sensor a trigger signal is sent to the MAT32.



This fictional MCU has the following APB3 bus interface. The ABP bus signals follow APB timing and protocol. Read and write cycles are provided on the next page. PSEL is configured to be “1” when memory locations **0x40050000-0x40050007** are accessed.

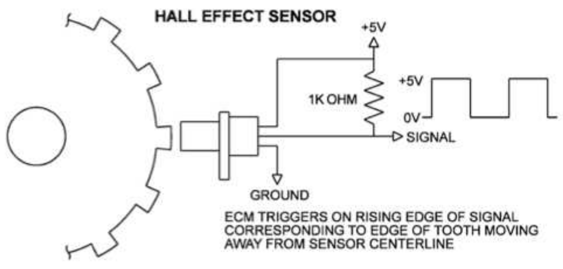
Integrated Hardware Timer

The hardware timer increments the value stored in its internal register on every rising edge of the *Timer Clock (CLK)*. When the *Timer Write Enable (TWE)* is set high the value on *Time Data In (TDI)* is latched in the timer module on the positive edge of the *Timer Clock (CLK)*. If the counter reaches its maximum value it wraps around to zero (*i.e.* it is a modulo counter). The current value of the timer is always available on *Timer Data Out (TDO)*.



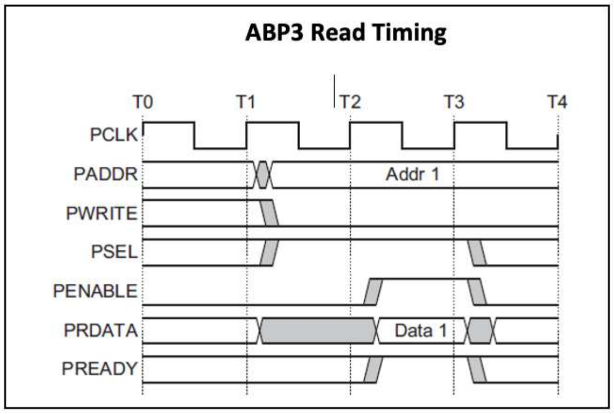
Hall-Effect Sensor - AH1815

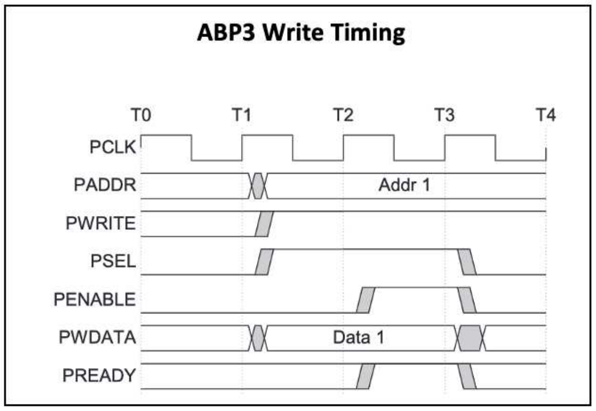
The AH1815 is an integrated Hall-Effect, non-latched sensor. When sufficient magnetic field strength is detected the sensor will output a high signal otherwise the sensor will output a low. Thus holding a magnet near the sensor will cause the output pin to toggle. If a magnet is attached to a rotating shaft a hall effect sensor is an easy way to measure the rotation.
https://en.wikipedia.org/wiki/Hall-effect_sensor


HALL EFFECT SENSOR
+5V
1K OHM
+5V
0V
SIGNAL
GROUND
ECM TRIGGERS ON RISING EDGE OF SIGNAL CORRESPONDING TO EDGE OF TOOTH MOVING AWAY FROM SENSOR CENTERLINE

APB Timing diagram

The following diagrams are provided as a reminder of the APB timing with no wait states





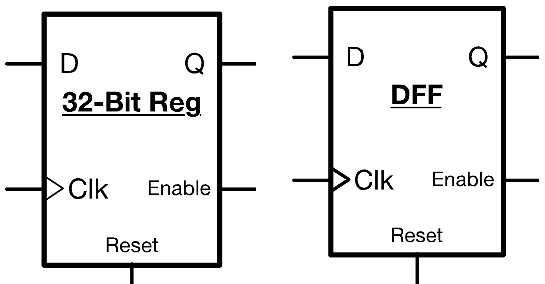
APB Bus Conventions

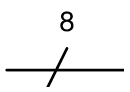
You may use standard gates such as ANDs, ORs, NOTs (as well as standard bubbles) and DFFs. Be sure to show all connections. You may use GND and VCC to indicate a logical 0 and 1 respectively. You may not use Boolean or Verilog expressions.

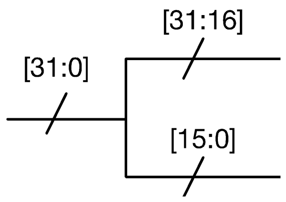
Example DFF and Register (enable is optional)

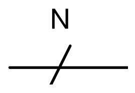
Label the number of lines in a bus.

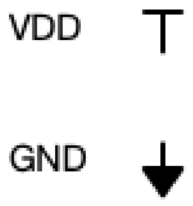
Bus concatenation and data labeling





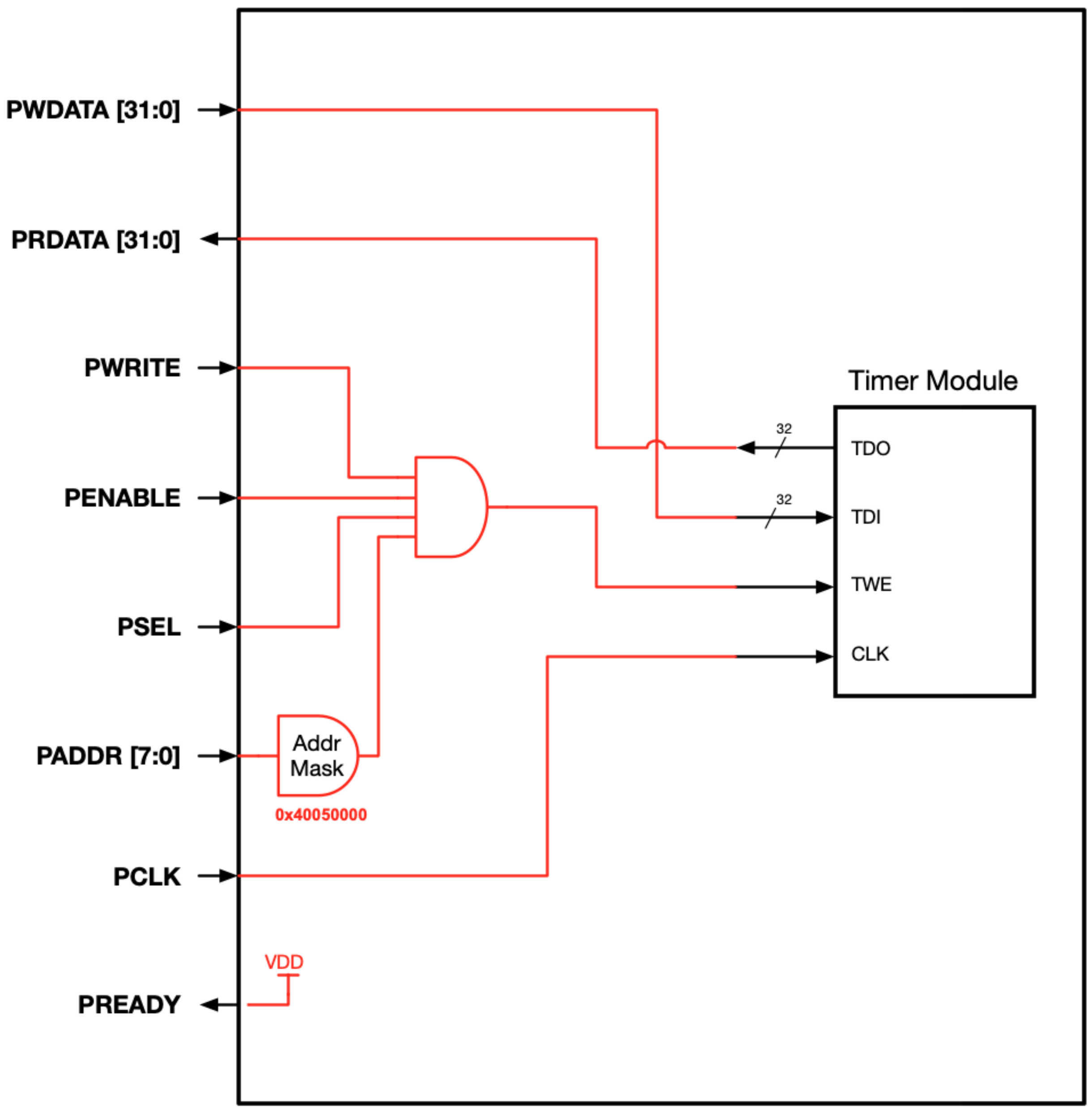



N



Part 1: Hardware Timer APB3 Bus Interface (16 points)

Provide a hardware interface between the APB bus and the hardware timer to allow the MCU to read and write the counter. Assume the timer's counter register is at address **0x40050000** *(and only that address)*. You may use standard gates such as ANDs, ORs, NOTs (as well as standard bubbles). Be sure to show all connections. You may use GND and VCC to indicate a logical 0 and 1 respectively. You may not use Boolean or Verilog expressions. You will lose points for having unneeded logic. ***Note:*** we have put all the APB connections on the left. *This means inputs and outputs are intermixed, be sure you are driving all outputs!*

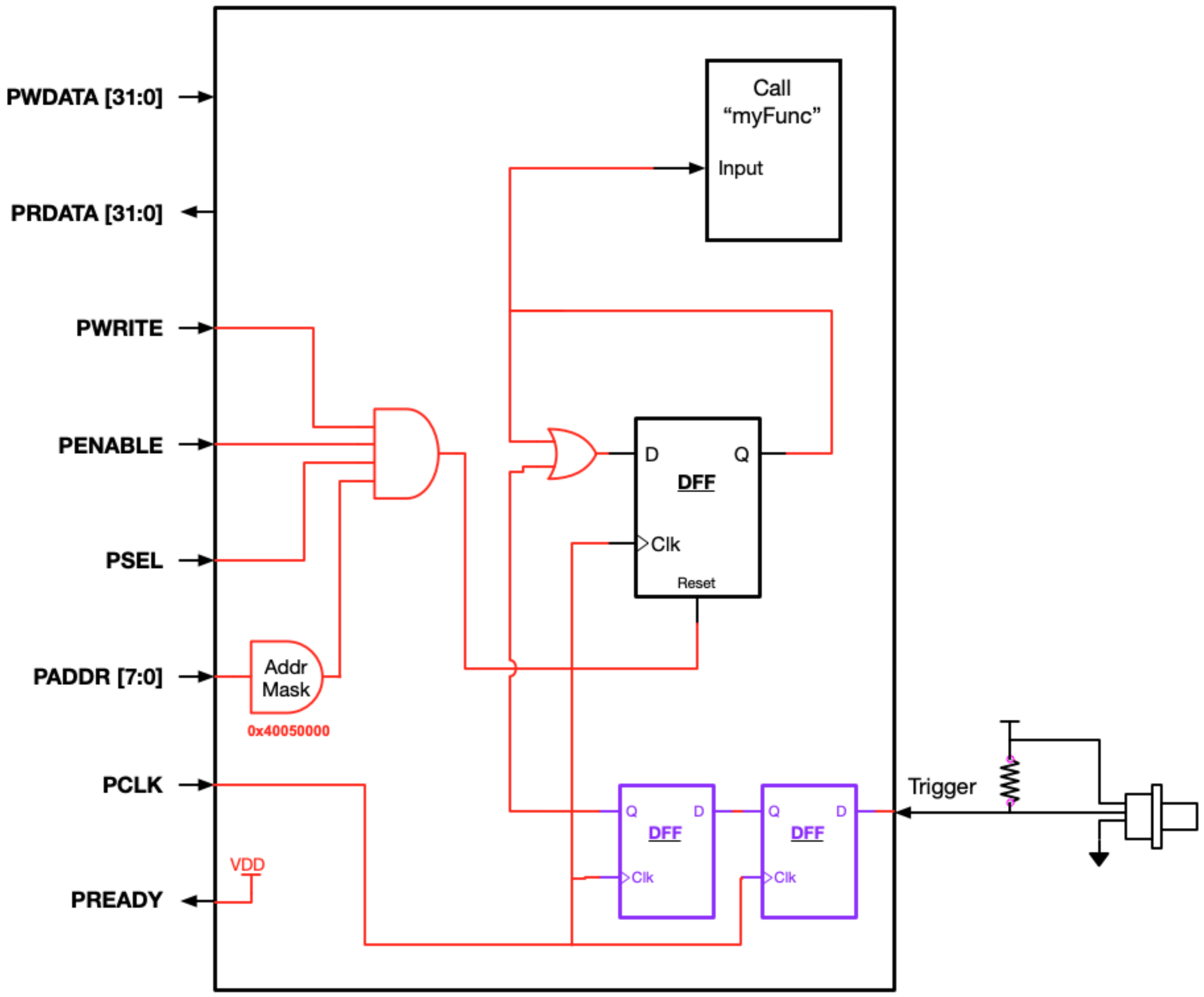


Part 2: Hall Effect Sensor Peripheral Module (16 points)

Each time a magnet passes in front of the hall effect sensor it provides a high pulse (*Trigger*) that is several clock cycles (*PCLK*) wide and is not synchronized to the APB clock. Matt has designed a special hardware block called "Call myFunc" which will cause the ARM Core to call a function when the input line is held high.

- Design logic using standard gates and D flip-flops that will hold the *input* to the *Call myfunc block* high when the *Trigger* from the hall effect sensor goes high. Your module must hold *input* high until it is cleared
- One D flip-flop is provided, though you may use additional ones as needed.

- Also, provide logic that will clear the high *Input* signal to the *Call myfunc block* when a write to memory address **0x40050004** *(and only that address)* occurs.



Part 3: "myFunc" (18 points)

Write a function *in C* called "myFunc" that will calculate and display the wind speed by reading the time of a rotation, scaling and then displaying the result using your hardware designed above.

- Assume that you have a function called **scale** that will convert the time of rotation to KPH. Scale accepts time per rotation in microseconds.
- Assume that you have another function called **display** that accepts KPH and provides display. It is acceptable if the first speed value is incorrect. All subsequent values should be correct.
- Assume the APB bus runs at 1 MHz.

Function Prototypes:

(void) display (int kph)

(int kph) scale (int ms)

Providing comments may improve your chance for partial credit.

```
void anemometer_interupt (void) {
  * int timer = (int *) 0x40005000;
  * int interupt = (int *) 0x40005004;

  display (scale (*timer)); //assume first reading might
                            // be wrong

  *timer = 0; // clear timer
  *interrupt = 0; // clear interrupt
                  // data value doesn't mater

 return;
}
```