Name: **Lab Section:**

CprE 288 Fall 2018 – Homework 4

Due: Not collected, or graded.

Notes:

- Homework must be typed and submitted as a PDF or Word Document (i.e. .doc or .docx) only.
- If collaborating with others, you must document who you collaborate with, and specify in what way you collaborated (see last page of homework assignment), review the homework policy section of the syllabus: http://class.ece.iastate.edu/cpre288/syllabus.asp for further details.
- Review University policy relating to the integrity of scholarship. See ("Academic Dishonesty"): http://catalog.iastate.edu/academic_conduct/#academicdishonestytext
- Late homework is accepted within two days from the due date. *Late penalty is 10% per day.* ***Except on Exam weeks***.
- ***Note:*** *Code that will not compile is a typo. Answering a question as "will not compile"* ***will be marked incorrect****. Contact the Professor if you think you have found a typo.*

Note: Unless otherwise specified, all problems assume the TM4C123 is being used

Question 1 (10 pts): I/O Ports

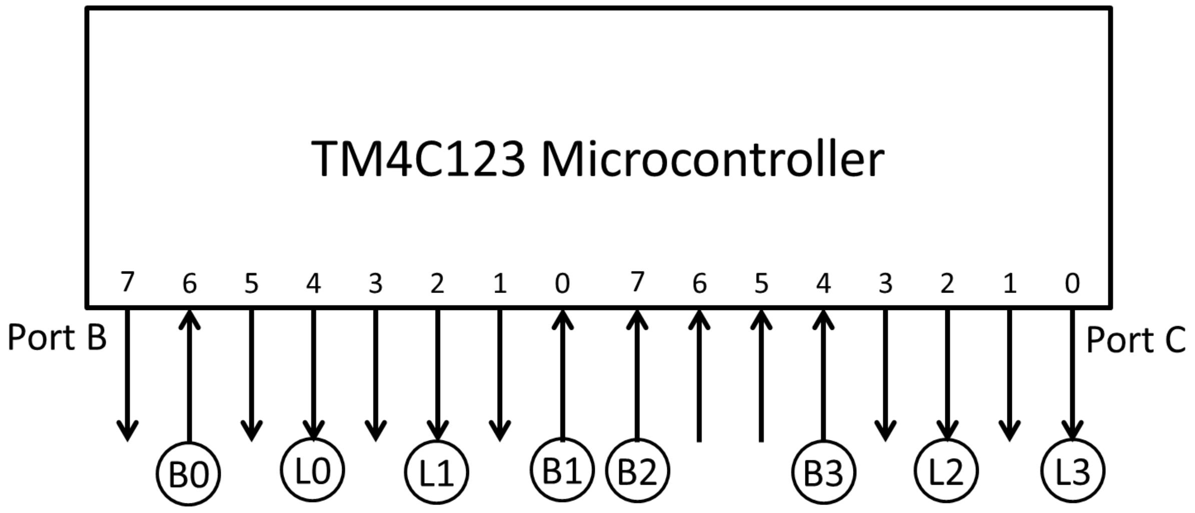
a. Complete the function `init_ports` based on the comments and figure below to configure Port B and Port C, while preserving the other Ports' configuration. (5pts)

```
void init_ports(void){
  // Start Port B and Port C clock
SYSCTL_RCGCGPIO_R = 0b000110;   // 0x03

  // Enable Digital functionality of Port B and Port C
GPIO_PORTB_DEN_R = 0xFF;
GPIO_PORTC_DEN_R = 0xFF;

  //Set direction of Port B's pins, based on the figure
GPIO_PORTB_DIR_R = 0b1011_1110;   // 0xBE

  //Set direction of Port C's pins, based on the figure
GPIO_PORTC_DIR_R = 0b0000_1111;   // 0x0F
}
```



Name: **Lab Section:**

b. Write C code that will read in button values, and send those values to their corresponding LEDs (i.e. B0 corresponds to L0), based on the figure in part a. Add any variables that you think you may need. (5pts)

There are 4 buttons (B0, B1, B2, B3), whose values should be stored in elements 0 to 3 of buttons (i.e. button B0 should be stored in buttons[0]). There are four LEDs (L0, L1, L2, L3) that can have a value of 1 or 0 written to them. All wires that do not have an LED or button connected should have their value persevered (i.e. your code should not change their value).

```
void main(void)
{
unsigned char buttons[4];
init_ports();  // from part a

// Read in Buttons into button array
buttons[0] =  (GPIO_PORTB_DATA_R >> 6) & 0x01;
buttons[1] =  (GPIO_PORTB_DATA_R >> 0) & 0x01;
buttons[2] =  (GPIO_PORTC_DATA_R >> 7) & 0x01;
buttons[3] =  (GPIO_PORTC_DATA_R >> 4) & 0x01;

// Give button values to LEDs

  // LED 0: Port B, bit 4
  if(button[0] == 1) // check if B0 is a 0 or 1
  {  // B0 == 1, so write a 1 to L0
    GPIO_PORTB_DATA_R = (GPIO_PORTB_DATA_R | 0b00010000);
  }
  else
  {  // B0 == 0, so write a 0 to L0
    GPIO_PORTB_DATA_R = (GPIO_PORTB_DATA_R & 0b11101111);
  }


  // LED 1: Port B, bit 2
  if(button[1] == 1) // check if B1 is a 0 or 1
  {  // B1 == 1, so write a 1 to L1
    GPIO_PORTB_DATA_R = (GPIO_PORTB_DATA_R | 0b00000100);
  }
  else
  {  // B1 == 0, so write a 0 to L1
    GPIO_PORTB_DATA_R = (GPIO_PORTB_DATA_R & 0b11111011);
  }
```

Name: **Lab Section:**

```
  // LED 2: Port C, bit 2
  if(button[2] == 1) // check if B2 is a 0 or 1
  {  // B2 == 1, so write a 1 to L2
    GPIO_PORTC_DATA_R = (GPIO_PORTC_DATA_R | 0b00000100);
  }
  else
  {  // B2 == 0, so write a 0 to L2
    GPIO_PORTC_DATA_R = (GPIO_PORTC_DATA_R & 0b11111011);
  }


  // LED 3: Port C, bit 0
  if(button[3] == 1) // check if B3 is a 0 or 1
  {  // B3 == 1, so write a 1 to L3
    GPIO_PORTC_DATA_R = (GPIO_PORTC_DATA_R | 0b00000001);
  }
  else
  {  // B3 == 0, so write a 0 to L3
    GPIO_PORTC_DATA_R = (GPIO_PORTC_DATA_R & 0b11111110);
  }

}
```

```
}
```

Name: **Lab Section:**

Question 2: Memory-Mapped Registers (10 pts)

Based on the information and figure given, provide C code that will perform a 512-point FFT (Fast Fourier Transform) of the musical sound stored in `Music[]`. And transmit the transformed version of the musical sound out of Port B.

Some Background:

Fast Fourier Transform (FFT): The FFT is the most widely used algorithm for converting a signal represented in the time domain (i.e. amplitude vs. time) in to its frequency spectrum representation (i.e. amplitude vs. frequency).

Applications of FFT: The Fourier transform is commonly used to analysis what frequency components compose a signal. It can be useful for a wide range of tasks. A typical example would be identifying frequency components that one may want to remove from a system. Two specific examples of this usage would be a) identifying a frequency component causing a vehicle to vibrate to damping that frequency by means of mechanical modifications, and b) post processing of music to remove unwanted noise from a recording (a free app: http://www.free-audio-editor.com/index.php)

Name: **Lab Section:**

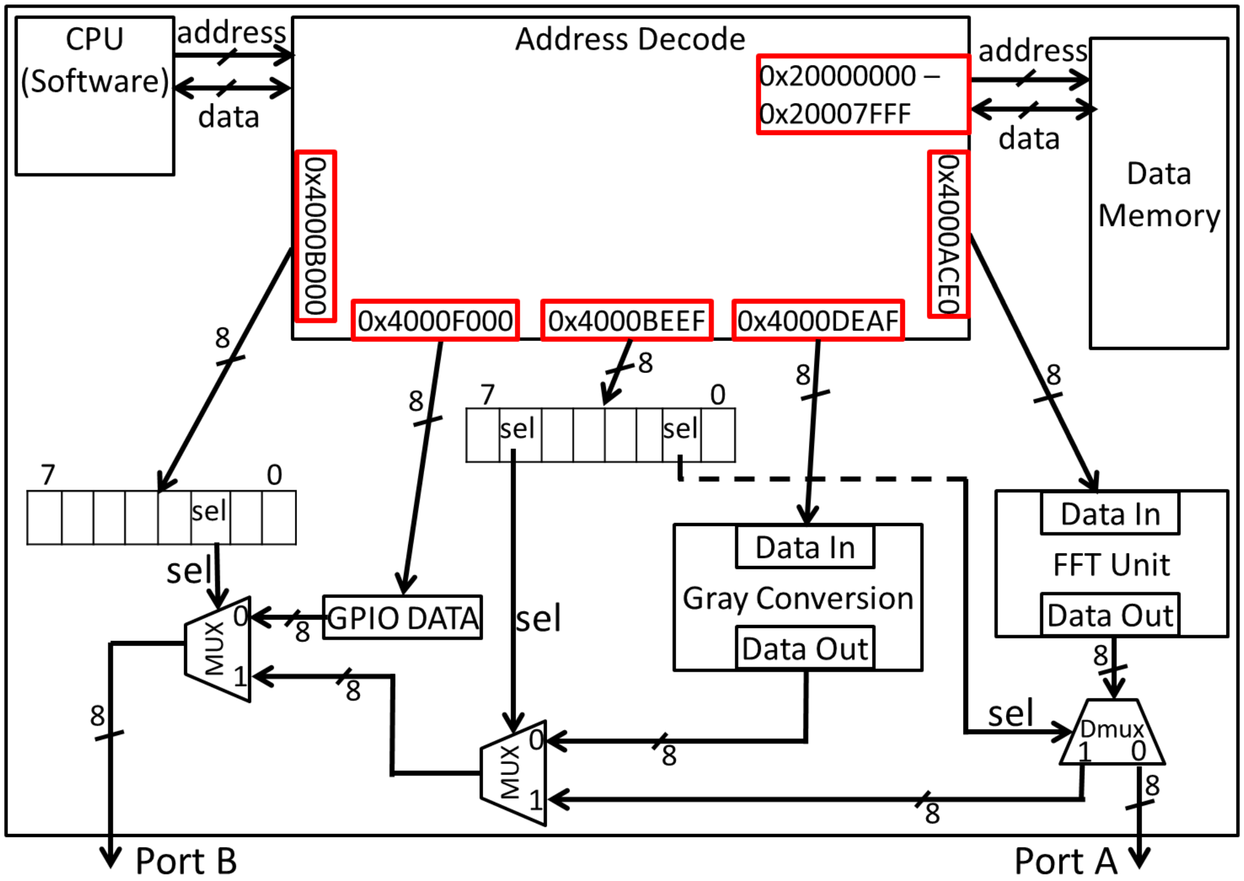
Problem Information:

1) `Music[]`, an array that contains 512 8-bit sound samples

2) A 512-point FFT takes as input 512 time-domain 8-bit samples one-by-one, and outputs one-by-one 512 8-bit values in the frequency domain. The box labeled FFT Unit in the figure performs the FFT, where each sample is given to **Data In,** and the transformed values are output one-by-one to **Data Out**.

3) The box labeled Gray Conversion takes as input 8-bit color encoded pixels and outputs Gray encoded pixels.

Problem Figure: Internal architecture for which your C code will be run.



Name: **Lab Section:**

// Convert 512 music samples to the frequency domain, and output the
// converted samples out of Port B
// Note: Add any variables you wish

```
main()
{

// Variable Declarations
char Music[512]; // Holds music samples

int i;  // loop variable

// Location of register that selects if GPIO data or
// an alternative type of functionality is sent to Port B
char *SEL_GPIO_ALT = 0x4000B000;

// Location of register that selects if Gray Conversion or FFT
// Unit is sent to Port B, and if FFT output is sent to Port A
// or Port B
char *SEL_GRAY_FFT_PORTA_PORTB = 0x4000BEEF;

// Location of register for sending data to the FFT Unit
char *FFT_DATA_IN = 0x4000ACE0;

  Initialize_Music(Music); // Assume this function fills Music[]
                           // with 512 samples for you.

  // Initialize the Hardware

  // Send data from alternative function to Port B
  *SEL_GPIO_ALT =  0b00000100;  // 0x04

  // Send FFT Unit data to Port B instead of Gray Conversion
  // and Send FFT Unit data to Port B instead of Port A
  *SEL_GRAY_FFT_PORTA_PORTB = 0b01000010; // 0x42


  // Send Music Samples to the FFT Unit
  for(i=0; i<512; i++)
  {
    *FFT_DATA_IN = Music[i];
  }

}
```

Name: **Lab Section:**

Question 3: GPIO Architecture (A CPRE 281, Digital Logic, view) (10 pts)

All the devices (also called on-chip peripherals) on our Microcontroller must have their signals routed through the GPIO module if they want to interact with the outside world, via the package pins (i.e. chip's external wires). The hardware structure of the GPIO module that allows a physical path from a device to a package pin (wire) can be illustrated using basic Digital Logic (i.e. CPRE 281) components.

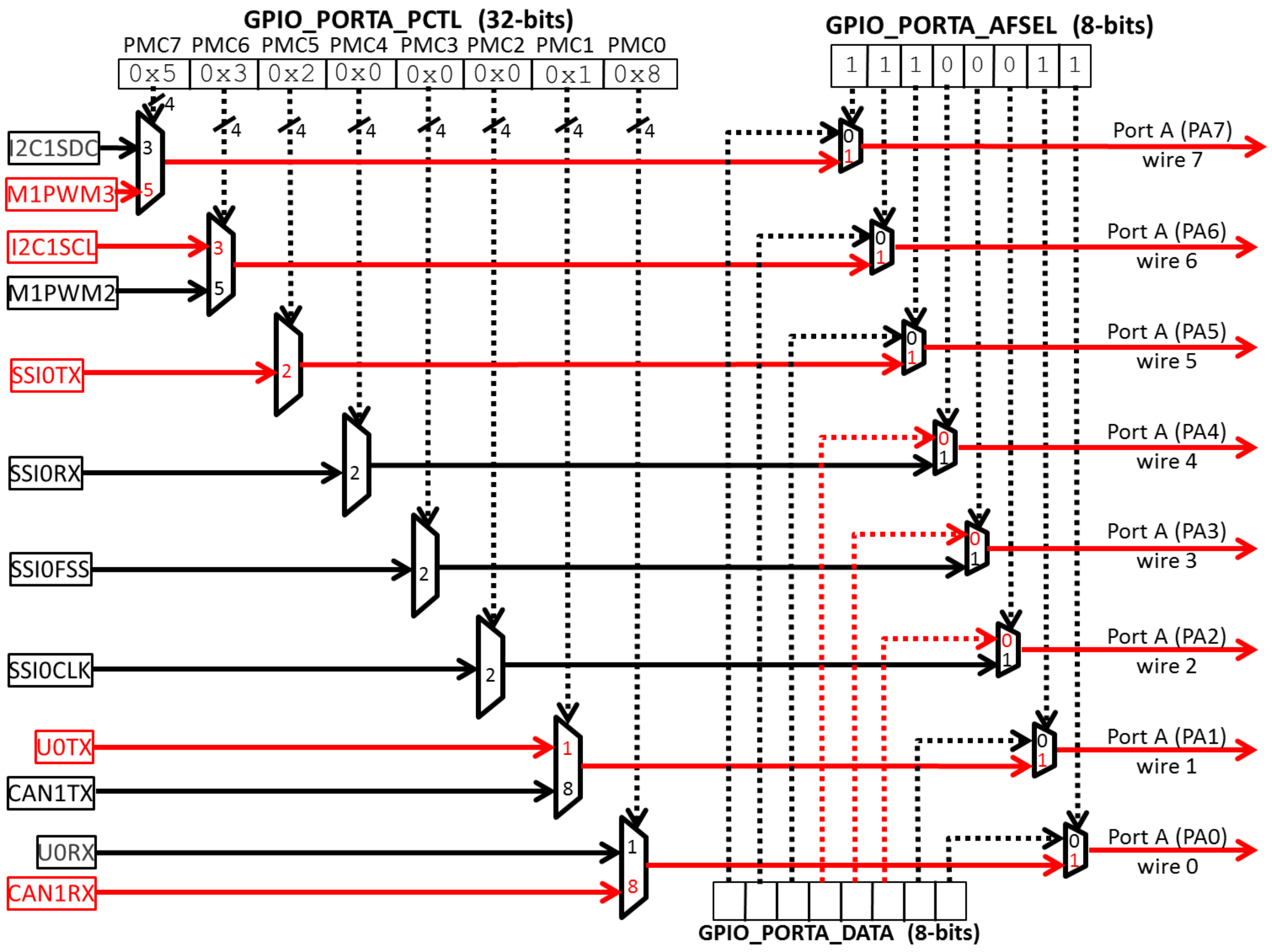
For Port A, using only registers and multiplexers (MUXs), provide a detailed diagram that shows:

- The connection between multiplexers associated with Port A's Alternative Function Select (AFSEL), Port Control (PCTL) registers, and Port A GPIO_DATA.
- For each Device associate with **Port A**, show how they connect to the multiplexers of your diagram. **For simplicity**, assume all devices are used for output (i.e. no demultiplexers needed)
- For every input of the multiplexers in your diagram, provide the value that the select line must have to allow a given input to be selected.
- Show in detail how the AFSEL and PCTRL registers connect to the MUXs in your design.
- **Note:** Port A has 8 package pins (wires), thus your diagram needs to show i) all 8 of these wires, and ii) for each of the 8 wires (package pins) how all the devices associated has data routed from the device to the package pin (wire).
- Fill in specific values of your AFSEL, and PCTRL registers that will configure your diagram so that.
    - Port A wire 0: Connects to CAN1RX
    - Port A wire 1: Connects to U0TX
    - Port A wire 2 – 4: Connected as GPIO_DATA
    - Port A wire 5: Connected as SSI0TX
    - Port A wire 6: Connected as I2C1SCL
    - Port A wire 7: Connected as M1PWM3

See Next Page for Figure

Name: **Lab Section:**

- Fill in specific values of your AFSEL, and PCTRL registers that will configure your diagram so that.
    - Port A wire 0: Connects to CAN1RX
    - Port A wire 1: Connects to U0TX
    - Port A wire 2 – 4: Connected as GPIO_DATA
    - Port A wire 5: Connected as SSI0TX
    - Port A wire 6: Connected as I2C1SCL
    - Port A wire 7: Connected as M1PWM3



Name: **Lab Section:**

Question 4: Pointers (10 pts)

Complete the table below (i.e. fill in the memory map) to show the state of memory after the following C fragment has been executed. Assume the **TM4C123** platform, and all the variables are in the stack (and thus the storage is allocated from high address to low address). Remember the **TM4C123** is a little endian architecture (i.e. least significant byte of a variable is stored in the lowest address)

Be very careful with this question. **(-.5pts for each incorrect value)**

```
typedef struct coord
{
  char x;
  char y;
} coord_t;

coord_t  *coord_ptr;
int *num_ptr;
int **p_ptr = &num_ptr;
char a = 0x07;
coord_t my_coord[2];
int num_array[2]={1,4};

int main()
{
  coord_ptr = my_coord;

  num_ptr = num_array;

  my_coord[1].x = 0x33;

  coord_ptr++;

  coord_ptr->y = 0x44;

  num_ptr = num_ptr + 2;

  *num_ptr = 0x5040;

  p_ptr++;

  *p_ptr = 0xFEC0;
}
```

Address	Variable Name	Value
0xFFFF_FFFF	coord_ptr	~~0xFF~~ 00
0xFFFF_FFFE		~~0xFF~~ 00
0xFFFF_FFFD		~~0xFF~~ FE
0xFFFF_FFFC		~~0xEF~~ ~~F1~~ C0
0xFFFF_FFFB	num_ptr	0xFF
0xFFFF_FFFA		0xFF
0xFFFF_FFF9		0xFF
0xFFFF_FFF8		~~0xE7~~ EF
0xFFFF_FFF7	p_ptr	0xFF
0xFFFF_FFF6		0xFF
0xFFFF_FFF5		0xFF
0xFFFF_FFF4		~~0xF8~~ FC
0xFFFF_FFF3	a	0x07
0xFFFF_FFF2	my_coord[1].y	~~0x44~~ 00
0xFFFF_FFF1	my_coord[1].x	~~0x33~~ 00
0xFFFF_FFF0	my_coord[0].y	0x50
0xFFFF_FFEF	my_coord[0].x	0x40
0xFFFF_FFEE	num_array[1]	0x00
0xFFFF_FFED		0x00
0xFFFF_FFEC		0x00
0xFFFF_FFEB		0x04
0xFFFF_FFEA	num_array[0]	0x00
0xFFFF_FFE9		0x00
0xFFFF_FFE8		0x00
0xFFFF_FFE7		0x01

Name: **Lab Section:**

<u>Collaboration Documentation</u>

List the people (First and Last name) you collaborated with: ___________________________.

For each collaborator, describe the manner in which you collaborated:

1)

2)