CPE403 – Advanced Embedded Systems

Design Assignment

DO NOT REMOVE THIS PAGE DURING SUBMISSION:

Name: Sai Balaji Jai Kumar

Email: jaikumar@unlv.nevada.edu

Github Repository link (root):
https://github.com/saibalaji1997/githubfiles/tree/main/TIVAC/Assignment%202

Youtube Playlist link (root): https://www.youtube.com/shorts/wxMhIFKZKtU

Follow the submission guideline to be awarded points for this Assignment.

Submit the following for all Assignments:

1. In the document, for each task submit the modified or included code (from the base code) with highlights and justifications of the modifications. Also include the comments. If no base code is provided, submit the base code for the first task only.
2. Create a private Github repository with a random name (no CPE/403, Lastname, Firstname). Place all labs under the root folder TIVAC, sub-folder named Assignment1, with one document and one video link file for each lab, place modified c files named as asng_taskxx.c.
3. If multiple c files or other libraries are used, create a folder asng1_t01 and place these files inside the folder.
4. The folder should have a) Word document (see template), b) source code file(s) with startup_ccs.c and other include files, c) text file with youtube video links (see template).
5. Submit the doc file in canvas before the due date. The root folder of the github assignment directory should have the documentation and the text file with youtube video links.
6. Organize your youtube videos as playlist under the name “cpe403”. The playlist should have the video sequence arranged as submission or due dates.
7. Only submit pdf documents. Do not forget to upload this document in the github repository and in the canvas submission portal.

1. Code for Tasks. for each task submit the modified or included code (from the base code) with highlights and justifications of the modifications. Also include the comments. If no base code is provided, submit the base code for the first task only. Use separate page for each task.

Code for IMU Sensor to determine Euler angles:

```
#include "TM4C123GH6PM.h"
#include <stdlib.h>
#include <string.h>
#include <stdio.h>
#define XG_OFFS_TC        0x00
#define YG_OFFS_TC        0x01
#define ZG_OFFS_TC        0x02
#define X_FINE_GAIN       0x03
#define Y_FINE_GAIN       0x04
#define Z_FINE_GAIN       0x05
#define XA_OFFS_H         0x06
#define XA_OFFS_L_TC      0x07
#define YA_OFFS_H         0x08
#define YA_OFFS_L_TC      0x09
#define ZA_OFFS_H         0x0A
#define ZA_OFFS_L_TC      0x0B
#define XG_OFFS_USRH      0x13
#define XG_OFFS_USRL      0x14
#define YG_OFFS_USRH      0x15
#define YG_OFFS_USRL      0x16
#define ZG_OFFS_USRH      0x17
#define ZG_OFFS_USRL      0x18
#define SMPLRT_DIV        0x19
#define CONFIG            0x1A
#define GYRO_CONFIG       0x1B
#define ACCEL_CONFIG      0x1C
```

```
#define FF_THR            0x1D
#define FF_DUR            0x1E
#define MOT_THR           0x1F
#define MOT_DUR           0x20
#define ZRMOT_THR         0x21
#define ZRMOT_DUR         0x22
#define FIFO_EN           0x23
#define I2C_MST_CTRL      0x24
#define I2C_SLV0_ADDR     0x25
#define I2C_SLV0_REG      0x26
#define I2C_SLV0_CTRL     0x27
#define I2C_SLV1_ADDR     0x28
#define I2C_SLV1_REG      0x29
#define I2C_SLV1_CTRL     0x2A
#define I2C_SLV2_ADDR     0x2B
#define I2C_SLV2_REG      0x2C
#define I2C_SLV2_CTRL     0x2D
#define I2C_SLV3_ADDR     0x2E
#define I2C_SLV3_REG      0x2F
#define I2C_SLV3_CTRL     0x30
#define I2C_SLV4_ADDR     0x31
#define I2C_SLV4_REG      0x32
#define I2C_SLV4_DO       0x33
#define I2C_SLV4_CTRL     0x34
#define I2C_SLV4_DI       0x35
#define I2C_MST_STATUS    0x36
#define INT_PIN_CFG       0x37
#define INT_ENABLE        0x38
#define DMP_INT_STATUS    0x39
#define INT_STATUS        0x3A
#define ACCEL_XOUT_H      0x3B
```

```
#define ACCEL_XOUT_L       0x3C
#define ACCEL_YOUT_H       0x3D
#define ACCEL_YOUT_L       0x3E
#define ACCEL_ZOUT_H       0x3F
#define ACCEL_ZOUT_L       0x40
#define TEMP_OUT_H         0x41
#define TEMP_OUT_L         0x42
#define GYRO_XOUT_H        0x43
#define GYRO_XOUT_L        0x44
#define GYRO_YOUT_H        0x45
#define GYRO_YOUT_L        0x46
#define GYRO_ZOUT_H        0x47
#define GYRO_ZOUT_L        0x48
#define EXT_SENS_DATA_00   0x49
#define EXT_SENS_DATA_01   0x4A
#define EXT_SENS_DATA_02   0x4B
#define EXT_SENS_DATA_03   0x4C
#define EXT_SENS_DATA_04   0x4D
#define EXT_SENS_DATA_05   0x4E
#define EXT_SENS_DATA_06   0x4F
#define EXT_SENS_DATA_07   0x50
#define EXT_SENS_DATA_08   0x51
#define EXT_SENS_DATA_09   0x52
#define EXT_SENS_DATA_10   0x53
#define EXT_SENS_DATA_11   0x54
#define EXT_SENS_DATA_12   0x55
#define EXT_SENS_DATA_13   0x56
#define EXT_SENS_DATA_14   0x57
#define EXT_SENS_DATA_15   0x58
#define EXT_SENS_DATA_16   0x59
#define EXT_SENS_DATA_17   0x5A
```

```
#define EXT_SENS_DATA_18    0x5B
#define EXT_SENS_DATA_19    0x5C
#define EXT_SENS_DATA_20    0x5D
#define EXT_SENS_DATA_21    0x5E
#define EXT_SENS_DATA_22    0x5F
#define EXT_SENS_DATA_23    0x60
#define MOT_DETECT_STATUS   0x61
#define I2C_SLV0_DO         0x63
#define I2C_SLV1_DO         0x64
#define I2C_SLV2_DO         0x65
#define I2C_SLV3_DO         0x66
#define I2C_MST_DELAY_CTRL  0x67
#define SIGNAL_PATH_RESET   0x68
#define MOT_DETECT_CTRL     0x69
#define USER_CTRL           0x6A
#define PWR_MGMT_1          0x6B
#define PWR_MGMT_2          0x6C
#define BANK_SEL            0x6D
#define MEM_START_ADDR      0x6E
#define MEM_R_W             0x6F
#define DMP_CFG_1           0x70
#define DMP_CFG_2           0x71
#define FIFO_COUNTH         0x72
#define FIFO_COUNTL         0x73
#define FIFO_R_W            0x74
#define WHO_AM_I            0x75
void I2C3_Init(void);
char I2C3_Wr(int slaveAddr, char memAddr, char data);
char I2C3_Rd(int slaveAddr, char memAddr, int byteCount, char* data);
void Delay(unsigned long counter);
void uart5_init(void);
```

```
void UART5_Transmitter(unsigned char data);
void printstring(char *str);
void MPU6050_Init(void);

char msg[20];
int main(void)
{
  int  accX, accY, accZ, GyroX, GyroY, GyroZ, Temper;
    float AX, AY, AZ, t, GX, GY, GZ;
    char sensordata[14];
    I2C3_Init();
    Delay(1000);
    MPU6050_Init();
    Delay(1000);
  uart5_init();
    while(1)
    {
 I2C3_Rd(0x68,ACCEL_XOUT_H, 14, sensordata);
 accX = (int) ( (sensordata[0] << 8 ) |sensordata[1] );
 accY = (int) ( (sensordata[2] << 8 ) |sensordata[3] );
 accZ = (int) ( (sensordata[4] << 8 ) |sensordata[5] );
 Temper = (int) ( (sensordata[6] << 8 ) |sensordata[7] );
 GyroX = (int) ( (sensordata[8] << 8 ) |sensordata[9] );
 GyroY = (int) ( (sensordata[10] << 8 ) |sensordata[11] );
 GyroZ = (int) ( (sensordata[12] << 8 ) |sensordata[13] );


   // Convert The Readings
  AX = (float)accX/16384.0;
  AY = (float)accY/16384.0;
  AZ = (float)accZ/16384.0;
```

```
 GX = (float)GyroX/131.0;
 GY = (float)GyroX/131.0;
 GZ = (float)GyroX/131.0;
 t = ((float)Temper/340.00)+36.53;
   sprintf(msg,"Gx = %.2f \t",GX);
   printstring(msg);
      sprintf(msg,"Gy = %.2f \t",GY);
   printstring(msg);
      sprintf(msg,"Gz  = %.2f \t",GZ);
   printstring(msg);
      sprintf(msg,"Ax  = %.2f \t",AX);
   printstring(msg);
      sprintf(msg,"Ay  = %.2f \t",AY);
   printstring(msg);
      sprintf(msg,"Ax  = %.2f \r\n",AZ);
   printstring(msg);
     // sprintf(msg,"Temp  = %.2f \r\n",t);
  // printstring(msg);
   Delay(1000);
  }
}

void MPU6050_Init(void)
{
 I2C3_Wr(0x68,SMPLRT_DIV, 0x07);
 I2C3_Wr(0x68,PWR_MGMT_1,  0x01);
 I2C3_Wr(0x68,CONFIG, 0x00);
 I2C3_Wr(0x68,ACCEL_CONFIG,0x00);
 I2C3_Wr(0x68,GYRO_CONFIG,0x18);
 I2C3_Wr(0x68,INT_ENABLE, 0x01);
```

```
}

void uart5_init(void)
{

    SYSCTL->RCGCUART |= 0x20;  /* enable clock to UART5 */
   SYSCTL->RCGCGPIO |= 0x10;  /* enable clock to PORTE for PE4/Rx and RE5/Tx */
   Delay(1);
   /* UART0 initialization */
   UART5->CTL = 0;        /* UART5 module disbable */
   UART5->IBRD = 104;     /* for 9600 baud rate, integer = 104 */
   UART5->FBRD = 11;      /* for 9600 baud rate, fractional = 11*/
   UART5->CC = 0;         /*select system clock*/
   UART5->LCRH = 0x60;    /* data lenght 8-bit, not parity bit, no FIFO */
   UART5->CTL = 0x301;    /* Enable UART5 module, Rx and Tx */

   /* UART5 TX5 and RX5 use PE4 and PE5. Configure them digital and enable alternate function
*/
   GPIOE->DEN = 0x30;     /* set PE4 and PE5 as digital */
   GPIOE->AFSEL = 0x30;   /* Use PE4,PE5 alternate function */
   GPIOE->AMSEL = 0;    /* Turn off analg function*/
   GPIOE->PCTL = 0x00110000;     /* configure PE4 and PE5 for UART */

}
void I2C3_Init(void)
{
SYSCTL->RCGCGPIO  |= 0x00000008 ; // Enable the clock for port D
SYSCTL->RCGCI2C   |= 0x00000008 ; // Enable the clock for I2C 3
GPIOD->DEN |= 0x03; // Assert DEN for port D
// Configure Port D pins 0 and 1 as I2C 3
GPIOD->AFSEL |= 0x00000003 ;
```

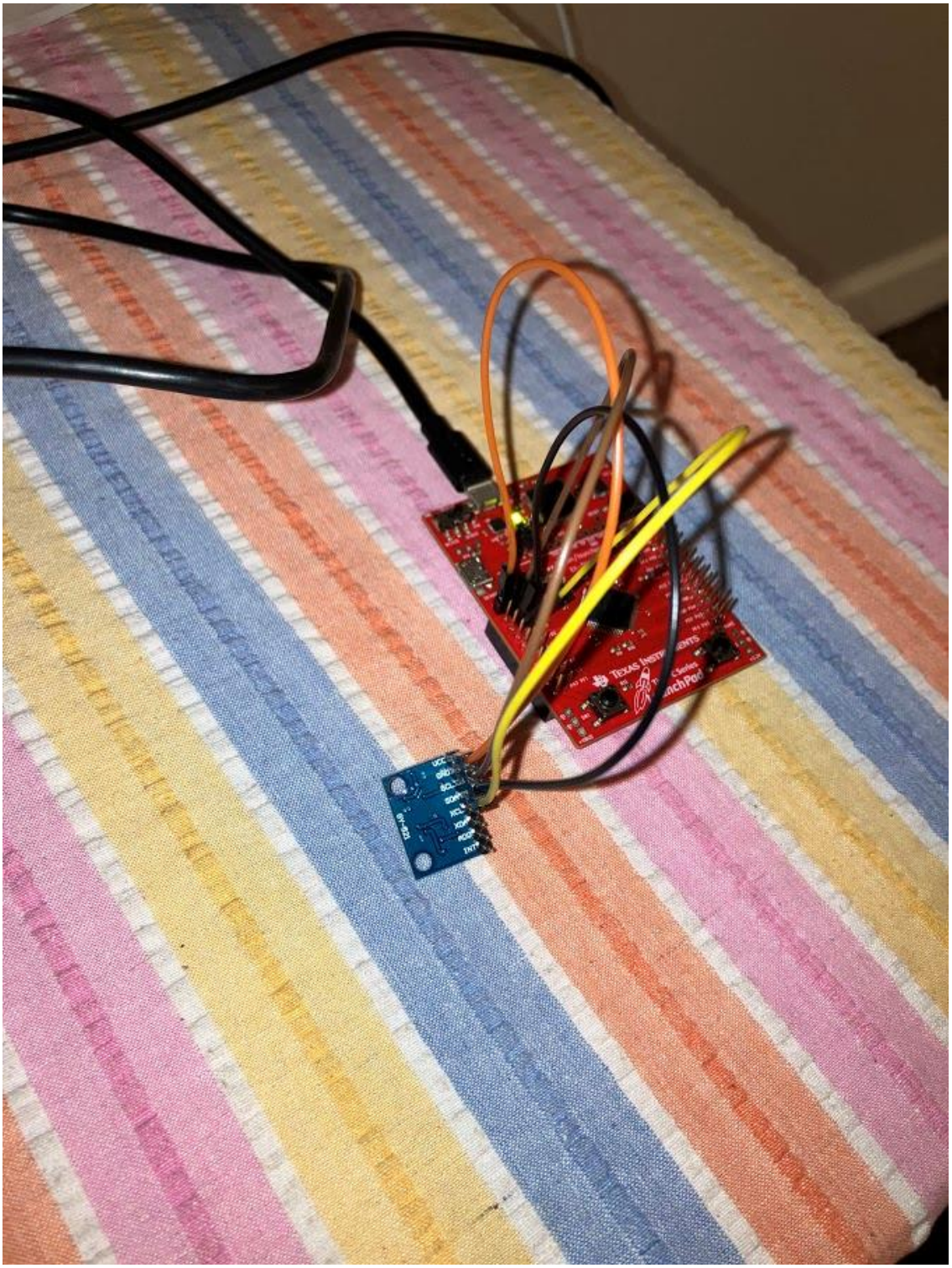
```
GPIOD->PCTL |= 0x00000033 ;
GPIOD->ODR |= 0x00000002 ; // SDA (PD1 ) pin as open darin
I2C3->MCR  = 0x0010 ; // Enable I2C 3 master function
/* Configure I2C 3 clock frequency
(1 + TIME_PERIOD ) = SYS_CLK /(2*
( SCL_LP + SCL_HP ) * I2C_CLK_Freq )
TIME_PERIOD = 16 ,000 ,000/(2(6+4) *100000) - 1 = 7 */
I2C3->MTPR  = 0x07 ;
}

/* Wait until I2C master is not busy and return error code */
/* If there is no error, return 0 */
static int I2C_wait_till_done(void)
{
   while(I2C3->MCS & 1);   /* wait until I2C master is not busy */
   return I2C3->MCS & 0xE; /* return I2C error code */
}

/* Write one byte only */
/* byte write: S-(saddr+w)-ACK-maddr-ACK-data-ACK-P */
char I2C3_Wr(int slaveAddr, char memAddr, char data)
{

   char error;

   /* send slave address and starting address */
   I2C3->MSA = slaveAddr << 1;
   I2C3->MDR = memAddr;
   I2C3->MCS = 3;                  /* S-(saddr+w)-ACK-maddr-ACK */

   error = I2C_wait_till_done();       /* wait until write is complete */
```

```
  if (error) return error;

  /* send data */
  I2C3->MDR = data;
  I2C3->MCS = 5;                  /* -data-ACK-P */
  error = I2C_wait_till_done();     /* wait until write is complete */
  while(I2C3->MCS & 0x40);          /* wait until bus is not busy */
  error = I2C3->MCS & 0xE;
  if (error) return error;

  return 0;     /* no error */
}
char I2C3_Rd(int slaveAddr, char memAddr, int byteCount, char* data)
{
   char error;

  if (byteCount <= 0)
     return -1;       /* no read was performed */

  /* send slave address and starting address */
  I2C3->MSA = slaveAddr << 1;
  I2C3->MDR = memAddr;
  I2C3->MCS = 3;     /* S-(saddr+w)-ACK-maddr-ACK */
  error = I2C_wait_till_done();
  if (error)
     return error;

  /* to change bus from write to read, send restart with slave addr */
  I2C3->MSA = (slaveAddr << 1) + 1;   /* restart: -R-(saddr+r)-ACK */

  if (byteCount == 1)           /* if last byte, don't ack */
```

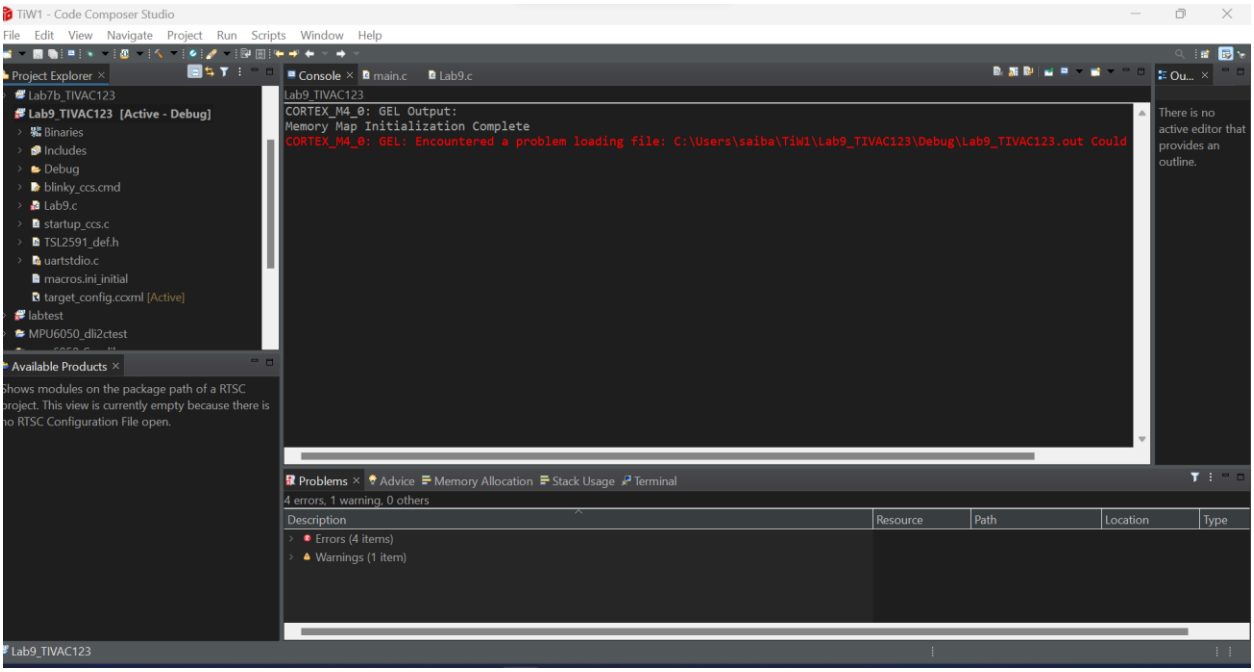
```
    I2C3->MCS = 7;            /* -data-NACK-P */
  else                        /* else ack */
    I2C3->MCS = 0xB;          /* -data-ACK- */
  error = I2C_wait_till_done();
  if (error) return error;

  *data++ = I2C3->MDR;        /* store the data received */

  if (--byteCount == 0)       /* if single byte read, done */
  {
    while(I2C3->MCS & 0x40);  /* wait until bus is not busy */
    return 0;     /* no error */
  }

  /* read the rest of the bytes */
  while (byteCount > 1)
  {
    I2C3->MCS = 9;            /* -data-ACK- */
    error = I2C_wait_till_done();
    if (error) return error;
    byteCount--;
    *data++ = I2C3->MDR;      /* store data received */
  }

  I2C3->MCS = 5;              /* -data-NACK-P */
  error = I2C_wait_till_done();
  *data = I2C3->MDR;          /* store data received */
  while(I2C3->MCS & 0x40);    /* wait until bus is not busy */

  return 0;     /* no error */
}
```

```
void UART5_Transmitter(unsigned char data)
{
   while((UART5->FR & (1<<5)) != 0); /* wait until Tx buffer not full */
   UART5->DR = data;                 /* before giving it another byte */
}

void printstring(char *str)
{
  while(*str)
   {
      UART5_Transmitter(*(str++));
   }
}
void Delay(unsigned long counter)
{
   unsigned long i = 0;

   for(i=0; i< counter*10000; i++);
}
```

2. Block diagram and/or Schematics showing the components, pins used, and interface.



3. Screenshots of the IDE, physical setup, debugging process - Provide screenshot of successful compilation, screenshots of registers, variables, graphs, etc.



4. Declaration

I understand the Student Academic Misconduct Policy -
http://studentconduct.unlv.edu/misconduct/policy.html

"This assignment submission is my own, original work".

Name of the Student

Sai Balaji Jai Kumar