Lab Assignment 6

Jose Monroy Villalobos

San José State University
Department of Computer Engineering
CMPE 146-01, Real-Time Embedded System Co-Design, Fall 2024

Exercise 1.1 Send text string:

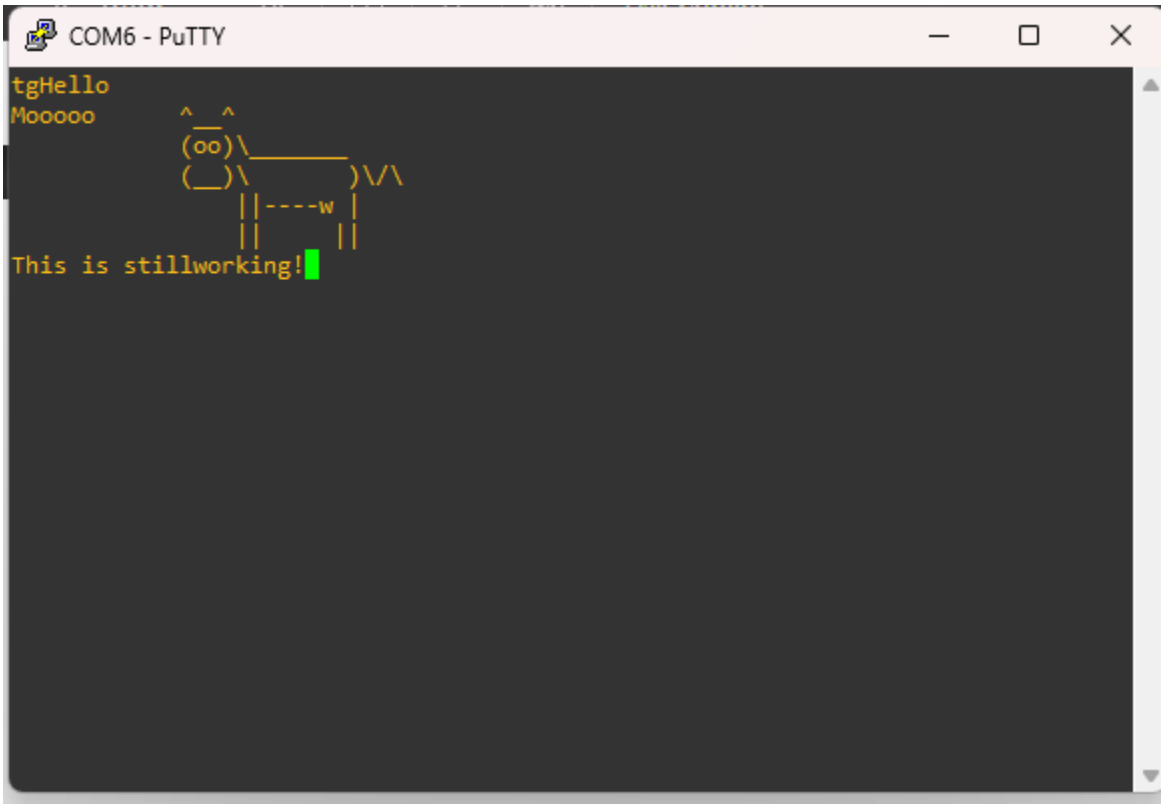
Code:

```
//Found values in documents
#define UCA0TXBUF_OFFSET   0x0E
#define UCA0STATW_OFFSET   0x0A
#define UART_BUSY_FLAG     0x01


void send_message(char* msg)
{

    volatile uint16_t* tx_buffer = (uint16_t*)(EUSCI_A0_BASE + UCA0TXBUF_OFFSET);
    volatile uint16_t* status = (uint16_t*)(EUSCI_A0_BASE + UCA0STATW_OFFSET);

    while (*msg) {
        while (*status & UART_BUSY_FLAG);  //Wait until the UART is ready
        *tx_buffer = *msg++;               //Send character
    }


}
```

Brief Explanation:

The way the function works is that the pointers are initialized for the Transmit Buffer Register and Status Register. The outer while loop goes through the whole message and the inner loop waits for UART to be ready. Once it is ready the character is written to and it advances. There was an issue when copying and pasting the message in the lab document, I had to align everything, so the hex numbers were properly together. If it happened to me, I assume when you run this code it could perhaps happen to you as well. By aligned I mean that some hex numbers were split because it went to a new line in the middle of a hex number.

Terminal Window Screenshot:



Exercise 1.2 Emulate UART:

Code:

```
#define  SYSTEM_CLOCK    12000000        //
#define DELAY_100MS (SYSTEM_CLOCK / 10)  //Cycles for 100 ms
#define BAUD_RATE 9600
//Within main the pins are configured as follows:
```

```
    MAP_GPIO_setAsOutputPin(GPIO_PORT_P1, GPIO_PIN3);   //Port 1.3 is tx and is set
as ouput
    MAP_GPIO_setAsInputPin(GPIO_PORT_P1, GPIO_PIN2);    //Port 1.2 is rx and is set
as input


    MAP_GPIO_setOutputHighOnPin(GPIO_PORT_P1, GPIO_PIN3); //Seting tx to its idle
sttate
    __delay_cycles(DELAY_100MS);                          //100ms delay
//This the function implementation outside of main
void send_message(char* msg)
{
    const int bit_time = SYSTEM_CLOCK / BAUD_RATE; //Calculate the delay for one bit
time
    int i =0 ;
    while (*msg)                                        //Traverse message
    {

        //Start bit (Logic Low)
        MAP_GPIO_setOutputLowOnPin(GPIO_PORT_P1, GPIO_PIN3);
        __delay_cycles(bit_time);

        //Send 8 data bits (LSB first)
        for ( i = 0; i < 8; i++)
        {
            if (*msg & (1 << i)) //Check if the bit is 1
            {
                MAP_GPIO_setOutputHighOnPin(GPIO_PORT_P1, GPIO_PIN3);
            }
            else //Otherwise, it's 0
            {
                MAP_GPIO_setOutputLowOnPin(GPIO_PORT_P1, GPIO_PIN3);
            }
            __delay_cycles(bit_time);
        }

        //Stop bit (Logic High)
        MAP_GPIO_setOutputHighOnPin(GPIO_PORT_P1, GPIO_PIN3);
        __delay_cycles(bit_time);

        //Move to the next character
        msg++;
    }
}
```

Explanation:

Function works similarly to what I previously implemented in Exercise 1.1, instead we use the tx pin to send the message. It calculates the bit time and sets the pin to low because its idle is high which counts as the start bit. And delays the bittime cycles Then it sends the 8 bits data while delaying the bit time cycles. Once done, it return to high for the pin and move to the next character and loop back. There was an issue when trying to run at 9600 baud and the way I got it to work was to change the frequency the board is running from 3 MHz to 12 MHz.

Terminal Window Screenshot:

Exercise 1.3 Parity bit:

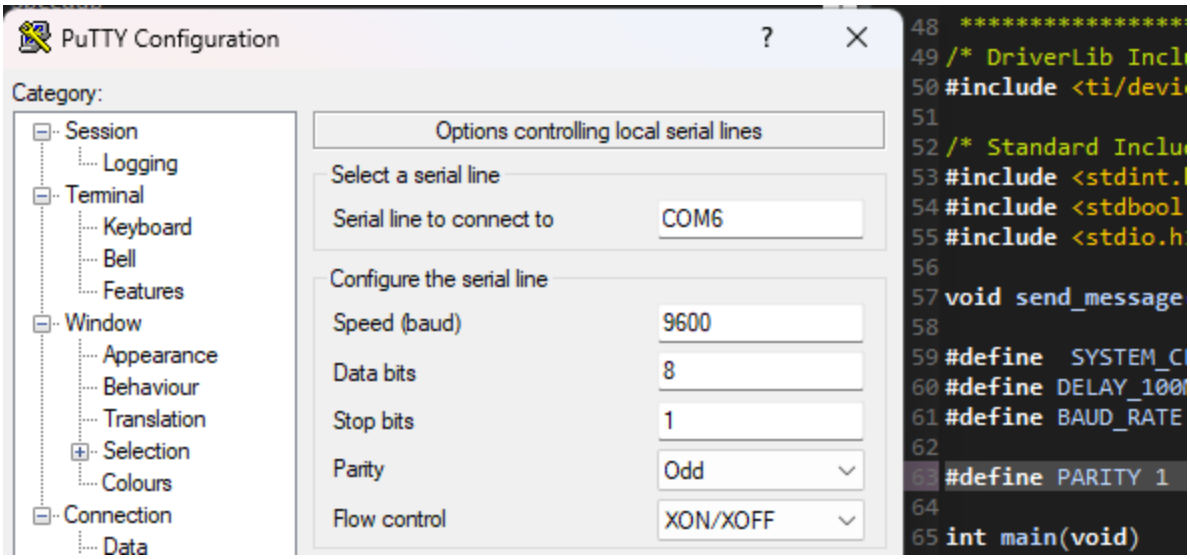
Code that generates parity bit:

```
int parity = 1; //Initialize parity bit

        if (PARITY > 0) //Check if parity is enabled
        {
            //Calculate the parity bit
            for (i = 0; i < 8; i++)
            {
                if (*msg & (1 << i)) //Check if the bit is 1
                {
                    parity ^= 1; //Toggle parity bit
                }
            }

            if (PARITY == 2) //Even parity mode
            {
                parity = !parity;
            }
```

Explanation:

The parity bit is initialized and then if the PARITY macro is greater than 0 that means that odd or even has been selected, it goes through the data and toggles when it encounters a 1 and then once it is done it checks the macro and flips the bit if the parity is set to even. Then before the stop bit, I have a conditional that handles the outputting of the bit based on the value of the macro.

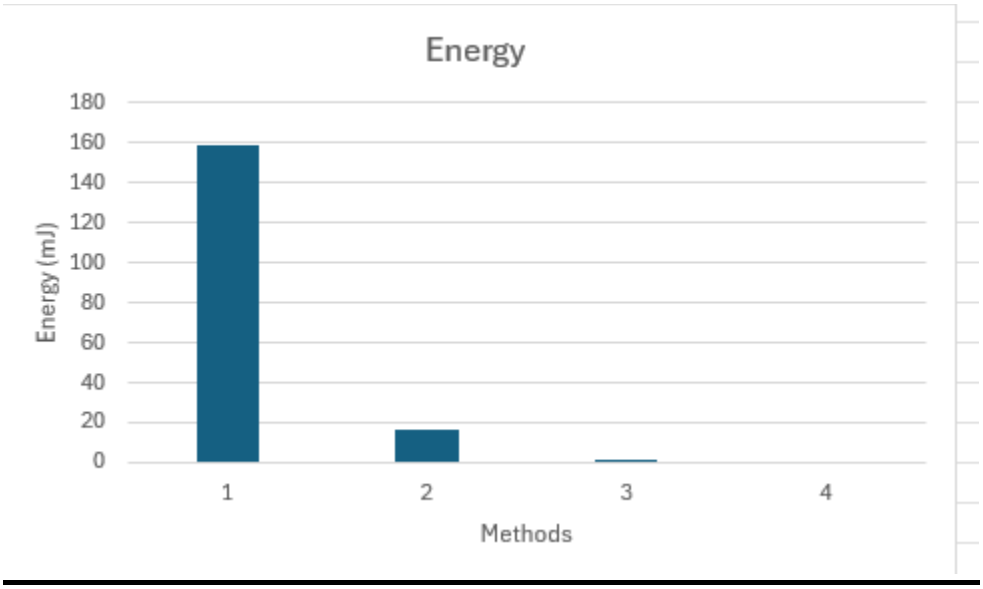
Setting for Odd parity Screenshot:

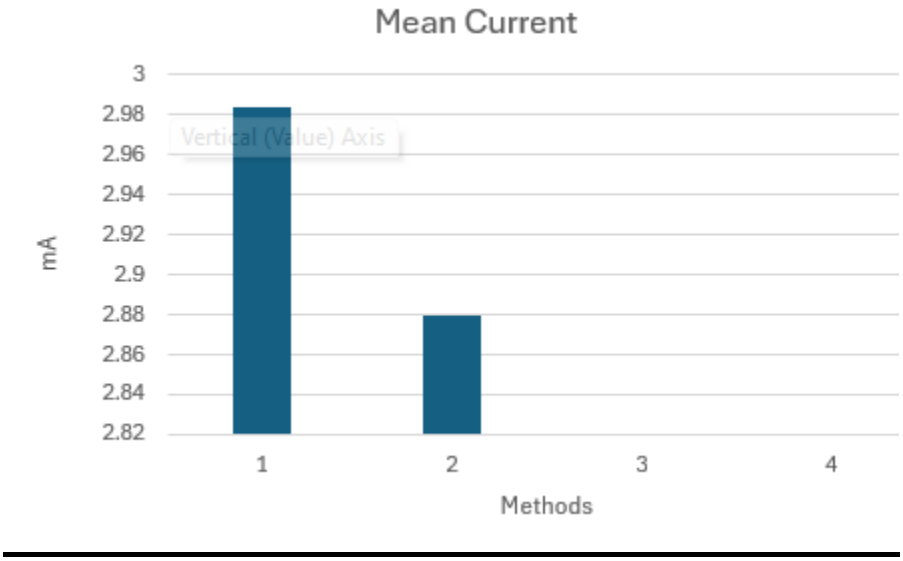


Terminal Window Screenshot:

Exercise 2.1 Energy measurements:

Method	M1 mJ	M2 mJ	M3 mJ	M4 mJ	Averages
USE_SW_METHOD	163.286	156.912	157.8	157.048	158.7615
USE_HW_METHOD	16.586	16.263	15.971	15.991	16.20275
USE_DMA_METHOD	0.796				0.796
USE_DMA_LPM_METHOD					#DIV/0!
Method	M1 mA	M2 mA	M3 mA	M4 mA	#DIV/0!
USE_SW_METHOD	3.0735	2.949	2.9657	2.9476	2.98395
USE_HW_METHOD	2.9205	2.9015	2.8901	2.805	2.879275
USE_DMA_METHOD	2.9235				
USE_DMA_LPM_METHOD					



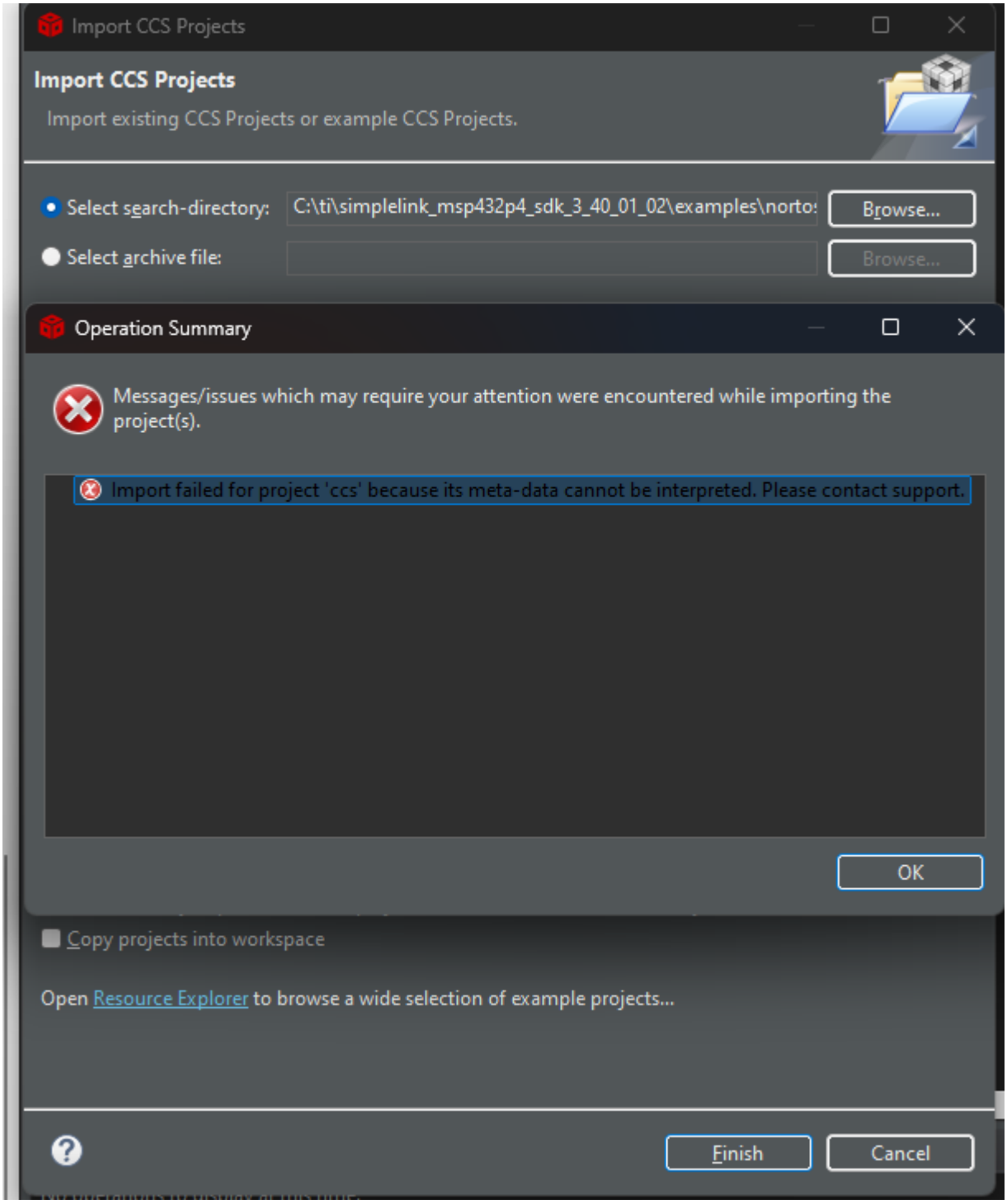


I tried extensively to measure the other two DMA methods but failed after one successful attempt. I would click the green arrow button and it appeared that they ran two fast to measure so then if I clicked the green arrow it ran for a long time before I stopped because I deemed it to be not right. I can demo this in person and am very confused if this was not the intended behavior. I noticed a trend deownward in the methods in time and energy consumption a SW the slowest and the DMA presumably being too fast to measure I assume. My assumption was that the methods were setup from least efficient to most efficient. I contacted other classmates that had similar results so I think something might be wrong that I do not understand.

Exercise 2.2 LPM4.5:

Had persistent issue with energy trace tool. I think the root of the issue might have perhaps been with importing the empty project. I attempted importing the empty project and had weird error

messages and I think this might be messing with my use of energy trace.



I plan to go to lab in person to get your insight on this and show what was going on. I also heard a report from another student that his board got bricked while doing this assignment so I ceased attempting after this.

Appendix:

Exercise 1.1 Send text string:

```
/* DriverLib Includes */
#include <ti/devices/msp432p4xx/driverlib/driverlib.h>

/* Standard Includes */
#include <stdint.h>
#include <stdbool.h>
#include <stdio.h>

//![Simple UART Config]
/* UART Configuration Parameter. These are the configuration parameters to
 * make the eUSCI A UART module to operate with a 9600 baud rate. These
 * values were calculated using the online calculator that TI provides
 * at:
 *http://software-
dl.ti.com/msp430/msp430_public_sw/mcu/msp430/MSP430BaudRateConverter/index.html
 */
const eUSCI_UART_ConfigV1 uartConfig =
{
        EUSCI_A_UART_CLOCKSOURCE_SMCLK,          // SMCLK Clock Source
        78,                                     // BRDIV = 78
        2,                                       // UCxBRF = 2
        0,                                       // UCxBRS = 0
        EUSCI_A_UART_NO_PARITY,                  // No Parity
```

```
        EUSCI_A_UART_LSB_FIRST,                  // LSB First
        EUSCI_A_UART_ONE_STOP_BIT,               // One stop bit
        EUSCI_A_UART_MODE,                       // UART mode
        EUSCI_A_UART_OVERSAMPLING_BAUDRATE_GENERATION,  // Oversampling
        EUSCI_A_UART_8_BIT_LEN                  // 8 bit data length
};
//![Simple UART Config]

void send_message(char* msg);

//Found values in documents
#define UCA0TXBUF_OFFSET   0x0E
#define UCA0STATW_OFFSET   0x0A
#define UART_BUSY_FLAG     0x01

int main(void)
{
    /* Halting WDT  */
    MAP_WDT_A_holdTimer();

    /* Selecting P1.2 and P1.3 in UART mode */
    MAP_GPIO_setAsPeripheralModuleFunctionInputPin(GPIO_PORT_P1,
            GPIO_PIN2 | GPIO_PIN3, GPIO_PRIMARY_MODULE_FUNCTION);

    /* Setting DCO to 12MHz */
    CS_setDCOCenteredFrequency(CS_DCO_FREQUENCY_12);

    //![Simple UART Example]
    /* Configuring UART Module */
    MAP_UART_initModule(EUSCI_A0_BASE, &uartConfig);

    /* Enable UART module */
    MAP_UART_enableModule(EUSCI_A0_BASE);

    /* Enabling interrupts */
    MAP_UART_enableInterrupt(EUSCI_A0_BASE, EUSCI_A_UART_RECEIVE_INTERRUPT);
    MAP_Interrupt_enableInterrupt(INT_EUSCIA0);
    MAP_Interrupt_enableSleepOnIsrExit();
    MAP_Interrupt_enableMaster();
    //![Simple UART Example]

    static char something[] =

{0x0d,0x0a,0x4d,0x6f,0x6f,0x6f,0x6f,0x6f,0x20,0x20,0x20,0x20,0x20,0x20,0x5e,0x5f,0x5f
,0x5e,0x0d,0x0a,0x20,0x20,0x20,0x20,0x20,0x20,0x20,0x20,0x20,0x20,0x20,0x20,0x28,0x6f
,0x6f,0x29,0x5c,0x5f,0x5f,0x5f,0x5f,0x5f,0x5f,0x5f,0x0d,0x0a,0x20,0x20,0x20,0x20,0x20
,0x20,0x20,0x20,0x20,0x20,0x20,0x20,0x28,0x5f,0x5f,0x29,0x5c,0x20,0x20,0x20,0x20,0x20
,0x20,0x20,0x29,0x5c,0x2f,0x5c,0x0d,0x0a,0x20,0x20,0x20,0x20,0x20,0x20,0x20,0x20,0x20
,0x20,0x20,0x20,0x20,0x20,0x20,0x20,0x7c,0x7c,0x2d,0x2d,0x2d,0x2d,0x77,0x20,0x7c,0x0d
,0x0a,0x20,0x20,0x20,0x20,0x20,0x20,0x20,0x20,0x20,0x20,0x20,0x20,0x20,0x20,0x20,0x20
,0x7c,0x7c,0x20,0x20,0x20,0x20,0x20,0x7c,0x7c,0x0d,0x0a,0x00};
    send_message(something);
    while(1)
    {
        MAP_PCM_gotoLPM0();
```

```
    }
}

/* EUSCI A0 UART ISR - Echoes data back to PC host */
void EUSCIA0_IRQHandler(void)
{
    uint32_t status = MAP_UART_getEnabledInterruptStatus(EUSCI_A0_BASE);

    if(status & EUSCI_A_UART_RECEIVE_INTERRUPT_FLAG)
    {
        MAP_UART_transmitData(EUSCI_A0_BASE, MAP_UART_receiveData(EUSCI_A0_BASE));
    }

}

void send_message(char* msg)
{

    volatile uint16_t* tx_buffer = (uint16_t*)(EUSCI_A0_BASE + UCA0TXBUF_OFFSET);
    volatile uint16_t* status = (uint16_t*)(EUSCI_A0_BASE + UCA0STATW_OFFSET);

    while (*msg) {
        while (*status & UART_BUSY_FLAG);  //Wait until the UART is ready
        *tx_buffer = *msg++;               //Send character
    }


}
```

Exercise 1.2 Emulate UART:

```
/* DriverLib Includes */
#include <ti/devices/msp432p4xx/driverlib/driverlib.h>

/* Standard Includes */
#include <stdint.h>
#include <stdbool.h>
#include <stdio.h>

void send_message(char* msg);

#define  SYSTEM_CLOCK    12000000        //Changed to 12 MHZ frequency for the 9600
baud to work properly
#define DELAY_100MS (SYSTEM_CLOCK / 10)  //Cycles for 100 ms
#define BAUD_RATE 9600
int main(void)
{
    /* Stop Watchdog  */
    MAP_WDT_A_holdTimer();
```

```
    MAP_GPIO_setAsOutputPin(GPIO_PORT_P1, GPIO_PIN3);   //Port 1.3 is tx and is set
as ouput
    MAP_GPIO_setAsInputPin(GPIO_PORT_P1, GPIO_PIN2);    //Port 1.2 is rx and is set
as input

    MAP_GPIO_setOutputHighOnPin(GPIO_PORT_P1, GPIO_PIN3); //Seting tx to its idle
sttate
    __delay_cycles(DELAY_100MS);                          //100ms delay

    //send same message
    static char something[] =

{0x0d,0x0a,0x4d,0x6f,0x6f,0x6f,0x6f,0x6f,0x20,0x20,0x20,0x20,0x20,0x20,0x5e,0x5f,0x5f
,0x5e,0x0d,0x0a,0x20,0x20,0x20,0x20,0x20,0x20,0x20,0x20,0x20,0x20,0x20,0x20,0x28,0x6f
,0x6f,0x29,0x5c,0x5f,0x5f,0x5f,0x5f,0x5f,0x5f,0x5f,0x0d,0x0a,0x20,0x20,0x20,0x20,0x20
,0x20,0x20,0x20,0x20,0x20,0x20,0x20,0x28,0x5f,0x5f,0x29,0x5c,0x20,0x20,0x20,0x20,0x20
,0x20,0x20,0x29,0x5c,0x2f,0x5c,0x0d,0x0a,0x20,0x20,0x20,0x20,0x20,0x20,0x20,0x20,0x20
,0x20,0x20,0x20,0x20,0x20,0x20,0x20,0x7c,0x7c,0x2d,0x2d,0x2d,0x2d,0x77,0x20,0x7c,0x0d
,0x0a,0x20,0x20,0x20,0x20,0x20,0x20,0x20,0x20,0x20,0x20,0x20,0x20,0x20,0x20,0x20,0x20
,0x7c,0x7c,0x20,0x20,0x20,0x20,0x20,0x7c,0x7c,0x0d,0x0a,0x00};
    send_message(something);

    while(1)
    {

    }
}

void send_message(char* msg)
{
    const int bit_time = SYSTEM_CLOCK / BAUD_RATE; //Calculate the delay for one bit
time
    int i =0 ;
    while (*msg)                                    //Traverse message
    {

        //Start bit (Logic Low)
        MAP_GPIO_setOutputLowOnPin(GPIO_PORT_P1, GPIO_PIN3);
        __delay_cycles(bit_time);

        //Send 8 data bits (LSB first)
        for ( i = 0; i < 8; i++)
        {
            if (*msg & (1 << i)) //Check if the bit is 1
            {
                MAP_GPIO_setOutputHighOnPin(GPIO_PORT_P1, GPIO_PIN3);
            }
            else //Otherwise, it's 0
            {
                MAP_GPIO_setOutputLowOnPin(GPIO_PORT_P1, GPIO_PIN3);
            }
            __delay_cycles(bit_time);
        }

        //Stop bit (Logic High)
```

```
        MAP_GPIO_setOutputHighOnPin(GPIO_PORT_P1, GPIO_PIN3);
        __delay_cycles(bit_time);

        //Move to the next character
        msg++;
    }
}
```

Exercise 1.3 Parity bit:

```
/* DriverLib Includes */
#include <ti/devices/msp432p4xx/driverlib/driverlib.h>

/* Standard Includes */
#include <stdint.h>
#include <stdbool.h>
#include <stdio.h>

void send_message(char *msg);

#define  SYSTEM_CLOCK    12000000        //Changed to 12 MHZ frequency for the 9600
baud to work properly
#define DELAY_100MS (SYSTEM_CLOCK / 10)  //Cycles for 100 ms
#define BAUD_RATE 9600

#define PARITY 1                         // 0: none, 1: odd parity, 2: even parity

int main(void)
{
    /* Stop Watchdog  */
    MAP_WDT_A_holdTimer();

    const int mclk_freq = MAP_CS_getMCLK();

    MAP_GPIO_setAsOutputPin(GPIO_PORT_P1, GPIO_PIN3); //Port 1.3 is tx and is set as
ouput
    MAP_GPIO_setAsInputPin(GPIO_PORT_P1, GPIO_PIN2); //Port 1.2 is rx and is set as
input

    MAP_GPIO_setOutputHighOnPin(GPIO_PORT_P1, GPIO_PIN3); //Seting tx to its idle
sttate
    __delay_cycles(DELAY_100MS);                          //100ms delay

    //send same message
    static char something[] = { 0x0d, 0x0a, 0x4d, 0x6f, 0x6f, 0x6f, 0x6f, 0x6f,
                                0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x5e, 0x5f,
                                0x5f, 0x5e, 0x0d, 0x0a, 0x20, 0x20, 0x20, 0x20,
                                0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20,
                                0x28, 0x6f, 0x6f, 0x29, 0x5c, 0x5f, 0x5f, 0x5f,
                                0x5f, 0x5f, 0x5f, 0x5f, 0x0d, 0x0a, 0x20, 0x20,
                                0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20,
```

```
                                0x20, 0x20, 0x28, 0x5f, 0x5f, 0x29, 0x5c, 0x20,
                                0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x29, 0x5c,
                                0x2f, 0x5c, 0x0d, 0x0a, 0x20, 0x20, 0x20, 0x20,
                                0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20,
                                0x20, 0x20, 0x20, 0x20, 0x7c, 0x7c, 0x2d, 0x2d,
                                0x2d, 0x2d, 0x77, 0x20, 0x7c, 0x0d, 0x0a, 0x20,
                                0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20,
                                0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x7c,
                                0x7c, 0x20, 0x20, 0x20, 0x20, 0x20, 0x7c, 0x7c,
                                0x0d, 0x0a, 0x00 };
    send_message(something);

    while (1)
    {

    }
}

void send_message(char *msg)
{
    const int bit_time = SYSTEM_CLOCK / BAUD_RATE; // Calculate the delay for one bit
time
    int i = 0;
    while (*msg) //Traverse message
    {
        int parity = 1; //Initialize parity bit

        if (PARITY > 0) //Check if parity is enabled
        {
            //Calculate the parity bit
            for (i = 0; i < 8; i++)
            {
                if (*msg & (1 << i)) //Check if the bit is 1
                {
                    parity ^= 1; //Toggle parity bit
                }
            }

            if (PARITY == 2) //Even parity mode
            {
                parity = !parity;
            }
        }

        //Start bit (Logic Low)
        MAP_GPIO_setOutputLowOnPin(GPIO_PORT_P1, GPIO_PIN3);
        __delay_cycles(bit_time);

        //Send 8 data bits (LSB first)
        for (i = 0; i < 8; i++)
        {
            if (*msg & (1 << i)) //Check if the bit is 1
            {
                MAP_GPIO_setOutputHighOnPin(GPIO_PORT_P1, GPIO_PIN3);
            }
```

```
            else //Otherwise, it's 0
            {
                MAP_GPIO_setOutputLowOnPin(GPIO_PORT_P1, GPIO_PIN3);
            }
            __delay_cycles(bit_time);
        }

        //Send Parity Bit
        if (PARITY > 0) //Only send parity bit if enabled
        {
            if (parity) //Check if the parity bit is 1
            {
                MAP_GPIO_setOutputHighOnPin(GPIO_PORT_P1, GPIO_PIN3);
            }
            else //Otherwise, it's 0
            {
                MAP_GPIO_setOutputLowOnPin(GPIO_PORT_P1, GPIO_PIN3);
            }
            __delay_cycles(bit_time);
        }

        //Stop bit (Logic High)
        MAP_GPIO_setOutputHighOnPin(GPIO_PORT_P1, GPIO_PIN3);
        __delay_cycles(bit_time);

        //Move to the next character
        msg++;
    }
}
```

Exercise 2.1 Energy measurements:

Exercise 2.2 LPM4.5: