An-Najah National University
Faculty of Engineering and IT

جامعة النجاح الوطنية
كلية الهندسة وتكنولوجيا المعلومات

Computer Engineering Department
Microcontroller Lab (10636496)
Report Grading Sheet

Instructor Name: Hikmat Darawsheh	**Experiment: 3**
Academic Year: 2024	**Performed on:**
Semester:	**Submitted on:**
Student Names:	
1- Sadeen Hawash	2- Walaa Ashqar
3-	4-
5-	6-

Evaluation Criterion	CLO	Grade	Points
Abstract and Aims Aims and idea of the experiment are clearly stated in simple words		10	
Introduction, Apparatus and Procedures Introduction is complete and well-written, all grammar/spelling correct, Appropriate background information related to the principles of the experiment is provided. The list of apparatus and procedures are also provided		15	
Experimental Results, Calculations and Discussion Results analyzed correctly. Experimental findings adequately and specifically summarized, in graphical, tabular, and/or written form. Comparison of theoretical predictions to experimental results, including discussion of accuracy and error analysis as needed.		50	
Conclusions Conclusions summarize the major findings from the experimental results with adequate specificity. Highlighting the most important results		15	
Appearance Title page is complete, page numbers applied, content is well organized, correct spelling, fonts are consistent, good visual appeal. You have also to use reference for the information you provide		10	
Total		100	

Table of Contents

Contents

Abstract 2
Introduction 2
Materials 3
Methods 3
Experimental Results 4
Discussion 7
Conclusion 7

Abstract

The purpose of this experiment is to learn about asynchronous communications and how to communicate with a microcontroller using a terminal emulation program to implement a point-to point-serial link between the ChipKITTM MX7 and a PC.

Introduction

In this project, we used the USB port to connect a PC and a Pic Kit microcontroller to create a basic communication channel. The communication was facilitated using Tera Term software, and data was transmitted asynchronously using the UART serial communication protocol.

Materials

- ChipKITTM Pro MX7 processor board with USB cable.
- Microchip MPLAB R X IDE.
- MPLAB R XC32++ Compiler.
- MPLAB Harmony Framework.
- Tera Term application

Methods

Part 1:

In the first section, we have to transmit the microcontroller a byte that has been added to it after receiving it as a character from the terminal. In the receive state, we must first determine whether the buffer is empty. If it is, the microcontroller receives the byte, which we then store in an input variable before moving on to the send state. Simply add one to the x, transmit it, and then return to the receive state when in the send state.

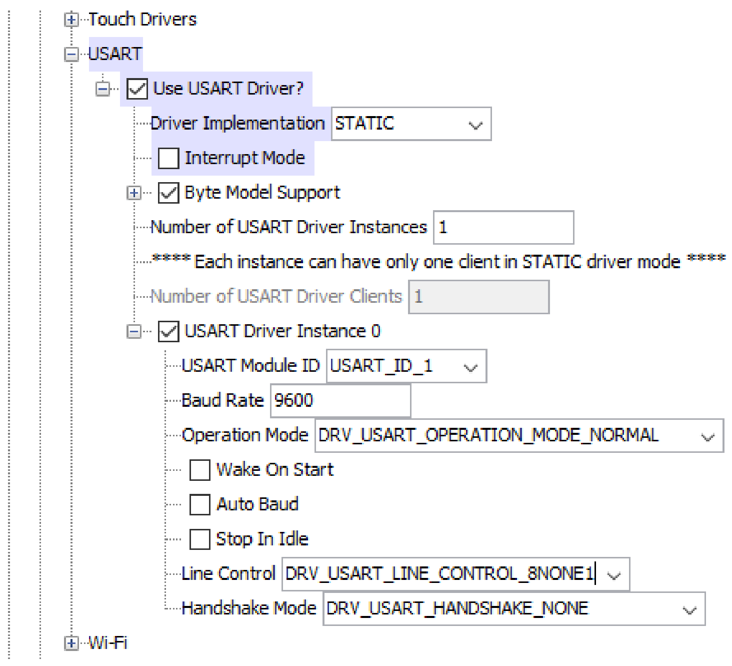
Part 2:

Part 2 requires us to receive a command from the Tera term, which can be either LED1 or LED2. If LED1 is given, the microcontroller should turn on LED 1 (RG12) and send a message to the Tera Term indicating that LED 1 is on. Similarly, if LED2 is given, the microcontroller should turn on LED 2 (RG13). and send a message to the Tera Term indicating that LED 2 is on, or if the command was ALL turn all the leds on, otherwise turn off the leds and send a message indicating that both leds are off. To accomplish that, we must read it character by character, store it as an array of characters, and then determine whether we receive '\n', '\r', or '*', which indicate the end of the line or, in other words, that the command has ended. Next, we must insert a null after the final character, store it in the character array, and proceed to the send state.

In send state we need to check the command by comparing it to the two commands (LED1 or LED2 or ALL) and make action on Leds accordingly and otherwise turn off the leds and send the result to the Tera Term.

Experimental Results

First, we configure the UART to use it and then we initiate the states for the parts in the app.h file as shown:



```
Touch Drivers
USART
  [✓] Use USART Driver?
      Driver Implementation STATIC
      [ ] Interrupt Mode
      [✓] Byte Model Support
      Number of USART Driver Instances 1
      **** Each instance can have only one client in STATIC driver mode ****
      Number of USART Driver Clients 1
      [✓] USART Driver Instance 0
          USART Module ID USART_ID_1
          Baud Rate 9600
          Operation Mode DRV_USART_OPERATION_MODE_NORMAL
          [ ] Wake On Start
          [ ] Auto Baud
          [ ] Stop In Idle
          Line Control DRV_USART_LINE_CONTROL_8NONE1
          Handshake Mode DRV_USART_HANDSHAKE_NONE
Wi-Fi
```

```
typedef enum
{
        APP_STATE_INIT=0,
        APP_STATE_SERVICE_TASKS,
        APP_STATE_RECEIVE,
        APP_STATE_ECHO,
        RX_State,
        TX_State,

} APP_STATES;
```

Then we initiate the variables in app.h and give its values in app.c in the as shown:

```
typedef struct
{
        APP_STATES state;
        char input;
        char * rx_byte

} APP_DATA;
```

Then we write the code for the first part to communicate between the UART and Tera term:

```
case APP_STATE_INIT: {
     bool appInitialized = true;
     if (appInitialized) appData.state = APP_STATE_RECEIVE;
     break;
 }
 case APP_STATE_RECEIVE:{
     if (!DRV_USART0_ReceiverBufferIsEmpty()){
        appData.inputt = DRV_USART0_ReadByte();
        appData.state = APP_STATE_ECHO;
     }
      break;
  }
```

```
case APP_STATE_ECHO:
{
   DRV_USART0_WriteByte((appData.inputt +1));
   appData.state = APP_STATE_RECEIVE;

   break;
}
```

The next part we write two states one for receive and another for sending:

```
case APP_STATE_SERVICE_TASKS:
{
      if (!DRV_USART0_ReceiverBufferIsEmpty())
      appData.state=RX_State;
      break;
}
case RX_State:
{
      appData.rx_byte=DRV_USART0_ReadByte();
      if (appData.rx_byte=='\n'|| appData.rx_byte=='\r'){
              appData.buffer[i++] = '\0';
              if(strcmp(appData.buffer,"led1") ==0){
                      LATG = 0x1000;
                      DRV_USART0_WriteByte('\n');
              }
              else if(strcmp(appData.buffer,"led2")==0){
                      LATG = 0x2000;
                      DRV_USART0_WriteByte('\n');
              }
              else if(strcmp(appData.buffer,"led3")==0){
                      LATG = 0x4000;
                      DRV_USART0_WriteByte('\n');
              }
```

```
                    else if (strcmp(appData.buffer,"leds are off")==0){
                            LATG = 0X0000;
                    }
                    else if(strcmp(appData.buffer,"led4")==0){
                            LATG = 0x8000;
                            DRV_USART0_WriteByte('\n');
                    }
            i=0;
            appData.state = APP_STATE_SERVICE_TASKS;
    }
    appData.buffer[i++] = appData.rx_byte;
    appData.state=TX_State;
    break;
}
case TX_State:
{
    DRV_USART0_WriteByte(appData.rx_byte);
    appData.state = APP_STATE_SERVICE_TASKS;
 }
```

Discussion

In this experiment we found that we can make can communicate with Tera Term using UART and also take commands from tera term and for example turn leds on or off.

Conclusion

We learned how to use the UART to send and receive text (command) and depending on that text (commands) we can make different actions such as turning certain lights on or off and sending different messages each time to Tera Term.