EE445L – Lab10: Wireless Serial Communication

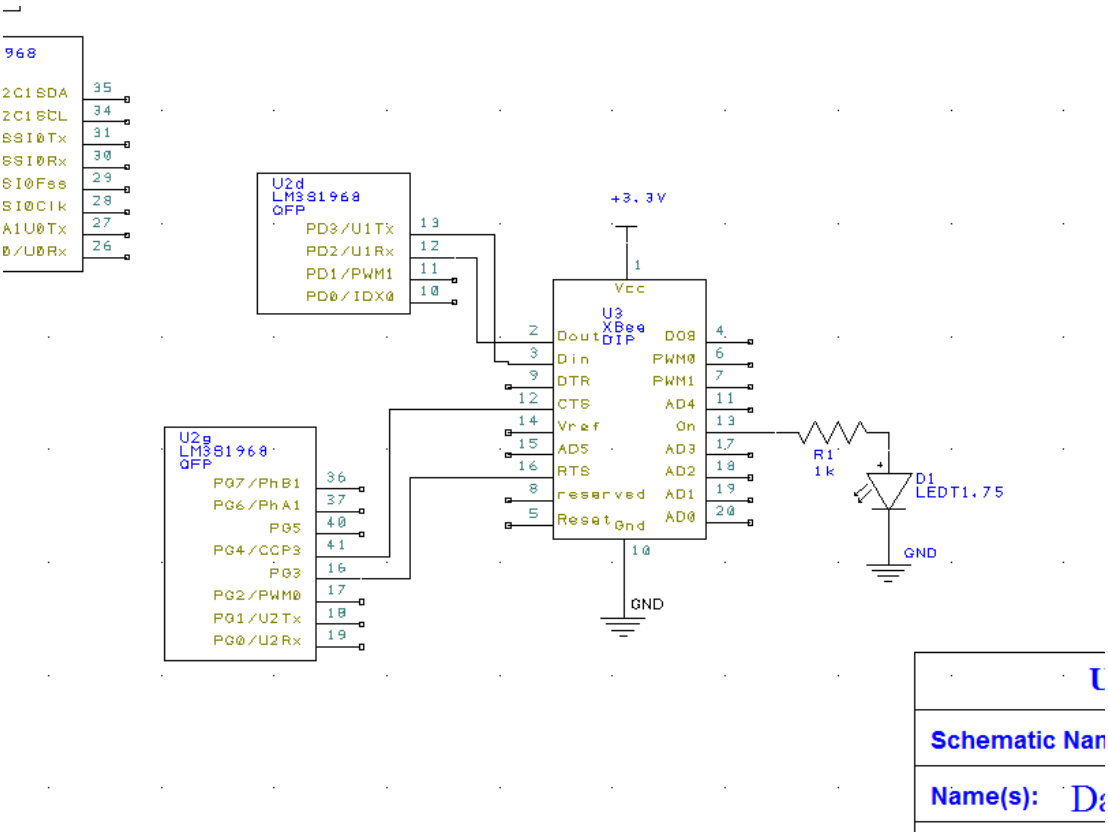
Harley Ross and Dalton Altstaetter
4/15/14

OBJECTIVES

The goals of this lab are to develop debugging techniques for transmitter and receiver systems. We will also learn how to use FIFO and other data structures to implement communications. Lastly, we will implement a text communication system between a PC and an OLED display using IEEE 802.15.4-ZigBee wireless module.

HARWARE DESIGN

SCH



SOFTWARE DESIGN

ZigBee

```
// XBee.c
#include "Xbee.h"
#include "systick.h"
#include "string.h"
#include "UART2.h"

static void sendATCommand(char* input);

unsigned char destination[2] = {0x00,0x4F};
unsigned char startDelimiter = 0x7E;
```

```
unsigned char opt = 0x00;
unsigned short length;
unsigned short numBytes;
unsigned char checksum;

static void check(char* r, int* flagPtr)
{
	int i;

	char checkString[4] = "OK";
	checkString[2] = CR;

	for(i=0; i < 3; i++)
	{
		if(r[i] != checkString[i])
		{
			*flagPtr = 1;
		}
	}
}

void XBee_Init(void)
{
	int flag;
	char response[20];

	flag = 0;
	OutCRLF_UART0();
	UART1_OutChar('X');	// send to XBee
	UART0_OutChar('X');	// echo to user
	SysTick_Wait10ms(110);  // guard time delay
	UART1_OutString("+++"); // send to XBee for AT cmd mode
	UART0_OutString("+++"); OutCRLF_UART0();// echo to user

	UART1_InString(&response[0], 5);
	SysTick_Wait10ms(110);  // guard time delay

	UART0_OutString(&response[0]); OutCRLF_UART0();

	//check(&response[0], &flag);
	if(strcmp(response,"OK"))   //flag)
	{
		UART0_OutString("Error entering AT Command Mode");
	}

	sendATCommand("ATDL50"); OutCRLF_UART1();  // sets destination address to 79

	sendATCommand("ATDL"); OutCRLF_UART1();  // sets destination address to 79
	//UART1_InString(&response[0],20); UART0_OutString(response);
	sendATCommand("ATDH0");  OutCRLF_UART1();  // sets destination high address to 0

	sendATCommand("ATDH"); OutCRLF_UART1();  // sets destination address to 79
	//UART1_InString(&response[0],20); UART0_OutString(response);
	sendATCommand("ATMY51"); OutCRLF_UART1();  // sets my address to 78
                                        sendATCommand("ATMY"); OutCRLF_UART1();  // sets
destination address to 79
	//UART1_InString(&response[0],20); UART0_OutString(response);
	sendATCommand("ATAP1");  OutCRLF_UART1();  // set for API mode 1
```

```
        sendATCommand("ATAP"); OutCRLF_UART1();  // sets destination address to 79

        sendATCommand("ATBD"); OutCRLF_UART1();  // sets destination address to 79

        sendATCommand("ATID"); OutCRLF_UART1();  // sets destination address to 79

        sendATCommand("ATCH"); OutCRLF_UART1();  // sets destination address to 79

        sendATCommand("ATCN");   OutCRLF_UART1();  // ends the AT Command mode
        OutCRLF_UART0();
        UART0_OutString("Done with AT Cmd Mode");   OutCRLF_UART0();
        // also check ATBD == 3 to make sure the baud rate is set at 9600 bits/sec

        //ATCH parameter<CR> changes the channel range between 0x0B and 0x1A (default
0x0C)
        //ATID parameter<CR> changes the personal area network ID range between 0x0000 and
0xFFFF (default 0x3332)
}


//sendATCommand – sends an AT command repeatedly until it receives a reply that it was
correctly received
//This routine receives the various parameters associated with an AT command as input
then transmits the formatted
//command to the XBee module. After a blind-cycle delay, the routine checks if the
command has been successfully
//received by determining if the module has returned the 'OK' character string.
static void sendATCommand(char* input)
{
        int flag;
        char r[20];
        flag = 0;

        UART1_OutString(input);      // send at cmd
        OutCRLF_UART1();

        SysTick_Wait10ms(2); // delay

        UART1_InString(r,19); // get OK<CR> response
        UART0_OutString("Should get OK :");
//      OutCRLF_UART0();

        if(!strcmp(r,"OK"))
        {
                UART0_OutString(r);//"We got Correct Response");
                OutCRLF_UART0();
                return;
        }

                UART0_OutString(r);//"We got Correct Response");
                OutCRLF_UART0();
                return;

        check(r, &flag); // check for the OK<CR> response

        while(flag)
        {
                flag = 0;
```

```
                UART1_OutString(input);
                OutCRLF_UART1();
                SysTick_Wait10ms(2);
                UART1_InString(r,19);
                UART0_OutString("Should get OK :");
                UART0_OutString(r); // think this get the OK<CR> response
                check(r, &flag);
        }
}



//XBee_TxStatus – determine transmit status
//When the XBee module transmits an API transmit data frame it will receive an
acknowledgement from the
//destination module if the frame was received without errors. The status of the
transmission will be sent to the
//LM3S1968 via an API transmit status frame. This routine returns a ‘1’ if the
transmission was successful and a ‘0’
//otherwise. The following figure shows a response the XBee returns after the transmitter
sends a TxFrame that was
//properly received by the other computer, measured on XBee pin 2 Dout.
int XBee_TxStatus(void)
{
        //length + 4
        char buffer[50];
        int x, y, length, returnedChecksum, ok, checksum;
        int id, dest_high, dest_low, success, api;
        api = 0x01;

        UART1_InString(buffer,19); // need to change this to InChar, till CR, maybe just
null char

        //get byte values
        id = buffer[4];
        dest_high = buffer[5];
        dest_low = buffer[6];
        success = buffer[8];

        //calculate length of message
        x = buffer[1];
        y = buffer[2];
        length = (x << 8) + y;

        //calculate check sum
        checksum = 0xFF - (api + id + dest_high + dest_low + success);

        //find returned checksum value
        returnedChecksum = buffer[length + 3];

        //find ok value
        ok = buffer[length + 2];

        //check ok and if the check sums match
        if(ok == 0 && returnedChecksum == checksum)
        {
                return 1;
        }
```

```
	return 0;
}
//-------------------------------------------------------------------------------------
----------
void XBee_SendTxFrame(void)
{
	int i;
	char* XbeeString;
	char string[25] = {0};

	UART0_OutString("InString0: ");
  UART0_InString(&string[0],19); OutCRLF_UART0();// get the message to send from the user
	XbeeString = XBee_CreateTxFrame(&string[0]); // create message
	//UART1_OutString(XbeeString); OutCRLF_UART1(); // send to XBee Wireless
	UART0_OutString("Sent: ");
	for(i=0; i < numBytes+9; i++)
	{
		UART1_OutChar(XbeeString[i]);  // send to XBee Wirelessy
		UART0_OutChar(XbeeString[i]);  // echo to user
	}
	OutCRLF_UART1();
	OutCRLF_UART0();

	//UART0_OutString("Sent: "); UART0_OutString(XbeeString); OutCRLF_UART0(); // tell
us what we sent

	if(!XBee_TxStatus())
	{
		UART0_OutString("Error, acknowledge not received"); OutCRLF_UART0();
	}
	else
	{
		UART0_OutString("Message sent successfully"); OutCRLF_UART0();
	}
}
//-------------------------------------------------------------------------------------
----------
char* XBee_CreateTxFrame(char* string)
{
	static unsigned char ID = 1;
	int i;
	static char message[25];
	unsigned char sum;

	checksum = 0;

	// zero message buffer
	for(i=0;i<25;i++)
	{
		message[i] = 0;
	}

	numBytes = 0;
	i=0;
	while(*(string+i) != 0)
	{
		numBytes++;
		i++;
```

```
	}
	length = numBytes+5; // 5 counts for the API, ID, Destination, & OPT bytes

	//for(numBytes = 0; *string != NULL; numBytes++){}

	message[0] = startDelimiter;
	message[1] = (unsigned char)((length & 0xFF00)>>8); // get top byte of length
	message[2] = (unsigned char)(length & 0x00FF); // get lower byte of length
	message[3] = 0x01; // API mode 1
	message[4] = ID;
	message[5] = destination[0];
	message[6] = destination[1];
	message[7] = opt;

	for(i=0; i<numBytes; i++)
	{
		// fill the message array
		message[8+i] = string[i];
	}

	ID = (ID+1)%256; // keep in range of an unsigned char
	if(ID == 0)
	{
		ID = 1; // make sure the ID never equals zero
	}

	sum = message[3]+message[4]+message[5]+message[6]+message[7];

	for(i=0; i < numBytes; i++)
	{
		sum += message[i+8];
	}
	checksum = 0xFF-sum;
	message[numBytes+9] = checksum;
	return &message[0];
}
//-------------------------------------------------------------------------------------
----------
```

MEASUREMENT DATA

Estimated Maximum Bandwidth: 2MHz
Estimated range: 30m

ANALYSIS AND DISCUSSION

An important part of this lab was understanding how the UART works to communicate to the Zigbee and the PC. Then using the UART, we had to understand the communication protocols for communication to occur between the Zigbees. Calculating the checksum and making sure the protocols were followed precisely were the most difficult parts and required the most debugging.