Andrew Ang 300302845

ECEN301 Lab Session 7: Serial Communication

Objectives

Methodology

Part 1: Communication between micro and PC via HyperTerminal

In this part, communication is established between the microcontroller and the PC through serial communication via the HyperTerminal application.

For this part of the laboratory, a program is written that will:

1. Initialise the serial port to a baud rate of 1200Hz
2. Read incoming characters
3. Display the characters on the LCD display
4. Echo characters back to the PC

With the phys340libkeil library, there are serial port functions. These functions are:

1. initSerial(unsigned int baud)
2. writeCharSerial(char c)
3. writeLineSerial(char * s)
4. readCharSerial()
5. readLineSerial(char * s, int max)

initSerial() initialise the serial port to a given baud rate. It uses timer 1 which is 16bit. It takes in a unsigned int to set the baud rate of the serial port. writeCharSerial writes a single character to the hyperterminal application. It takes a char character. writeLineSerial writes an array of characters to the hyperterminal until null character. readCharSerial reads a single character from the hyperterminal application. ReadLineSerial read an array of characters and store result in provided array. Keeps blocking until a newline or ‘max’ characters received. Destination array must have size of at least ‘max’ +1.

To set the baud rate of 1200Hz by using the initSerial method. A parameter of 1200 is used to set the baud rate.

To read incoming characters, the readCharSerial() method is called. It returns the value typed on the hyperterminal application.

To display the characters on the LCD display, the phys340livkeil.h library contains a writeCharLCD(), which takes the character returned from readCharSerial method and is used as an argument for the writeCharLCD() method.

To echo characters back to the PC, the writeCharSerial method is called to write an array of characters back to the hyperterminal application.

For this part, the micro was connected to COM5.

Part 2: Implementation of the initLCD.

For this part, a program is written to initialise the serial port to the baud rate. The procedure is as follows:

1. Configure Timer 1 as an 8 bit timer with automatic reload.
2. Set SM0 and SM1 to provide an 8 bit serial UART with variable Baud rate.
3. Set The PCON register, setting SMOD so that the Baud rate is doubled.
4. Set TH1 to set the baud rate.
5. Turn on timer 1 in the TCON register.
6. Enable serial reception.

The first step requires the TMOD register to be set to 0x20. This sets the bits M11 and M01 high and low respectively to set the timer 1 as an 8-bit auto-reload timer.

The second step requires the bits SM0 and SM1 of the SCON register to be low and high respectively to provide an 9 bit serial UART with variable baud rate.

In the third step, the TCON register is used and a value of 0x80 is set. This turns only bit SMOD1 on to select double baud rate in mode 1.

Setting the baud rate requires the TH1 register to be set a certain value for a particular baud rate. That value can be calculated using the following equation:

$$Baud = \frac{F_{OSC}}{192(256 - TH1)}$$

Rearranging for TH1, the value to be placed across the TH1 register can be calculated for a baud rate. F_{OSC} in this case is the clock frequency of 12MHz.

$$TH1 = 256 - \frac{F_{OSC}}{192Baud}$$

To turn on timer 1, in the TCON register, TR1 is set to 1.

Lastly to enable the serial reception, bit REN in the SCON register is set high.

Part 3: Implementation of Read() and Write().

To perform a read function, the following procedure is used:

1. Wait until the RI interrupt flag goes high
2. Read the data from the SBUF register
3. Clear the RI interrupt.

Within the while loop of the main function, an if statement is only initialised if the bit RI in the SCON register is set to high. The RI flag is a receive interrupt flag. This indicates that the interrupt flag goes high. By fulfilling the if statement condition, the data is take from the SBUF register. The SBUF register handles data sent and received by the serial I/O port. This value corresponds to the data coming from the hyperterminal. It is stored in a value. After the reading the data, the RI interrupt flag is set back to 0.

To perform a write function, the procedure is similar to that of the read function, but it is in the reverse order:

1. Clear the TI interrupt flag
2. Put the har to send into the SBUF variable
3. Wait until the TI interrupt flag goes high.

After when the reading procedure, the TI flag is set false to indicate the beginning of the writing procedure. The TI flag is a transmit interrupt flag. The value is stored back into the SBUF register. A while loop is then initiated and it is not terminated until the flag TI is set to true. This indicates that the writing procedure is finished. Once the TI is set true, the while loop breaks and the RI is set to 0. This is not initiated again until the RI interrupt is set.

Questions

Q1 What is connected to COM ports 1,2 and 3

Through the device manager, com ports 1,2 and 3 connection can be determined. Com11 is a communications port. Com 3 is an Intel® Active management tech –SOC. Whereas Com 2 is a serial port at the back of the pc.

Q2 Investigate what happens when the variable TH1 is altered. How stringently must the baud rate be defined.

Varying the baud rate resulted in either the hyperterminal not registering any input from the micro, or the micro not registering correct output from the PC. Using the baud rate of 1200Hz and putting it in the second formula above, leads to a TH1 value of 203.64. Using a value of 204 resulted in an incorrect initialisation as printed characters did not correspond to what was typed on the keyboard. But a value of 203 allowed correct reading.

Q3 Explain where the formula for the baud rate comes from.

Depends on the number of bits in timer 1 and the clock frequency.

Q4 Explain what the bits of SCON do.

Bit	Description
FE	Framing error bit
SM0	Serial port mode bit 0
SM1	Serial mode bit 1
SM2	Serial port mode 2 bit/ multiprocessor communication enable bit
REN	Reception enable bit
TB8	Transmitter bit 8/Ninith bit to transmit in modes 2 and 3
RB8	Receiver Bit 8/ Ninth bit received n modes 2 and 3
TI	Transmit Interrupt flag
RI	Receive interrupt flag.

Andrew Ang 300302845

CODE

PART 1

```
#include <t89c51ac3.h>

#include <phys340libkeil.h>

#include <string.h>

#include <stdio.h>


char value;


void main()

{

	initLCD();

	initSerial(1200);

	while(1)

	{

		//get char from hyperterminal

		value = readCharSerial();

		//write to LCD

		writeCharLCD(value);

		//write to hyperterminal

		writeCharSerial(value);

	}

}
```

Andrew Ang 300302845

PART 2

```
#include <t89c51ac3.h>

#include <phys340libkeil.h>

#include <string.h>

#include <stdio.h>


char value;

void myInitSerial()

{

        //configuring timer 1 pg 71

        TMOD = 0x20;

        //setting SM0 and SM1 to provide an 8 bit serial UART with variable baud rate pg 63

        SM0 = 0;

        SM1 = 1;

        //setting PCON register setting SMOD so that the baud rate is doubled pg 65

        PCON = 0x80;

        //setting TH1 to set the baud rate pg 72


        TH1 = 203;

        //turning on timer 1 pg 70

        TR1 = 1;

        //enable serial reception pg 63

        REN = 1;


}


void main()

{

        initLCD();

        myInitSerial();

        while(1)
```

```
    {
        //get char from hyperterminal
        value = readCharSerial();
        //write to LCD
        writeCharLCD(value);
        //write to hyperterminal
        writeCharSerial(value);
    }
}
```

Andrew Ang 300302845

PART 3 read and write combined

```
#include <t89c51ac3.h>

#include <phys340libkeil.h>

#include <string.h>

#include <stdio.h>


char value;



void main()

{

        initLCD();

        initSerial(1200);

        while(1)

        {

                //checking for RI interrupt flag

                if(RI)

                        {

                                //get char from hyperterminal

                                value = SBUF;

                                //value = readCharSerial();

                                //write to LCD

                                writeCharLCD(value);

                                //write to hyperterminal


                                TI = 0; //clear TI interrupt flag

                                SBUF = value;

                                while(TI == 0){}

                                //writeCharSerial(value);

                                //clear RI interrupt flag
                        Rl = 0;
                    }
        }
}
```