Infrared Data Transmission with the PIC Microcontroller

Matthew Beckler
beck0778@umn.edu
EE2361 Lab 007

May 5, 2006

Abstract

To create an infrared communications system, utilizing a transmitter and receiver, you will need one IR diode, similar to a standard LED, and one IR receiver. Two project boards are combined into one project, utilizing two computers running terminal emulation software, such as HyperTerminal.

Introduction

There are two main sections to this experiment, the transmitter and receiver. The transmitter uses the pulse width modulation (PWM) module in combination with an IR diode to produce a 38KHz carrier wave. This wave is then modulated at a slower frequency to send data. This is both similar and dissimilar to the standard RS-232 transmission protocol, in that we will be using a start bit, 8 data bits, and a stop bit. The IR scheme is different in that to send a pulse, you cannot merely sent a solid IR signal. You must instead send a 38KHz square wave, with a duty cycle of approximately 40-50%. The signal will be transmitted with the least-significant bits sent first.

To receive data, the processor has a specialized receiving module that converts an active-low signal consisting of a modulated 38KHz IR square wave into a standard TTL logic level signal. This can be easily read by the processor, using the timer 2 module to mark apart the signal timings. The data is received least-significant bit first, and the resulting value must be constructed by summing the respective powers of two.

Data

For the transmitter, two functions have been crafted to handle timings. The `wait600us` and `wait1000us` functions provide a waiting time for the respective bits. The processor first initializes the serial communications registers, along with timer 2 set-up for PWM operation. Each time through the main loop, the processor waits for a character to be sent from the computer. The processor then enters into an 8-execution loop. For each bit in the input byte, the processor shifts a value of 1 to the left the requesite number of spots, producing the

appropriate power of two. Each power of two is then bitwise-and'ed with the input byte, to determine if a particular bit of the input is set or cleared. Depending on the bit's value, the PWM module is either turned-on or turned-off, and a delay function is entered, to produce the requisite signal length. After all 8 bits have been sent, the processor sends a stop bit, and resumes waiting for the next input bit.

For the receiver, the processor enters a busy loop waiting for the RB0 pin to go low. This signifies the start bit of the transmission. The processor then waits until 1.5 times the bit's time period, and starts sampling (hopefully in the middle of each bit), recording the value of each bit. It also computes the appropriate power of two to add (or not) to the total, depending on the received bit's value. Upon receiving all 8 bits, including the stop bit, the processor transmits the total value to the computer using the serial connection.

Conclusion

This laboratory experiment is very similar to the procedure a computer's modem uses to connect to a computer network using a dial-up connection over the voice telephone network. The modem, much like the IR transmitter processor, modulates a higher frequency signal to transfer the data. The receiving modem would demodulate the signal, but in our experiment, the IR receiver automatically demodulates and inverts the signal, which makes it much easier to decode.

C Language Program - Transmitter

```
#include <p18f452.h>
#pragma config OSC=HSPLL, WDT=OFF, BOR=OFF, PWRT=ON

void putch(char byte)
{
    while (!PIR1bits.TXIF);
    TXREG = byte;
}

char getch(void)
{
    while (!PIR1bits.RCIF);
    return RCREG;
}

void wait600us(void)
{
    int x = 453;
    while(x--);
}

void wait800us(void)
{
    int x = 610;
    while(x--);
}

void wait1000us(void)
{
    int x = 762;
    while(x--);
}

void main(void)
{
    //initialize variables
    char input, whichBit, temp;

    //initialize serial registers
    TXSTAbits.TXEN = 1;
    RCSTAbits.SPEN = 1;
    RCSTAbits.CREN = 1;
    SPBRG = 0x40;

    TRISC = 0;

    //initialize timer2 for PWM use
    T2CON = 0b00000101;
```

```
    PR2 = 65;

    //initialize PWM module
    CCPR1L = 32;
    //CCP1CON = 0b00001100; //start off

    putch('T');
    putch('X');
    putch(0x0d);
    putch(0x0a);

    while(1)
    {
        putch('-');
        putch('>');
        putch(' ');
        //wait for input from the computer
        input = getch();
        //input = 0x41; // A
        putch(input);
        putch(0x0d);
        putch(0x0a);

        CCP1CON = 0x0C;
        wait800us();

        whichBit = 0;
        while(whichBit < 8)
        {
            temp = 1 << whichBit;
            if ((input & temp) == 0)
            {
                CCP1CON = 0x0C;
                wait800us();
            }
            else
            {
                CCP1CON = 0x00;
                wait800us();
            }
            whichBit++;
        }

        CCP1CON = 0x00;
        wait800us();
    }
}
```

C Language Program - Receiver

```
#include <p18f452.h>
#pragma config OSC=HSPLL, WDT=OFF, BOR=OFF, PWRT=ON

void putch(char byte)
{
    while (!PIR1bits.TXIF);
    TXREG = byte;
}

char getch(void)
{
    while (!PIR1bits.RCIF);
    return RCREG;
}

void wait800us(void)
{
    int x = 610;
    while(x--);
}

void main(void)
{
    //initialize variables
    char input, whichBit, temp, total;

    //initialize serial registers
    TXSTAbits.TXEN = 1;
    RCSTAbits.SPEN = 1;
    RCSTAbits.CREN = 1;
    SPBRG = 0x40;

    //Setup PORTB
    INTCON2bits.RBPU = 0;
    TRISB = 0x0F;
    PORTB = 0;

    //Setup Timer2
    T2CON = 0x79; //PRE = 4 POST = 16
    PR2 = 187;
    putch('R');
    putch('x');
    putch(0x0D);
    putch(0x0A);
    while(1)
    {
        while(!PORTBbits.RB0);
        T2CONbits.TMR2ON = 1;
```

```
        whichBit = 0;
        total = 0;
        while(whichBit < 8)
        {
            while(!PIR1bits.TMR2IF);
            PR2 = 125;
            PIR1bits.TMR2IF = 0;


            if(PORTBbits.RB0 == 0)
            {
                temp = 1 << whichBit;
                total += temp;
            }

            whichBit++;
        }
        putch(total);
        T2CONbits.TMR2ON = 0;
        PR2 = 187;
    }
}
```

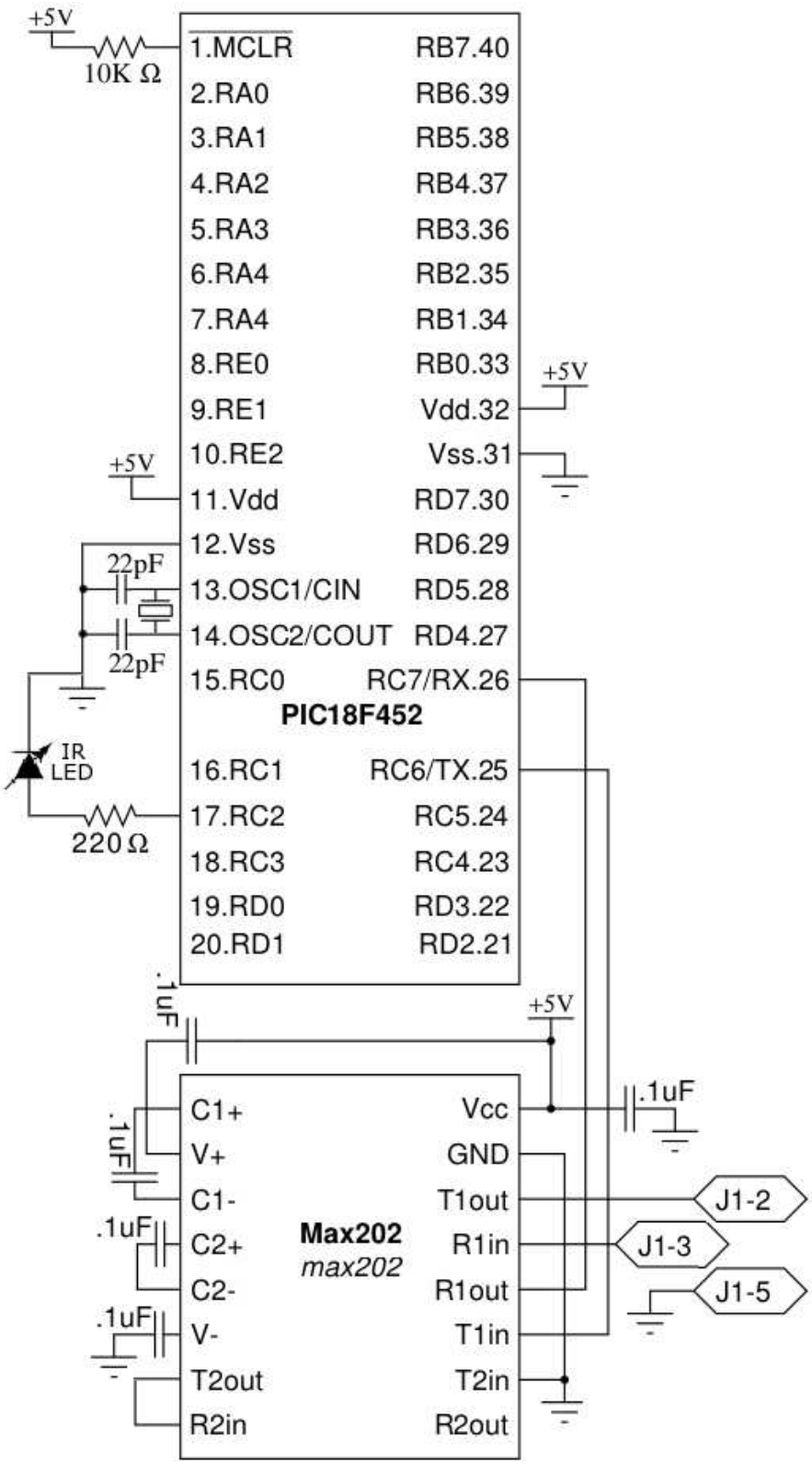

Figure 1: Transmitter Circuit Schematic

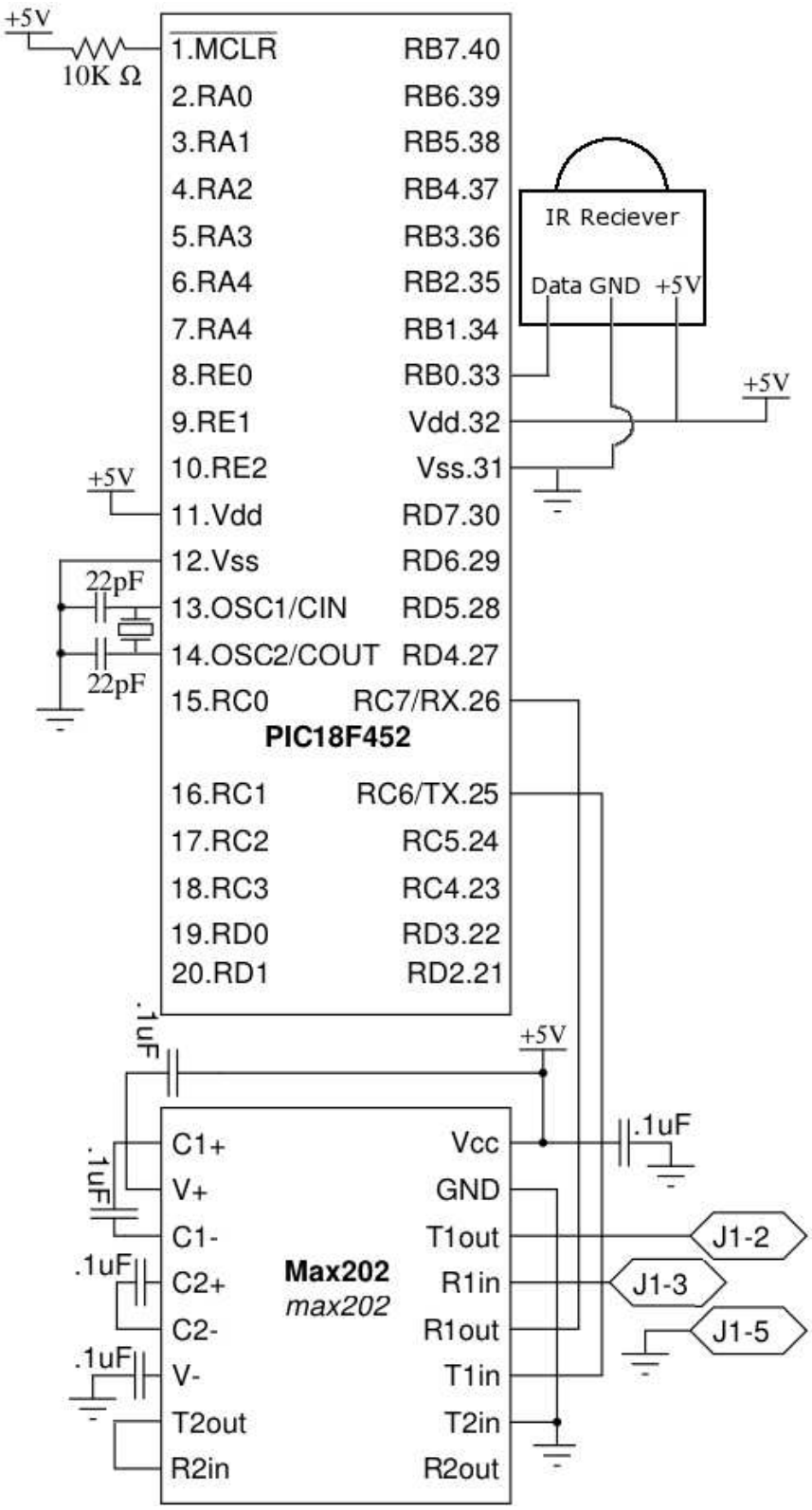

Figure 2: Receiver Circuit Schematic