Group #

1

RS 232 LASER TRANSCIEVER

Interface Synthesis [summer 2003] project

Mohteshim Hussain
Hussain Caraballo
Iman Khalid
Shaddab Amjad
Usman Tariq

INTERFACE SYNTHESIS PROJECT

RS-232 Laser Transceiver Documentation

Submitted to:

Dr. Saeid Moslehpour

Table of Contents

Chapter

1

RS-232 standards

RS-232 is a standard for transferring data in serial format. Information is sent in small packets of data called data frames. A data frame consists of the following sequence: a start bit, the actual data word, an optional parity bit and ends with one or two stop bits. The data word can be 7 or 8 bits long. RS-232 offers asynchronous communication with the combination of start and stop bits of being used to synchronize each data frame. The parity bit is used by the receiver to determine if an odd number of bits were corrupted during transmission. There are two types of parity, odd and even. For example, if even parity is used the transmitter makes the parity bit a 1 anytime there is an odd number of 1's in the data word. This makes a total even number of bits in the data frame. If an odd number of bits arrives at the receiver then the data frame was corrupted.

The standard not only specifies the order of bits but also specifies the voltage levels used to send the data. Bipolar signaling is used in the RS-232 protocol to support long cabling with minimum noise. A logic 0 is represented by a positive voltage between +3VDC and +15VDC and a logic 1 is represented by a negative voltage between -3VDC and -15VDC.

The IBM PC serial port contains a number of handshaking lines that are used to indicate the willingness of the receiver to receive data and the sender to send data. These are not strictly needed and so I will not cover them here.

The Transceiver

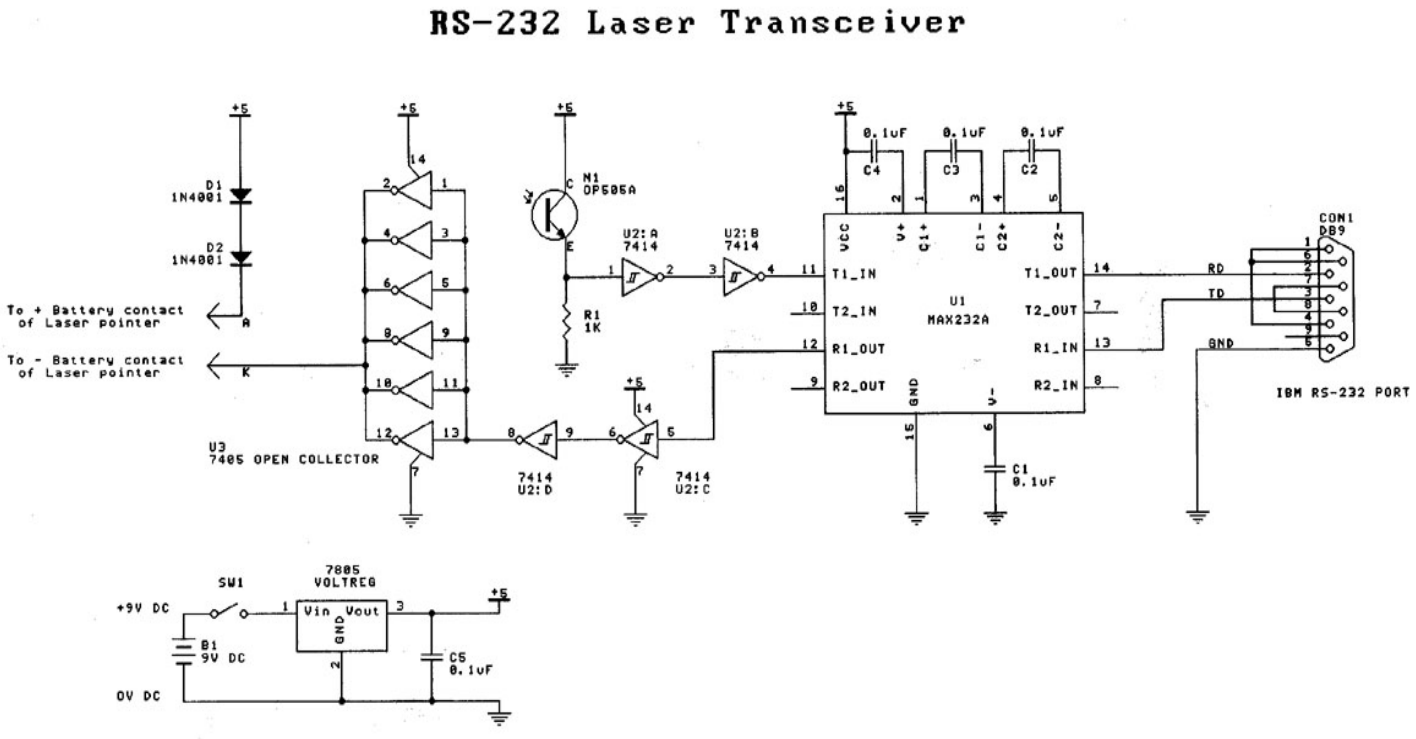
The transceiver is based on the MAX232A IC for generating and receiving RS-232 compatible voltage signals. The receiving sensor is an NPN infrared photo-transistor (OP505A). I chose an infrared photo-transistor to minimize ambient light interference. Although the laser wavelength is in the visible spectrum (~670nm) the photo-transistor's broad response band (550nm to 1050nm) is wide enough to sense the intense laser beam. The signal from the photo-transistor is buffered via a pair of Schmitt trigger buffers to clean up and square the signal. The output of the second buffer is then directly converted to a RS-232 standard signal via the MAX232A.

The MAX232A generates +10V and -10V voltage swings using a dual charge-pump voltage converter from a single +5VDC rail. Several different versions of the MAX232 chip exist. The A version requires only 0.1 uF capacitors for the charge-pump and inverter, whereas the MAX232 requires 1uF

capacitors. The advantage of the A version is that it has faster response times, and allows for faster data rates.

The laser diode driver consists of a 7405 open-collector hex inverter IC. All the outputs of the inverters are coupled together to provide enough drive current for the laser diode which draws around 35mA @ 3V. A 7805 voltage regulator is used to provide the IC and laser diode with a stable 5V voltage source. The two 1N4001 diodes, in series with the laser diode, step down the voltage from +5VDC to around 3.6VDC which is close to the nominal voltage for the laser diode.

The transceiver is designed in such a way that when no signal is present the laser is on. This helps you see where the laser is pointing during the laser-detector alignment. The transceiver is powered by a 9V battery and draws approximately 80mA (laser on) and 40mA (laser off).



The schematic of the transceiver. The MAX232A IC provides the interface to the PC, and the 74LS05 is used to drive the laser diode inside the laser pointer.

Chapter

2

Construction

Construction of the transceiver is fairly straight forward. First start by checking the PCB to make sure it is clean and free from dirt. Next mount all the passive components, this includes the resistors and capacitors. You can now mount the diodes taking note of their polarity. Next fit the active components, this includes all the ICs and the voltage regulator. The voltage regulator does not require a heat sink, so it can be placed flush against the PCB. The ICs may be mounted in sockets or soldered directly to the PCB. Now fit the pin-headers to the appropriate holes on the PCB. If you prefer not to use pin- headers and connectors, you may solder the wires of the external components directly to the PCB.

Now we are ready to start attaching the external components. These include the laser pointer, photo-transistor, battery connector, switch and the DB-9 connector. Take particular care with the orientation of the photo-transistor when clipping the pins and soldering wires to it.

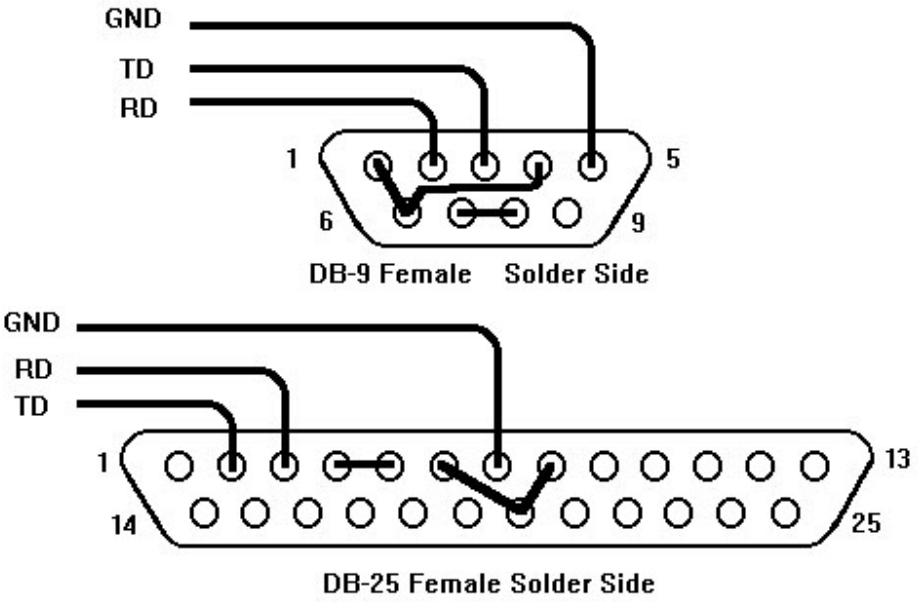
Then prepare the serial connector. We may use a standard (female) DB-9 or DB-25 connector depending on your needs. the connections for the DB-9 connector as this is found on most IBM-PCs. The IBM PC serial port contains several data and handshake lines. We will only be using the Transmit Data (TD), Receive Data (RD) and common ground (GND) lines. Handshaking will be done in software. In order to make the serial port happy we need to connect the Data Terminal Ready (DTR) line to the Data Set Ready (DSR) and Data Carrier Detect (DCD) lines. We also need to connect the Request To Send (RTS) line to the Clear To Send (CTS) line. This has the effect of tricking the serial port into thinking that it is always ready to receive and send data. These links should be soldered inside the connector itself. Only 3 wires are required for the connection to the transceiver. Connect the three wires to the RD (pin 2), TD (pin 3) and GND (pin 5) lines of the connector.

Next connect the black wire of the 9V battery clip to the PCB and the red wire to one contact on the switch. The other switch contact should then be wired to the PCB. You can use light duty hook-up wire to achieve this.

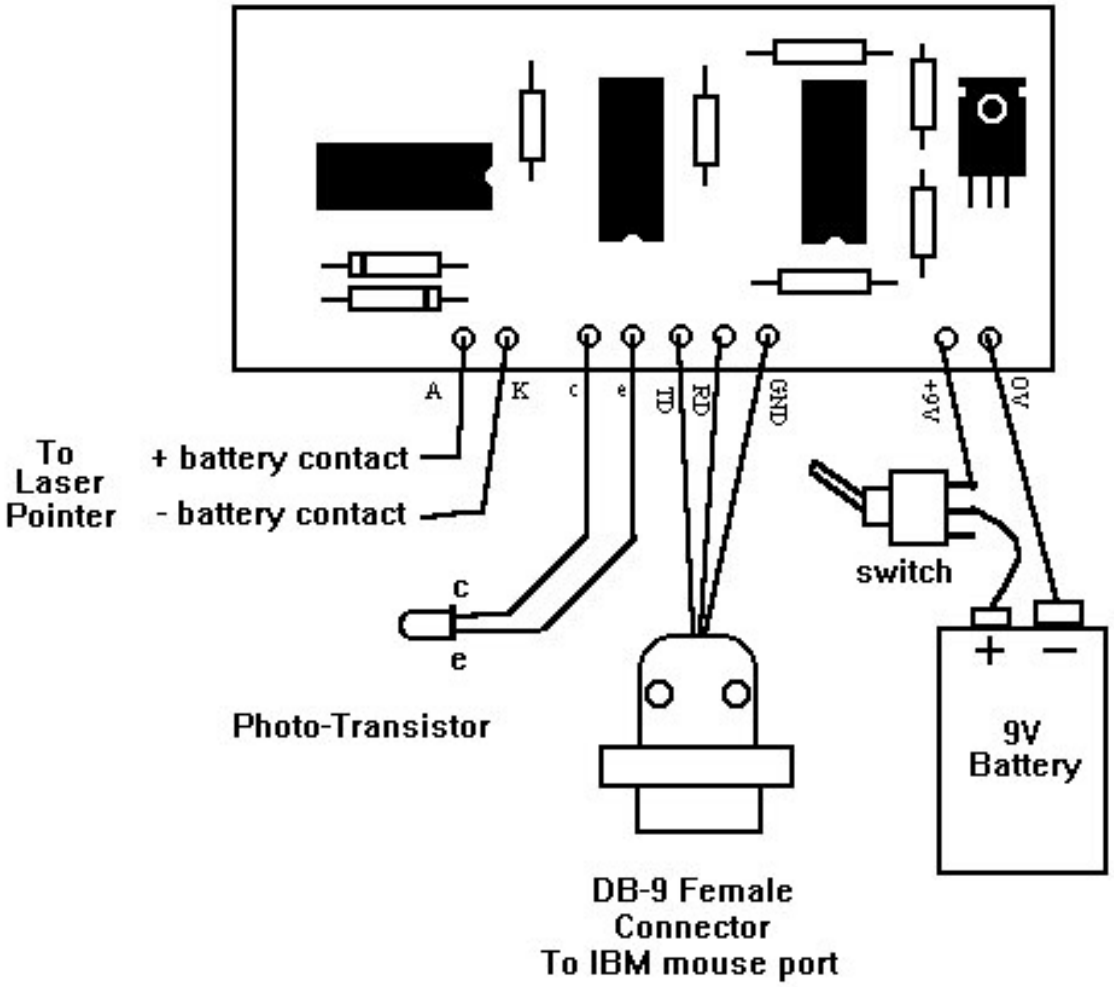
The last, and most interesting, component needs to be wired to the circuit - The Laser. We must first prepare the laser pointer since almost none have wires already hanging from them. The preparation will vary on the laser pointer you have, but most should have access to the battery compartment. The most suitable laser pointers are the ones that require 2 AAA or 2 AA batteries. First remove the batteries carefully noting the polarity of the contacts. You now need to connect a wire to each battery

contact. Depending on the laser pointer case you may need to create a conductive *dummy battery* in order to reach the contacts. For the laser pointer used in this design, in order to reach the negative contact deep inside the case, a metal rod was cut to the length of both batteries and a wire was soldered to the end of it. The rod can be made from a bolt or large diameter nail with the head cut off. The rod was wrapped in electrical tape to insulate it from the aluminum case which formed the other contact. An exposed wire was taped to the insulation of the rod so that it made contact with the case when the rod was inserted. This made the positive contact.

One more step and we are ready to use the laser. Usually most laser pointers have a push- on switch to turn on the laser. This switch can simply be taped down with electrical tape to hold it closed. The laser pointer is now ready for use. Again it is very important that you get the correct orientation of the wires from the laser pointer when connecting them to the PCB.



This solder side view shows how to wire the DB-9 (female) or DB-25 (female) connector for an IBM PC compatible computer



Connections to the external components are shown here for both the transceiver and the transmitter.

Testing

You will need a PC to test the circuit. The program listing at the end of this article gives an example of a test communications program. You will need a C compiler to compile it. The code was compiled using Borland C++ 3.1.

To test the circuit, plug the DB-9 connector into the mouse port on your PC. Turn on power to the circuit and the laser should switch on. Now turn on the PC and make sure a mouse driver is not loaded. A TSR(Terminate and Stay Resident) mouse driver will interfere with the operation of the circuit. Also make sure that no other TSRs are attempting to use the serial port. Now point the pointer directly at the photo-transistor. Next run the test program from a DOS prompt by typing LASER 1 and pressing the key. Where 1 represents the COM port number the circuit is connected to. Anything you type on the keyboard should appear at the top of the screen as well as the bottom. The top part of your screen displays the data sent out over the laser pointer while the bottom part shows the received data. Press the ESC key anytime to end the program.

To test communication between two computers simply repeat the steps above for each computer except that the lasers are pointed towards the other transceiver. Also depending on the laser pointer, beam intensity and beam spread will vary which will affect the distance over which reliable communication can be achieved. Most laser pointers should achieve a minimum of 100 meters. And if all goes well you will be sending data over a laser beam!

Chapter

3

Parts Listing:

Component	Value	Description
R1	1k	1/4 W resistor
C1-5	0.1uF	Capacitor (Ceramic)
U1	MAX232A	RS-232 line driver
U2	74LS05	Hex open-collector buffer
U3	74LS14	Schmitt trigger hex inverter
D1-2	1N4001	Power diode
P1	OP505A	Photo-transistor
V1	7805	Voltage regulator
L1	LX1000	Laser Pointer
.	.	9V battery clip
.	.	DB-9 female connector with backshell.
.	.	2 m shielded 3 core cable.
.	.	Switch
.	.	PCB
.	.	Light duty hook-up wire

Program Code:

```
/*************************************************
**
**        Laser Pointer RS-232 Transceiver
**         Data Transmitted @ 9600 bps
**
*************************************************/

#include <dos.h>
#include <stdlib.h>
#include <conio.h>
#include <bios.h>
#include <stdio.h>

#define COM1       0
#define COM2       1
#define DATA_READY 0x100
#define TRUE       1
#define FALSE      0
#define ESC_KEY    '\x1b'

#define SETTINGS ( _COM_9600 | _COM_CHR8 | _COM_NOPARITY | _COM_STOP1)

void clear_line(int line)
{
    int i;

    gotoxy(1,line);     // clear a whole line
    for(i=0;i<80;i++)
        printf(" ");
}

int main(int argc, char *argv[])
{
   int in, out, status, done = FALSE;
   int curs_rx=0,curs_ry=15,curs_tx=0,curs_ty=4;
   int com_port=COM1;

   if (!(argc == 2 && (argv[1][0] == '2' || argv[1][0] == '1')))
   {
      printf("Usage: LASER [1|2]\nwhere 1 = Com port 1\n      2 = Com port 2\n");
      exit(-1);
   }

   if (argv[1][0]=='2') // select com port
      com_port = COM2;
   else
      com_port = COM1;

   bioscom(_COM_INIT, SETTINGS, com_port);    // Initialize serial port

   clrscr();
   printf("                        Laser Transceiver Communicator V1.1\n");
   printf("                          Press [ESC] to exit program\n");
   printf("_____________________________________ Sent Data
_____________________________________");
   gotoxy(1,13);
```

```
   printf("_____ Recieved Data
_____");

   while (!done) {
      status = bioscom(_COM_STATUS, 0, com_port);   // recieved data?
      if (status & DATA_READY)
         if ((out = bioscom(_COM_RECEIVE, 0, com_port) & 0x7F) != 0)  {  // get data
            if (curs_rx < 78)         // move cursor
               curs_rx++;
            else {
                curs_rx = 1;          // at end of line
                if (curs_ry < 23)
                    curs_ry++;
                else
                    curs_ry = 15;

                clear_line(curs_ry);
            }
            gotoxy(curs_rx,curs_ry);  // goto correct screen location
            putch(out);               // print the character
         }

      if (kbhit())  {
          if ((in = getch()) == ESC_KEY) // check for ESC key
             done = TRUE;

          if(!in)                   // read an extended character
              in = getch();

          if (curs_tx < 78)         // position cursor
              curs_tx++;
          else   {
              curs_tx = 1;          // at end of line
              if (curs_ty < 12)
                  curs_ty++;
              else
                  curs_ty = 4;

              clear_line(curs_ty);
          }
          gotoxy(curs_tx,curs_ty);  // goto correct screen location
          putch(in);                // print the character
          bioscom(_COM_SEND, in, com_port); // output data
      }
   }
   clrscr();
   return (0);
}
```