Port Interfacing In Java Through Microcontroller

MUHAMMAD AJMAL P.

A serial port interfacing circuit built around AT89C51 microcontroller is presented here for testing and understanding a serial port emulator. It can be used to control a USB port also by using a serial to USB converter.

This project can be used to control a device through the serial COM port or USB port of a PC. You can send data to the device from the PC and then receive it back on the PC through the serial port. Here we describe control of eight LEDs from the PC. The LEDs are connected across the output port of the microcontroller. The user interface program is written in Java.

Circuit description

Port P1 (refer Fig. 1) of the microcontroller is the input port and P0 the output port. The input is given through input switches S1 through S8. The corresponding output is indicated on the LEDs (LED1 through LED8). The microcontroller is synchronised with the software at a speed of 9.6 kbps. For this, a crystal of 11.0592 MHz frequency is used. This value can be set at either the AT89C51 or by the software easily and accurately.

The microcontroller has TTL input/output (I/O) logic signal level, whereas a computer has RS232 I/O. The TTL logic works with 0V as 'low' and +5V as 'high'. In RS232 protocol, 'high' is represented by the range -3V to -25V and 'low' by +3V to +25V. The range -3V to +3V is undefined.

So the output signals available through RS232 from the PC are not compatible with the signal requirements of AT89C51. Therefore we have used a MAX232, an RS232-to-TTL converter. Thus serial data communication can be done efficiently.

Power supply

The power supply section is shown in Fig. 2. The circuit is powered by 5V

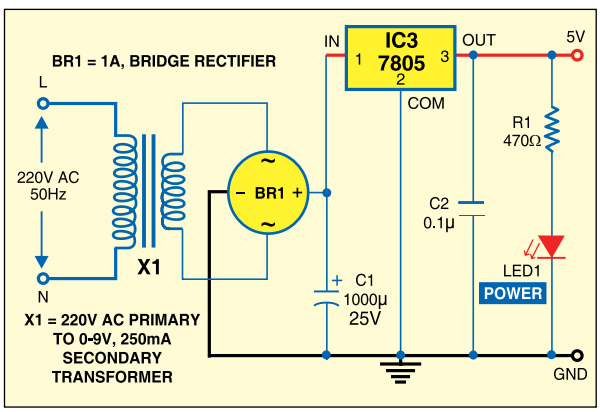

Fig. 2: Power supply circuit

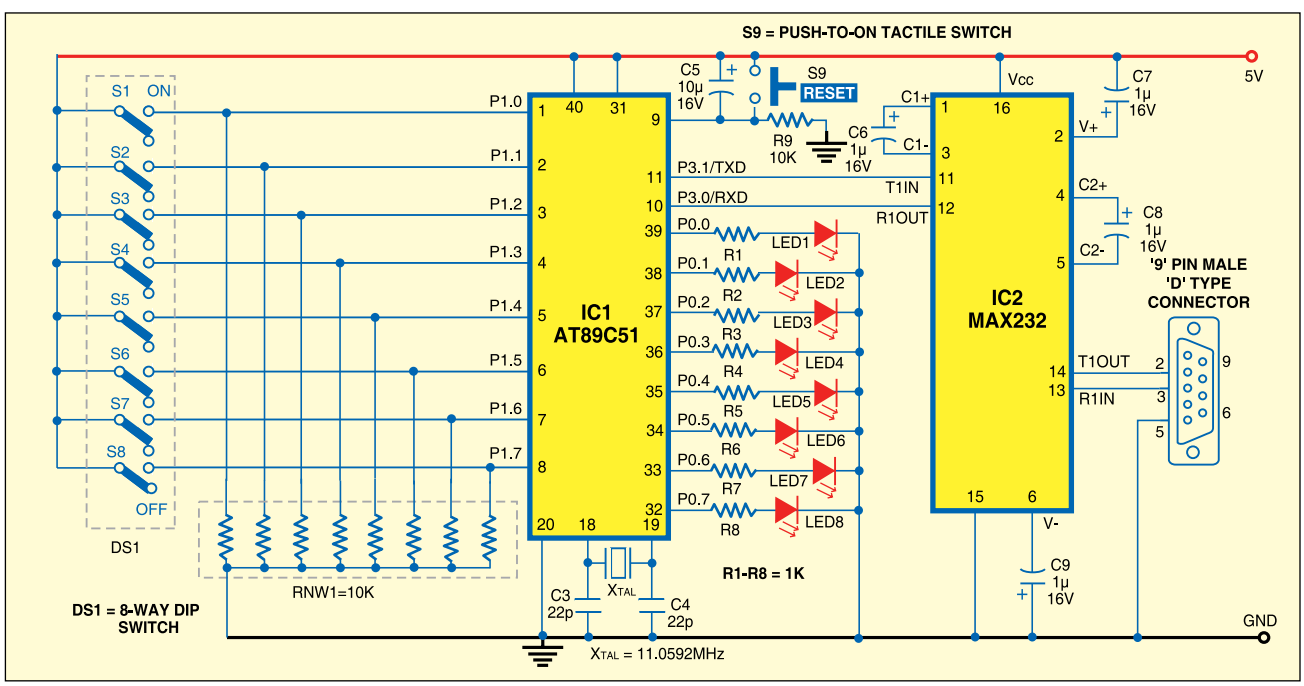

Fig. 1: Circuit for serial port interfacing with LEDs in Java through microcontroller

PARTS LIST

Semiconductors:

IC1	- AT89C51 microcontroller
IC2	- MAX232 driver
IC3	- 7805, 5V regulator
BR1	- 1A bridge rectifier
LED1-LED8	- 3mm light-emitting diode
LED9	- 5mm light-emitting diode

Resistors (all ¼-watt, ±5% carbon):

R1-R8	- 1-kilo-ohm
R9	- 10-kilo-ohm
R10	- 470-ohm
RNW1	- 10-kilo-ohm resistor network

Capacitors:

C1, C2	- 22pF ceramic
C3	- 10µF, 16V electrolytic
C4	- 1000µF, 25V electrolytic
C5	- 0.1µF ceramic
C6-C9	- 1µF, 16V electrolytic

Miscellaneous:

X1	- 220V AC primary to 9V, 250mA secondary transformer
X_{TAL}	- 11.0592MHz crystal
S1-S8	- 8-way DIP switch
S9	- Tactile switch
	- 9-pin D-type COM port connector

DC using a 7805 voltage regulator. To derive the power supply, the 220V, 50Hz AC mains is stepped down by transformer X1 to deliver a secondary output of 9V, 250 mA. The transformer output is rectified by bridge rectifier BR1, filtered by capacitor C1 and regulated by IC 7805 (IC3). Capacitor C2 bypasses the ripples from the regulated supply. LED1 acts as the power-'on' indicator. Resistor R1 limits the current through LED1.

Software

Java is a platform-independent language. So this software can be used in any operating system with the corresponding Java Runtime installed. The project contains Java source codes, Java classes, SIIJ_setup file and firmware.

We have used JCreator for developing the software. To use JCreator, you need JDK 1.6 or higher version installed in your system. The serial port is accessed by the Java program using third-party packages such as 'rxtxSerial.dll' and 'gnu/io'. These files can be downloaded from the Internet.

All Java source files are in 'src' folder. Compile these using a Java editor such as JCreator ve4.5 Pro. Compile Java files and generate a jar file in the folder.

On successful compilation of Java program, ensure that the following files are in 'Classes' folder:

1. 'Gnu' folder with 52 files
2. 'Pic' folder having two sub-folders with three files in the first folder and seven files in the second folder
3. 'SIIJ' folder with nine files
4. rxtxSerial.dll file of 75.9kB size
5. Manifest file of 71 bytes containing the text: "Main-Class: SIIJ.classes.SIIJ. SplashScreen-Image: pic/EG_Icon/EG.png"

Ensure that in the Manifest file there is a hard return (blank) at the end. That is, there should be three lines present with the last line blank.

The software for this project includes nine class files and a package for controlling the serial port. The main class file is SIIJ.class. This file creates the user-interface window for controlling the serial port. The main Java file, SIIJ.java, is listed at the end of this article for reference.

The MenubarCreator.java file creates the menu bar and adds it to the window screen. The TrayCreator.java file creates an icon in the tray so that the program stays in the tray (refer Fig. 3) until you exit it. Clicking 'Close' button will not exit the program unless you click 'Exit' button in File menu.

The SendData.java file includes the code for controlling the serial port. This file uses the package 'gnu.io.*;' that includes the class files for controlling the serial port. Here the code at the end of the SendData.java file is the standard code for initialising the serial port.

Fig. 3: Tray icon

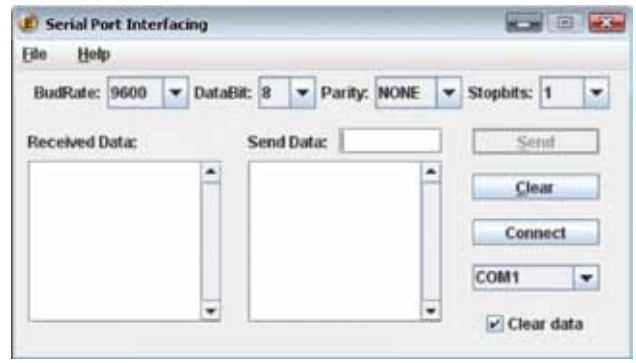

Fig. 4: Screenshot of Java program output

The getPortIds( ) method returns the available ports. The SimpleWrite( ) method sends the data to the hardware via the corresponding serial port. The 'public void recValue(int tempprecval)' method is called whenever serial data is received by the corresponding port. When data is received, the work to be done should be included in this function.

Third-party packages are included in 'Classes' folder for easy compilation.

It is necessary that the rxtxSerial.dll file is present in the same directory as 'Serial Interfacing in Java.jar' file. Or else, the .dll file should be put in 'java/bin' directory in Microsoft Windows. For working in Linux or Macintosh, you need other files, which are not described here.

SIIJ_setup file. This is the installer file for Windows operating system. You can run this file and choose a directory for installation. On successful installation, 'Serial Port Interfacing' folder is created, which includes files 'Serial Interfacing In Java.jar' and 'rxtxSerial.dll'. If you are using Windows 7 OS, some folders such as 'Program Files' need additional write permission. So install the file in another location such as the desktop. Ensure that you get a successful installation message.

'Serial Interfacing In Java.jar' is the final executable file. With JRE 1.6+ you can run jar file on Windows. The rxtxSerial.dll file is needed along with the executable file.

The Java software allows one to set the baud rate, data bit, etc. The screenshot of the program output is shown in Fig. 4.

Firmware

The program for the microcontroller is written in Assembly language, in serial interrupt mode. It is a simple program to test and study the working of serial COM port of a PC, and serially interface it with a microcontroller. It has the code for sending serial data and displaying the received signal. The Assembly program code (SIIJ.asm) is listed at the bottom of this page.

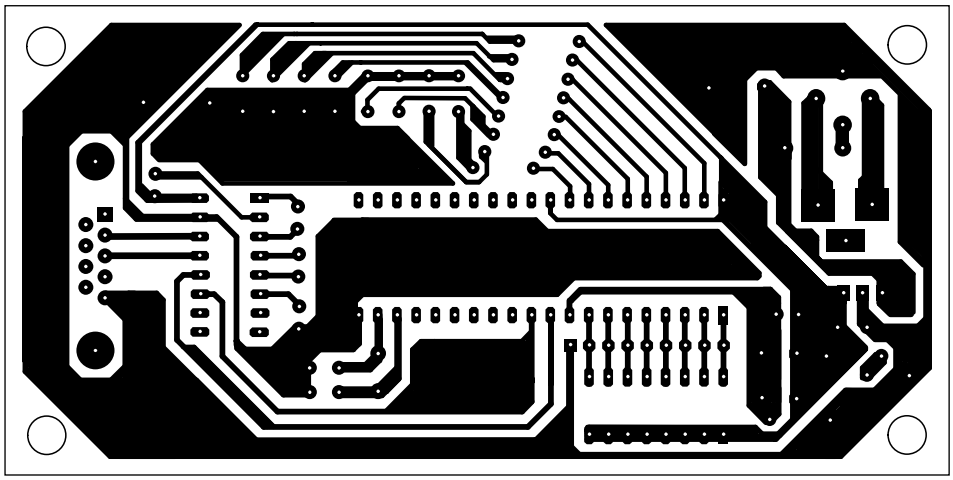

Fig. 5: An actual-size, single-side PCB layout of the circuit for serial port interfacing with LEDs in Java

Construction and testing

An actual-size, single-side PCB layout of the circuit for serial port interfacing with LEDs in Java is shown in Fig. 5 and its component layout in Fig. 6.

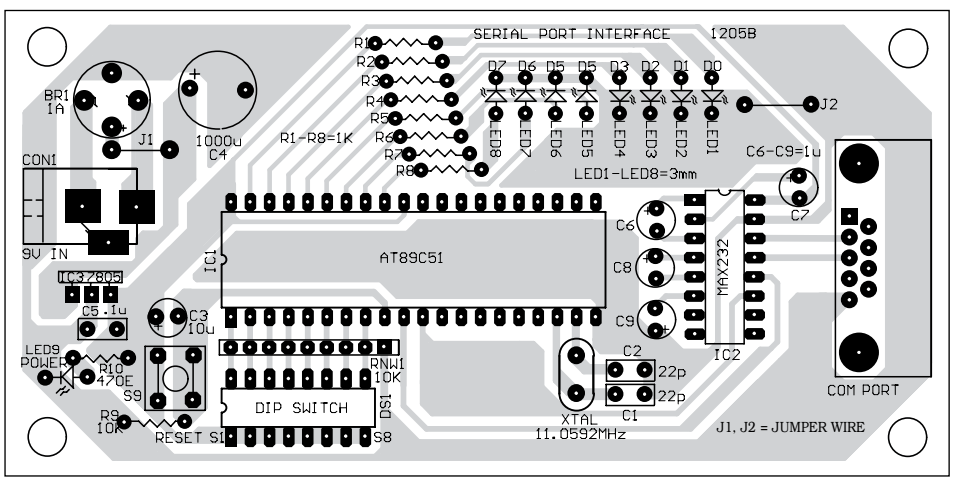

Fig. 6: Component layout for the PCB

Step-by-step procedure for testing the project follows:

1. Compile the Java codes using JCreator.

2. Compile the Assembly code using a suitable software such as ASM51 cross-assembler. Burn the hex code into the microcontroller chip and then place the chip in the circuit.

3. Set port P1 of microcontroller to FF (or 255) by pulling all the switches S1 through S8 to +5V. This will enable handshaking.

4. Run the Java program. Select COM port. Here COM1 port is used (refer Fig. 4).

5. Set baud rate to 9600, data bit to 8, parity bit to none and stop bit to 1.

6. Now type a value between 0 and 255 in 'Send Data' field.

7. Click 'Connect' button. After sending data for the first time, the software waits for handshaking signal FF (Hex) from the hardware device. If port P1 is set to FF (Hex), the handshaking will be successful. The data sent from the hardware device will be displayed in 'Received Data' text box on the screen. In this case, FFH is displayed. Any value between 0 and 255 entered in the software is displayed in 'Received Data' text box and also on the LEDs (LED1 through LED8). Similarly, changes made in the input switches should be displayed in 'Received Data' box as well as on the LEDs.

EFY note. 1. The rxtxSerial.dll file can be downloaded from:

http://rapidlibrary.com/files/rxtxserial-dll_ulz99nee8ci89on.html

2. The download links for gnu/io folder are:

www.jcontrol.org/download/files/rxtx-2.1-7-bins-r2.zip

http://www.jcontrol.org/download/rxtx_en.html

3. The source codes of the project and related files are available at www.efymag.com website.

The author is a software developer at Software Associates, Kozhikode (Calicut)

SIIJ.ASM

```
            ORG 00H
            SJMP MAIN
            ORG 23H
            LJMP SER_INT
            ORG 30H
MAIN:       MOV P3,#0FFH
            MOV P2,#00H
            MOV P1,#0FFH
            MOV P0,#00H
            LCALL SER_INIT
BEGIN:      MOV A,P1
            MOV SBUF,A
            SJMP BEGIN
SER_INIT:   MOV TMOD,#20H
            MOV TH1,#0FDH
            MOV SCON,#50H
            MOV IE,#90H
            SETB IP.4
            SETB TR1
            RET
SER_INT:    JB TI,TRANS
            PUSH 0E0H
            MOV A,SBUF
            MOV P0,A
            CLR RI
            POP 0E0H
            RETI
TRANS:      CLR TI
            RETI
            END
```

SIIJ.JAVA

```
/*  Serial Interfacing In Java application */
package SIIJ.classes;
import javax.swing.JFrame;
import javax.swing.JTextArea;
import javax.swing.JTextField;
import javax.swing.JButton;
import javax.swing.JCheckBox;
import javax.swing.JComboBox;
import javax.swing.JPanel;
import javax.swing.JLabel;
import javax.swing.JScrollPane;
import javax.swing.JOptionPane;
import javax.swing.ImageIcon;
import java.awt.Dimension;
import java.awt.Toolkit;
import java.awt.TrayIcon;
import java.awt.BorderLayout;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;
import java.awt.event.ItemEvent;
import java.awt.event.ItemListener;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;
import javax.swing.JToggleButton;

public class SIIJ implements ActionListener {
    public JFrame frame;
    private JTextArea rectext,sndtext;
    private JTextField sndfld;
    private TrayCreator tray;
    private JButton sndbut,clrbut;
    private JToggleButton connectbut;
    private JCheckBox clrchk;
    private JComboBox serialport,budrate,databit,parity,stopbit;
    private SendData sdata;
    private ClassLoader cl;
  public SIIJ() {
    String[] budval = { "110","300","600","1200","2400","4800","9600","14400","19200","28800","38400","56000" };
    String[] dataval = {"5   ","6   ","7   ","8   "};
    String[] parityval = {"NONE","ODD","EVEN","MARK","SPACE"};
    String[] stopval = {"1     ","2   ","1.5   "};
    frame = new JFrame("Serial Port Interfacing");
    rectext = new JTextArea();
    sndtext = new JTextArea();
    sndfld = new JTextField();
    sndbut = new JButton("Send");
    clrbut = new JButton("Clear");
    connectbut = new JToggleButton("Connect");
    clrchk = new JCheckBox("Clear data");
    sdata = new SendData(frame,rectext,sndtext);
    serialport = new JComboBox(sdata.getPortIds());
    budrate = new JComboBox(budval);
    databit = new JComboBox(dataval);
    parity = new JComboBox(parityval);
    stopbit = new JComboBox(stopval);
    sdata.setPortId((String) serialport.getSelectedItem());
    new MenuBarCreator(frame);
    cl = getClass().getClassLoader();
    tray = new TrayCreator(frame,"Serial Port Interfacing");
    go();
  }
  private void go() {
    JPanel frmpan = new JPanel();
    JPanel serialpan = new JPanel();
    JScrollPane recscrl = new JScrollPane(rectext);
    JScrollPane sndscrl = new JScrollPane(sndtext);
    JLabel reclab = new JLabel("Received Data:");
    JLabel sndlab = new JLabel("Send Data:");
    JLabel budlab = new JLabel("BudRate:");
    JLabel datalab = new JLabel("DataBit:");
    JLabel paritylab = new JLabel("Parity:");
    JLabel stoplab = new JLabel("Stopbits:");
    Dimension scrn = Toolkit.getDefaultToolkit().getScreenSize();
    frame.setBounds(scrn.width/2-200,scrn.height/2-130,480,275);
    frame.setIconImage(new ImageIcon(cl.getResource("pic/EG_Icon/EG_Icon.jpg")).getImage());
    frame.setDefaultCloseOperation(JFrame.DO_NOTHING_ON_CLOSE);
    frame.setResizable(false);
    frame.addWindowListener(new WindowAdapter() {
      public void windowClosing(WindowEvent e) {
        frame.setVisible(false);
        tray.getTrayIcon().displayMessage("Serial Port Interfacing","Program is running",TrayIcon.MessageType.INFO);
      }
    });
    frame.getRootPane().setDefaultButton(sndbut);
    recscrl.setHorizontalScrollBarPolicy(JScrollPane.HORIZONTAL_SCROLLBAR_NEVER);
    recscrl.setVerticalScrollBarPolicy(JScrollPane.VERTICAL_SCROLLBAR_ALWAYS);
    sndscrl.setHorizontalScrollBarPolicy(JScrollPane.HORIZONTAL_SCROLLBAR_NEVER);  sndscrl.setVerticalScrollBarPolicy(JScrollPane.VERTICAL_SCROLLBAR_ALWAYS);
    rectext.setEditable(false);
    sndtext.setEditable(false);
    clrchk.setSelected(true);
    sndbut.setEnabled(false);
    budrate.setMaximumRowCount(5);
    budrate.setSelectedIndex(6);
    databit.setSelectedIndex(3);
    reclab.setBounds(10,10,85,20);
    sndlab.setBounds(180,10,70,20);
    recscrl.setBounds(10,35,150,125);
    sndscrl.setBounds(180,35,150,125);
    sndfld.setBounds(250,10,80,20);
    sndbut.setBounds(353,10,98,20);
    clrbut.setBounds(353,45,98,20);
    connectbut.setBounds(353,80,98,20);
    serialport.setBounds(353,115,98,20);
    clrchk.setBounds(360,150,90,20);
    sndbut.setMnemonic('S');
    clrbut.setMnemonic('C');
    frmpan.setLayout(null);
    frmpan.add(reclab);
    frmpan.add(recscrl);
    frmpan.add(sndscrl);
    frmpan.add(sndlab);
    frmpan.add(sndfld);
    frmpan.add(sndbut);
    frmpan.add(clrbut);
    frmpan.add(connectbut);
    frmpan.add(serialport);
    frmpan.add(clrchk);
    serialpan.add(budlab);
    serialpan.add(budrate);
    serialpan.add(datalab);
    serialpan.add(databit);
    serialpan.add(paritylab);
    serialpan.add(parity);
    serialpan.add(stoplab);
    serialpan.add(stopbit);
    sndbut.addActionListener(this);
    clrbut.addActionListener(this);
    connectbut.addActionListener(this);
    serialport.addItemListener(new comboboxListener());
    frame.getContentPane().add(frmpan,BorderLayout.CENTER);
    frame.getContentPane().add(serialpan,BorderLayout.NORTH);
    frame.getContentPane().add(new JLabel(" Developed by Muhammad Ajmal P"),BorderLayout.SOUTH);
    frame.setVisible(true);
    sndfld.requestFocus();
  }
  public static void main(String[] args) {
    SIIJ siij = new SIIJ();
  }
  private void setComboBox(boolean chs) {
    budrate.setEnabled(chs);
    databit.setEnabled(chs);
    stopbit.setEnabled(chs);
    parity.setEnabled(chs);
    serialport.setEnabled(chs);
  }
  public void actionPerformed(ActionEvent ae) {
    if(ae.getSource() == sndbut) try {
      int recvalue = Integer.parseInt(sndfld.getText());
      if(recvalue >= 0 && recvalue <= 255) sdata.sendValue(recvalue);
      else JOptionPane.showMessageDialog(frame,"Invalied Input\nEnter a value between 0 - 255","Error",JOptionPane.ERROR_MESSAGE,new ImageIcon(cl.getResource("pic/SIIJ/err.png")));
    } catch(Exception e) {
      JOptionPane.showMessageDialog(frame,"Invalied Input\nEnter a value between 0 - 255","Error",JOptionPane.ERROR_MESSAGE,new ImageIcon(cl.getResource("pic/SIIJ/err.png")));
    }
    finally {
      if(clrchk.isSelected()) sndfld.setText("");
    }
    else if(ae.getSource() == clrbut) {
      rectext.setText("");
      sndtext.setText("");
    }
    else if(ae.getSource() == connectbut) {
      if(connectbut.isSelected()) {
        sdata.setPortData(budrate.getSelectedItem().toString()
        ,databit.getSelectedItem().toString(),parity.getSelectedIndex()
        ,stopbit.getSelectedIndex());
        connectbut.setText("Disconnect");
        sdata.setConnection(true);
        setComboBox(false);
        sndbut.setEnabled(true);
      }
      else {
        connectbut.setText("Connect");
        sdata.setConnection(false);
        setComboBox(true);
        sndbut.setEnabled(false);
      }
    }
    sndfld.requestFocus();
  }
class comboboxListener implements ItemListener {
  public void itemStateChanged(ItemEvent ie) {
    if(ie.getSource() == serialport && ie.getStateChange() == ItemEvent.SELECTED) {
      sdata.setPortId(serialport.getSelectedItem().toString());
      System.out.println ("COM port changed - "+serialport.getSelectedItem().toString());
    }
  }
}
}
```