

```

/*
 * Created on Jul 7, 2004
 *
 */
package pony;
import javax.comm.*;

import java.io.*;

/**
 * @author rocha
 *
 */
public class SerialControl implements SerialPortEventListener {

    static CommPortIdentifier portId;
    static final int TIMEOUT = 100; /* (1/100) seconds */
    static final int SPACE_SIZE = 6;
    static final String PORT = "COM1";

    SerialPort serialPort;
    OutputStream outputStream;
    InputStream inputStream;

    boolean wait = true;

    SerialControl() {
        try {
            portId = CommPortIdentifier.getPortIdentifier(PORT);
            serialPort = (SerialPort) portId.open("SerialControl", 1000);

            serialPort.setSerialPortParams(
                19200,
                SerialPort.DATABITS_8,
                SerialPort.STOPBITS_1,
                SerialPort.PARITY_NONE);

            serialPort.setDTR(false);
            serialPort.setRTS(false);

            outputStream = serialPort.getOutputStream();
            inputStream = serialPort.getInputStream();

            serialPort.addEventListener(this);
            serialPort.notifyOnDataAvailable(true);

        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    public void serialEvent(SerialPortEvent event) {
        switch (event.getEventType()) {
            case SerialPortEvent.BI :
            case SerialPortEvent.OE :
            case SerialPortEvent.FE :
            case SerialPortEvent.PE :
            case SerialPortEvent.CD :
            case SerialPortEvent.CTS :
            case SerialPortEvent.DSR :
            case SerialPortEvent.RI :
            case SerialPortEvent.OUTPUT_BUFFER_EMPTY :
                break;
            case SerialPortEvent.DATA_AVAILABLE :
                StringBuffer readBuffer = new StringBuffer();
                try {
                    while (inputStream.available() > 0) {
                        int c = inputStream.read();
                        //System.out.println("SERIAL got: " + c);
                        if (c == 13) {

```

```

        tick();
    }
} catch (IOException e) {
}

break;
}
}

public void tick() {
    wait = false;
}

public void waitResponse() {
    //System.out.println("waiting response...");
    int c = 0;
    while (wait) {
        try {
            Thread.sleep(100);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }

        if (c > TIMEOUT) {
            //System.out.println("timeout");
            wait = false;
        }

        c++;
    }
    wait = true;
    try {
        Thread.sleep(500);
    } catch (InterruptedException e) {
        e.printStackTrace();
    }
    return;
}

public void writeString(String string) {
    string = string.toLowerCase();
    int length = string.length();
    for (int i = 0; i < length; i++) {
        char c = string.charAt(i);
        writeChar(c);
    }
}

public void writeChar(char c) {
    byte[] column = new byte[10];

    int count = makeLetter(c, column);

    for (int i = 0; i < count; i++) {
        writeByte(column[i]);
        waitResponse();
    }
    writeByte((byte) 0);
    waitResponse();
}

public void writeByte(byte b) {
    try {
        outputStream.write(b);
        outputStream.flush();
    } catch (IOException e) {
        e.printStackTrace();
    }
}
}

```

```

public int makeLetter(char c, byte[] column) {
    int[] value = new int[10];
    int count = 0;

    switch (c) {

        case 'a' :
            {
                value[0] = 1000;
                value[1] = 111000;
                value[2] = 11010000;
                value[3] = 111000;
                value[4] = 1000;
                count = 5;
                break;
            }
        case 'b' :
            {
                value[0] = 10001000;
                value[1] = 11111000;
                value[2] = 10101000;
                value[3] = 10101000;
                value[4] = 1010000;
                count = 5;
                break;
            }
        case 'c' :
            {
                value[0] = 1110000;
                value[1] = 10001000;
                value[2] = 10001000;
                value[3] = 11010000;
                count = 4;
                break;
            }
        case 'd' :
            {
                value[0] = 10001000;
                value[1] = 11111000;
                value[2] = 10001000;
                value[3] = 10001000;
                value[4] = 1110000;
                count = 5;
                break;
            }
        case 'e' :
            {
                value[0] = 10001000;
                value[1] = 11111000;
                value[2] = 10101000;
                value[3] = 10001000;
                value[4] = 11011000;
                count = 5;
                break;
            }
        case 'f' :
            {
                value[0] = 10001000;
                value[1] = 11111000;
                value[2] = 10101000;
                value[3] = 10000000;
                value[4] = 11000000;
                count = 5;
                break;
            }
        case 'g' :
            {
                value[0] = 1110000;
                value[1] = 10001000;
                value[2] = 10101000;
            }
    }
}

```

```

        value[3] = 10111000;
        count = 4;
        break;
    }
case 'h' :
{
    value[0] = 10001000;
    value[1] = 11111000;
    value[2] = 10101000;
    value[3] = 100000;
    value[4] = 11111000;
    value[5] = 10001000;
    count = 6;
    break;
}
case 'i' :
{
    value[0] = 10001000;
    value[1] = 11111000;
    value[2] = 10001000;
    count = 3;
    break;
}
case 'j' :
{
    value[0] = 1000;
    value[1] = 10001000;
    value[2] = 11110000;
    value[3] = 10000000;
    count = 4;
    break;
}
case 'k' :
{
    value[0] = 10001000;
    value[1] = 11111000;
    value[2] = 10101000;
    value[3] = 100000;
    value[4] = 11011000;
    value[5] = 10001000;
    count = 6;
    break;
}
case 'l' :
{
    value[0] = 10001000;
    value[1] = 11111000;
    value[2] = 10001000;
    value[3] = 1000;
    value[4] = 11000;
    count = 5;
    break;
}
case 'm' :
{
    value[0] = 10001000;
    value[1] = 11111000;
    value[2] = 1001000;
    value[3] = 100000;
    value[4] = 1001000;
    value[5] = 11111000;
    value[6] = 10001000;
    count = 7;
    break;
}
case 'n' :
{
    value[0] = 10001000;
    value[1] = 11111000;
    value[2] = 1001000;
    value[3] = 100000;

```

```

        value[4] = 10010000;
        value[5] = 11111000;
        value[6] = 10000000;
        count = 7;
        break;
    }
case 'o' :
    {
        value[0] = 1110000;
        value[1] = 10001000;
        value[2] = 10001000;
        value[3] = 1110000;
        count = 4;
        break;
    }
case 'p' :
    {
        value[0] = 10001000;
        value[1] = 11111000;
        value[2] = 10101000;
        value[3] = 10100000;
        value[4] = 1000000;
        count = 5;
        break;
    }
case 'q' :
    {
        value[0] = 1110000;
        value[1] = 10001000;
        value[2] = 10001000;
        value[3] = 1110000;
        count = 4;
        break;
    }
case 'r' :
    {
        value[0] = 10001000;
        value[1] = 11111000;
        value[2] = 10101000;
        value[3] = 10110000;
        value[4] = 1001000;
        value[5] = 1000;
        count = 6;
        break;
    }
case 's' :
    {
        value[0] = 1011000;
        value[1] = 10101000;
        value[2] = 10101000;
        value[3] = 11010000;
        count = 4;
        break;
    }
case 't' :
    {
        value[0] = 11000000;
        value[1] = 10001000;
        value[2] = 11111000;
        value[3] = 10001000;
        value[4] = 11000000;
        count = 5;
        break;
    }
case 'u' :
    {
        value[0] = 10000000;
        value[1] = 11110000;
        value[2] = 10001000;
        value[3] = 1000;
        value[4] = 11110000;
    }

```

```

        value[5] = 10000000;
        count = 6;
        break;
    }
    case 'v' :
    {
        value[0] = 10000000;
        value[1] = 11100000;
        value[2] = 11000;
        value[3] = 11100000;
        value[4] = 10000000;
        count = 5;
        break;
    }
    case 'w' :
    {
        value[0] = 10000000;
        value[1] = 11000000;
        value[2] = 111000;
        value[3] = 11000000;
        value[4] = 111000;
        value[5] = 11000000;
        value[6] = 10000000;
        count = 7;
        break;
    }
    case 'x' :
    {
        value[0] = 10001000;
        value[1] = 11011000;
        value[2] = 100000;
        value[3] = 11011000;
        value[4] = 10001000;
        count = 5;
        break;
    }
    case 'y' :
    {
        value[0] = 10000000;
        value[1] = 11001000;
        value[2] = 111000;
        value[3] = 11001000;
        value[4] = 10000000;
        count = 5;
        break;
    }
    case 'z' :
    {
        value[0] = 11011000;
        value[1] = 10101000;
        value[2] = 11001000;
        value[3] = 10011000;
        count = 4;
        break;
    }
    default :
    {
        for (int i = 0; i < 10; i++) {
            value[i] = 00000000;
        }
        count = SPACE_SIZE;
        break;
    }
}

for (int i = 0; i < count; i++) {
    column[i] = convert(value[i]);
}

return count;

```

```
}  
  
private byte convert(int val) {  
    int bin = 0x0000;  
    for (int i = 8; i >= 0; i--) {  
        int dec = (int) Math.pow(10, i);  
        if ((val / dec) > 0) {  
            bin = (bin << 1) | 1;  
        } else {  
            bin = (bin << 1) | 0;  
        }  
        val = val % dec;  
    }  
    return (byte) bin;  
}  
}
```