

SV TEHS SIA

Development Tools for Java™

IPVES Application Note 11:

PPP connection

SV TEHS SIA

IPJV-ES Application Note 11: PPP connection

V 1.0

© SV TEHS SIA

Ruses 14-24 • LV1029 • Riga • Latvia

Phone: +371-9237495 +371-9223895 • Fax: +371-7332773

Email: info@svtehs.com • Web: <http://www.svtehs.com>

IPJVM and IPJV-ES are trademarks of SV TEHS SIA. ipStack, ipOS are trademarks of Ubicom, Inc. Java™ and all Java™-based marks are trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S. and other countries. All other trademarks are property of their respective owners.



Introduction

IPJV-ES Development Board can be used in different applications.

The IPJV-ES Development Board with embedded virtual machine for Java™ offers an Ethernet based connection to the Internet and numerous interface possibilities to other equipment, include serial RS-232 DTE interface, serializer module with UART, SPI, GPSI and 10BASE-T Ethernet support, 6-channel 10-bit A/D inputs, analog comparator and 16 I/O pins.

The IPJVM virtual machine for Java is a clean room implementation, that has been specially optimized to run on device with limited amount of internal memory and designed for Java™ 2 Platform, Micro Edition (J2ME™) Connected Device Configuration (CDC) Foundation Profile.

A complete development toolkit available for application development with IPJVM platform. The IPJVM platform provide system designers and software developers simple, flexible and cost-effective solution for embedded Internet application rapid development and prototyping. The platform is combination of Uvicom IP2022 Internet Processor and a Java programmable runtime environment.

The IPJV-ES Development Board based on Uvicom IP2022 Internet Processor, optimized for Internet-edge applications. It handles protocol processing in software instead of in hard-wired logic, making the whole solution more adaptable to evolving standards and allow designer to use the same solution across a wide variety of internet-edge products simply by changing the software, thereby significantly reducing nonrecurring engineering (NRE) costs.

Typical IPJV-ES applications include Includes HTTP/FTP/SMTP/SNMP/Telnet servers, PPP support on embedded UARTs, encryption, security and authentication tools, reporting and alarming via e-mail, remote monitoring, control, management and maintenance.

Updates

New versions of the IPJV-ES software and applications can be obtained from the manufacturer's web site at:

<http://www.svtehs.com/ipjv.htm>

PPP connection

Serial bridge with PPP connection.

This application present serial bridge on two serial ports. You need to download firmware with two serial ports and PPP support to run this application. This firmware freely available from the www.svtchs.com/ipjv/ Please use IPJV User's Guide Chapter 6 for firmware upgrade instruction. You also need to use two serial ports on the IPJV.

RS232 DTE Connector

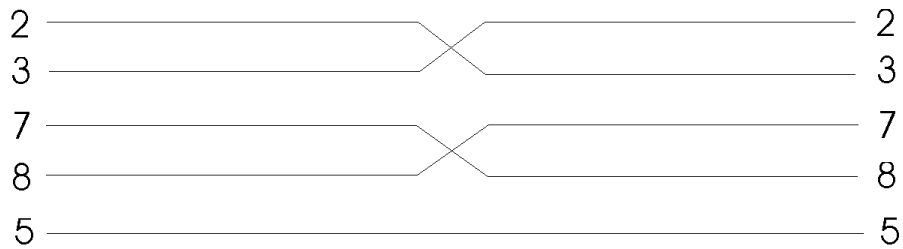
For most serial applications, IPJV-ES control some serial devices, like modems or communication equipment, or acts as network bridge, so it have build-in DTE serial port with hardware handshake (flow control) lines. Port have standard $\pm 10V$ input and output signal levels. RS232 connector have the following pinout:

Pin	Signal Name	DTE Sense	Description
2	RD (Receive Data)	Input	Data receive from DCE
3	TD (Transmit Data)	Output	Data transmit to DCE
5	Common		0 Volt reference
7	RTS	Output	Asserted by DTE to request permission to transmit data
8	CTS	Input	Asserted by DCE to grant permission to DTE to transmit data

For equipment with DCE serial port, straight-through serial cable with 9-pin male connector on one end and 9-pin female connector on other end should be used.

For connection to the equipment with DTE serial port, like computer or other IPJV-ES board, null-modem cable with 9-pin female connectors on both ends should be used. This cable should have at least the following connections:

PPP CONNECTION



Second UART use the following extension connector pins:

Pin	Signal Name	DTE Sense	Description
18	RD (Receive Data)	Input	Data receive from DCE
22	TD (Transmit Data)	Output	Data transmit to DCE
29,30	Common		0 Volt reference
21	RTS	Output	Asserted by DTE to request permission to transmit data
20	CTS	Input	Asserted by DCE to grant permission to DTE to transmit data

When using firmware with two UARTs, pull pin 18 (RD) to +5V with external 1K...10K pullup resistor, if the second UART is not actually using. 22 (TD) and 21 (RTS) pins have 0...3.3V output levels, 18 (RD) and 20 (CTS) pins allow 0...3.3V or 0...5V input levels. Use external level converter, for example MAX232 or compatible, if standard $\pm 10V$ signal levels is necessary.

Configuration file

All serial ports and PPP configuration parameters stored in `/etc/serial.conf` configuration file. This file created from `sercgi.java` CGI script, runned on the WWW server.

```
import java.util.*;
import java.io.*;
import com.decaf.*;
import jbvm.ip2k.*;
class sercgi extends getCGI
{
public sercgi(Properties Info,PrintStream Out) {
    super(Info,Out);
    outOK();
    PrintHd("Serial configuration");
    boolean isRoot=true;
    VMProperties ep=new VMProperties();
    File f=new File("/etc/serial.conf");
```

PPP CONNECTION

For each serial port the following parameters can be set: baudrate (1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200); stopbits (1,2); parity (N,E,O); bits (7,8); hardware flow control on the Rx (RTS) and on the Tx(CTS).

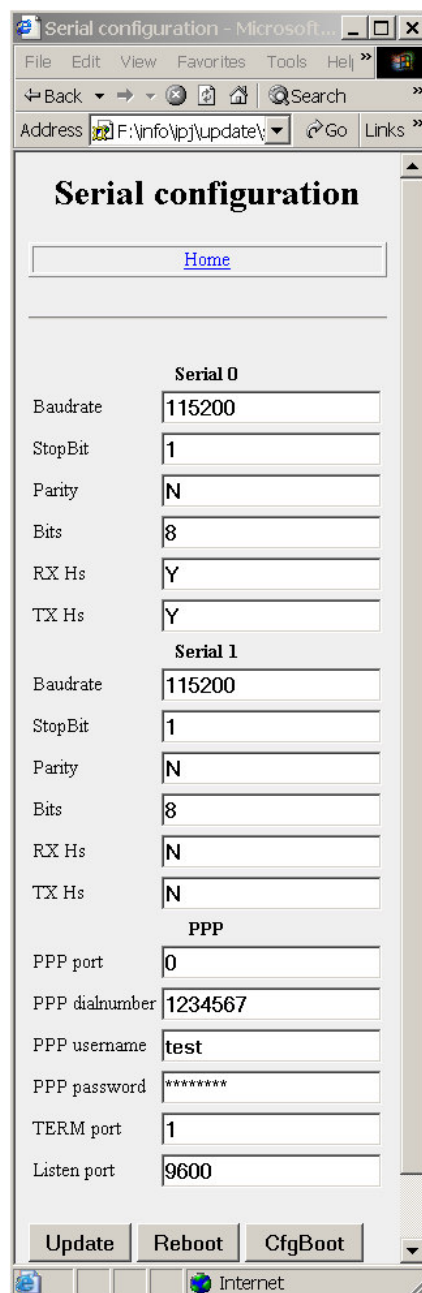
PPP can be switched to one of the serial ports – serial 0 (build-in UART) or serial 1 (second UART on the extension connector). PPP dialup script `DialScr.java` use the following connection parameters: port, user, password and dialup number.

Dialup script intended for simple general modem. Parameters and timeouts can be changed in real application.

```
int modemReset()
{
    int rc;
    String ok[]=new String[1]; ok[0]="OK";
    String err[]=new String[1]; err[0]="ERROR";
    doMdmCmd("+++", null, null, 2000);
    rc=doMdmCmd("ATZ\r", ok, err, 5000);
    if (rc<0) return rc;
    return 0;
}

int modemDial(String dnm)
{
    int rc;
    String ok[]=new String[1]; ok[0]="CONNECT";
    String err[]=new String[4]; err[0]="ERROR";
    err[1]="NO DIALTONE"; err[2]="NO CARRIER";
    err[3]="BUSY";

    String cmd="ATD"+dnm+"\r";
    rc=doMdmCmd(cmd, ok, err, 90000);
    if (rc<0) return rc;
    doMdmCmd("\r", null, null, 3000);
    return 0;
}
```



Bridge connection

When the PPP connection on one serial port established, another serial port can be used for terminal connection. `ListenThread.java` connect both serial ports in bridge mode. It will listen inbound connection for data on the Listen port, set in the configuration file. This data will be received with `RecvTrd.java` and send to another serial port. `SendTrd.java` will send data in another direction – from the serial port with terminal connection to the serial port with inbound connection, established with PPP.

To stop PPP connection, `link_up` variable should be set to false.

```
import java.io.*;
import java.net.*;
import java.util.*;

public class ListenThread implements Runnable
{
    ServerSocket lSocket;
    String port;
    int tty;
    Thread thread;

    public ListenThread(String pconfig,String tt)
    {
        port = pconfig;
        tty=Integer.parseInt(tt);
        try {
            lSocket = new ServerSocket (Integer.parseInt(port));
        } catch(IOException ee) { ; return; }
        thread=new Thread(this);
        thread.start();
    }

    public void run()
```

```

    {
    Socket client;
    try {
    while (true)
        {
        client=lSocket.accept();
        new RecvTrd(client,tty);
        }
        } catch(IOException e) {;}
    }

static public void main(String args[])
    {
    try {

    if ( args.length == 0)
        new ListenThread("9600","1");
    else if ( args.length == 1)
        new ListenThread(args[0],"1");
    else
        new ListenThread(args[0],args[1]);

        } catch (Throwable e) { ; System.exit(0); }
    }
}

```

Table of Contents

1. Introduction	1
Updates	1
2. PPP connection	2
RS232 DTE connector	2
Configuration file	3
Bridge connection	5