

SV TEHS SIA

Development Tools for Java™

IPVES Application Note 16:

Real-time clock

SV TEHS SIA

IPJV-ES Application Note 16: Real-time clock

V 1.0

© SV TEHS SIA

Ruses 14-24 • LV1029 • Riga • Latvia

Phone: +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>

Real-time clock

How to work with real-time clock.

Build-in real-time clock (RTC) can be accessible from the terminal in configuration mode. Command **time** will change and display current date/time settings. Command format is the following: **time 20021008183500** will set 2002 year, October 8, 18:35:00. **time** without parameters will display current settings. For additional details about terminal please check IPJV-ES User's Guide Chapter 6.

For Java applications interaction with RTC will be through the `Date` class in the `java.util` package. For fast access to the system clock two classes in the `java.lang` package also available: `System.currentTimeMillis()` to read current time and `System.setCurrentTimeMillis(long millis)` to set new time. The `millis` parameter is the difference between the current time and midnight, January 1, 1970 UTC (coordinated universal time) in milliseconds. For practical purposes UTC is equivalent to GMT. More details available in the documentation for `Date` class.

This application describe simple method to convert system time in milliseconds from the Jan 1, 1970 to the standard date format and vice versa. It also show, how to change system time.

```
import java.util.*;
import java.text.*;
public class TimeTest
{
    public static void main (String[] aArg)
    {
        long CurTime=System.currentTimeMillis();
        c_localtime ct=new c_localtime(CurTime);
        String aCurTime=ct.timestr();
    }
}
```

```

        System.out.println("Current      GMT      Time:      "+aCurTime+"
0x"+Long.toHexString(CurTime)+" ms");
        /* Convert TimeString to ms */
        CurTime=Date.parse(aCurTime);
        /* Set Current Time + 1s */
        System.setCurrentTimeMillis(CurTime+1000);
        /* Get Current Time */
        CurTime=System.currentTimeMillis();
        ct.setTime(CurTime);
        aCurTime=ct.timestr();
        System.out.println("New      Current      GMT      Time      :      "+aCurTime+"
0x"+Long.toHexString(CurTime)+" ms");
    }
}

```

Conversion from the system time to standard date and time usually need to be done faster (for example, for web server or file applications), therefore special class `c_localtime` used in this example. For conversion in another direction standard `Date` class used.

```

class c_localtime
{
int tm_sec;
int tm_min;
int tm_hour;
int tm_mday;
int tm_mon;
int tm_year;
int tm_wday;
int tm_yday;

```

```

long ctime;
static int mon_lengths[][] = {
    {31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31},
    {31, 29, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31}};
public c_localtime(long tm)
    {
        setTime(tm);
    }
void setTime(long timep)
    {
        int days, rem;
        int y, yleap;
        int wday;
        ctime=timep;
        timep/=1000L;
        if (timep<0L) timep=-timep;
        days = (int) (timep / 86400L);
        rem = (int) (timep % 86400L);
        /* compute hour, min, and sec */
        tm_hour = rem / 3600;
        rem %= 3600;
        tm_min = rem / 60;
        tm_sec = rem % 60;
        /* compute day of week */
        tm_wday = ((4 + days) % 7);
        /* compute year & day of year */
        for (y=1970;;y++)

```

```

        {
            int yd = (((y) % 4) == 0 && ((y) % 100) != 0 ||
((y) % 400) == 0)) ? 366 : 365;
            if (days < yd) break;
            days -= yd;
        }
        tm_year = y;
        tm_yday = days;
        int ip[] = mon_lengths[(((y) % 4) == 0 && ((y) % 100) != 0 ||
((y) % 400) == 0)] ? 1 : 0;
        for (tm_mon = 0; days >= ip[tm_mon]; tm_mon++) days -=
ip[tm_mon];
        tm_mday = days + 1;
    }
static String mon_name[] = {"Jan", "Feb", "Mar", "Apr", "May", "Jun",
"Jul", "Aug", "Sep", "Oct", "Nov", "Dec"};
static String day_name[] = {"Sun", "Mon", "Tue", "Wed", "Thu", "Fri",
"Sat"};
String timestr()
{
    StringBuffer bf = new StringBuffer();
    bf.append(day_name[tm_wday]); bf.append(' ');
    bf.append(mon_name[tm_mon]); bf.append(' ');
    toDeSt(bf, tm_mday, 2); bf.append(' ');
    toDeSt(bf, tm_hour, 2); bf.append(':');
    toDeSt(bf, tm_min, 2); bf.append(':');
    toDeSt(bf, tm_sec, 2); bf.append(' ');
    toDeSt(bf, tm_year, 4);
}

```

```
        return new String(bf);
    }
void toDeSt(StringBuffer sb,int val,int ncn)
    {
        int rem,res;
        rem=val%1000; res=((val-rem)/1000)+'0'; val=rem; if (ncn>3)
sb.append((char)res);
        rem=val%100; res=((val-rem)/100)+'0'; val=rem; if (ncn>2)
sb.append((char)res);
        rem=val%10; res=((val-rem)/10)+'0'; val=rem; if (ncn>1)
sb.append((char)res);
        res=rem+'0'; sb.append((char)res);
    }
}
```

Table of Contents

| | |
|---------------------------|---|
| 1. Introduction | 1 |
| Updates | 1 |
| 2. Real-time clock | 2 |