OpenNetCom Documentation 1.0

Vision Systems GmbH

November 16, 2006

Contents

1 Hardware

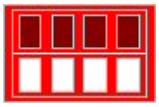
1.1 Components

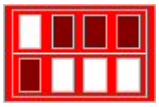
The *OpenNetCom* consists of the following hardware:

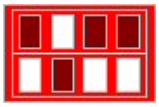
- *CPU:* Micrel KS8695P ARM9/166MHz
- *SDRAM:* 16MB
- *Flash:* 2MB
- *LAN:* one 100MBit LAN port
- *Wireless:* 802.11b/g RaLink card (RT2560)
- *Serial:* from one up to sixteen ports (16C950 with rs232/rs422/rs485)
- *Optional:* Service-Board with a console port, diagnostic leds and a 20pin JTAG connector

1.2 Startup

You can configure the startup modes of the *OpenNetCom* with the switches on the back of the device. Therefor you've the following choices:

1:  kernel messages and the console on the internal port (Service-Board)

2:  kernel messages and the console on the first serial port

3:  KGDB on the internal port and the console on the first serial port

It's possible to use the switches for your own tasks. Therefor you can switch off their usage in RedBoot when you insert „**noswitch**“ into the kernel command line. Then you've the option to control the behaviour of the switches over kernel parameters:

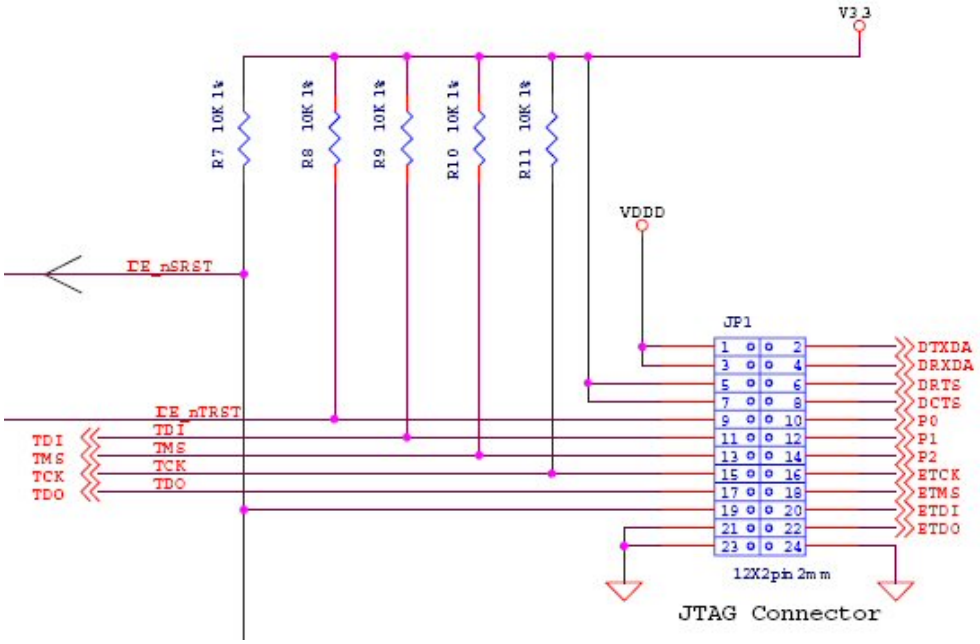
1: console=ttyKS0 *(default)*
2: console=ttyS0
3: console=ttyKGDB kgdb

1.3 RS-driver

The serial drivers will be configured into rs232 mode at startup of the system. If you don't want this for any reason, you can switch the RS-drivers off with the kernel parameter „**rsoff**". This could be useful if you've a rs485 device at a serial port and it would not be a good idea to put rs232 voltage levels on the lines.

Than you can configure the operation mode of the drivers over linux with ioctls or over the proc-filesystem. Take a look at the software linux section.

1.4 Service-Board-Connector



P0, P1, P2 are the status leds on the *Service-Board*. VDD is supplied with 5 Volt and the RxD, TxD, RTS and CTS belong to the internal console port *(TTL level)*. The other lines are for the original JTAG interface.

2 Software

2.1 Bootloader

RedBoot[1] is used as the main bootloader for the system. If you wanna work directly with *RedBoot*, you'll need the *Service-Board*. Then it's possible to get connected over a terminal program[2] with the following serial parameters: **115200,8,n,1**. Therefor you must restart the system and press „**Control-C**“ to get into *RedBoot's* console.

Now you should see someting like this:

```
FLASH 1 SST39VF1601 device found, AMD command set

RedBoot(tm) bootstrap and debug environment [REDBOOT]
Non-certified release, version UNKNOWN - built 10:13:57, Jul  6 2006

Platform: Kendin/Micrel KS8695 system (ARM9)
Copyright (C) 2000, 2001, 2002, Red Hat, Inc.

RAM: 0x00000000-0x01000000, [0x00008000-0x00fdc000] available
FLASH: 0x02800000 - 0x02a00000, 512 blocks of 0x00001000 bytes each.
== Executing boot script in 0.100 seconds - enter ^C to abort
^C
RedBoot> fis list
Name              FLASH addr  Mem addr    Length      Entry point
RedBoot           0x02800000  0x02800000  0x00010000  0x00000000
File System       0x02810000  0x00400000  0x000E1000  0xFFFFFFFF
Kernel            0x028F1000  0x00400000  0x000C8000  0x00008000
Data              0x029B9000  0x029B9000  0x00045000  0xFFFFFFFF
RedBoot config    0x029FE000  0x029FE000  0x00001000  0x00000000
FIS directory     0x029FF000  0x029FF000  0x00001000  0x00000000
RedBoot>
```

When you type the command „**fis list**“, you'll see the partition listing of the flash. In the first partition resides *RedBoot*. **This partition is write-protected!**

The partition „File System“ contains the CramFs (Compressed ROM File System). The CramFs is a very good compressed file system, in which the main applications and libraries reside (e.g. busybox, libc, ...). When you call an application like „socat“, then it would be decompressed into ram and gets called. **The CramFs itself is read-only.**

The next partition includes the linux kernel. It is also compressed and loaded from *RedBoot* into RAM at boot time.

The „Data“ partition is the place for the JFFS2 (Journalling Flash File System Ver2). It is a log-structured file system designed for use on flash devices in embedded systems. It is also writeable, so the configuration and your own files will reside there. JFFS2 is a compressed file system but not as good as the CramFs.

The next two partitions include the *RedBoot* configuration and the partition table.

There are many commands available in *RedBoot*. Some of the important once are: *help, fconfig, fis and load*.

If you wanna flash a partition with *RedBoot*, e.g. you've damaged the kernel partition, you could do this over the console port. Therefor you must load the data with „**load -r -m xmodem -b 0x400000**“ and write it to the destination partition with „**fis create -b 0x400000 -l 0xC8000 -e 0x8000 Kernel**“.

[1] http://sourceware.org/redboot

[2] ZOC / Hyperterminal

You've also the choice to program a complete flash image. Therefor load the image with „**load -r -m xmodem -b 0x400000**“ and write it back with „**fis write -f 0x2800000 -b 0x400000 -l 0x200000**“. You'll find such an image for your *OpenNetCom* on the CD.

If you're interested in changing some kernel parameters, you can do this over the command „**fconfig**“. When you come to the line „*alias/linux_exec: exec -c "root=/dev/mtdblock2"*“, you could enter your additional stuff there. It is also possible to change *RedBoot's* configuration under linux with the command „**rbcfg**“.

2.2 Linux

The operating system of the *OpenNetCom* is *uClinux*[3]. It consists of a kernel which is adapted to support many embedded architectures - also some without a MMU (Memory Management Unit). And it includes many applications which are very small and will satisfy most of the tasks a user could do. You find the *uClinux* distribution on the the CD and you've also the possibility to checkout or update the sources with *subversion*[4] from *svn.vscom.de*. For example: „**svn checkout svn://svn.vscom.de/ OpenNetCom/trunk OpenNetCom/trunk**“. For more information take a look at http://svn.vscom.de.

The main configuration files of the *OpenNetCom* reside in „*/etc*“ on the data partition. So if you wanna configure another ip-address at bootup, you can do this in the file „*/etc/rc*“ - there you can start also the dhcp client daemon or do other initializing things.

2.3 Development

2.3.1 Cross-Compiler

For the compilation of *uClinux* or your own applications, you'll need a cross-compiler for the ARM platform. You find one on the CD in the archive „*toolchain_arm.tar.bz2*“. Please decompress this archive into your directory „*/opt*“. For example: „*root@development:/opt*> **tar -xjf /cdrom/development/toolchain_arm.tar.bz2**“.

Additionaly, put the directory of the cross-compiler tools in your path environment variable (e.g. „**export PATH=/opt/uClinux/bin:$PATH**“).

2.3.2 uClinux Compilation

First of all please ensure, that you've also installed the needed packages for your development system - like:

- *compiler tools (gcc, make)*
- *ncurses development*
- *zlib development*

[3] http://www.uClinux.org
[4] http://subversion.tigris.org

If you wanna compile the kernel and user tools for your own, you must decompress the sources from the CD into a directory of your choice. For example: „**tar -xjf /cdrom/development/OpenNetCom_Sources_1_0.tar.bz2**“. *(The decompressed sources occupy more than one gigabyte on your harddisk; the main reason is, that* subversion *holds a copy of the origin version of each file.)*

Then you can switch into the „*uClinux-dist*“ directory and call „**make menuconfig**“. Now go to the vendor and product selection dialog and select the vendor „*VisionSystems*“ and the product „*OpenNetCom*“. Then you can leave the menu and run „**make dep**“. After that you can start the main compilation by calling „**make**“. When everything is finished, you'll find the images in the folder „*images*“. *zImage* is the compressed kernel, *cramdisk* the CramFs and *jffs2* includes the data partition filesystem.

If you wanna flash one of these images to your device, you can do this with the following commands:

NetCom	**nc -l -p 2400 > /var/image.bin**
PC	**netcat 192.168.254.254 2400 < images/zImage**
NetCom	**cat /var/image.bin > /dev/mtdblock3**

Take a look at „*./proc/mtd*“, if you like to know where the several images reside (e.g. „**cat /proc/mtd**“).

2.3.3 User-Application

When you're programming your own applications it's the same thing as under your normal linux PC - except that you must use a cross-compiler. If you wanna program a hello world example, you must create the source and compile it with the arm-linux-gcc, like this:

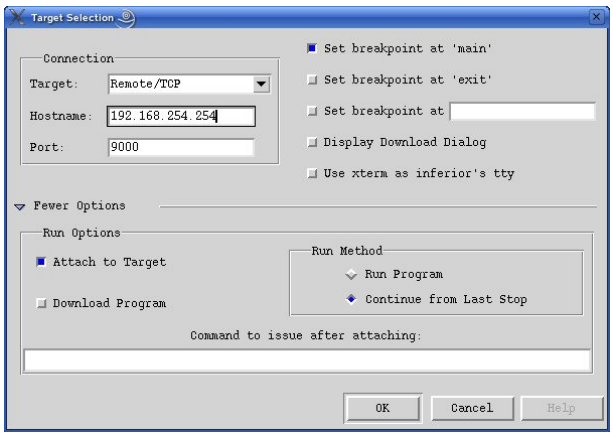
```
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char *argv[])
{
    printf("Hello World\n");
    return 0;
}
```

and than do the compilation with „**arm-linux-gcc -O0 -g3 -o hello hello.c**“. To reduce the size for the target system, you can call strip which will exclude all stuff that is not needed for the final binary (debugging information, strings, . . .). Do it with „**arm-linux-strip -o hello.strip hello**“.

For uploading your own applications to the *OpenNetCom*, you can use a ftp client of your choice. Use the Internet Explorer or Konqueror for example. Therefor you type in the ip address of your *OpenNetCom* in the address field like „`ftp://root@192.168.254.254`“. The standard password for the system is „**linux**“.

2.3.4 Debugging

You have the possibility to debug your own applications with the gdbserver on the target. To debug your application start the server with the following command: „**gdbserver :9000 /data/hello**“. Make a connection to the server with the insight debugger: „**arm-linux-insight hello**“. Go to the menu „*Target Settings*“ and enter your destination data. Than you can get connected to the target with „*Run Connect to target*“. The rest of the debugging is up to you.



2.3.5 Kernel-Debugging

You must start the system with the switch configuration number three. After that, the KGDB will listen on the serial console port (*Service-Board*). Then you can get connected to the kernel with „**arm-linux-insight linux-2.4.x\vmlinux**“. There you must configure the connection type *remote serial* in the target selection dialog. The baudrate is „**115200**“ for the system.

The console will be on the first serial port. And in the kernel debugging mode, you've also the possibility to use the *SysRq* features. Therefor you must send a serial break and the special *SysRq* key - for example: „**h = help**“, „**d = break into the kernel**“. *Please note: to send a serial break, the terminal program ZOC should be the first choice.*

2.3.6 Examples

There are example programs on the CD from which you can learn how to use ioctls for the leds, switches, network and serial ports. The programs are preinstalled in the „*./data*“ directory on the *OpenNetCom*.

2.4 Miscellaneous Applications

2.4.1 Proc-Filesystem

You can configure many *OpenNetCom* specific things over the proc-filesystem - like:

/proc/netcom/epld_ttySx	read and write the rs-modes from/to the EPLD of the *OpenNetCom*
/proc/netcom/leds	switch the leds or read the actual status of them
/proc/netcom/switch	read the switch positions
/proc/netcom/tstjmp	read the status of the test jumper
/proc/gpio	read and write from/to the gpio
/proc/gpio_ctrl	read and write from/to the gpio control register

While reading most of the values, you'll get some help. Here are examples:

cat /proc/netcom/epld_ttyS2	retrieve the status of the mode for the second serial port
echo rs422 > /proc/netcom/epld_ttyS2	sets the mode of the second port to rs422
echo GREEN red > /proc/netcom/leds	switch the green led on and the the red one off
echo 0x03 0x01 > /proc/gpio	switch the first gpio bit on and the second off

When you're interested in the direct use of the gpio data and control register, take a look at the *KS8695 datasheet* on the CD.

2.4.2 Daemons (Telnet, Ftp)

There are several daemons running on the *OpenNetCom*. The telnet daemon gives you the possibility for user logins from a remote workstation. The ftp daemon is used for the file transfer to and from the *OpenNetCom*. The upload functionality *(writing to the device)*, is only supported for the data partition *(mounted on „/data")* or the self growing ramfs *(mounted on „/var")*. *(Note: The default username for the system is **root** and the password **linux**)*

2.4.3 rbcfg

rbcfg gives you the chance to manipulate the data of *Redboot's* configuration, which is stored in the „*RedBoot config*" partition from linux.

2.4.4 socat

Socat is a specialized version of netcat - for advanced users *;-)*. A snippet from the readme:

```
socat can be used, e.g., as TCP port forwarder (one-shot or daemon), as an
external socksifier, for attacking weak firewalls, as a shell interface to UNIX
sockets, IP6 relay, for redirecting TCP oriented programs to a serial line, to
logically connect serial lines on different computers.
```

The most interesting thing would be to emulate the raw server functionality, that is also implemented as an example application. To do this, you must only call: „**socat TCP4-LISTEN:3000 ,fork,reuseaddr /dev/ttyS1,raw,echo=0**" - than you have a transparent raw tunnel between the tcp connection on port 3000 and the first serial port. Another example is the communication between the stdin from your console and the first serial port: „**socat -,raw,echo=0 /dev/ttyS1,raw,echo=0**".

2.4.5 setserial and stty

With *setserial* you can get and set many serial port informations. These informations include things like the I/O port, IRQ, baud base and some more. For example call „**setserial -a /dev/ttyS1**“ to get the information of the first serial port. You will see a output like this:

```
/dev/ttyS1, Line 0, UART: 16950/954, Port: 0x0000, IRQ: 4
        Baud_base: 921600, close_delay: 50, divisor: 0
        closing_wait: 3000
        Flags: spd_normal
```

To get some help which parameters you can set and how to do this, call „**setserial –help**“.

With *stty* you can change and print terminal line settings. This is usefull, if you wanna see the actual baudrate or set it to a different value. To see all the informations of the first serial port, call „**stty -a < /dev/ttyS1**“. Setting the baudrate to another value will be optained with „**stty 115200 < /dev/ttyS1**“.

2.4.6 iwconfig, iwpriv and iwlist

You can view and change the wireless configuration with these programs. Calling „**iwconfig ra0**“, you'll get all the wireless information for the WLAN-card in the *OpenNetCom*. Here are some examples for the configuration of the wireless settings:

```
Example I: Config STA to link with AP which is OPEN/NONE(Authentication/Encryption)
1. iwconfig ra0 mode managed
2. iwconfig ra0 key open
3. iwconfig ra0 key off
4. iwconfig ra0 essid "AP's SSID"

Example II: Config STA to link with AP which is SHARED/WEP(Authentication/Encryption)
1. iwconfig ra0 mode managed
2. iwconfig ra0 key restricted
3. iwconfig ra0 Key [1] "s:AP's wep key"
4. iwconfig ra0 key [1]
5. iwconfig ra0 essid "AP's SSID"

Example III: Config STA to create/link as adhoc mode
1. iwconfig ra0 mode ad-hoc
2. iwconfig ra0 key off
3. iwconfig ra0 essid "AP's SSID"

Example IV: Config STA to link with AP which is WPAPSK/TKIP(Authentication/Encryption)
1. iwconfig ra0 mode managed
2. iwconfig ra0 essid "AP's SSID"
3. iwpriv ra0 set AuthMode=WPAPSK
4. iwpriv ra0 set EncrypType=TKIP
5. iwpriv ra0 set WPAPSK="AP's wpa-preshared key"

Example V: Config STA to link with AP which is WPAPSK/AES(Authentication/Encryption)
1. iwconfig ra0 mode managed
2. iwconfig ra0 essid "AP's SSID"
3. iwpriv ra0 set AuthMode=WPAPSK
4. iwpriv ra0 set EncrypType=AES
5. iwpriv ra0 set WPAPSK="AP's wpa-preshared key"
```

Please note, that you must configure the right country region if you wanna configure some channel numbers. Take also a look at the „*iwpriv_usage.txt*“ file in the documentation directory on

the CD and also at the example configuration in „*./etc/rc*“.

If you want to find wireless networks in your range, you can do this with „**iwlist ra0 scanning**“.

3 General Information

3.1 System

- *Username:* root
- *Password:* linux

3.2 LAN

- *Ip-address:* 192.168.254.254
- *Running services:* telnet, ftp

3.3 WLAN

- *Ip-address:* 192.168.127.254
- *SSID:* OpenNetCom
- *Channel:* 7
- *Encryption:* WEP
- *Password:* linux