Netconsole support in My Book Live (MBL)

First read about netconsole here: https://www.kernel.org/doc/Documentation/networking/netconsole.txt

The great thing about our MBL's is that both uboot and 4.x Linux kernel support netconsole.
This means that to monitor a system boot from hardware, you don't need to solder an UART/serial device.
It's technically even possible to interact with uboot (versus readonly view), but this is beyond the scope of this guide.

The MBL netconsole implementation requires 3 things:

1. netconsole enabled uboot files
2. netconsole enabled kernel or netconsole enabled Linux kernel patches and config file (if you want to compile your own kernel)
3. a system (Windows or Linux) to view uboot and kernel console messages

DO NOT TEST THIS WITHOUT ABILITY TO TAKE YOUR DRIVE OUT AND HOOK TO A LINUX AND/OR WINDOWS SYSTEM

Test using a prebuild kernel without modifying uboot boot file
To quickly test the netconsole functionality on a Linux kernel

1. Download and unpack a precompiled kernel with netconsole support

```
cd /tmp
wget "https://drive.google.com/uc?export=download&id=1WlNAptzyrNe9s0w-ZXeaA-qr8UQ4YI-B" -O
kernel_4.9.119_hdd_led.tgz
cd /
tar xvfz netconsole.tgz
```

2. install prebuild kernel

```
cd /boot
cp uImage uImage.save
cp uImage_4.9.119_hdd_led uImage
```

3. reboot (and hope for the best)

```
systemctl reboot    # Debian 8.x jessie
```

4. after the system comes up we have a netconsole enabled kernel which we can now test
Basically there are two ways to enable netconsole in the kernel:
- via kernel boot command line (e.g. by uboot)
- via sysfs, a filesystem for exporting kernel objects

Since we don't want to modify uboot files for now, let's test sysfs. The first thing we will do is create a netconsole device. Typically, each netconsole client (e.g. a Windows PC, a Linux box) will get its own netconsole directory. In this example, linuxConsole1 is our first system

```
mkdir /sys/kernel/config/netconsole/linuxConsole1
cd /sys/kernel/config/netconsole/linuxConsole1
```

When a new console is created, it's disabled by default. We can check this as follows, the result should be 0

```
cat enabled
```

Now we can define the settings for this netconsole. LinuxConsole1 is Linux box with IP address 192.168.1.18 and MAC address cc:5f:f4:cb:2f:2c. The default port is 6666.

```
echo 0 > enabled                    # disable the target (if required)
echo eth0 > dev_name                # set local interface
echo 192.168.1.18 > remote_ip       # update some parameter
echo cc:5f:f4:cb:2f:2c > remote_mac  # update more parameters
echo 1 > enabled                    # enable target
```

5. on your linux server (there are solutions for Windows too) start the monitoring using netcat

```
nc -u -l -p 6666
```

7. send a test message from the MBL, if all is well we will see it on our Linux box

```
echo "test message" >/dev/kmsg
```

8. test on a windows client
Download a windows 32/64 bit netcat client (plus useful batch scripts) from here
Repeat these test for your Windows system

Test using a prebuild kernel by modifying uboot boot file
Now we will boot the kernel with NETCONSOLE enabled

1. test if you have the mkimage command, if not install it using "apt install u-boot-tools"

```
mkimage -?
apt install u-boot-tools
```

2. create a new uboot boot file with your netconsole settings
Basically, we will create a new */boot/boot.scr* file
In */tmp*, create *boot.netconsole.txt* with your editor of choice of with the cat command.
The content should be as follows:

```
setenv md0_args 'setenv bootargs netconsole==[+][ [src-port]@[src-ip]/[dev],[tgt-port]@[tgt-ip]/[tgt-macaddr]
root=/dev/sda1 rw rootfstype=ext3 rootflags=data=ordered'
setenv load_sata 'sata init; ext2load sata 1:1 ${kernel_addr_r} /boot/uImage; ext2load sata 1:1 ${fdt_addr_r}
/boot/apollo3g.dtb'
setenv boot_sata 'run load_sata; run md0_args addtty; bootm ${kernel_addr_r} - ${fdt_addr_r}'
echo ==== Loading Linux kernel, Device tree, Root filesystem ====
run boot_sata
```

where

+	if present, enable extended console support
src-port	source for UDP packets (defaults to 6665)
src-ip	source IP to use (IP address of your MBL)
dev	network interface (always eth0)
tgt-port	port for logging agent (defaults to 6666)
tgt-ip	IP address for netconsole logging agent/client
tgt-macaddr	ethernet MAC address for logging agent (broadcast)

in our example above for our Linux client, this would do the job if our MBL has IP address 192.168.1.5:

```
cat >/tmp/boot.netconsole.txt <<EOF
setenv md0_args 'setenv bootargs netconsole=6663@192.168.1.5/eth0,6664@192.168.1.28/cc:5f:f4:cb:2f:2c
root=/dev/sda1 rw rootfstype=ext3 rootflags=data=ordered'
setenv load_sata 'sata init; ext2load sata 1:1 ${kernel_addr_r} /boot/uImage; ext2load sata 1:1 ${fdt_addr_r}
/boot/apollo3g.dtb'
setenv boot_sata 'run load_sata; run md0_args addtty; bootm ${kernel_addr_r} - ${fdt_addr_r}'
echo ==== Loading Linux kernel, Device tree, Root filesystem ====
run boot_sata
EOF
```

Note: this example uses EXT3 as root file system. If yours is EXT2, please change
Note: this example uses port 6664 for the client, so on your client use

```
nc -u -l -p 6664
```

3. now we will enable the kernel to boot with netconsole enabled

```
cd /tmp
mkimage -A powerpc -O linux -T script -C none -a 0 -e 0 -n 'Execute uImage' -d boot.netconsole.txt
boot.netconsole.scr
mv boot.netconsole.scr /boot
```

4. make a copy of your original uboot file (just in case...) and install the new uboot loader

```
cd /boot
cp boot.scr boot.scr.bck
cp boot.netconsole.scr boot.scr
```

5. reboot and watch the whole log come through :-)

```
systemctl reboot
```

Double NETCONSOLE test using a prebuild kernel by modifying uboot boot file

Now we will boot the kernel with NETCONSOLE enabled while also logging the uboot hardware console
So, for this, we will need to run two netcat client in parallel.

1. In */tmp*, create *boot.netconsole2.txt* with your editor of choice of with the cat command.
The content should be as follows:

```
setenv ncip tgt-ip
setenv ipaddr src-ip
saveenv
run nc
setenv md0_args 'setenv bootargs netconsole==[+][ [src-port]@[src-ip]/[dev],[tgt-port]@[tgt-ip]/[tgt-macaddr]
root=/dev/sda1 rw rootfstype=ext3 rootflags=data=ordered'
setenv load_sata 'sata init; ext2load sata 1:1 ${kernel_addr_r} /boot/uImage; ext2load sata 1:1 ${fdt_addr_r}
/boot/apollo3g.dtb'
setenv boot_sata 'run load_sata; run md0_args addtty; bootm ${kernel_addr_r} - ${fdt_addr_r}'
echo ==== Loading Linux kernel, Device tree, Root filesystem ====
run boot_sata
```

2. as above, now we will enable the kernel and uboot with netconsole

```
cd /tmp
mkimage -A powerpc -O linux -T script -C none -a 0 -e 0 -n 'Execute uImage' -d boot.netconsole2.txt
boot.netconsole2.scr
mv boot.netconsole2.scr /boot
cd /boot
cp boot.netconsole2.scr boot.scr
systemctl reboot
```

3. now run two netcat sessions in 2 windows

```
nc -u -l -p 6664
nc -u -l -p 6666
```

for windows, you can leverage *uboot_netconsole.bat* and *netconsole.bat* from the package downloaded

Compile your own netconsole enabled kernel

Download patches here and extract to /tmp
Apply the patches as follows from the base directory of your kernel sources:

```
for i in $(ls /tmp/patches/[0-9]*)
do
  echo #### $i ####
  patch -p 1 -b -i $i
done
cp /tmp/patches/config_4.1.119 .config
make uImage
make modules_install
```

copy the new kernel to /boot, as above copy to uImage and reboot

```
cp arch/powerpc/boot/uImage /boot/uImage_4.9.119
```