

Installing Debian on the Marvell SheevaPlug and OpenRD

These instructions cover how to install Debian onto the NAND of the Marvell SheevaPlug [<http://www.plugcomputer.org/>] and OpenRD [<http://www.open-rd.org/>] (tested on the Client and Ultimate flavours) platforms; another approach is to use the Marvell Easy Plug [<http://www.marvell.com/solutions/assets/Marvell-Easy-Plug-Computer-Installer.pdf>] ESIA [<http://sourceforge.net/projects/esia/>] tool.

Before you continue you must have a pre-built 'armel' based Debian rootfs to hand, using an approach such as the one I detail on my page titled [Cross debootstrap Debian Root Filesystem with User Mode QEMU](#); these instructions continue from where that page leaves off. You will additionally need the following packages installed:

- mtd-utils - do not use 20100907 [<http://bugs.debian.org/592485>]
- uboot-mkimage
- tftpd-hpa
- binfmt-support
- qemu-user-static - for Ubuntu weenies this package is called 'qemu-arm-static'

Well, we start off by adding the Linux kernel package to the root filesystem:

```
berk:/usr/src/kirkwood# cat /etc/resolv.conf > rootfs/etc/resolv.conf
berk:/usr/src/kirkwood# cp /usr/bin/qemu-arm-static rootfs/usr/bin/
berk:/usr/src/kirkwood# chroot rootfs
berk:/# apt-get update
berk:/# aptitude install linux-image-2.6-kirkwood uboot-mkimage tftpd-hpa mtd-utils
[snipped, ignore any qemu errors and other warnings]
berk:/# apt-get clean
berk:/# exit
berk:/usr/src/kirkwood# rm rootfs/usr/bin/qemu-arm-static
berk:/usr/src/kirkwood# find rootfs/var/lib/apt/lists -type f -delete
berk:/usr/src/kirkwood# : > rootfs/etc/resolv.conf
```

Now our root filesystem is ready to go, we need to create suitable snapshots (approximate output sizes show) that U-Boot can boot and also the final UBIFS image that we will burn to the NAND. To get this data to the platform, we run a TFTP server on the host workstation:

```
berk:/usr/src/kirkwood# cd rootfs
berk:/usr/src/kirkwood/rootfs# find . | cpio -H newc -o | gzip -n > ../initrd.img.gz
berk:/usr/src/kirkwood/rootfs# cd ..
berk:/usr/src/kirkwood# mkimage -A arm -O linux -T kernel -C none -n uImage -a 0x00008000 -e 0x00008000 -d rootfs/vmlinuz uImage
berk:/usr/src/kirkwood# mkimage -A arm -O linux -T ramdisk -C none -n uInitrd -d initrd.img.gz uInitrd

berk:/usr/src/kirkwood# cat <<EOF > ubi.cfg
[ubifs]
mode=ubi
image=ubifs.img
vol_id=0
vol_size=128MiB
vol_type=dynamic
vol_name=rootfs
vol_flags=autoresize
EOF

berk:/usr/src/kirkwood# # NAND properties can be lifted from UBI dmesg or /sys/class/mtd/mtd*/size or mtdinfo
berk:/usr/src/kirkwood# # These values match an [[OpenRD]] or a sheevaplug
berk:/usr/src/kirkwood# SUBPAGE=512 # /sys/class/mtd/mtd*/subpagesize
berk:/usr/src/kirkwood# MIN_IO_SIZE=2048 # /sys/class/mtd/mtd*/writesize
berk:/usr/src/kirkwood# PEB_SIZE=131072 # /sys/class/mtd/mtd*/erasesize
berk:/usr/src/kirkwood# FLASH_MB=512 # Full Flash chip
berk:/usr/src/kirkwood# UBOOT_MB=1 # U-Boot partition size
berk:/usr/src/kirkwood# KERNEL_MB=4 # Kernel partition size
berk:/usr/src/kirkwood# PART_MB=$(( FLASH_MB-UBOOT_MB-KERNEL_MB ))

berk:/usr/src/kirkwood# # LEB=PEB-overhead, below assumes NAND w/subpages. See http://www.linux-mtd.infradead.org/doc/ubi.html#L_overhead
berk:/usr/src/kirkwood# [ "${2*SUBPAGE})" -lt "$MIN_IO_SIZE" ] && LEB_SIZE=$(( PEB_SIZE-MIN_IO_SIZE )) || LEB_SIZE=$(( PEB_SIZE- 2*MIN_IO_SIZE ))
berk:/usr/src/kirkwood# MAX_LEB_COUNT=$(( (PART_MB*1024*1024)/LEB_SIZE + 1 ))
berk:/usr/src/kirkwood# mkfs.ubifs -v -r rootfs -m $MIN_IO_SIZE -e $LEB_SIZE -c $MAX_LEB_COUNT -o ubifs.img
berk:/usr/src/kirkwood# ubinize -v -m $MIN_IO_SIZE -s $SUBPAGE -p $PEB_SIZE ubi.cfg -o ubi.img

berk:/usr/src/kirkwood# ls -lh uImage uInitrd ubi.img
-rw-r--r-- 1 root src 113M Feb 28 11:51 ubi.img
-rw-r--r-- 1 root src 1.4M Feb 28 11:50 uImage
-rw-r--r-- 1 root src 80M Feb 28 11:50 uInitrd

berk:/usr/src/kirkwood# ip addr add 10.4.50.5/24 dev eth0
berk:/usr/src/kirkwood# in.tftpd --foreground --secure /usr/src/kirkwood
```

Now, over to the platform it's-self. You will need to hook up using the serial port over USB that it provides. Set up 'minicom' at 115200,8n1 or you could use 'screen' by running "screen /dev/ttyUSB0 115200".

N.B. bear in mind, the serial port/USB device only functions when powered up so strange things happen if you power cycle the platform whilst connected to it over USB...please try to avoid doing so otherwise you will quickly become irritated at having to repeatedly restart minicom/screen.

When you power up the device you will see the following rolling by, interrupt the process when prompted so you get to U-Boot's command line:

```
[snipped]
PCI 0: PCI Express Root Complex Interface
PEX interface detected Link X1
Net: egiga0 [PRIME], egiga1
Hit any key to stop autoboot: 0
Marvell> resetenv
Marvell> reset

[snipped]
PCI 0: PCI Express Root Complex Interface
PEX interface detected Link X1
Net: egiga0 [PRIME], egiga1
Hit any key to stop autoboot: 0
Marvell> setenv run_diag no
Marvell> setenv mainlineLinux yes
Marvell> setenv arcNumber 2884 <--- [[OpenRD]]-Ultimate, consult http://www.arm.linux.org.uk/developer/machines/ (ie. 2097 for [[SheevaPlug]], 2361 for [[OpenRD]]-Client, etc)

Marvell> setenv mtdids nand0=nand_mtd
Marvell> setenv mtdparts mtdparts=nand_mtd:0x100000@0x000000(uboot),0x400000@0x100000(uImage),0x1fb00000@0x500000(root)

Marvell> setenv x_bootcmd_kernel nand read 0x1000000 uImage
Marvell> setenv x_bootargs console=ttyS0,115200 panic=10 ubi.mtd=root
Marvell> setenv x_bootargs_root root=ubi0:rootfs rootfstype=ubifs rw

Marvell> setenv bootcmd '${x_bootcmd_kernel}; setenv bootargs ${x_bootargs} ${x_bootargs_root}; bootm 0x1000000'
```

```

Marvell>> saveenv
Marvell>> reset

[snipped]
PCI 0: PCI Express Root Complex Interface
PEX interface detected Link X1
Net:   egiga0 [PRIME], egiga1
Hit any key to stop autoboot:  0
Marvell>> tftp 0x1000000 uImage
Marvell>> nand erase uImage
Marvell>> nand write 0x1000000 uImage

Marvell>> tftp 0x2000000 uInitrd
Marvell>> setenv bootargs ${x_bootargs} root=/dev/ram0 rdinit=/sbin/init
Marvell>> bootm 0x1000000 0x2000000

[snipped Linux booting]

Debian GNU/Linux squeeze/sid (none) ttyS0

(none) login: root
Password:
Linux (none) 2.6.32-5-kirkwood #1 Thu Aug 26 03:31:56 UTC 2010 armv5tel

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
root@(none):~# # replace with the output of 'date' from your own workstation
root@(none):~# date -s "Thu Dec 30 15:58:00 GMT 2010"

root@(none):~# ip addr add 10.4.50.165/24 dev eth0
root@(none):~# ip link set dev eth0 up
root@(none):~# mount /tmp/ -o remount,size=256M
root@(none):~# tftp -m binary 10.4.50.5 -c get ubi.img /tmp/ubi.img
root@(none):~# # ubidetach /dev/ubi_ctrl -m 2
root@(none):~# ubiformat -s 512 -f /tmp/ubi.img /dev/mtd2
root@(none):~# reboot

```

You should find your platform is up and running on it's own and you can use it like you would any x86/amd64 installed copy of Debian. All that remains is that you kill the TFTP server with a Ctrl-C and remove the temporary IP you gave your host:

```

berk:/usr/src/kirkwood# ip addr del 10.4.50.5/24 dev eth0

```

Second Serial Port (UART1)

The OpenRD lets you choose between enabling either [http://groups.google.com/group/openrd/browse_frm/thread/7ba48fd7916a88f/2b89c877c4f39098?vc=1&fwc=1] the SDIO interface (SD card slot) or alternatively the UART1 port (the RS-232 [<http://en.wikipedia.org/wiki/RS-232>] or RS-485 [<http://en.wikipedia.org/wiki/EIA-485>] external serial port).

To pick what you want you pass the kernel parameter 'kw_openrd_init_uart1' on boot [<http://git.kernel.org/?p=linux/kernel/git/nico/orion.git;a=commitdiff;h=f2ac38dbcb95c19ce97207f1e02d0623c7052610>], this functionality is included in Debian 'squeeze' linux-image-2.6.32-5-kirkwood [<http://packages.debian.org/squeeze/linux-image-2.6.32-5-kirkwood>], like so:

```

Marvell>> setenv x_bootargs console=ttyS1,115200 panic=10 ubi.mtd=root kw_openrd_init_uart1=232
Marvell>> saveenv
Marvell>> boot

```

The parameter takes two options, either '232' or '485' depending on whether you want the RS-232 port or the RS-485 port enabled; passing anything else (or not using the parameter) results in the default behaviour where SDIO interface left enabled.

N.B. easy to overlook but in Documentation/serial-console.txt [<http://git.kernel.org/?p=linux/kernel/git/torvalds/linux-2.6.git;a=blob;f=Documentation/serial-console.txt>] it states "Note that you can only define one console per device type (serial, video)". This means you are unable to have two serial driven 'console's; more clearly explained in the Remote Serial Console HOWTO [<http://tldp.org/HOWTO/Remote-Serial-Console-HOWTO/configure-kernel.html>]

N.B. unfortunately there does not seem to be any way to get the factory installed u-boot to use UART1 (RS232 or RS485 port) as the console instead of UART0 (USB port), I do have some instructions on how to do this though that I will get around to typing up one day so do pester me if you need them

Updating the Kernel

When the kernel updates, the kernel image needs to be separately written to the NAND with the following:

```

# mkimage -A arm -O linux -T kernel -C none -n uImage -a 0x00008000 -e 0x00008000 -d /vmlinuz /tmp/uImage
# flash_eraseall /dev/mtd1
# nandwrite -p /dev/mtd1 /tmp/uImage

```

Once done, you can reboot whenever you wish.

Of course this gets annoying, so to automate the process, you can install the 'initramfs-tools' package and use:

```

# cat <<'EOF' > /etc/kernel/postinst.d/local-kirkwood
#!/bin/sh -e

version="$1"
bootopt=""

# passing the kernel version is required
[ -z "${version}" ] && exit 0

# kernel-package passes an extra arg
if [ -n "$2" ]; then
    if [ -n "${KERNEL_PACKAGE_VERSION}" ]; then
        # exit if custom kernel does not need an initramfs
        [ "$INITRD" = 'No' ] && exit 0
        bootdir=$(dirname "$2")
        bootopt="-b ${bootdir}"
    else
        # official Debian linux-images take care themself
        exit 0
    fi
fi

# avoid running multiple times

```

```

if [ -n "$DEB_MAINT_PARAMS" ]; then
    eval set -- "$DEB_MAINT_PARAMS"
    if [ -z "$1" ] | [ "$1" != "configure" ]; then
        exit 0
    fi
fi

[ -x /usr/bin/mkimage ] | { echo "missing mkimage" 2>&1; exit 1; }
[ -x /usr/sbin/flash_eraseall ] | { echo "missing flase_eraseall" 2>&1; exit 1; }
[ -x /usr/sbin/nandwrite ] | { echo "missing nandwrite" 2>&1; exit 1; }


/usr/bin/mkimage -A arm -O linux -T kernel -C none -n uImage -a 0x00008000 -e 0x00008000 -d /boot/vmlinuz-${version} /tmp/uImage
/usr/sbin/flash_eraseall /dev/mtd1
/usr/sbin/nandwrite -p /dev/mtd1 /tmp/uImage
rm /tmp/uImage
EOF

# chown root:root /etc/kernel/postinst.d/local-kirkwood
# chmod 755 /etc/kernel/postinst.d/local-kirkwood

```

Now when a new Debian kernel is installed (typically via 'aptitude upgrade') the kernel should be automatically written to the NAND.

Updating U-Boot

 this is a works in progress but does work, however I do not personally feel there is much point in doing so

Gleamed from <http://thread.gmane.org/gmane.comp.boot-loaders.u-boot/79733> [http://thread.gmane.org/gmane.comp.boot-loaders.u-boot/79733], <http://www.cyrius.com/debian/kirkwood/sheevaplug/uboot-upgrade.html> [http://www.cyrius.com/debian/kirkwood/sheevaplug/uboot-upgrade.html] and http://www.plugincomputer.org/pluginwiki/index.php/Re-building_the_kernel_and_U-Boot#Building_U-Boot [http://www.plugincomputer.org/pluginwiki/index.php/Re-building_the_kernel_and_U-Boot#Building_U-Boot].

Download [uboot-openrd.patch](#).

```

berk:/usr/src/kirkwood# git clone git://git.denx.de/u-boot-marvell.git
berk:/usr/src/kirkwood# cd uboot-marvel
berk:/usr/src/kirkwood/uboot-marvell# patch -p1 < ../uboot-openrd.patch
berk:/usr/src/kirkwood/uboot-marvell# make mrproper
berk:/usr/src/kirkwood/uboot-marvell# make openrd_client_config
berk:/usr/src/kirkwood/uboot-marvell# make u-boot.kwb CROSS_COMPILE=arm-linux-gnueabi-

```

```

Marvell>> tftp 0x1000000 u-boot.kwb
Marvell>> nand erase uboot
Marvell>> nand write 0x1000000 uboot
Marvell>> reset

```

Except where otherwise noted, content on this wiki is licensed under the following license: CC Attribution-NonCommercial-Share Alike 3.0 Unported [http://creativecommons.org/licenses/by-nc-sa/3.0/] debian/debootstrap/kirkwood.txt · Last modified: 2012/03/24 20:28 by alex