Wandboard porting guide / release notes

Tapani

May 15, 2013

1 Booting

1.1 Boot order

The Wandboard features two SD card slots, one on the CPU module and one on the baseboard. The default boot order is to boot from the CPU module first, and if no SD card is detected – try booting from the baseboard slot.

The boot order is determined by resistors on the module (and might be changeable with a soldering iron and a voided warranty).

1.2 Boot order issues

The iMX6 CPU support up to four SD card (MMC) slots; the Wandboard has two. The SD card slot on the CPU module is number three, and the slot on the baseboard is number one.

By default the linux kernel would initialize slot one first, and slot three afterwards – and then look for a root filesystem on SD1. There is a hack in the board file to initialize the SD-card slots in reverse order, but still, if the SD card in SD3 (module) is slow to be detected, the kernel can misdetect the order of the cards.

1.3 SD card layout

The iMX6 ROM will look for a bootloader at byte offset 1024. For instance a u-boot binary can be copied to the SD card by:
`# sudo dd if=u-boot.imx of=/dev/$dev bs=1k seek=1`
(where $dev is the SD card device).

If a u-boot bootsplash is used, it should be placed 512k into the SD card (see section about boot splash, further down).

The default u-boot expects a kernel one megabyte into the card. Hence, the kernel can be installed to SD by:
`# dd if=arch/arm/boot/uImage of=/dev/$dev bs=1M seek=1`

Of course, a custom u-boot can change the location where it will attempt to load a kernel.

WARNING: Make sure you use the right device for your SD! Using the wrong device name can overwrite the content on your hard drive!

NOTE: When partitioning the SD card, reserve some space on the card for u-boot + kernel by leaving some "unused" space before the first partition. That is, start your first partition a few megabytes into the card. A large kernel can be ≈5 Mb in size, so leaving, say 8MB unpartitioned before the first partition should be safe.

2 Compiling

2.1 Setting up a cross-compiler

In order to compile the kernel or u-boot bootloader you need a *cross-compiler*, that is a compiler running on one machine but producing executables for another.

The recommended compiler for the Wandboard u-boot and kernel is some of the older CodeSourcery compilers. For instance version CodeSourcery G++ Lite 2010.09-50 has so far stood the test of time and provided stable kernel binaries. Unfortunately all cross compiler versions are not as stable.

To set up a cross compiler on a linux host, one way is to follow the steps:

1. Install a cross compiler somewhere, like /opt/arm-2010.09
2. Set the environment variable CROSS_COMPILE to the prefix of your cross compiler, i.e. `export CROSS_COMPILE=arm-none-linux-gnueabi-`.
3. Add the cross compilers bin folder to your PATH. i.e.
   `export PATH=$PATH:/opt/arm-2010.09/bin`
   The /opt/arm-2010.09/bin folder is an example location of `arm-none-linux-gnueabi-gcc`.

For software packages, often setting the environment variable ARCH to arm (by `export ARCH=arm`) is needed.

2.2 u-boot

The u-boot included in this release is a slightly modified version of the mainline 2013.04 release. Essentially a HDMI splash screen has been added.

To compile u-boot for the Wandboard DualLite, issue the following command in the u-boot folder:

```
% make -j4 wandboard_dl
```

or, for the Wandboard Solo

```
% make -j4 wandboard_solo
```

Note that a Wandboard Solo SD card should work on a DualLite (but using only half of the memory).

The the u-boot binary can be installed to SD card by: `sudo dd if=u-boot.imx of=/dev/$dev bs=1k seek=1`

2.3 u-boot splash screen

The Wandboard u-boot enables a u-boot splash-screen displayed on an attached HDMI monitor. In the boot_logo folder there is a script that converts

and resizes common image formats to the expected bmp.gz file format.

The simple approacs would be:

```
% ./mkbootlogo.sh wandboard-720x480-bw.png
# dd if=out.bmp.gz of=/dev/\$dev bs=512k seek=1 count=1
```

Some colour distortions can be present when splash screens are displayed in u-boot. Any remedies are welcone.

2.4 Linux kernel

To compile the linux kernel configure the kernel with

```
% make wandboard_defconfig
% make -j4 uImage modules
```

The Ubuntu kernel configuration is present as a file in the kernel folder (wandboard_ubuntu_config)

The Wandboard board file is in `/arch/arm/mach-mx6/board-wand.c`, and is the natural starting point for kernel hacking.

Note: the boardfile is for the module only. Components (that require drivers) on the baseboard should be placed in the corresponding baseboard file. As of now there is just one baseboard file (baseboard-wand.c) containing the initialization of the sgtl5000 codec.

The wireless driver should be compiled as modules so firmware can be loaded runtime. Also note that the Wandboard kernel uses the new fullmac (brcmfmac) driver instead of the old (bcm4329) driver.

3 Drivers

3.1 Bluetooth

The bluetooth part of bcm4329 requires a userspace utility to load firmware. The utility, named brcm_patchram_plus, is included (with source code) in

this release.

As example, a bluetooth scan can be performed by:

```
#  brcm_patchram_plus --timeout=6.00 --patchram /lib/firmware/
 brcm/bcm4329.hcd --baudrate 921600 --use_baudrate_for_download
 /dev/ttymxc2
# hciattach /dev/ttymxc2 any 921600
# hciconfig hci0 up
# hcitool -i hci0 scan
```

Where /lib/firmware/brcm/bcm4329.hcd is the bluetooth firmware.

3.2 Wireless

The procedure for WiFi is less compilcated than for Bluetooth. Just make sure the firmware and nvram files are located in /lib/firmware/brcm/, with filenames:

- brcmfmac-sdio.bin
- brcmfmac-sdio.txt

This way the firmware will be loaded by the kernel module upon insertion.

3.3 Audio

The wandboard features three audio devices

- The sgtl5000 codec
- S/PDIF
- HDMI audio

One possible ALSA configuration (.asoundrc) for these is:

```
pcm.sgtl5000audio {
        type hw
        card 0
        channels 2
}
ctl.sgtl5000audio {
        type hw
        card 0
        channels 2
}

pcm.imxspdif {
        type hw
        card 1
        channels 2
}
ctl.imxspdif {
        type hw
        card 1
        channels 2
}

pcm.imxhdmisoc {
        type hw
        card 2
        channels 2
}
ctl.imxhdmisoc {
        type hw
        card 2
        channels 2
}

pcm.copy {
        type plug
        slave {
                pcm hw
        }
        route_policy copy
}
```

To play a WAV over s/pdif can then be done by: `% aplay -Dimxspdif file.wav`