Mini2440



From LinuxMCE wiki

Contents

[hide]

- 1 About
- 2 Preparations
  - 2.1 Toolchain
- 3 Setting up the bootloader
  - 3.1 Compiling uboot
  - 3.2 Flashing uboot
  - 3.3 Configure uboot
- 4 Kernel
  - 4.1 Compilation
  - 4.2 Flash kernel
  - 4.3 Configure uboot
- 5 Setting up embedian
- 6 Troubleshooting
  - 6.1 Write errors
- 7 Links

About

This is a howto install uboot and a new kernel on the mini2440. Big thanks to BusError for the instructions and guidance. DISCLAIMER: This worked for me, if it eats your cat or burns your kitchen sink, don't blame me.

Preparations

Toolchain

I used this one:

http://www.codesourcery.com/sgpp/lite/arm/portal/package3696/public/arm-none-linux-gnueabi/arm-2008q3-72-arm-none-linux-gnueabi-i686-pc-linux-gnu.tar.bz2

Add the bin dir to the $PATH. If you have a toolchain with another prefix (like arm-linux-) adapt the CROSS_COMPILE environment variable in the next steps.

Setting up the bootloader

Compiling uboot

Clone the git tree:

```
mkdir uboot ; cd uboot
git clone git://repo.or.cz/u-boot-openmoko/mini2440.git
```

Setup cross compiling:

```
export CROSS_COMPILE=arm-none-linux-gnueabi-
```

or edit your .bashrc eg.

```
export CROSS_COMPILE=arm-none-linux-gnueabi-
export PS1="\[\033[1;34m\]\u \[\033[0m\]\w:\$: \[\033[0m\]"
#export PS1="\u:\w\$>"
LS_OPTIONS="--color=auto"
alias ls="ls $LS_OPTIONS"
```

Prepare the mini2440 target:

```
cd mini2440
make mini2440_config
```

Compile:

```
make all
```

This will give you a "u-boot.bin". If you get an error about a missing whatever-gcc, verifiy if there is a hardcoded CROSS_COMPILE setting in the Makefile

Flashing uboot

We use the installed vivi bootloader to load uboot via usb. Power the board off, set the NOR switch to on, power the board on. You'll get the vivi output on the serial console. Press 'q' to go to the shell and do the "load flash" on address 0 with the size of "u-boot.bin" (238232 bytes in this example):

```
[q] Goto shell of vivi
Enter your selection: q
Supervivi> load flash 0 238232 u
USB host is connected. Waiting a download.
```

Use the "s3c2410_boot_usb" utility (http://mini2440.googlecode.com/files/s3c2410_boot_usb-20060807.tar.bz2) to upload the "u-boot.bin" via usb:

```
./s3c2410_boot_usb u-boot.bin
```

You'll see something like this as output, ignore the error message:

```
csum = 0x9a6e
send_file: addr = 0x33f80000, len = 0x0003a298
Error downloading program
```

and this on the serial console:

```
Now, Downloading [ADDRESS:30000000h,TOTAL:238242]
RECEIVED FILE SIZE:  238242 (232KB/S, 1S)
Downloaded file at 0x30000000, size = 238232 bytes
Found block size = 0x0003c000
Erasing...    ... done
Writing...    ... done
Written 238232 bytes
Supervivi>
```

Then you see

```
MINI2440 # saveenv

Saving Environment to NAND...

Erasing Nand...nand_erase: attempt to erase a bad block at page 0x00000260
```

Try

```
MINI2440 # nand scrub
...
Really scrub this NAND flash? <y/N>
```

press **y and enter**.

After erasing make

```
MINI2440 # nand createbbt
```

press **y and enter**.

Now you can save the environment

```
 MINI2440 # saveenv
```

Now power off the board, set the NOR switch back to off, and power on the board. We verify the version and then prepare config storage in flash:

```
MINI2440 # version

U-Boot 1.3.2-dirty-moko12 (Mar  8 2009 - 13:50:19)

MINI2440 # dynenv set 40000
```

```
device 0 offset 0x40000, size 0x3fc0000
45 4e 56 30 - 00 00 04 00

MINI2440 # saveenv
Saving Environment to NAND...
Erasing Nand...Writing to Nand... done

MINI2440 # reset
```

Configure uboot

Do a "***printenv***" to show active settings. Adjust the IP for uboot and the tftp server. We need proper IP connectivity to load the kernel with tftp. "***setenv***" is your friend. "***saveenv***" stores the setting to flash.

Kernel

Compilation

Fetch from git:

```
mkdir kernel ; cd kernel
git clone git://repo.or.cz/linux-2.6/mini2440.git
cd mini2440
mkdir -p ../kernel-bin
```

Generate .config from mini2440 template:

```
CROSS_COMPILE=arm-none-linux-gnueabi- ARCH=arm make O=../kernel-bin/
mini2440_defconfig
```

Compile kernel and modules:

```
CROSS_COMPILE=arm-none-linux-gnueabi- ARCH=arm make O=../kernel-bin/
```

Generate "uImage" with the "mkimage" tool:

```
../uboot/mini2440/tools/mkimage -A arm -O linux -T kernel -C none -a
0x30008000 -e 0x30008000 -d ../kernel-bin/arch/arm/boot/zImage
/tftpboot/uImage
```

Flash kernel

Verify partitions, we need the offset for the kernel partition (probably 0x60000):

```
mtdparts
```

Erase the kernel partition:

```
nand erase 60000 <padded size>
```

Load the kernel with uboot into ram:

```
tftp 0x32000000 uImage
```

Note the size, pad it to 512 byte and use the nand write command to flash it:

```
nand write 32000000 60000 <padded size>.
```

Configure uboot

We use a nfs chroot for now:

```
setenv bootargs console=ttySAC0,115200 noinitrd init=/sbin/init
mini2440=1tb ip=192.168.80.12 root=/dev/nfs rw
nfsroot=192.168.80.1:/home/public/arm
```

This is helpful for automatic booting:

```
setenv bootcmd 'nboot.e kernel ; bootm'
```

Setting up embedian

Follow this nice howto from BusError http://code.google.com/p/mini2440/wiki/Emdebian

Troubleshooting

Write errors

When you have bad blocks in NAND, you'll get write errors. This gives more information: http://wiki.openmoko.org/wiki/NAND_bad_blocks

Clear the flash (do not reset the board till uboot is reinstalled, or you need to use NOR/vivi to reinstall uboot via usb):

```
nand scrub
```

Create new bad blocks table:

```
nand createbbt
```

Reflash uboot (use the size in the output from the tftp command and pad it to hit 512 byte boundaries for the nand write command)

```
tftp 0x32000000 u-boot.bin
nand write 32000000 0 <size of u-boot.bin>
reset
```

Create dynamic partitions:

```
dynpart
```

```
dynenv set u-boot_env
saveenv
reset
```

And then use write.e to write to the mtdparts, e.g.:

```
tftp 0x32000000 uImage
nand write.e 32000000 kernel
nboot.e kernel
```