Building U-Boot and the Linux Kernel

Chris Jenkins - Genesi USA, Inc

Terminology

- “Das U-Boot” or “U-Boot”: Universal Bootloader, used to boot Linux kernel on Drónov Slimbook
- “u-boot.bin”: name of the executable U-Boot binary
- “uImage”: Linux kernel image in U-Boot bootable format
- “boot script”: U-Boot shell scripts used to boot Linux on the Drónov Slimbook

Prerequisites: Software

- This presentation assumes that you are cross-compiling on an Ubuntu-based workstation
- GNU Make: Build tool, allows configuring which sources are included in the kernel
  - Install: `sudo apt-get install make`
- GCC Cross-Compiler and Linker: For compiling / linking software for architecture other than native.
  - For best results, use gcc version 4.4
  - Can be downloaded from Mentor Graphics here, version `arm-2010q1`
    - You will need to provide a valid email to get the download link
  - Add `${PATH_TO_DOWNLOAD}/arm-2010q1/bin` to system `PATH` variable
  - Test with: `arm-none-eabi-gcc --version`
    - May need to set executable permissions for programs in bin folder:

      `chmod u+x arm-2010q1/bin/*`
- Debian kernel packager (`make-kpkg`)

Prerequisites: GitHub

- U-Boot & Drónov Slimbook Linux Kernel are available on open GitHub repositories
- U-Boot:
  - `git clone https://github.com/genesi/u-boot-upstream.git`
  - `git checkout -t efikamx-2013.04`
- Linux Kernel:
  - `git clone https://github.com/genesi/linux-shortbus.git`
  - `git checkout -t slimbook`

Building U-Boot

- Configure U-Boot for mx53_efikasb
  - `make ARCH=arm CROSS_COMPILE=arm-none-eabi- mx53_efikasb_config`
  - `ARCH` is system architecture, `CROSS_COMPILE` is the name of the cross-compiler you are using
- Build U-Boot:
  - `make ARCH=arm CROSS_COMPILE=arm-none-eabi- -j3`
  - Option “-j*n*” to use *n* threads
- Should create `u-boot.bin` at the root of the repo, this is the file you need
- We don’t anticipate that you will need to make changes to the code here or need to build this yourself unless you wish to customize the boot process extensively.

Building the Linux Kernel

- Configure the kernel for `mx53_efikasb`
  - `make ARCH=arm CROSS_COMPILE=arm-none-eabi- mx53_efikasb_defconfig`
  - The current config file lives in `arch/arm/configs`
- Build uImage
  - `make ARCH=arm CROSS_COMPILE=arm-none-eabi- uImage -j3`
- Build kernel modules
  - `make ARCH=arm CROSS_COMPILE=arm-none-eabi- modules -j3`
- Install the kernel modules
  - `make ARCH=arm CROSS_COMPILE=arm-none-eabi- INSTALL_MOD_PATH=./install modules_install -j3`
- `uImage` will be in `arch/arm/boot/`
- If you encounter compile-time issues, first make sure the `arm-none-eabi-*` commands are in your `PATH` variable

Building Kernel `.debs`

If you want to package your changes to the Linux kernel for other users of your system to download, you will need to use `make-kpkg`.

- Install: `apt-get install kernel-package`
- Command for building `.debs`:

```
make distclean
cp arch/arm/configs/mx53_efikasb_defconfig .config
DEB_HOST_ARCH=armel fakeroot make-kpkg --arch arm --cross-compile arm-none-
eabi- --initrd kernel_image kernel_headers modules_image kernel_source
```

  - The arguments at the end specify what to build to build
  - `DEB_HOST_ARCH` gives the arch. family, `--arch` gives the subdirectory of `arch/` to use
  - Produced `.debs` will be in the parent directory

Testing U-Boot on SD

We do not anticipate that you will need this - the Bamboo plans creates an image with U-Boot already installed correctly. If however you do need to test booting with a modified U-Boot, use the command below. You will need an inserted (and unmounted) SD card with the correct file system partitioning.

- `sudo dd if=${PATH_TO}/u-boot.bin of=/dev/mmcblk0p2 bs=1024 seek=32`
  - “bs” is block size, in this case 1,024 bytes or one kilobyte
  - “seek” says to skip over the first 32 blocks (in our case kilobytes) of the output device. The first 32 kilobytes is where the STM Loader lives.

It is **very** important that you execute this command correctly, otherwise you may overwrite the STM loader or the Linux Kernel boot partition on the SD card!

Testing the Kernel on SD

- Mount an SD card with the `maverick-installer` image and open MMCBoot partition in file browser
- Create new file `boot.scr` in MMCBoot by making a copy of `boot.ubi.scr`
  - From terminal: `cp boot.ubi.scr boot.scr`
  - Our U-Boot will automatically run any file named `boot.scr` on a recognized boot medium, checking SD first and then NAND.
- Make a backup of the existing uImage
  - `mv uImage-2.6.38.3-slimbook uImage-2.6.38.3-slimbook.bak`
- Copy locally built uImage to MMCBoot
  - `cp ${PATH_TO_KERNEL}/arch/arm/boot/uImage uImage-2.6.38.3-slimbook`
  - You **must** name your uImage `uImage-${current_kver}` (which right now is `2.6.38.3-slimbook`) or it will not work: `boot.ubi.scr` will automatically try to load the uInitrd and uImage in the same directory it lives in with the correct name

Editing Bootscripts

When you are testing changes to the Linux kernel, you may find it useful to change the boot args to e.g. enable early prints, increase the ring buffer, or increase the log level at boot. To do this:

- Open `MMCSystem/home/installer` of installer image in a file browser
- Make a copy of file `boot.ubi.script` (e.g. `boot.ubi-debug.script`) and open in your favorite text editor
  - Note the file extension - our convention is `.script` for the source and `.scr` for compiled
- Make changes to variable `bootargs_base` (usually adding an argument)
- Open a terminal and type this command (requires `u-boot-tools`):
  - `mkimage -A arm -T script -C none -d boot.ubi-debug.script boot.scr`
  - Make sure you make a backup of your own `.scr` file (`boot.ubi-debug.scr`)
  - For more details, read the “U-Boot on Efika MX53” documentation

genesi