Setting up Slackware ARM 14.0 on the OLinuXino A10S Micro from scratch

This document can also be found at
http://www.malaya-digital.org/setting-up-slackware-arm-14-0-on-the-olinuxino-a10s-from-scratch/

NOTE: The documentation below needs updating. I recommend that you use the link I've just given above as it leads to updated documentation. I'll have the text below updated when time permits.

Setup of development environment

- Create a VirtualBox VM for Debian 7.2.0 i386. I allocated 1024MB of RAM to it. As for the virtual hard drive space, I allocated 8GB. I also recommend that you disable audio support for the VM.
- Download the Net Install ISO image for Debian 7.2.0 i386. Boot the mentioned ISO in the VirtualBox VM you've created. I used the “Install” option in the installer boot menu.
- This set of instructions will work with everything deselected under “Software selection.” You may select software to your taste during installation.
- After the installation, the guest OS will automatically get an IP Address via DHCP.
- As root in your guest OS:

```
# apt-get update
# apt-get upgrade
```

- Install OpenSSH client and server software as root in your guest OS if this will make file transfers and access to your development environment convenient for you:

```
# apt-get install openssh-client openssh-server
```

- As root in your guest OS:

```
# apt-get install ncurses-dev uboot-mkimage build-essential git
# cd ~
# wget
https://launchpad.net/linaro-toolchain-binaries/trunk/2013.01/+download/gcc-
linaro-arm-linux-gnueabihf-4.7-2013.01-20130125_linux.tar.bz2
# tar xjfv gcc-linaro-arm-linux-gnueabihf-4.7-2013.01-20130125_linux.tar.bz2
```

- Add toolchain directory to $PATH in your guest OS(You may want to add this to /etc/profile . Appending it will do.):

```
export PATH=/root/gcc-linaro-arm-linux-
gnueabihf-4.7-2013.01-20130125_linux/bin:$PATH
```

- Restart your guest OS as root:

```
# shutdown -r now
```

Compiling the kernel (Copied from http://olimex.wordpress.com/2013/06/19/building-linux-for-a10s-from-scratch/ and http://olimex.wordpress.com/2013/10/28/building-debian-sd-card-for-linux-with-kernel-3-4-from-scratch-for-a10s-olinuxino-micro/ with some modifications)

- In the development environment you've made above, login as root. Then get the kernel source code:

```
# cd ~
# git clone https://github.com/linux-sunxi/linux-sunxi
```

- Note that I'm using the revision below:

```
# cd ~/linux-sunxi/
# git rev-parse --verify HEAD
9ee9fc5f0988df5677f0f142b5b88a8988d283d7
```

So, to checkout the mentioned revision:

```
# cd ~/linux-sunxi/
# git checkout 9ee9fc5f0988df5677f0f142b5b88a8988d283d7
```

- Do a “make clean”:

```
# cd ~/linux-sunxi
# make clean
```

- Download a10s_defconfig:

```
# cd ~/linux-sunxi
# wget http://www.malaya-digital.org/a10s_defconfig
```

- Move this file to ~/linux-sunxi/arch/arm/configs/ . These are to be done in your development environment as root:

```
# mv a10s_defconfig ~/linux-sunxi/arch/arm/configs/
```

- Execute the following under the linux-sunxi directory in your development environment as root:

```
# cd ~/linux-sunxi/
# make ARCH=arm a10s_defconfig
```

- Configure the kernel in your development environment as root:

```
# make ARCH=arm menuconfig
```

- HINT: If you're having problems with Logitech wireless keyboards and mice, don't build the following (for Linux 3.4.61):
  - Device Drivers→HID Devices→Special HID drivers→< > Logitech Unifying receivers full support

- HINT: To verify OTG is enabled(for Linux 3.4.61):
  - Under Kernel Configuration, this must be built in the kernel:
    - Device Drivers→USB support→[*] SUNXI USB2.0 Dual Role Controller Support
  - Also, this must be built in the kernel, too:
    - Device Drivers→USB support→SUNXI USB2.0 Dual Role Controller Support→[*] SUNXI USB2.0 Manager
  - And make sure of the following:
    - Device Drivers→USB support→SUNXI USB2.0 Dual Role Controller Support→SUNXI USB2.0 Manager→USB0 Controller support (otg support)→(X) otg support
- HINT: To verify Ethernet is enabled(for Linux 3.4.61):
  - Under Kernel Configuration, this must be built in the kernel:
    - Device Drivers→Network device support→Ethernet driver support→<*> Allwinner Ethernet MAC support
- Note that before compiling kernel, you have to patch it:
- Download the patch hcd_axp-md.patch from http://www.malaya-digital.org/hcd_axp-md.patch :

```
# cd ~/linux-sunxi/
# wget http://www.malaya-digital.org/hcd_axp-md.patch
```

- Copy files:

```
# cp drivers/usb/sunxi_usb/hcd/hcd0/sw_hcd0.c
drivers/usb/sunxi_usb/hcd/hcd0/sw_hcd0a.c
# cp drivers/usb/sunxi_usb/hcd/hcd0/sw_hcd0.c
drivers/usb/sunxi_usb/hcd/hcd0/sw_hcd0b.c
```

- Apply the patch:

```
# patch -p0 < hcd_axp-md.patch
```

- Copy a file again:

```
# cp drivers/usb/sunxi_usb/hcd/hcd0/sw_hcd0a.c
drivers/usb/sunxi_usb/hcd/hcd0/sw_hcd0.c
```

- Edit ~/linux-sunxi/arch/arm/plat-sunxi/include/plat/i2c.h . Find “I2C0_TRANSFER_SPEED”. Define “100000” for I2C1_TRANSFER_SPEED and up. These are to be done in your development environment as root. For example, the desired edit is:

```
#define I2C0_TRANSFER_SPEED (400000)
#define I2C1_TRANSFER_SPEED (100000)
#define I2C2_TRANSFER_SPEED (100000)
#define I2C3_TRANSFER_SPEED (100000)
#define I2C4_TRANSFER_SPEED (100000)
```

- You can now compile the kernel and its modules in your development environment as root.

```
# cd ~/linux-sunxi/
# make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- -j4 uImage
# make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- -j4 INSTALL_MOD_PATH=out
modules
# make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- -j4 INSTALL_MOD_PATH=out
```

```
modules_install
```

- In your development environment, your kernel will be “~/linux-sunxi/arch/arm/boot/uImage”. And your modules will be found at “~/linux-sunxi/out/lib/modules/3.x.xx” where 3.x.xx is kernel version (eg: “3.4.61+”).

Compiling Uboot (Copied from http://olimex.wordpress.com/2013/06/19/building-linux-for-a10s-from-scratch/ and http://olimex.wordpress.com/2013/10/28/building-debian-sd-card-for-linux-with-kernel-3-4-from-scratch-for-a10s-olinuxino-micro/ with some modifications)

- Note: The Allwinner Linux-Sunxi community uboot is maintained by Henrik Nordstrom aka hno on Freenode irc. You can find him in #linux-sunxi or #olimex channels. If something with uboot is broken, he is your man.
- Download the uboot sources from GitHub repository.

```
# cd ~
# git clone https://github.com/linux-sunxi/u-boot-sunxi
```

- After the download, you should have a new directory.

```
# cd ~/u-boot-sunxi/
```

- Note that I'm using the revision below:

```
root@debian:~/u-boot-sunxi# git rev-parse --verify HEAD
8a4621c488f33089d831168bfa5bae210a5684c8
```

- Edit ~/u-boot-sunxi/include/configs/sunxi-common.h . Look for the following:

```
        "setargs=" \
          "if test -z \\\\\"$root\\\\\"; then"\
            " if test \\\\\"$bootpath\\\\\" = \"/boot/\"; then"\
              " root=\"/dev/mmcblk0p1 rootwait\";"\
            " else" \
              " root=\"/dev/mmcblk0p2 rootwait\";"\
            " fi;"\
          " fi;"\
```

- Change “ root=\”/dev/mmcblk0p2 rootwait\“;”\“ to the following:
  - ” root=\“/dev/mmcblk0p2 ro rootwait\”;“\
- With the following command, you can start the uboot build:

```
# cd ~/u-boot-sunxi/
# make distclean CROSS_COMPILE=arm-linux-gnueabihf-
# make a10s-olinuxino-m CROSS_COMPILE=arm-linux-gnueabihf-
```

- At the end of the process, you can check if everything is OK by:

```
# ls u-boot-sunxi-with-spl.bin
```

- If you got this file, well done so far.

Format and setup the SD-card (Copied from http://olimex.wordpress.com/2013/06/19/building-linux-for-a10s-from-scratch/ and http://olimex.wordpress.com/2013/10/28/building-debian-sd-card-for-linux-with-kernel-3-4-from-scratch-for-a10s-olinuxino-micro/ with some modifications)

- We suggest that you use a 4GB class 10 micro SD card. But you can use any micro SD card between 2GB and 16GB.
- First, we have to make the correct card partitions. This is done with fdisk.
- Plug the micro SD card into your SD card reader. Then, enter in the terminal:

```
# ls /dev/sd
```

- Then press TAB twice. You will see a list of your sd devices like sda, sdb, sdc, etc. Note that some of these devices may be your hard disk, so make sure you know which one is your micro SD card before you proceed. You can damage your HDD if you choose the wrong sd device. You can do this by unplugging your micro SD card reader, and identify which ?sd? devices was removed from the list.
- Once you know which device is your micro SD card, use this text instead of the sdX name in the references below:

```
# fdisk -u=sectors /dev/sdX
```

- Then do these steps:
  - This will list your partitions:
    - p
  - If there are already partitions on your card, do:
    - d 1
  - If you have more than one partitition, delete them all.
  - Create the first partition. It should start from 2048 and end at 34815:
    - n p 1
  - Create the second partition:
    - n p 2 enter enter
  - List the created partitions:
    - p
  - If you did everything correctly on a 4GB card, you should see something like:

```
Disk /dev/sdX: 3980 MB, 3980394496 bytes
123 heads, 62 sectors/track, 1019 cylinders, total 7774208 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x00000000

Device Boot Start End Blocks Id System
```

```
/dev/sdg1 2048 34815 16384 83 Linux
/dev/sdg2 34816 7774207 3869696 83 Linux
```

- Write changes to the micro SD card.
  - w
- Now, we have to format the file system on the card:
- The first partition should be vfat as this is the FS which the Allwinner bootloader understands.

```
# mkfs.vfat /dev/sdX1
```

- The second partition should be a normal Linux EXT4 FS:

```
# mkfs.ext4 /dev/sdX2
```

Installing the kernel, Slackware ARM 14.0 mini root filesystem, and kernel modules

- Proceed to copy the ”~/linux-sunxi/arch/arm/boot/uImage“ kernel you've compiled in the development environment into the first filesystem of the micro SD card.
  - Mount the first partition:

```
# mkdir /mnt/olinuxino0
# mount /dev/sdX1 /mnt/olinuxino0 # Substitute the appropriate value for X
in /dev/sdX1
```

- Then copy the kernel uImage to the first filesystem of the micro SD card.

```
# cp uImage /mnt/olinuxino0 # Get the uImage file from the environment
you've compiled the kernel.
```

- Copy the script.bin file in /mnt/olinuxino0 . The mentioned file can be found here: https://drive.google.com/file/d/0B-bAEPML8fwlYkItdU1TTm1VN0E/edit?usp=sharing
  - Or iff you need to compile your own script.bin, you can get the fex files here in a scripts_A10s.7z archive: https://drive.google.com/file/d/0B-bAEPML8fwlY3llVDJxelY3d28/edit?usp=sharing
- Unmount /mnt/olinuxino0 :

```
# umount /mnt/olinuxino0
```

- Mount the second partition:

```
# mkdir /mnt/olinuxino1
# mount /dev/sdX2 /mnt/olinuxino1 # Substitute the appropriate value for X
in /dev/sdX1
```

- Extract the Slackware ARM 14.0 mini root filesystem in /mnt/olinuxino1 . Slackware mini root filesystem can be found here: ftp://ftp.arm.slackware.com/slackwarearm/slackwarearm-devtools/minirootfs/roots
- Delete all contents of /mnt/olinuxino1/dev/*
- Extract this file in /mnt/olinuxino1/dev : http://www.malaya-digital.org/dev.tar.gz
- Proceed to copy the generated kernel modules (”~/linux-sunxi/out/lib/modules/3.x.xx“ in the

development environment where you've compiled the kernel) in the second filesystem of the micro SD card.
    - If the /mnt/olinuxino1/lib/modules directory does not exist, create it.

```
# mkdir /mnt/olinuxino1/lib/modules
# cp -rf 3.x.xx+ /mnt/olinuxino1/lib/modules # Get the modules directory
from the environment you've compiled the kernel.
```

- Append the following in /mnt/olinuxino1/etc/fstab:

```
/dev/mmcblk0p2   /               ext4    errors=remount-ro 0       1
```

- Unmount /mnt/olinuxino1 :

```
# umount /mnt/olinuxino1
```

Write Uboot (Copied from http://olimex.wordpress.com/2013/06/19/building-linux-for-a10s-from-scratch/ and http://olimex.wordpress.com/2013/10/28/building-debian-sd-card-for-linux-with-kernel-3-4-from-scratch-for-a10s-olinuxino-micro/ with some modifications)

- Note that you have to write u-boot-sunxi-with-spl.bin in /dev/sdX (not sdX1 or sdX2).

```
# dd if=u-boot-sunxi-with-spl.bin of=/dev/sdX bs=1024 seek=8
# sync
```

NB: When you boot OLinuXino using the micro SD card with Slackware 14.0 for ARM, the "root" user has the password "password" by default.

Sources

- Original source: http://olimex.wordpress.com/2013/06/19/building-linux-for-a10s-from-scratch/
- Original source: https://github.com/linux-sunxi/u-boot-sunxi/wiki
- Original source: http://olimex.wordpress.com/2013/10/28/building-debian-sd-card-for-linux-with-kernel-3-4-from-scratch-for-a10s-olinuxino-micro/

- Originally written by Michael Balcos

howtos, hardware, arm, author michael balcos