April 28, 2017

Setting up a BBB for network boot

This is largely a matter of finding the file uEnv.txt and modifying it.

On a recent Debian based system, the file you want is /uEnv.txt. It is **not** /boot/uEnv.txt. That file is just there to mislead you. There is probably **not** a file /uEnv.txt on a pristine system, but you are going to put one there. Make it look like this:

```
optargs=quiet
uenvcmdx=echo Booting via tftp (Xinu); setenv saloadaddr 0x81000000; setenv ipaddr 192.168.0.54; setenv serverip 192.168.0.5; tftpboot ${salo
uenvcmd=run uenvcmdx
```

Or maybe:

```
optargs=quiet
uenvcmdx=echo Booting via tftp (kyu.bin); setenv saloadaddr 0x80000000; setenv ipaddr 192.168.0.12; setenv serverip 192.168.0.5; tftpboot ${s
uenvcmd=run uenvcmdx
```

Some early Debian systems had a FAT partition that held uEnv.txt. MLO and u-boot.img were not in here, and eventually they realized that this partition was not needed at all and did away with it. But this is a place to look if the above does not seem to make sense. One advantage of FAT partitions like this is that they can be "exposed" via USB and mounted (once linux is running). This setup also has a misleading file in /boot/uEnv.txt. There is even more confusion in this setup to be had. The fat partition with the uEnv.txt you want is not even mounted normally. If you want to access the uEnv.txt file of interest from inside the linux running on the BBB, you need to mount the partition by something like this:

```
mount /dev/mmcblk0p1 /mnt
```

On old Angstrom based systems there was a FAT partition with MLO and uEnv.txt in it (and U-boot.img as well for that matter). The contents of uEnv.txt are the same, only the file location is changed.

Load address

This has to correspond to how the executable is linked (usually specified in an "lds" file these days). I used to avoid the start of RAM for no particular reason other than general paranoia. Now I let U-boot (which lives at 0x80800000 on the BBB) just load it right to the start of RAM at 0x80000000 and it works fine. I used to load it to 0x80300000, which also worked fine, but left a big (3M) hole at the start of RAM.

Running a static image

On my Debian based boards, I make /uEnv.txt look like this:

```
uenvcmdx=echo Booting kyu (kyu.bin) from eMMC; setenv saloadaddr 0x80000000; load mmc 1:1 ${saloadaddr} /kyu.bin; go ${saloadaddr}
uenvcmd=run uenvcmdx
```

And I put kyu.bin right in the root at /kyu.bin alongside the /uEnv.txt file. It works great. This gives you a "turnkey" embedded system that doesn't need the network to boot. The bin file is the exact same file I would have used for TFTP booting. The bytes are the same, and they end up in the same place.

Getting back to running linux again

It is always good to know you haven't burned your bridges. And sometimes you need to do maintenance (like changing IP numbers in the network boot setup).

You need a serial console. When U-boot tells you to type a key to interrupt it, type a key. Then at the U-boot prompt, type:

```
setenv bootenv x.txt
boot
```

Any file that does not exist will do; x.txt is just a suggstion. You can use "linux" or "abracadabra" or any name that does not exist for this purpose. This blocks it from running uEnv.txt, so it does its normal thing and boots linux. And just one time only. The next time you reset or power up it will go back to network booting. Unless of course you modify uEnv.txt while you are in there.

Note that this scheme can also be used to select alternate files that exist, and this could be useful, though I have never exploited it.

Getting back in tricky cases

I have taken U-boot images from a unit with the old partitioning scheme (partition 1 is a FAT partition holding MLO and u-boot.img, partition 2 is linux) and installed them using "dd" on a unit with the new partition scheme (partition 1 is linux and that is all there is). This works fine for network booting, but it will **not** boot linux. It looks for linux on partition 2, which does not exist. To make a system like this boot linux again, you need to use an SD card, mount the eMMC partitions, and install a U-boot that looks for Linux on partition 1).

But if you do monkey business like this, you deserve what you get. This is a dirty hack as an interim measure for network booting only.

Fiddling with U-boot

You need to have a serial console hooked up if you want to fiddle. U-boot has an interactive prompt with an extensive repertoire of commands.

When it encourages you to "hit any key", do so, and you will be presented with the U-boot prompt. To resume the boot sequence, type "boot" A worthy command to type is "env_print".

If you know that the "bootcmd" variable is the "script" that runs when you type boot (or when you let U-boot rattle along unmolested) then you can figure out why the above works. And you can study the following output and get other ideas.

- Output from "print env"
