RSS Subscribe: **RSS feed**
Freedom Embedded
Balau's technical blog on open hardware, free software and security

Booting Linux with U-Boot on QEMU ARM

Posted on 2010/04/12

137

In recent months I played with QEMU emulation of an ARM Versatile Platform Board (http://infocenter.arm.com/help/topic/com.arm.doc.dui0224i/index.html), making it run bare metal programs (https://balau82.wordpress.com/2010/02/28/hello-world-for-bare-metal-arm-using-qemu/), the U-Boot (https://balau82.wordpress.com/2010/03/10/u-boot-for-arm-on-qemu/) boot-loader and a Linux kernel (https://balau82.wordpress.com/2010/03/22/compiling-linux-kernel-for-qemu-arm-emulator/) complete with a Busybox-based file system (https://balau82.wordpress.com/2010/03/27/busybox-for-arm-on-qemu/). I tried to put everything together to emulate a complete boot procedure, but it was not so simple. What follows is a description of what I've done to emulate a complete boot for an emulated ARM system, and the applied principles can be easily transferred to other different platforms.

Prerequisites

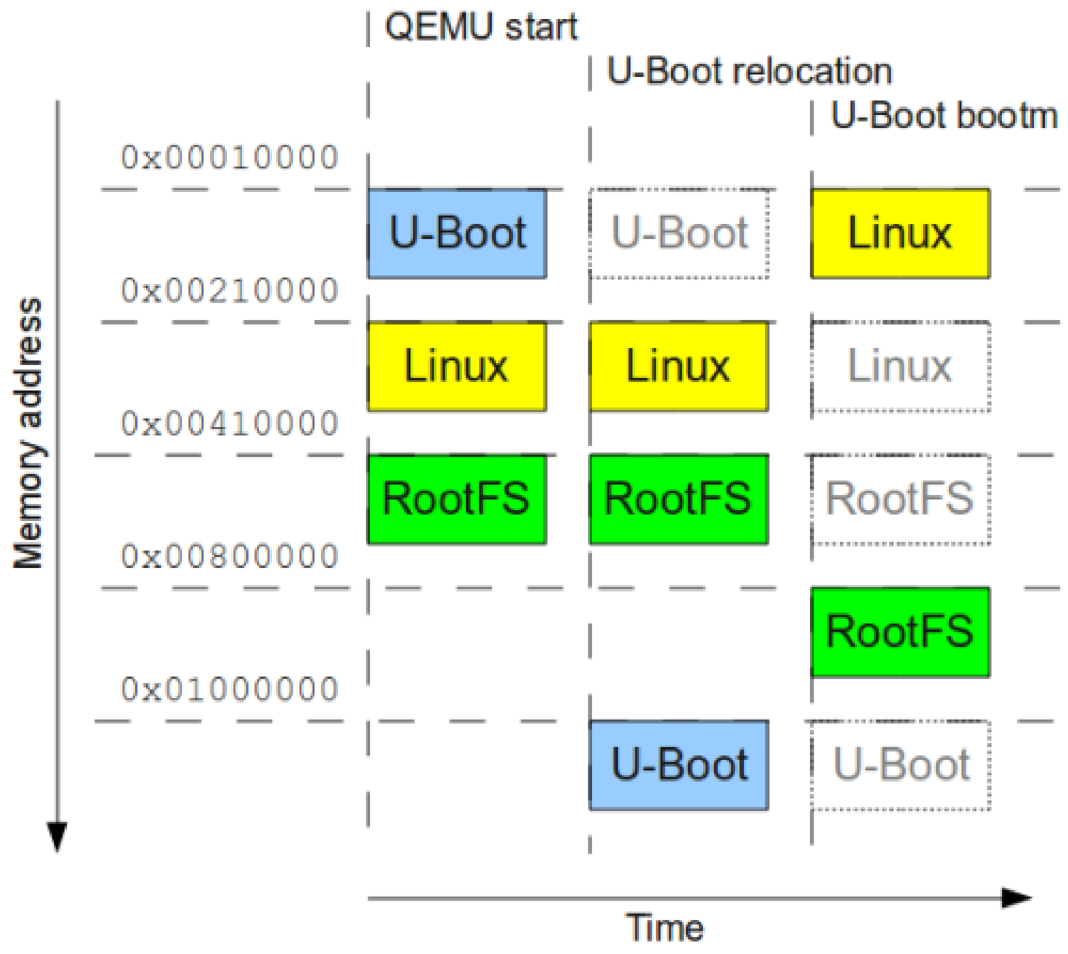
- `qemu-system-arm`: can be installed on Ubuntu with "`sudo apt-get install qemu-kvm-extras`", on Debian with "`aptitude install qemu`" as root.
- `mkImage`: can be installed with the package `uboot-mkimage`. Alternatively, it is compiled from U-Boot source.
- `arm-none-eabi` toolchain: can be downloaded from the the CodeSourcery ARM EABI toolchain (http://www.codesourcery.com/sgpp/lite/arm/portal/subscription?@template=lite) page
- `zImage`: the Linux kernel created in my previous post here (https://balau82.wordpress.com/2010/03/22/compiling-linux-kernel-for-qemu-arm-emulator/)
- `rootfs.img.gz`: the Busybox-based file system created in my previous post here (https://balau82.wordpress.com/2010/03/27/busybox-for-arm-on-qemu/)

The boot process

On real, physical boards the boot process usually involves a non-volatile memory (e.g. a Flash) containing a boot-loader and the operating system. On power on, the core loads and runs the boot-loader, that in turn loads and runs the operating system. QEMU has the possibility to emulate Flash memory on many platforms, but not on the VersatilePB. There are patches ad procedures available (http://thomas.enix.org/Blog-20081002153859-Technologie) that can add flash support, but for now I wanted to leave QEMU as it is.

QEMU can load a Linux kernel using the `-kernel` and `-initrd` options; at a low level, these options have the effect of loading two binary files into the emulated memory: the kernel binary at address `0x10000` (64KiB) and the ramdisk binary at address `0x800000` (8MiB). Then QEMU prepares the kernel arguments and jumps at `0x10000` (64KiB) to execute Linux. I wanted to recreate this same situation using U-Boot, and to keep the situation similar to a real one I wanted to create a single binary image containing the whole system, just like having a Flash on board. The `-kernel` option in QEMU will be used to load the Flash binary into the emulated memory, and this means the starting address of the binary image will be `0x10000` (64KiB).

Understanding memory usage during the boot process is important because there is the risk of overwriting something during memory copy and relocation. One feature of U-Boot is self-relocation, which means that on execution the code copies itself into another address, which by default is `0x1000000` (16MiB). This feature comes handy in our scenario because it frees lower memory space in order to copy the Linux kernel. The compressed kernel image size is about 1.5MiB, so the first 1.5MiB from the start address must be free and usable when U-Boot copies the kernel. The following figure shows the solution I came up with:



(https://balau82.files.wordpress.com/2010/04/u-boot.png)
Timeline of the memory usage

At the beginning we have three binary images together: U-Boot (about 80KiB), Linux kernel (about 1.5MiB) and the root file system ramdisk (about 1.1MiB). The images are placed at a distance of 2MiB, starting from address `0x10000`. At run-time U-boot relocates itself to address `0x1000000`, thus freeing 2MiB of memory from the start address. The U-Boot command `bootm` then copies the kernel image into `0x10000` and the root filesystem into `0x800000`; after that then jumps at the beginning of the kernel, thus creating the same situation as when QEMU starts with the `-kernel` and `-initrd` options.

Building U-Boot

The problem with this solution is that U-Boot, when configured to be built for VersatilePB, does not support ramdisk usage, which means that it does not copy the ramdisk during the `bootm` command, and it does not give any information about the ramdisk to the kernel. In order to give it the functionality I need, I patched the original source code of U-Boot before compilation. The following code is the patch to apply to `u-boot-2010.03` source tree:

```
diff -rupN u-boot-2010.03.orig/common/image.c u-boot-2010.03/common/imag
--- u-boot-2010.03.orig/common/image.c  2010-03-31 23:54:39.000000000 +0
+++ u-boot-2010.03/common/image.c   2010-04-12 15:42:15.911858000 +0200
@@ -941,7 +941,7 @@ int boot_get_ramdisk (int argc, char *ar
             return 1;
         }
 
-#if defined(CONFIG_B2) || defined(CONFIG_EVB4510) || defined(CONFIG_ARM
+#if defined(CONFIG_B2) || defined(CONFIG_EVB4510) || defined(CONFIG_ARM
         /*
          * We need to copy the ramdisk to SRAM to let Linux boot
          */
diff -rupN u-boot-2010.03.orig/include/configs/versatile.h u-boot-2010.0
--- u-boot-2010.03.orig/include/configs/versatile.h 2010-03-31 23:54:39.
+++ u-boot-2010.03/include/configs/versatile.h  2010-04-12 15:43:01.5147
@@ -124,8 +124,11 @@
 #define CONFIG_BOOTP_SUBNETMASK
 
 #define CONFIG_BOOTDELAY   2
-#define CONFIG_BOOTARGS        "root=/dev/nfs mem=128M ip=dhcp "\
-                "netdev=25,0,0xf1010000,0xf1010010,eth0"
+/*#define CONFIG_BOOTARGS      "root=/dev/nfs mem=128M ip=dhcp "\
+                "netdev=25,0,0xf1010000,0xf1010010,eth0"*/
+#define CONFIG_BOOTARGS        "root=/dev/ram mem=128M rdinit=/sbin/ini
+#define CONFIG_BOOTCOMMAND   "bootm 0x210000 0x410000"
+#define CONFIG_INITRD_TAG   1
 
 /*
  * Static configuration when assigning fixed address
```

I also changed the boot arguments (`CONFIG_BOOTARGS`)so that they are the same as those given from QEMU command line, and then added a command (`CONFIG_BOOTCOMMAND`) to start the Linux boot automatically. To apply the patch:

1. save the patch to a file, for example `~/u-boot-2010.03.patch`
2. download u-boot-2010.03 source tree (ftp://ftp.denx.de/pub/u-boot/u-boot-2010.03.tar.bz2) and extract it, for example in `~/u-boot-2010.03`
3. `cd` into the source tree directory
4. apply the patch, for example with "`patch -p1 < ~/u-boot-2010.03.patch`"

After applying the patch, U-Boot can be built as seen in my previous post (https://balau82.wordpress.com/2010/03/10/u-boot-for-arm-on-qemu/):

```
make CROSS_COMPILE=arm-none-eabi- versatilepb_config
make CROSS_COMPILE=arm-none-eabi- all
```

The building process will create a `u-boot.bin` image that supports ramdisks for the VersatilePB. Incidentally, it will also build the `mkimage` executable in the `tools` directory; it can be used instead of the one installed with Debian/Ubuntu packages.

Creating the Flash image

As I said earlier, I need to create a flash image in which the three binary images are placed at a distance of 2MiB. U-Boot needs to work with binary images wrapped with a custom header, created using the `mkimage` tool. After creating the Linux and root file system images, we can write them inside a big binary at a given address with the `dd` command. Assuming that we have in the same directory: `u-boot.bin`, `zImage` and `rootfs.img.gz`, the list of commands to run are:

```
mkimage -A arm -C none -O linux -T kernel -d zImage -a 0x00010000 -e 0x00
mkimage -A arm -C none -O linux -T ramdisk -d rootfs.img.gz -a 0x00800000
dd if=/dev/zero of=flash.bin bs=1 count=6M
dd if=u-boot.bin of=flash.bin conv=notrunc bs=1
dd if=zImage.uimg of=flash.bin conv=notrunc bs=1 seek=2M
dd if=rootfs.uimg of=flash.bin conv=notrunc bs=1 seek=4M
```

These commands do the following:

1. create the two U-Boot images, `zImage.uimg` and `rootfs.uimg`, that contain also information on where to relocate them
2. create a 6MiB empty file called `flash.bin`
3. copy the content of `u-boot.bin` at the beginning of `flash.bin`
4. copy the content of `zImage.uimg` at 2MiB from the beginning of `flash.bin`
5. copy the content of `rootfs.uimg` at 4MiB from the beginning of `flash.bin`

At the end we have a binary image, `flash.bin`, containing the memory layout that I had in mind.

Booting Linux

To boot Linux we can finally call:

```
qemu-system-arm -M versatilepb -m 128M -kernel flash.bin -serial stdio
```

The U-Boot-related messages will appear on the console:

```
U-Boot 2010.03 (Apr 12 2010 - 15:45:31)
```

```

DRAM:   0 kB
## Unknown FLASH on Bank 1 - Size = 0x00000000 = 0 MB
Flash:  0 kB
*** Warning - bad CRC, using default environment

In:    serial
Out:   serial
Err:   serial
Net:   SMC91111-0
Hit any key to stop autoboot:  0
## Booting kernel from Legacy Image at 00210000 ...
   Image Name:
   Image Type:   ARM Linux Kernel Image (uncompressed)
   Data Size:    1492328 Bytes =  1.4 MB
   Load Address: 00010000
   Entry Point:  00010000
## Loading init Ramdisk from Legacy Image at 00410000 ...
   Image Name:
   Image Type:   ARM Linux RAMDisk Image (uncompressed)
   Data Size:    1082127 Bytes =  1 MB
   Load Address: 00800000
   Entry Point:  00800000
   Loading Kernel Image ... OK
OK

Starting kernel ...

Uncompressing Linux... done, booting the kernel.
```

Then the Linux kernel will execute inside the emulated screen and the message "`Please press Enter to activate this console`" will appear, indicating that the root file system is working and so the boot process completed successfully. If something doesn't work, one can always check that the system works without U-Boot, with the following command:

```
qemu-system-arm -M versatilepb -m 128M -kernel zImage -initrd rootfs.img.
```

The kernel should uncompress and execute up to the activation of the console.

This procedure has room for improvements and optimizations, for example there's too much memory copying here and there, where mostly everything can be executed in place. It is anyway a nice exercise and a good starting point that reveals interesting details about the boot process in embedded systems. As usual, this is possible mainly due to the fact that all the tools are free and open source.

About these ads (http://wordpress.com/about-these-ads/)

Tagged: ARM, busybox, codesourcery, debian, linux, open source, qemu, u-boot, uboot, ubuntu
Posted in: Embedded (https://balau82.wordpress.com/category/software/embedded-software/), Hardware (https://balau82.wordpress.com/category/hardware/)

137 Responses "Booting Linux with U-Boot on QEMU ARM" →

Howimboe

2010/04/19

Thx for the tuto, works perfectly with custom rootfs, host running last Ubuntu.

Jon Rios

2010/09/29

First of all, thank you very much for all the tutorials you have written in the blog, they have been very useful for me.

I'm trying to configure an ARM emulation enviroment consisting in QEMU+U-Boot+Busybox.

I followed all the steps to create the linux kernel image (linux kernel version 2.6.34.7) and the busybox file system (busybox version 1.17.2) with the other tutorials you have in the blog.

Also I created the flash.bin image as described in this post.

The problem is that when I try to execute qemu with the flash as kernel, I get nothing but a black screen.

No errors
No u-boot messages
No kernel messages

Do you know what can I do or where the problem can be?

Thank you for your attention

Balau

2010/09/29

The screen should be black at first, but the terminal should show the autoboot countdown and some other messages. If you see nothing on the terminal, then surely the problem is not in Linux or Busybox. Are you using u-boot version 2010.03? Try to make just u-boot work, once you have created `u-boot.bin`, with the following command:

```
$ qemu-system-arm -M versatilepb -m 128M -kernel u-boot.bin -serial stdio
```

The autoboot should fail and it should display a prompt on the terminal.

Check also if the following command works (it skips u-boot):

```
$ qemu-system-arm -M versatilepb -m 128M -kernel zImage -initrd
rootfs.img.gz -append "root=/dev/ram mem=128M rdinit=/sbin/init" -serial
stdio
```

Jon Rios

2010/09/29

Hi, thanks for the fast response.

I am using U-boot 2010-03 and also I tried with 2010-09.

Each component separately works fine. I mean, executing only u-boot fails on loading the image but it shows the prompt.

Also running busybox with the linux kernel works fine.

With this info, the only thing I think it could be wrong may be the flash.bin file, but I'm sure I made it right as explained in this post.

Balau

2010/09/29

Try to run:

```
$ hexdump -C u-boot.bin |head >u-boot.hex
$ hexdump -C flash.bin |head >flash.hex
$ diff u-boot.hex flash.hex && echo OK
```

The two dumps should be equal.

Is any of the binaries composing the flash bigger than 2MB? Because I assumed they were smaller and spaced them accordingly on the flash.

Jon Rios

2010/09/29

Ok, I found the problem. It was I was using the dd command wrong. When the pc is creating the 6M file, the console doesn't write anything. I interpreted this as I must enter then the rest of dd commands. And when I end entering this, I was terminating the first proccess.

The result was a blank file.

Thank you for helping me realize this and congratulations for the great info you have in the blog. All is workin fine now-

Balau

2010/09/30

Glad to help!

Robert Smith

2010/10/01

Hello,

Thank you for your tutorial.

It seems to me that something is missing in the text of your u-boot patch.
My browser shows 29 lines and last two of them are open comment:

28 /*
29 * Static configuration when assigning fixed address

May be something wrong with my browser, I use Firefox 3.6.10 under Ubuntu 9.10?
Can you clarify.

Thanks

Balau

2010/10/01

It’s just context that helps the “patch” program to verify that it is indeed modifying the right piece of code. The lines that are actually changed in the patch are those with “+” or “-” as the first character of the line. See http://en.wikipedia.org/wiki/Diff#Context_format

satya prakash

2010/12/29

hi everyone,

Currently i’m working on arm-linux(embedded system).
The process of my booting up is that initially i have got a bootloader(u-boot) which initializes kernel and then kernel takes care of rest. But can anyone please tell me a more regarding the basics:
1) What happens when initially the board is powered on or reset(beginning from cpu)?
2)bootloader initializes kernel, but who initializes bootloader? I mean something should be there which is activated by default on being powered up or reset, which in turn must be starting bootloader? (this is what i think)

I have tried to search in google regarding this, but everywhere i get the results directly beginning from uboot, but who starts uboot, that i haven’t found upto now.
Can anyone please help me regarding these basics?
If possible, please provide me with the links where i can get these details both from hardware and software point of view.

thanking in advance,

With regards,
sattu

Balau

2010/12/29

The answer depends on the hardware system you are using. For example if you are using a BeagleBoard it’s different than using a RealView versatile board. QEMU has its own implementation of the boot, that prepares the minimum for a Linux kernel and then jumps to address `0x10000`, and is very different than what real hardware does.
Usually you have a ROM at a particular address (can be `0x00000000` or `0xFFFF0000`, …) that executes when the hardware is reset. It should turn on the clocks and it should configure the memory interfaces, then it can jump to a fixed address or load some code from Flash and execute it. This procedure and its code is very specific to the architecture so you should find the information on the manual of the hardware platform.
Here are a couple of links for the Beagleboard:

The Android boot process from power on

http://www.hindawi.com/journals/wcn/2011/530354.fig5.html

satya prakash

2010/12/30

Oh sorry balau, thanks for your reply but i forgot to mention the details regarding the board. It's using a soc called S3C2440 having arm9 architecture. Can you please send me a mail i.d of yours(yahoo, gmail or rediff) so that i can mail you the introductory pdf regarding the board i'm working upon.

with regards,
sattu

satya prakash

2010/12/30

By the way balu, you have sent me the link regarding android boot process. As far as i know, Android is nothing but a different flavour of linux. So, can you send me any links where it will be mentioned regarding the boot process of embedded linux(2.6.30.4) specifically. The link that you have sent regarding the android is really conceptual and to the core. As far as i feel, almost the same principle would be working for linux. Still, if you can explain me regarding the exact booting process of embedded linux or atleast send me some links, dat would really be great. Looking forward for your help.

Balau

2010/12/30

Dear Sattu,
my mail is in my About Me page.
I never looked in details the internal process of Linux booting, but I'm expecting that it's very similar (if not the same) for Android and for any Linux distribution for ARM.
The Linux kernel itself contain some information that can be useful:
ARM Booting
Samsung-S3C24XX documents

nguyễn văn đạt

2011/01/18

i thank you, but why when i do follow your instruction then it error message:

"Failed to execute /init
Kernel panic – not syncing: No init found. Try passing init= option to kernel."

you can see picture

Balau

2011/01/18

First things that come to mind:
– have you set the execution bit of "sbin/init" with "cmod +x" ?
– have you tried this simpler exercise, and does it work?

Adithya

2011/01/25

Hello,

Can u plz suggest me a way, by which we can pass a Device Tree from U-Boot to the linux kernel on ARM platform. I want to pass only a subset of the available hardware to the linux kernel.

Thank you in advance.

nguyễn văn đạt

2011/01/25

- have you set the execution bit of "sbin/init" with "cmod +x" ?

by how set execution when Qemu cant input keyboard ?

Balau

2011/01/25

I don't know how to do it currently but I know they are actively working on it, especially Grant Likely of Secretlab.
The last update I have read for Device Tree support on ARM is the following presentation: ARM Device Tree status report.

Balau

2011/01/25

nguyễn văn đạt, sorry, I meant on your host computer, before creating the filesystem with "cpio".

Mohd Anuar

2011/02/21

Nice tutorial! Good job Balau! Even myself is a MS Windows programmer/researcher (allergies to Linux) capable to complete this short course. huh.. it's take 1 1/2 weeks to complete (+ understand Linux + Embedded).

william estrada

2011/03/01

Hello,

I'm trying to test my embedded kernel under qemu. Having problems creating the DRAM file.
I am using this script:

```
sudo virsh net-start default

rm DRAM > /dev/null 2>&1

dd if=/dev/zero of=DRAM bs=1024 count=256

sudo qemu-system-arm -M versatilepb -m 256 \
-pflash DRAM \
-nographic -kernel u-boot \
-net nic,macaddr=00:16:3e:00:00:01 \
-net tap,vlan=0,ifname=vnet1
```

But I get this error:

```
U-Boot 1.1.6 (Feb 27 2011 – 12:25:20)

DRAM: 0 kB
Flash: 0 kB
*** Warning – bad CRC, using default environment

In: serial
Out: serial
Err: serial
```

Do you have any suggestions??

Balau

2011/03/01

If your problem is the "-pflash" support, you can try these instructions : Using U-Boot and Flash emulation in Qemu
Other than that, you can try a newer u-boot version like the 2010.03, since you are using a version 1.1.6) of 5 years ago and maybe they fixed something.
You can also see that I have similar errors in my output (no DRAM and no Flash), but the execution continues correctly because everything is in RAM.

Vidur Garg

2011/03/13

Hello Balau ,
Thanks a lot for all the posts ! They've been highly informative.
Well, I was trying out the different meathods booting linux , and for the most part it worked fine .
In this case however , while execting qemu , i don't get the auto boot message but get the uboot promt instead.
So I thought i'll enter bootm 0x0021000 after looking into the figure. This worked and kernel bootup continued. When it came to the rootfs part , the rootfs wasn't detected. So instead i used bootm 0x0021000 0x0041000 .. didnt solve the problem .
Could you please help me out ?

Balau

2011/03/13

I think you miss some "zeros" at the end of the address you are using. they should be "0x210000″ and "0x410000″
In case you have only typed it wrong in this comment but you are using the right addresses in your tests:
What does it say inside the U-Boot prompt if you run "iminfo 0x210000″ and "iminfo 0x410000″?
If U-Boot doesn't auto-boot maybe there's something wrong in the patching or in the compilation. It should at least give you the same messages that you get when you run "bootm 0×00210000 0×00410000″ by yourself. Can you check by using "-kernel u-boot.bin" in the qemu-system-arm command instead of "flash.bin"? It should fail but give you some error messages about booting. In this case, the problem is the creation of the "flash.bin" image with "dd" command. Also, are all the three files that compose "flash.bin" below 2MiB in size?

Vidur Garg

2011/03/14

Hello , thanks for the reply . The zeros were just a typo.
Its now working .. the patching was the issue. instead this is what i did :
After looking through the patch file i modified the image.c file and added : || defined (CONFIG_VERSATILE)
and in the versatile.h i added
#define CONFIG_BOOTARGS "root=/dev/ram mem=128M rdinit=/sbin/init"
#define CONFIG_INITRD_TAG 1
didn't add the bootm command as i had to create a larger flash.bin file as the rootfs.uimg was over 3 MB , and so had different memory values.
Nevertheless , it works fine . Thanks a ton !

Adithya

2011/04/05

Hey Balau,

What is the difference between ".axf" file and ".bin". Will the above mentioned procedure work for the u-boot image with .axf extension.

Thanks in Advance.

Regards
B. Adithya

Balau

2011/04/05

".axf" is an executable file, with ELF format, that contains the code and data, but also information about the loading address, sections information, debugging symbols… It is usually run by a simulator or a debugger.
".bin" is a pure binary file, containing the code and data that can be written inside a Flash to be run by an hardware platform.
If you run QEMU passing an ELF file with the kernel option, QEMU should be able to recognize the executable. But if you create a binary flash image like I do in my example, and put the .axf file inside it, it could not work.
If you have an ".axf" file, you can generate easily a ".bin" file using the "arm-*-objcopy" (in case of GCC toolchains) or "fromelf" (in case of RVDS) tools.

Adithya

2011/04/21

Hello Balau, I get this error when i try to boot the flash.bin. Both u-boot and zImage boot properly individually. Could you please tel me where i am going wrong.

```
Release AEL-5.0
Remote commit cloned UNKNOWN
Latest commit locally UNKNOWN
git state UNKNOWN
DRAM: 0 Bytes
## Unknown FLASH on Bank 1 – Size = 0x00000000 = 0 MB
Flash: 0 Bytes
*** Warning – bad CRC, using default environment

In: serial
Out: serial
Err: serial
Net: LAN9118 ethernet chip detected
This board has no MAC address auto-loaded
SMC_RV-9118-0
Hit any key to stop autoboot: 0
## Booting kernel from Legacy Image at 00210000 …
Image Name:
Image Type: ARM Linux Kernel Image (uncompressed)
Data Size: 1763072 Bytes = 1.7 MB
Load Address: 00010000
Entry Point: 00010000
Wrong Ramdisk Image Format
Ramdisk image is corrupt or invalid
```

Adithya

2011/04/21

Contd … from above,
I think i forgot to add the file system in the previos comment, Now when i add the filesystem, i get the following error:

```
Release AEL-5.0
Remote commit cloned UNKNOWN
Latest commit locally UNKNOWN
git state UNKNOWN
DRAM: 0 Bytes
## Unknown FLASH on Bank 1 – Size = 0x00000000 = 0 MB
Flash: 0 Bytes
*** Warning – bad CRC, using default environment
```

```
In: serial
Out: serial
Err: serial
Net: LAN9118 ethernet chip detected
This board has no MAC address auto-loaded
SMC_RV-9118-0
Hit any key to stop autoboot: 0
## Booting kernel from Legacy Image at 00210000 …
Image Name:
Image Type: ARM Linux Kernel Image (uncompressed)
Data Size: 1763072 Bytes = 1.7 MB
Load Address: 00010000
Entry Point: 00010000
## Loading init Ramdisk from Legacy Image at 00410000 …
Image Name:
Image Type: ARM Linux RAMDisk Image (uncompressed)
Data Size: 1118941 Bytes = 1.1 MB
Load Address: 00800000
Entry Point: 00800000
Loading Kernel Image … OK
OK

Starting kernel …

Uncompressing Linux….qemu: fatal: Bad mode 0

R00=73200000 R01=00000000 R02=00000000 R03=c035c804
R04=00000000 R05=e91d7679 R06=00000000 R07=00000000
R08=ffffffff R09=00000000 R10=73200000 R11=73200000
R12=001be704 R13=fffff0d0 R14=700100f4 R15=70023160
PSR=200001db –C- A und32
Aborted (core dumped)
```

Can u help me plz !!!

Balau

2011/04/21

It seems to me that the code goes into an "undefined" exception, as the PSR value say (The 32 bit PSR) and then the code somehow tries to change the mode to 0 (the code that exits QEMU can be found in the QEMU source, file "target-arm/helper.c"
The registers R14 (link register) and R15 (program counter) have strange values: at memory address 0x70023160 should not be any memory and so any code.
Does the zImage still work individually?

Dan

2011/07/16

Hi, Balau,

In your example, u-boot, Linux kernel, and rootfs are placed at the distance of 2MiB in flash.bin. Is this 2MiB required by u-boot or QEMU. I suspect it is u-boot, right? But I am a little confused because this would limit the kernel size to be less than 2MiB. Also will the flash be loaded at 0x10000 (64KiB) so that QEMU can load u-boot?

Thanks for your contributions!

Dan

Balau

2011/07/16

Actually the 2MiB size is not required, neither by U-Boot nor by Linux. I chose that size because the images of the three components were all less than 2MiB and I needed an easy way to know where they are. You can change that distance if you want, and it doesn’t need to be the same, it could be 3MiB and 5MiB, the only requirement is that it has enough space to contain the binary files. Once you create a flash.bin file with different placement, you must change the U-Boot “bootm” command with the right addresses.

Dan Guo

2011/07/16

Hi, Balau,

Thanks for your reply!

But how could QEMU know these three binaries are 2MiB apart in flash.bin?

Thanks a lot,

Dan

Balau

2011/07/17

QEMU doesn't need to know. The binary you give to QEMU with the "-kernel" option is placed at 0x10000, then the execution starts at that address. It is U-Boot that needs to know where the kernel and root filesystem are placed, and you need to pass this information with the "bootm" command.

Dan Guo

2011/07/17

Dear Balau,

Thanks a lot for your explanation! I believe "bootm 0x210000 0x410000″ in your patch will let u-boot know that the kernel and file system are placed at a distance of 2MiB.

Bests,
Dan

Gareth Ferneyhough

2011/07/21

Thanks very much for the tutorials; they all work wonderfully! Now I hope to get an open core Microblaze clone (openfire2) working on my Spartan3e starter kit board. Then I will hopefully reproduce these experiments on real hardware and be on my way!

-Gareth

srinivas

2011/09/12

Hi Balau,

I am very grateful to your well organized and informative posts.

I am facing an issue while using qemu for booting with flash.bin.
Error:
"can't open /dev/tty3: no such file or directory" message is being displayed repetitively.
If use 'ls' command File system directories are displaying.

Best Regards
Srinivas

Balau

2011/09/12

Maybe the "/dev" directory is not populated correctly, so there could be a problem in "/etc/init.d/rcS" file. Can you check that your rcS file is executable?

srinivas

2011/09/14

Hi Balau,
Again On error after starting qemu with flash.bin
"can't open /dev/tty3: no such file or directory" message is being displayed repetitively.

On My Host PC(ubuntu-11.04) /etc/init.d/rcS is executable for root.
On QEMU terminal there is no /etc directory.

I tried to execute below command using root privileges but still problem exists.
command:
sudo qemu-system-arm -M versatilepb -m 128M -kernel flash.bin

Could you please tell me what might be the problem.

— Srinivas

Balau

2011/09/14

If you don't have an /etc directory inside QEMU terminal then the rootfs image file has not been generated correctly.

On your host PC the "/etc/init.d/rcS" is not important, and executing "QEMU" with sudo does not change things. What is important is your custom rcS file that I created in my previous post here: https://balau82.wordpress.com/2010/03/27/busybox-for-arm-on-qemu/

In my tutorial the file must be placed in "_install/etc/init.d/rcS" into the busybox source tree after busybox compilation. Then with the "cpio" command you create a root filesystem image from the _install directory, and it must contain the local "etc" directory that must appear in the QEMU terminal when you do "ls /"

I hope it's more clear now.

srinivas

2011/09/15

Hi Balau,
Thank you very much for your quick reply.
Now It's working. I didn't fallow your busybox post completely

Thanks
Srinivas

Srinivas

2011/09/26

Hi Balau,
I want to get familiarization with u-boot code for versatile PB.
For that I want use GDB on Qemu. Could you please provide info
for using GDB on QEMU(How to). And also could you please provide good URL'S or docs for learning u-boot functionality from scratch.

Balau

2011/09/26

QEMU can easily act as a GDB server. When QEMU is run with the "-s -S" options, it will start waiting for a GDB connection on port 1234. Then you can connect using the "target remote localhost 1234″ command inside a GDB session. See also my old post Hello world for bare metal ARM using QEMU for an example on debugging a bare metal ARM program, such as U-Boot.

Keep in mind that on Ubuntu the QEMU package does not support debugging very well. I had to compile it from source to make it work.

Debugging U-Boot is a little more complicated because it relocates itself. In this page there is a tutorial on how to debug it: Debugging of U-Boot. When you make the u-boot.elf program the debugging symbols should already be included by default (i mean the "-g" option of GCC).

There is much information in the "README" file inside the U-Boot source tree, and some other things in the "doc" directory, but I don't think there is documentation to explain the internals of the source code.

omer

2011/10/21

Hi Balau,
I want to install linux on arm6410 with sd card or usb.but,when i insert the sd card to arm6410 i can see the documents in the sd card but i can't start the boot linux.for that what i must do to

firstly.can you examine the first steps. or after insert sd card ,must i write some commands for starting install linux. thanks for answer.

Balau

2011/10/22

Unfortunately I am not familiar with the hardware you're working on. I suppose the hardware came with a manual, so maybe there's some information about the boot procedure there. The boards often have some configuration switches that change the way the processor behaves during boot (for example they might have a "Boot from SD" configuration).

I think that you need to prepare the SD card from a PC, using a procedure similar to the one I used in this post and then writing directly on the SD card block device instead of a binary file. Then you can insert the SD card in the board and try to boot. I never did this procedure myself, though.

Grant Likely

2011/11/01

For anyone trying to reproduce this, at least on a recent Ubuntu host, you may need to pass "-cpu all" or "-cpu cortex-a8″ to qemu. The libgcc that gets linked to u-boot appears to be compiled with thumb2 instructions which are not implemented in the Versatile cpu.

I don't get any u-boot console output without this flag, and using gdb I can see that the cpu takes an exception during __udivsi3() called from serial_init().

Grant Likely

2011/11/01

Oops, I got the option wrong. Make that "-cpu any".

Balau

2011/11/03

Thanks for commenting, Grant. As an aside, I really appreciate your work.

The toolchain has "multilibs" and should link the correct libraries based on compiler flags. If I have time I'll take a look, maybe it's just a matter of configuring U-Boot for the correct ARM architecture, because the default expects a newer (thumb2-capable) processor.

Ritu

2012/01/27

Hello Balau, This information is really helpful for getting started. I was trying trying to get the same up in Ubuntu. I have not been able to build the image u-boot.bin. I made the patch fixes mentioned above but I am getting some undefined reference errors. Some of the errors are pasted below for reference:

lib_arm/libarm.a(board.o): In function `start_armboot':
/home/ritu/qemu_test/arm_downloads/u-boot-2010.03/lib_arm/board.c:304: undefined reference to `flash_init'
/home/ritu/qemu_test/arm_downloads/u-boot-2010.03/lib_arm/board.c:414: undefined reference to `copy_filename'
/home/ritu/qemu_test/arm_downloads/u-boot-2010.03/lib_arm/board.c:434: undefined reference to `eth_initialize'
/home/ritu/qemu_test/arm_downloads/u-boot-2010.03/lib_arm/board.c:442: undefined reference to `BootFile'
lib_arm/libarm.a(board.o):(.data+0x8): undefined reference to `env_init'
lib_arm/libarm.a(board.o):(.data+0x10): undefined reference to `serial_init'
common/libcommon.a(cmd_bootm.o): In function `bootm_load_os':

Pls suggest if I am missing anything in the setup.

Thanks
Ritu

Balau

2012/01/27

In my "uboot.map" file that is generated I see that all the functions are present in linking stage:

```
flash_init: drivers/mtd/libmtd.a(cfi_flash.o)
copy_filename: net/libnet.a(net.o)
eth_initialize: net/libnet.a(eth.o)
BootFile: net/libnet.a(net.o)
env_init: common/libcommon.a(env_flash.o)
serial_init: drivers/serial/libserial.a(serial_pl01x.o)
```

I suggest re-trying again from a clean state using "make distclean" and then redoing the "make CROSS_COMPILE=arm-none-eabi- versatilepb_config" and "make CROSS_COMPILE=arm-none-eabi- all" commands in my post.
If even that doesn't work, you can retry by setting environmental variable "export ARCH=arm" and recompile.

Hope this helps.

tanaka

2012/02/02

is flash emulation now support in latest qemu.15.0 for arm versatilepbqemu platform?

Balau

2012/02/02

From a quick look at the source code it seems it's not been added. The VersatilePB is an old hardware so I suppose it may never gain flash support in QEMU.

eng trojan

2012/02/14

now i make the steps as good as i can but when i finally release flash.bin and try to simulate it on qemu i have this error

```
R00=00000000 R01=33fb9880 R02=00049868 R03=00000000
R04=00000000 R05=00000000 R06=00000000 R07=00000000
R08=00000000 R09=00000000 R10=00000000 R11=00000000
R12=000100fc R13=00000000 R14=000100fc R15=33f801f4
PSR=800001d3 N— A svc32
Aborted
```

but when i try to simulate without u-boot just with rootfs and zimage, it works good
first i was working with version of u-boot 1.7 , i expected that it was the error but i used the mentioned version in your explain and applied the patch and tried to make flash.bin again but i have the same error

Balau

2012/02/14

I replied in this comment.

Patrick

2012/03/14

Hi Balau, I am following your procedure to run the latest linux kernel (stable version, 3.2.10) on an imx51 Freescale board with a cassini root filesystem of GenIVI. The uImage that I generate from my kernel zImage is 2.3MB and I packed it into my boot partition. However it turns out that the rootfile system(a .tgz file) which I downloaded is 723MB and when I tar it into my rootfs partition it has a size of 1.7GB. Is there some mistake? I am not able to correctly set the u-boot parameters as I am confused. I have the u-boot.imx after compiling u-boot sources.This is what I have:
sudo dd if=u-boot.imx of=${DISK} seek=1 bs=1024
Followed by setting of the partitions as follows:
unallocated: 5MB
fat16:boot: 50MB
ext4:rootfs:3.6GB(rest of the 4GB SD card)
I then copy the uImage, boot.scr to the boot partition and then I tar &copy the rootfs.tgz and kernel sources to the rootfs.

Here is what I set in boot.scr:
setenv bootcmd 'fatload mmc 0:1 0x90800000 uImage; bootm 0x90800000'
setenv bootargs console=ttymxc0,115200 console=tty0 root=/dev/mmcblk0p2 rootwait ro rootfstype=ext4 mxcdi1fb:1280x720M@60
boot

How do I set the right addresses in the above file? I don't understand it
Perhaps because of this, while I boot up my imx device: I get the following message:

U-Boot 2011.12 (Mar 13 2012 – 14:15:41)

CPU: Freescale i.MX51 family rev3.0 at 800 MHz
Reset cause: POR
Board: MX51EVK
DRAM: 512 MiB
WARNING: Caches not enabled
MMC: FSL_SDHC: 0, FSL_SDHC: 1
MMC: no card present
MMC init failed
Using default environment

In: serial
Out: serial
Err: serial
Net: FEC
Warning: failed to set MAC address

Hit any key to stop autoboot: 0
MMC: no card present
Booting from net …

BOOTP broadcast 1
BOOTP broadcast 2

Balau

2012/03/14

I'm sorry but my method will not work with a root filesystem that big. My method can be used to boot an intermediate initrd (ramdisk) that is able to load some modules and boot the real root. Depending on your hardware you can place the root filesystem on a server on the network, on an SD card, an USB disk/flash or a SATA drive. More information on the usage of initial ramdisk can be found in kernel source in "Documentation/initrd.txt"

Amit kumar

2012/03/19

qemu-system-arm -M versatilepb -m 128M -nographic -kernel u-boot.bin

when i m giving this command then it is saying command not found….

Balau

2012/03/19

It means the "`qemu-system-arm`" program has not been correctly installed. The installation depends on the Linux distribution you are using, I already specified the steps in the "prerequisites" section. Be aware that this article has been written in 2010 so the way to install "`qemu-system-arm`" may have changed.

Amit

2012/03/27

Hi Balau,
I have build my toolchain through buildroot. while building uboot I am getting following errors
board.c: In function '__dram_init_banksize':
board.c:233: error: 'CONFIG_SYS_SDRAM_BASE' undeclared (first use in this function)
board.c:233: error: (Each undeclared identifier is reported only once
board.c:233: error: for each function it appears in.)
board.c: In function 'board_init_f':
board.c:279: error: 'CONFIG_SYS_INIT_SP_ADDR' undeclared (first use in this function)
board.c:312: error: 'CONFIG_SYS_SDRAM_BASE' undeclared (first use in this function)

make[1]: *** [board.o] Error 1
make[1]: Leaving directory `/home/timberline/Android_devel/u-boot-2011.03/arch/arm/lib'
make: *** [arch/arm/lib/libarm.o] Error 2

Balau

2012/03/27

It's not a problem of toolchain, it's a problem of u-boot version. There are some versions in which old hardware does not compile properly because they changed some of the internals. If you try to do the same with the 2010.03 or 2011.12 they should compile fine.

Jerzy

2012/04/02

qemu-system-arm -kernel file -initrd file …
What are the QEMU default load address's for the kernel and initrd files?

Balau

2012/04/02

In the post I already wrote:

> *QEMU can load a Linux kernel using the -kernel and -initrd options; at a low level, these options have the effect of loading two binary files into the emulated memory: the kernel binary at address 0x10000 (64KiB) and the ramdisk binary at address 0x800000 (8MiB).*

Sorry but I don't understand what information that you need is not present in this sentence.

Jerzy

2012/04/03

Hi Balau,
Thank for your reply.
My question is :
What are the QEMU default load address's into the emulated memory – not in this case but generally – for the kernel and initrd files without using U-Boot, in command like this
qemu-system-arm -kernel file -initrd file …

Balau

2012/04/03

The default addresses are indeed `0×10000` for the kernel and `0×800000` for the initrd.
I think I have confused you because in my example I used the same addresses for U-Boot booting.
My plan was:
– I see what QEMU does when I pass kernel and ramdisk from command line
– I recreate the same state using U-Boot
The result is that after the `bootm` command, the kernel and the ramdisk are in the same addresses that they would have been if I passed them to QEMU from the command line.
I hope I have clarified the situation.

Jerzy

2012/04/13

Is it possible to launch successfull QEMU in the way like this

qemu-system-arm -M versatilepb -m 128M -kernel flash.bin -initrd rootfs.img.gz -serial stdio

where flash.bin = u-boot.bin + zImage.uimg ?

Balau

2012/04/14

When U-Boot executes "`bootm 0x210000 0x410000`" it copies the two images into their load addresses and then launches Linux with some parameters.
If you run QEMU as you want to do, you already have the ramdisk in place. For this reason, you don't need the second argument to `bootm`, but you need to tell the kernel that the ramdisk is there. For this, I think you can append "`initrd=0x800000`" to the `BOOTARGS` that U-Boot passes to Linux.

Jerzy

2012/04/14

Did you try to do it?
I set of course
#define CONFIG_BOOTCOMMAND "bootm 0x210000"
in UBOOT versatile.h file, and prepared flash.bin
mkimage -A arm -C none -O linux -T kernel -d zImage -a 0x00010000 -e 0x00010000 zImage.uimg
dd if=/dev/zero of=flash.bin bs=1 count=4M
dd if=u-boot.bin of=flash.bin conv=notrunc bs=1

dd if=zImage.uimg of=flash.bin conv=notrunc bs=1 seek=2M
I launched qemu
qemu-system-arm -M versatilepb -m 128M -kernel flash.bin -initrd rootfs.img.gz -serial stdio
Next I stopped it in UBOOT and checked memory
VersatilePB# md.b 0x800000 1000
There wasn't contents of rootfs.img.gz but only 000…

Balau

2012/04/14

You are right: I just tried and the ramdisk is placed at `0xd00000` instead.
I discovered this address my doing "`md 0 64`" and inspecting the data that looked like an address.
Then I misinterpreted the initrd parameter, it should be something like "`initrd=0xd00000,2M`", where the value after the comma is the size of the ramdisk (I rounded by eccess).
With these modifications it works for me.

Jerzy

2012/04/16

Thanks Balau.
Any idea why in this case ramdisk is placed at 0xd00000 instead at 0x800000?

Balau

2012/04/16

No idea, but I haven't investigated either.
It might have something to do with the kernel binary size, but it may also have been changed in QEMU source code.
In my opinion it should not change anything relevant, we could just accept the fact that the ramdisk is placed at an arbitrary address.

Jerzy

2012/04/17

I'd like to launch linux in QEMU, in the way like this

qemu-system-arm –M versatilepb –kernel flash.bin –initrd rootfs.img.gz …

Suppose that I'd like to pass to the linux kernel the following parameters :
console=ttyAMA0 root=/dev/ram rw initrd=0xd00000,2M

Generally, I can pass these parameters, through :
– append option in qemu command
– bootargs in UBOOT environment
– Boot options->kernel command string (at the time of kernel configuration)

It is possible to pass them in every of this mentioned above way but only in one at once?

Sometimes I've kernel image compiled with some parameters in kernel command string and u-boot.bin compiled with other parameters in bootargs. At time of software developing the most comfortable way for me is to change the kernel parameters in qemu command in append option.
Can I launch linux in QEMU in the way mentioned above with different parameters in qemu line, u-boot.bin and kernel image? If yes, which parameters will be passed to linux kernel?

Balau

2012/04/17

The kernel parameters in your case are those passed by U-Boot. The one in QEMU "-append" option never reach the kernel.
This is because QEMU prepares ATAGS that U-Boot does not read, and then U-Boot prepares its own ATAGS (from bootargs) to be passed to the kernel.
See Documentation/arm/Booting.txt for information about what should be passed to Linux kernel.
In physical world scenario, U-Boot saves its environment in the flash, so you can have an U-Boot image with default parameters, and then a sector of the flash that contains your parameters.

Are you sure you need to use U-Boot? If you don't use U-Boot then you can pass the kernel parameters with QEMU without problems.

Jagan

2012/06/23

I have a problem with booting Linux on versatilepb through QEMU.

I have used root=/dev/ram rw…
but still my FS not mounted…Can you help me whether I am missing any bootargs..

Here is the tail logs:
——————–
List of all partitions:

```
1f00 65536 mtdblock0 (driver?)
No filesystem could mount root, tried: ext2 cramfs minix romfs
Kernel panic – not syncing: VFS: Unable to mount root fs on unknown-block(1,0)
[] (unwind_backtrace+0x0/0xf4) from [] (panic+0x74/0x1c0)
[] (panic+0x74/0x1c0) from [] (mount_block_root+0x1e8/0x228)
[] (mount_block_root+0x1e8/0x228) from [] (mount_root+0xcc/0xf0)
[] (mount_root+0xcc/0xf0) from [] (prepare_namespace+0x160/0x1b8)
[] (prepare_namespace+0x160/0x1b8) from [] (kernel_init+0x158/0x19c)
[] (kernel_init+0x158/0x19c) from [] (kernel_thread_exit+0x0/0x8)
```

Balau

2012/06/24

You can't know if you missed any bootargs by looking only at the tail of the log.
You could add "`console=ttyAMA0`" to the current bootargs (and QEMU must be launched with "`-serial stdio`") to display more info on the terminal.
Try to find a line near the beginning of the log starting with "`Kernel command line: `".
Those are the bootargs.
Then in the middle of the log you should find a line such as "`Trying to unpack rootfs image as initramfs...`". The lines around that could contain useful hints about why the kernel isn't mounting the filesystem.

Jagan

2012/06/24

Exactly..I missed to give you the details about boot args.
setenv bootargs 'console=ttyAMA0,115200n8 root=/dev/ram rw'
I have created uImage loaded at addr1 and created ramdisk of mkimage compatible loaded at addr2 (addr1 > addr2).
I did below command
$ bootm $addr1 $addr2

Balau

2012/06/24

Where's the "`rdinit=...`" bootarg? Why did you remove it?

Jagan

2012/06/24

No, I just need to mount ramdisk not any other app or init file.
I think rdinit required for explicit app running…correct me If am wrong

Balau

2012/06/25

The kernel has to run something (in userspace) when the boot ends, otherwise it panics. This "something" is usually the `init` program.
I don't know if using both "`root=/dev/ram`" and "`rdinit=/sbin/init`" is the cleanest way to do it, but I noticed that without "`rdinit`" the kernel does not try to mount the ramdisk and so it panics.

Jagan

2012/06/27

Let me clear the entire scenario.
I have uImage and ramdisk with mkimages.
like uramdisk.img

uImage – load and entry address are 0x800
uramdisk -load and entry address are 0x800000

$ tftp 0x100 uImage
$ tftp 0x4000000 uramdisk.img
$ setenv bootargs 'console=ttyAMA0,115200 root=/dev/ram rw'
$ bootm 0x100 0x4000000

Found the below issue :
— — — — — — — — –
No filesystem could mount root, tried: ext2 cramfs minix romfs
Kernel panic – not syncing: VFS: Unable to mount root fs on unknown-block(1,0)
[] (unwind_backtrace+0×0/0xf4) from [] (panic+0×74/0x1c0)
[] (panic+0×74/0x1c0) from [] (mount_block_root+0x1e8/0×228)
[] (mount_block_root+0x1e8/0×228) from [] (mount_root+0xcc/0xf0)
[] (mount_root+0xcc/0xf0) from [] (prepare_namespace+0×160/0x1b8)
[] (prepare_namespace+0×160/0x1b8) from [] (kernel_init+0×158/0x19c)
[] (kernel_init+0×158/0x19c) from [] (kernel_thread_exit+0×0/0×8)

Balau

2012/06/28

Let me clear my complete opinion.
I am convinced that if you try "`setenv bootargs 'console=ttyAMA0,115200 root=/dev/ram rw rdinit=/sbin/init'`", it will work.
This is because, as said in Linux "`Documentation/early-userspace/README`" and in other parts of the web, the "initramfs" way of booting Linux expects that the ramdisk is a cpio archive, it mounts it and then tries to execute "`/init`". In our case we don't have "`/init`" so we have two options:
1. creating a link such as "`ln -s ./sbin/init ./init`" in the busybox `_install` directory before creating the cpio archive (I haven't tried it actually)
2. adding "`rdinit=/sbin/init`" to the kernel parameters (as specified in my blog post and in my past replies to you)

I think "`root=/dev/ram`" is superfluous, it should work without it because we don't reach the point where we mount the root filesystem.
But implementing one of the two ways above is necessary to boot Linux with the ramdisk.

If it still doesn't work, then you should also check the other parts of the kernel messages as I already said, because something could have gone wrong in mounting the initramfs.

Hope this helps.

Jerzy

2012/06/30

Hi Balau,

Why does Uboot informs on the console
DRAM: 0 kB
instead
DRAM: 128 MB ?

Balau

2012/06/30

I don't know, it seems to be printed by "`display_dram_config`" function in "`board.c`" file (in u-boot-2010.03 the file is in "lib_arm" directory).
Maybe the new u-boot versions fixed this information, but I don't remember if they still support VersatilePB.

Jagan

2012/06/30

copy the dram_init code on to
board/armltd/versatile/versatile.c
int dram_init (void)
{
/* dram_init must store complete ramsize in gd->ram_size */
gd->ram_size = get_ram_size((void *)CONFIG_SYS_SDRAM_BASE,
PHYS_SDRAM_1_SIZE);
return 0;
}

Jerzy

2012/07/01

Thanks for your replies.
And what about Uboot commands history ?

Balau

2012/07/02

What do you mean "what about Uboot commands history"? You wanted to ask why it does not work for you?
If that was the question, my answer is still "I don't know" as before, and I don't have time right now to check the source code to try to understand why it does not work.

You have (at least) two paths:

1. Ask U-Boot mailing list
2. Try to find your answers yourself by trying to understand the source code

I suggest trying 2 and then 1.

Jerzy

2012/07/02

Thanks Balau.
I understand that Uboot 2010.03 commands history on qemu not working at all?

Balau

2012/07/03

In my environment, it is clear that it does not understand the "arrow" keys as "go up in history of commands".
U-Boot is made to be small, I suppose giving it a command history is considered bloat for what it should do.
I tend to agree, because if everything works you should never need to access U-Boot command shell.

Terence

2012/07/17

This is great Balau, thanks for your articles. It serves as a great reference point for my project which is to get u-boot and a linux kernel up and running on the ST-E U8500 platform. Unfortunately QEMU seems to have issues…

Jerzy

2012/11/30

Hi,

Fortunately uboot linaro has commands history and u-boot.bin size is roughly the same.

psychesnet

2013/01/17

Hi Balau,

I try to practice Qemu by following your blog, but I face some problem, please help me, thanks a lot.

```
$ mkimage -A arm -C none -O linux -T kernel -d zImage -a 0x00010000 -e 0x00010000 uImage
$ mkimage -A arm -C gzip -O linux -T ramdisk -d rootfs.cpio.gz -a 0x00800000 -e 0x00800000
rootfs.uimg
$ dd if=/dev/zero of=flash.bin bs=1 count=10M
$ dd if=u-boot.bin of=flash.bin conv=notrunc bs=1
$ dd if=uImage of=flash.bin conv=notrunc bs=1 seek=2M
$ dd if=rootfs.uimg of=flash.bin conv=notrunc bs=1 seek=4M
```

```
VersatilePB # sete bootargs console=ttyAMA0 mem=128M root=/dev/ram rw rdinit=/sbin/init
VersatilePB # bootm 0x210000 0x410000
## Booting kernel from Legacy Image at 00210000 …
Image Name:
Image Type: ARM Linux Kernel Image (uncompressed)
```

```
Data Size: 1517816 Bytes = 1.4 MB
Load Address: 00010000
Entry Point: 00010000
## Loading init Ramdisk from Legacy Image at 00410000 …
Image Name:
Image Type: ARM Linux RAMDisk Image (gzip compressed)
Data Size: 2579307 Bytes = 2.5 MB
Load Address: 00800000
Entry Point: 00800000
Loading Kernel Image … OK
OK
Starting kernel …
Uncompressing Linux… done, booting the kernel.
…..
TCP: cubic registered
NET: Registered protocol family 17
VFP support v0.3: implementor 41 architecture 1 part 10 variant 9 rev 0
drivers/rtc/hctosys.c: unable to open rtc device (rtc0)
RAMDISK: Couldn't find valid RAM disk image starting at 0.
List of all partitions:
1f00 131072 mtdblock0 (driver?)
No filesystem could mount root, tried: ext2 cramfs squashfs vfat msdos romfs
Kernel panic – not syncing: VFS: Unable to mount root fs on unknown-block(1,0)
```

How do I fix this rootfs problem????

By the way, it is working when I use
```
$ qemu-system-arm -M versatilepb -kernel zImage -initrd rootfs.cpio.gz -nographic -append "console=ttyAMA0 mem=128M"
```

But why following command would fail with u-boot rootfs ?
```
$ qemu-system-arm -M versatilepb -kernel zImage -initrd rootfs.uimg -nographic -append "console=ttyAMA0 mem=128M root=/dev/ram rw"
```

It seem like first problem ?
Need your help, Thanks a lot~

Balau

2013/01/17

About your first question:

a. you could use the exact same versions that I used and the exact same configuration to make it work, and then little by little change from my setup to yours to see when things start to go bad. I used Linux 2.6.33, U-Boot 2010.03 and busybox 1.16.0. For example I see that your root filesystem is bigger than mine, in particular bigger than 2MiB. I don't know if that could be a problem.

b. you could launch QEMU with -s -S options and then attach with arm-…-gdb using "target remote localhost:1234", then put a breakpoint on the start of Linux execution (for example using "file vmlinux" and putting a breakpoint on start_kernel) and when the breakpoint is reached display the content of 0x00800000 to see if ramdisk has been corrupted (check if the data is the same as rootfs.cpio.gz).

About your second question:

rootfs.uimg is just rootfs.cpio.gz with a U-Boot header attached at the beginning. Linux can't understand U-Boot headers so the second command will not work and I did not expect otherwise.

hemal

2013/01/18

Hello,

I am using U-Boot(compressed) and two kernel Image(uImage). I want to add some code in U-Boot which will select kernel based of time stamp(or using any other way if you have in mind). I am using MIPS architecture.

For example:-

If kernel-1 is new, U-Boot will boot Kernel-1. and leave kernel-2 as it is.
If kernel-2 is new, U-Boot will boot kernel-2. and leave kernel-2 as it is.
Questions:-

Is it possible to do so?
How can I add such functionality in U-boot?
Where to chage the code for the same?

Balau

2013/01/18

I don't think U-Boot was made for something like that.
You could modify the source code of U-Boot around the autoboot functionality, and use the timestamp added by mkimage to choose.
I don't think it's simple, you could ask U-Boot mailing lists.

Take a look at this to understand what can be done without modifying the source code: http://omappedia.org/wiki/Multiboot_using_u-boot hemal

2013/01/18

thank you for your reply.
Can you just tell me from where u-boot put the kernel image into RAM?
so that I can tel u-boot to put the proper image of kernel to RAM.

Balau

2013/01/20

If I search the displayed message "Booting kernel from" in U-Boot source code (2010.03), it's present in "`common/cmd_bootm.c`", in function `boot_get_kernel`. Following back the calls in the same C file it's quite easy to find the point where the kernel is loaded.

Geo

2013/03/22

Hi Balau. I have been following your post to run linux via uboot on qemu. I followed your steps and when I run the "flashed" image on qemu, i always get the error "Uncompressing Linux… done, booting the kernel.
Bad ram offset 8000000″.

I can run u-boot by itself and kernel also by itself (although with kernel, i keep getting spew about /dev/ttyxxx not found). But when I create a flash image, i get this error.

Wondering if you knew anything about it.

thanks in advance

Balau

2013/03/23

The error says "8000000" (0x08000000), but in my post I talk about address 0x00800000. Are you sure you didn't put a zero more in the mkimage command or something like that?

Geo

2013/03/23

Thanks for the reply! One important thing i should have mentioned is that i am running osx qemu.

balaji

2013/04/16

Hi Balau,

I am using zynq_zc702 based U-Boot 2011.03 source for running on zynq's based qemu. Individually i am able to run the u-boot.bin and zimage with rootfs from the qemu. As you suggested I combined u-boot, zimage and rootfs into a single image(flash.bin) for supporting autoboot and I made the changes to include/configs/zynq_common.h.

```
#define CONFIG_BOOTARGS "root=/dev/ram mem=128M rdinit=/sbin/init"
#define CONFIG_BOOTCOMMAND "bootm 0x210000 0x410000"
#define CONFIG_INITRD_TAG 1
```

When I run with the following command

```
./arm-softmmu/qemu-system-arm -M xilinx-zynq-a9 -m 1024 -serial null -serial mon:stdio -
kernel flash.bin -nographic
```

I got the following error.

```
ram size=40000000
error reading QSPI block device
error no mtd drive for nand flash
a0mpcore_priv: smp_priv_base f8f00000
error no sd drive for sdhci controller (0)
error no sd drive for sdhci controller (1)
Number of configured NICs 0x1
ram_size 40000000, board_id d32, loader_start 0

U-Boot 2011.03 (Apr 16 2013 – 12:13:30)

DRAM: 256 MiB
MMC: SDHCI: 0
Using default environment

In: serial
Out: serial
Err: serial
Net: zynq_gem
Hit any key to stop autoboot: 0
Wrong Image Format for bootm command
ERROR: can't get kernel image!
```

when i give the following command at u-boot level

iminfo 0x210000 gave the following information.

Checking Image at 00210000 …
Unknown image format!.

I am not able to see any content at 0x210000 location with md command.

I created flash.bin as you suggest and cross check it with hexdump command. Nothing wrong with flash.bin.

Please help me in this if it is a relevant question to you.

Thanks
balaji

Balau

2013/04/22

Two possibilities:

1. U-boot relocation overwrote your image
2. The "-kernel" option does not place the binary at 0x00010000

My suggestion is to try to run QEMU with -s -S options, attach with ARM GDB, analyze step by step the first instructions and check the memory with "x" GDB command.

balaji

2013/04/23

Thanks Balau

Giridhar (@giridhart)

2013/07/25

Hi Balau,

Are there u-boot build config options to enable more verbose output from u-boot?
I could see CONFIG_TRACE but could not find how to enable this.

Regards,

Balau

2013/07/27

I believe it's just a matter of adding "`#define DEBUG 1`" somewhere like in "`include/config_defaults.h`".
U-Boot is full of "`debug(...)`" calls that are enabled by this macro to be expanded as `printf`.
You can increase the value of `DEBUG` to print also "`debugX(level, ...)`" messages.
These macros are defined in "`include/common.h`".

Shashi

2013/09/30

I'm new to work on u-boot, it will be very helpful if someone let us know how to simulate port on qemu u-boot. thanks in advance

Balau

2013/10/04

What do you mean by "port"? If you mean "serial port", you are probably looking for "`-serial stdio`" option when running qemu-system-arm.

ML

2013/12/29

How am I boot on QEMU WM8650-SD-linux-image (at http://kernelhacks.blogspot.com/2012/06/arch-linux-on-wm8650-netbook-ii.html)

Balau

2013/12/29

QEMU doesn't emulate WM8650 ("`qemu-system-arm -M ?`" doesnt' show it) so you can't do that.

ML

2013/12/30

on SD are 2 partitons:
1(fat)-uzImage.bin, wmt_script
2(ext3)-files from ArchLinuxARM-armv5te-latest.tar.gz, and from -xjf alarm-wm8650-modules.tar.bz2

and otherwise you can not emulate the filesystem of this uzImage.bin (or uImage.bin)?

Nicholas

2014/01/04

hi, when i use
qemu-system-arm -M versatilepb -m 128M -kernel zImage -initrd rootfs.img.gz -append "root=/dev/ram mem=128M rdinit=/sbin/init" -serial stdio,
everything works fine.
But if i use
qemu-system-arm -M versatilepb -m 128M -kernel flash.bin -serial stdio
uboot is able to find kernel image and ramdisk image, but when kernel starts, kernel is not able to find ramdisk.
Do you where is the problem?

Balau

2014/01/06

@ML
The first partition is probably used only by the boot (u-boot?) to run the kernel with the chosen parameters. The second seems to be the root filesystem.
You can't emulate u-boot, the kernel or the modules for the wm8650 with QEMU because they depend strictly on the hardware and the memory map, and QEMU does not emulate wm8650.

Balau

2014/01/06

@Nicholas
Is it possible that the ramdisk is too big (> 2MiB)?
You can also run QEMU with -s option, then when the kernel fails you connect to it with arm gdb, by executing "target remote localhost:1234" in the gdb prompt, and then check that the content of the memory containing the ramdisk is as expected, for example by executing "dump binary mem.bin 0x00800000 0x00A00000" and check that mem.bin corresponds to rootfs.img.gz (you can do an hexdump -C of both files and diff them graphically).

Sasq

2014/03/08

Hello,

i got some problems with patchng u-boot from code posted above. I'm using same u-boot version(u-boot-2010.03). Error from applying patch is :

```
patch -p1 < ~/u-boot-2010.03.patch
patching file common/image.c
Hunk #1 FAILED at 941.
1 out of 1 hunk FAILED — saving rejects to file common/image.c.rej
patching file include/configs/versatile.h
Hunk #1 FAILED at 124.
1 out of 1 hunk FAILED — saving rejects to file include/configs/versatile.h.rej
```

Any suggestions why this happened?

Balau

2014/03/08

I don't know why it happened but I recommend changing the source code manually since it' a few lines of code, and in the process try to see if the source code is different from what it's expected.
Another explanation is that you copied also the line numbers from the site, they don't have to be copied.

hackembed

2014/04/05

Hi Balau, i followed your tutorial to boot flash.bin from qemu. MY problem is that when i start the qemu i got his error

```
U-Boot 2010.03 (avril 05 2014 – 11:47:27)

DRAM: 0 kB

Unknown FLASH on Bank 1 – Size = 0x00000000 = 0 MB

Flash: 0 kB
*** Warning – bad CRC, using default environment

In: serial
Out: serial
Err: serial
```

```
Net: SMC91111-0
Hit any key to stop autoboot: 0
qemu: hardware error: pl011_write: Bad offset ff8

CPU #0:
R00=56190527 R01=00000000 R02=00000010 R03=00000000
R04=00210000 R05=00210000 R06=00fddef4 R07=00000003
R08=00fddfe0 R09=00000000 R10=01014ffc R11=01017e8c
R12=00fddd03 R13=101f4000 R14=010105ac R15=010105a4
PSR=600001d3 -ZC- A svc32
```

When i test without u-booot it works perfectlly. So, i wanna know waht's wrong.
Please, help me.
Regards.

Balau

2014/04/05

The error is generated while executing U-Boot, Linux is not yet started.
R15 is program counter, R14 is return address; you can see where the program crashed from U-Boot disassembly. You can disassemble it by running something like `arm-none-eabi-objdump -S u-boot >u-boot.dis`. U-Boot should already be compiled with debugging symbols, otherwise try to enable them by adding `-g` to `CFLAGS`.
You can also try to run QEMU with `-s -S` options and then attach to it with `arm-none-eabi-gdb` using `u-boot` as file for symbols and debug info. Then you can break at `do_bootm_linux` (that's the last function U-Boot should execute) or at `abort_boot` (that's the last thing your U-Boot prints) and try to debug from there.

Nitin

2014/05/05

Hi Balu
I try to debug decompress part of linux kernel. But my Breakpoint does not hit.
To run linux on qemu i ran following command.
qemu-system-arm -M versatilepb -m 128M -s -S -kernel arch/arm/boot/zImage
and on gdb side ran this command
arm-none-eabi-gdb target remote localhost:1234
after that
file ./arch/arm/boot/compressed/vmlinux
then add breakpoint
like b __setup_mmu
None of my breakpoint hit .
Only break point after MMU_ENABLE hit like start_kernel

Thanks,
Nitin

Balau

2014/05/05

The part of software that runs before the decompression is position independent code. In my case (probably also in yours) the zImage is loaded in 0x60010000, and that's the address containing `start`, but in the `vmlinux` ELF the address is 0. I tried the GDB command `add-symbol-file arch/arm/boot/compressed/vmlinux 0x60010000` instead of `file arch/arm/boot/compressed/vmlinux` and it seems to load the symbols to their right addresses, try it also in your setup.

Nitin

2014/05/06

Hi Balau,
I followed your steps as mentioned.But Still not able to hit the breakpoints.
I checked with gdb and dump the 0x60010000 location.
x/10wx 0x60010000
0x60010000: 0x00000000 0x00000000 0x00000000 0x00000000
0x60010010: 0x00000000 0x00000000 0x00000000 0x00000000
0x60010020: 0x00000000 0x00000000
It shows nothing . So might be in my case load address of zImage will be different.
I have question:-
1) How to find the address where zImage loaded in qemu.

Nitin

2014/05/06

Hi Balau,

I checked with gdb and dump the location 0x00010000 location. I got this location when i searched for zImage header magic number 0x016f2818.
I matched with my zImage Hexdump but when i load the elf file with add-symbol-file arch/arm/boot/compressed/vmlinux 0x00010000 and put some breakpoint.
It halt that place and show function input_data() with no code.

Thanks
Nitin

Nitin

2014/05/06

Hi Balu,

I checked with add-symbol-file arch/arm/boot/compressed/vmlinux 0x60010000
but still breakpoints not hitting. When i dump the 0x60010000 on gdb it shows
0x00000000 . When i dump the 0x00010000 on gdb it has same footprint as my zImage.
But when i add-symbol-file arch/arm/boot/compressed/vmlinux 0x00010000 and put
breakpoint on this address. It stops there but function it shows input_data(). Please let me
know how you find the load address 0x60010000 on your system.

Thanks,
Nitin

Balau

2014/05/06

I launched QEMU with `-s -S` (so it's stopped before executing anything) and then connected with gdb. With something like `x/10i $pc` you can see the code at current program counter. At the beginning QEMU puts a couple of its own instructions to jump to the user code. So I `stepi` a few times until it jumps, and at that point it jumps to the beginning of the binary passed by `-kernel` option. That address should be the offset that you pass to `add-symbol-file`. I tried with versatilepb to make it similar to your setup and I find your same problem, and it is strange, but it is easily worked around if you `stepi` until you reach `start` or if you `break start` and then `continue`.

Nitin

2014/05/10

Hi Balau,

i used this command :-
add-symbol-file ./arch/arm/boot/compressed/vmlinux 0x00010000 -s .piggydata 0x00014610
and now i am able to put break points and debug the code.
I found out that in arch/arm/boot/compressed/vmlinux
.piggydata 001d28e0 00004610 00004610 0000c610 2**0
which means .piggydata starthe at 0x4610 and length 001d28e0 which overlaps the 0x00010000
address. i.e. gdb not able put the breakpoints . so remap that section also .

Thanks for your help,
Nitin

Cruise

2014/05/28

Hi Balau,
I compiled U-Boot and run QEMU with "u-boot" (ELF file) and I can see u-boot message.
$ qemu-system-arm -M vexpress-a9 –kernel u-boot -nographic
But I just saw nothing when using "u-boot.bin":
$ qemu-system-arm -M vexpress-a9 –kernel u-boot.bin -nographic
The cross-compiler I use is "arm-linux-gnueabi" and my linux distrubution is Ubuntu 12.04.
I also tried to re-compiled U-Boot with "versatile_defconfig" and run qemu with "versatilepb". I still can't see any boot message.
Do you have any idea?

Thanks.

Balau

2014/05/29

I am quite sure that if you give QEMU an ELF it will load it using the information about the segments of data, code, etc. and then it will execute from the entry point. But if you run a binary it will simply place it as is into a given address (different for each machine) and then jump to that address. You can try to run QEMU with `-s -S` and attach to it with GDB by running `arm-linux-gnueabi-gdb u-boot` and then `(gdb) target remote localhost:1234`; in this way you can run step-by-step and see what is happening with program counter and memory content.

Cruise

2014/06/05

Hi Balau,
I ran QEMU with GDB but u-boot.bin just can't be executed.
I tried to decode u-boot ELF file and found entry point is 0×60800000.
So, I generated image file with mkimage and set entry point to 0×60800000. In this way, it worked.

Giab

2014/09/12

Hi Balau,

great work!
I used it with latest kernel (linux-3.16.2) and u-boot (u-boot-2014.07) with these patch:

```
— — — — — — — — — — — –- common/image.c — — — — — — — — — — — –
index 11b3cf5..4a92b6a 100644
@@ -933,6 +933,15 @@ int boot_get_ramdisk(int argc, char * const argv[], bootm_headers_t
images,
return 1;
}
}
+#if defined(CONFIG_VERSATILE)
+ /
+ * We need to copy the ramdisk to SRAM to let Linux boot
+ /
+ if (rd_data) {
+ memmove ((void *)rd_load, (uchar *)rd_data, rd_len);
+ rd_data = rd_load;
+ }
+#endif / CONFIG_VERSATILE */
} else if (images->legacy_hdr_valid &&
image_check_type(&images->legacy_hdr_os_copy,
IH_TYPE_MULTI)) {

— — — — — — — — — –- include/configs/versatile.h — — — — — — — — — –-
index 29c32fe..56e4818 100644
@@ -15,6 +15,8 @@
#ifndef __CONFIG_H
#define __CONFIG_H

+#define CONFIG_ARCH_VERSATILE_QEMU
+
/*
* High Level Configuration Options
* (easy to change)
@@ -90,6 +92,9 @@
#define CONFIG_CMD_NET
#define CONFIG_CMD_PING
#define CONFIG_CMD_SAVEENV
+#define CONFIG_CMD_RUN
+#define CONFIG_CMD_SOURCE
+#define CONFIG_CMD_BOOTZ

/*
* BOOTP options
@@ -100,9 +105,11 @@
#define CONFIG_BOOTP_SUBNETMASK
```

```
#define CONFIG_BOOTDELAY 2
-#define CONFIG_BOOTARGS “root=/dev/nfs mem=128M ip=dhcp “\
– “netdev=25,0,0xf1010000,0xf1010010,eth0 “\
– “console=ttyAMA0,38400n1″
+#define CONFIG_BOOTARGS “root=/dev/ram mem=128M rdinit=/sbin/init
console=ttyAMA0″
+#define CONFIG_BOOTCOMMAND “bootm 0x210000 0x410000″
+#define CONFIG_INITRD_TAG 1
+
+

/*
* Static configuration when assigning fixed address
```

10 Trackbacks For This Post

1. *Links 13/4/2010: KDE 4.5 Schedule, Fedora 13 Beta | Techrights* →
April 13th, 2010 → 10:36
[…] Booting Linux with U-Boot on QEMU ARM […]

2. *Statom arm cross compile | Maksim Norkin* →
August 3rd, 2010 → 07:49
[…] Šaltinis: Balau. […]

3. *Blog stats for 2010 « Balau* →
January 2nd, 2011 → 17:25
[…] Booting Linux with U-Boot on QEMU ARM April 2010 11 comments 4 […]

4. *therning.org/ magnus » Blog Archive » Compiling U-Boot for use in QEMU (VersatilePB)* →
January 12th, 2012 → 14:40
[…] for one of the ARM-based machines that QEMU supports. I settled for VersatilePB after finding this old-ish article. Rather optimistically I thought that maybe, just maybe things had change in a year and that the […]

5. *QEMU 1.5.0 released, a backward compatibility warning | Balau* →
May 21st, 2013 → 20:51
[…] Booting Linux with U-Boot on QEMU ARM […]

6. *Vishwas Sharma | Getting Started with Yocto : Part 1* →
July 18th, 2013 → 17:31
[…] Booting linux with U-boot on QEMU ARM […]

7. *【整理】QEMU使用心得 | 在路上* →
August 17th, 2013 → 12:46
[…] （需要翻墙）Booting Linux with U-Boot on QEMU ARM […]

8. *QEMU used to experiment ARM u-boot kernel rootfs boot chain | Embedded Soft2* →

October 10th, 2013 → 19:57
[…] https://balau82.wordpress.com/2010/04/12/booting-linux-with-u-boot-on-qemu-arm/ […]

9. *#Embedded development with Qemu, Beagleboard, Yocto, Ångström, Buildroot. Where to begin? | Mateusz Kaczanowski :: Personal Blog* →
March 18th, 2014 → 20:02
[…] Some useful links: 1. Bootloader in details: http://omappedia.org/wiki/Bootloader_Project 2. MLO and RBL in details: http://coherentmusings.wordpress.com/2012/09/05/what-is-mlo-file/comment-page-1/ 3. U-boot on qemu explained: https://balau82.wordpress.com/2010/04/12/booting-linux-with-u-boot-on-qemu-arm/ […]

.0. *Blog do Software Livre O que é Linux Embarcado » Blog do Software Livre* →
May 27th, 2014 → 08:11
[…] Fonte: https://balau82.wordpress.com/2010/04/12/booting-linux-with-u-boot-on-qemu-arm/ […]