Balau
Freedom Embedded

U-boot for ARM on QEMU

Posted on 2010/03/10 93

Das U-Boot, the universal bootloader, is a crucial piece of software that runs on embedded platforms: its role is to put in place and boot the linux kernel from a hard drive, a flash memory, network or serial line. Here I explain how to try U-Boot on QEMU, and in particular on the emulation of the VersatilePB platform.

First, install the necessary tools:

- qemu-system-arm: run "`apt-get install qemu`" on Debian, or "`sudo apt-get install qemu-kvm-extras`" on Ubuntu.
- mkimage: install the uboot-mkimage package from the Debian or Ubuntu repository.
- CodeSourcery ARM EABI toolchain toolchain: download from their website and install.

Grab the U-Boot source code from the U-Boot FTP site and decompress it. Go inside the created directory and run:

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

This command configures U-Boot to be compiled for the VersatilePB board. Then compile and build with:

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

The compilation will create a `u-boot.bin` binary image. To simulate, run:

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

The U-Boot prompt should appear:

```
U-Boot 1.1.6 (Mar 3 2010 - 21:46:06)

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

In: serial
Out: serial
Err: serial
Versatile #
```

You can have a list of commands by entering `help`, and then try out various commands (hit "Ctrl-a" and then "x" to exit QEMU). The `bootm` command in particular is used to boot a program that is loaded in memory as a special U-Boot image, that can be created with the tool `mkimage`. This program is usually an operating system kernel, but instead of running a full-blown Linux kernel, we can instead run the simple "Hello world" program described in a previous blog post. To do so, we create a single binary that contains both the U-Boot program and our "Hello world" program together. The initial address of the "Hello world" program must be changed with respect to the instructions present in the last blog post, because at `0x10000` (our last initial address) QEMU places the beginning of the U-Boot binary. Since the U-Boot binary is about 100KB, we can place our binary at `0x100000` (that is 1MB) to be safe.

Create `test.c`, `startup.s` and `test.ld` as last time, but change line 4 of `test.ld` from "`. = 0x10000`" to "`. = 0x100000`". Build the binary with:

```
arm-none-eabi-gcc -c -mcpu=arm926ej-s test.c -o test.o
arm-none-eabi-ld -T test.ld -Map=test.map test.o startup.o -o test.elf
arm-none-eabi-objcopy -O binary test.elf test.bin
```

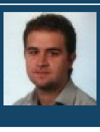
Now create the U-Boot image `test.uimg` with:

Search

Subscribe

Posts
Comments

Author



Top Posts & Pages

- Programming Arduino Uno in pure C
- Compile Linux kernel 3.2 for ARM and emulate with QEMU
- Booting Linux with U-Boot on QEMU ARM
- U-boot for ARM on QEMU
- Hello world for bare metal ARM using QEMU
- Simplest serial port terminal in C#
- Simplest bare metal program for ARM
- Trace and profile function calls with GCC
- Busybox for ARM on QEMU
- Using Newlib in ARM bare metal programs

Categories

- Hardware
- Links
- Software
  - Embedded
  - Internet
  - Security
- Working

Tags

android ansic arduino ARM avr backup bare metal boot busybox codesourcery cortex-m3 csharp debian debug debugging eda electronics embedded encryption facebook firefox fpga free electrons ftdi gcc gdb gnu google hacking Hardware hdl howto icarus

```
mkimage -A arm -C none -O linux -T kernel -d test.bin -a 0x00100000 -e
0x00100000 test.uimg
```

With these options we affirm that the image is for ARM architecture, is not compressed, is meant to be loaded at address `0x100000` and the entry point is at the same address. I use "linux" as operating system and "kernel" as image type because in this way U-Boot cleans the environment before passing the control to our image: this means disabling interrupts, caches and MMU.

Now we can create a single binary simply with:

```
cat u-boot.bin test.uimg > flash.bin
```

This binary can be run instead of the U-Boot binary with:

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

at the U-Boot prompt, we can check that the image is inside the memory: it should be exactly after the u-boot.bin code. To calculate the address, we must take the size of u-boot.bin and sum the initial address where flash.bin is mapped. From the `bash` prompt, the following script prints the command to be written inside U-Boot:

```
printf "bootm 0x%X\n" $(expr $(stat -c%s u-boot.bin) + 65536)
```

in my case it prints "`bootm 0x21C68`". In fact, if I run "`iminfo 0x21C68`" inside U-Boot prompt to check the memory content I get:

```
## Checking Image at 00021c68 ...
Image Name:
Image Type:   ARM U-Boot Standalone Program (uncompressed)
Data Size:    376 Bytes =  0.4 kB
Load Address: 00100000
Entry Point:  00100000
Verifying Checksum ... OK
```

I can then confidently run "`bootm 0x21C68`" (you should substitute your address in this command). This command copies the content of the image, that is actually `test.bin`, into the address `0x100000` as specified in the U-Boot image, and then jumps to the entry point. The emulator should print "Hello world!" as last time, and then run indefinitely (hit "Ctrl-a" and then "x" to exit). This is basically the same procedure that is used to boot a Linux kernel, with some modifications: for example, the Linux kernel accepts some parameters that must be received from U-Boot somehow. I plan to write a post about that in the future.

In a real world example, the binary file we created could be placed inside the parallel Flash memory of an embedded platform, and the boot process can be controlled from the serial port.

About these ads

Google | Twitter 3 | More

Like this:

Tagged: *ARM, boot, codesourcery, embedded, firmware, linux, qemu, serial, serial port, u-boot, uart, uboot*

Posted in: *Embedded*

← *Hello world for bare metal ARM using QEMU* *Scratchbox on Ubuntu* →

ip-xact java jtag kernel launchpad linux microcontrollers myhdl olimex opencores openocd openrisc open source open source hardware open source software privacy programming python qemu qemu-system-arm security self management serial serial port soc social soft work stm32 stmicroelectronics ted Texas Instruments truecrypt tutorial twitter u-boot uart ubuntu usb verilog vhdl wifi wordpress

License

93 Responses "U-boot for ARM on QEMU" →

Li Zhengke

2010/06/21

After success in excuteing cmd iminfo 0×25258, I did not see the output "hello world" by excuteing cmd bootm 0×25258.The output is below:

```
VersatilePB # iminfo 0×25258

## Checking Image at 00025258 ...
Legacy image found
Image Name:
Image Type: ARM U-Boot Standalone Program (uncompressed)
Data Size: 360 Bytes = 0.4 kB
Load Address: 00100000
Entry Point: 00100000
Verifying Checksum ... OK
VersatilePB # bootm 0×25258
## Booting kernel from Legacy Image at 00025258 ...
Image Name:
Image Type: ARM U-Boot Standalone Program (uncompressed)
Data Size: 360 Bytes = 0.4 kB
Load Address: 00100000
Entry Point: 00100000
Loading Standalone Program ... OK
OK

U-Boot 2010.03 (Jun 21 2010 – 00:41:30)

DRAM: 0 kB
## Unknown FLASH on Bank 1 – Size = 0×00000000 = 0 MB
Flash: 0 kB
*** Warning – bad CRC, using default environment

In: serial
Out: serial
Err: serial
Net: SMC91111-0
VersatilePB #
```

How can I get the correct output with hello world?

Balau

2010/06/22

Does it print "Hello world!" if you follow the instructions in the previous blog post?

If I can find time, I will try to replicate your problem.

From the U-Boot output, it seems to me that the system has been reset.

Balau

2010/06/22

Maybe I found the solution: the entry point was misplaced. Try to use this linker script file instead:

```
ENTRY(_Reset)
SECTIONS
{
. = 0x100000;
.startup . : { startup.o }
.text : { *(.text) }
.rodata : { *(.rodata) }
.data : { *(.data) }
.bss : { *(.bss) }
. = . + 0x1000; /* 4kB of stack memory */
stack_top = .;
}
```

Li Zhengke

2010/07/02

It does not work!Could you tell me which version of the qemu and u-boot you use?

Li Zhengke

2010/07/02

my qemu version is "QEMU PC emulator version 0.11.0 (qemu-kvm-0.11.0), Copyright (c) 2003-2008 Fabrice Bellard";and u-boot is u-boot-2010.03 and u-boot-2010.06.I tried on the 2 version of u-boot ,both are not OK.

Balau

I use u-boot-2010.03 as you do, but I have QEMU PC emulator version 0.12.3 (qemu-kvm-0.12.3), Copyright (c) 2003-2008 Fabrice Bellard

Could you take a look at the test.map file that is generated? It is important that

2010/07/02

the _Reset label is placed at 0×00100000. My test.map looks like this:

```
Memory Configuration

Name Origin Length Attributes
*default* 0x00000000 0xffffffff

Linker script and memory map

0x00100000 . = 0x100000

.startup 0x00100000 0xf8
startup.o()
.text 0x00100000 0x10 startup.o
0x00100000 _Reset
.data 0x00100010 0x0 startup.o
.bss 0x00100010 0x0 startup.o
.ARM.attributes
0x00100010 0x20 startup.o
.debug_line 0x00100030 0x39 startup.o
.debug_info 0x00100069 0x57 startup.o
.debug_abbrev 0x001000c0 0x14 startup.o
*fill* 0x001000d4 0x4 00
.debug_aranges
0x001000d8 0x20 startup.o
.glue_7 0x001000f8 0x0 startup.o
.glue_7t 0x001000f8 0x0 startup.o
.vfp11_veneer 0x001000f8 0x0 startup.o
.janus_2cc_veneer
0x001000f8 0x0 startup.o
.v4_bx 0x001000f8 0x0 startup.o

.text 0x001000f8 0x6c
*(.text)
.text 0x001000f8 0x6c test.o
0x001000f8 print_uart0
0x0010014c c_entry

.glue_7 0x00100164 0x0
.glue_7 0x00100164 0x0 test.o

.glue_7t 0x00100164 0x0
.glue_7t 0x00100164 0x0 test.o

.vfp11_veneer 0x00100164 0x0
.vfp11_veneer 0x00100164 0x0 test.o

.janus_2cc_veneer
0x00100164 0x0
.janus_2cc_veneer
0x00100164 0x0 test.o

.v4_bx 0x00100164 0x0
.v4_bx 0x00100164 0x0 test.o

.rodata 0x00100164 0x14
*(.rodata)
.rodata 0x00100164 0x14 test.o
0x00100164 UART0DR

.data 0x00100178 0x0
*(.data)
.data 0x00100178 0x0 test.o

.bss 0x00100178 0x4
*(.bss)
.bss 0x00100178 0x4 test.o
0x00100178 bkp
0x0010117c . = (. + 0x1000)
0x0010117c stack_top = .
LOAD test.o
LOAD startup.o
OUTPUT(test.elf elf32-littlearm)
...
```

Li Zhengke

2010/07/06

This is my test.map.the _Reset label is placed at 0×00100000 indeed.

Memory Configuration

Name Origin Length Attributes
default 0×00000000 0xffffffff

Linker script and memory map

0×00100000 . = 0×100000

.startup 0×00100000 0xf0
startup.o()

```
.text 0×00100000 0×10 startup.o
0×00100000 _Reset
.data 0×00100010 0×0 startup.o
.bss 0×00100010 0×0 startup.o
.ARM.attributes
0×00100010 0×24 startup.o
.debug_line 0×00100034 0×39 startup.o
.debug_info 0x0010006d 0x4e startup.o
.debug_abbrev 0x001000bb 0×14 startup.o
*fill* 0x001000cf 0×1 00
.debug_aranges
0x001000d0 0×20 startup.o

.text 0x001000f0 0x6c
*(.text)
.text 0x001000f0 0x6c test.o
0x001000f0 print_uart0
0×00100144 c_entry

.glue_7 0x0010015c 0×0
.glue_7 0×00000000 0×0 linker stubs

.glue_7t 0x0010015c 0×0
.glue_7t 0×00000000 0×0 linker stubs

.vfp11_veneer 0x0010015c 0×0
.vfp11_veneer 0×00000000 0×0 linker stubs

.v4_bx 0x0010015c 0×0
.v4_bx 0×00000000 0×0 linker stubs

.rodata 0x0010015c 0×14
*(.rodata)
.rodata 0x0010015c 0×14 test.o
0x0010015c UART0DR

.data 0×00100170 0×0
*(.data)
.data 0×00100170 0×0 test.o

.bss 0×00100170 0×0
*(.bss)
.bss 0×00100170 0×0 test.o
0×00101170 . = (. + 0×1000)
0×00101170 stack_top = .
LOAD test.o
LOAD startup.o
OUTPUT(test.elf elf32-littlearm)

.comment 0×00000000 0x2b
.comment 0×00000000 0x2b test.o

.ARM.attributes
0×00000000 0×30
.ARM.attributes
0×00000000 0×34 test.o
```

Balau

2010/07/06

I'm sorry I gave you incorrect information: for this exercise I tried U-Boot 1.1.6 (I was mislead by the compilation date that is displayed before the prompt) that is from 2006. I used your version of U-Boot only to boot Linux in this other post: http://balau82.wordpress.com/2010/04/12/booting-linux-with-u-boot-on-qemu-arm/

Try to see there if you find something that can help you. When I can, I will try to replicate it with the new u-boot.

Li Zhengke

2010/07/07

Thanks your help.Hoplefully,you will give us more excellent tutorials!

Balau

2010/07/10

Ok the problem was a mkimage option. The correct command is:
"mkimage -A arm -C none -O linux -T kernel -d test.bin -a 0×00100000 -e 0×00100000 test.uimg"

When booting a "standalone" program, the new U-Boot versions have a particular mechanism that run the program inside U-Boot environment. When booting any "kernel" U-Boot resets the environment, for example disabling interrupts, caches and MMUs. In this "clean" environment our program can do whatever it wants, while as a "standalone" program it was constrained by U-Boot settings. Maybe the MMU was active so the program couldn't access the UART memory address, or maybe the start address was fixed as another value different from 0×100000.

Note: the “linux” option in the makefile is only to fool U-Boot. Actually, we don’t run an operating system but a bare metal program.

simon
2010/10/10

Thanks for sharing your experience, really a good guide into uboot and qemu! Also I have some suggestion on booting “hello world” with uboot, I think we doesn’t need to modify the test.ld, although uboot launched to 0×1000 by qemu, but uboot will copy it self to 0×01000000, so it will be OK set the address of helloworld at 0×10000, and I have pass the test.

```
——————————
qemu-system-arm -M versatilepb -m 128M -nographic -kernel flash.bin
U-Boot 1.1.6 (Oct 10 2010 – 20:38:27)
DRAM: 0 kB
Flash: 0 kB
*** Warning – bad CRC, using default environment
In: serial
Out: serial
Err: serial
Versatile # bootm 0×90000
## Booting image at 00090000 …
Image Name:
Image Type: ARM Linux Kernel Image (uncompressed)
Data Size: 400 Bytes = 0.4 kB
Load Address: 00010000
Entry Point: 00010000
OK
Starting kernel …
Hello world!
——————————
```

the only question is we need to ensure the space from 0×10000 to address where helloworld should be larger than the size itself.

Adithya
2011/01/24

I enter this command in the U-boot directory:

```
make versatilepb_config ARCH=arm
CROSS_COMPILE=/home/aditz/CodeSourcery/Sourcery_G++_Lite/bin/arm-none-eabi-
```

Output:

```
Generating include/autoconf.mk
include/common.h:269:29: fatal error: asm/mach-types.h: No such file or directory
compilation terminated.
Generating include/autoconf.mk.dep
include/common.h:269:29: fatal error: asm/mach-types.h: No such file or directory
compilation terminated.
Configuring for versatile board…
Variant:: PB926EJ-S
```

Can u plz tell me where is this “asm” directory.
And how can i solve this problem

Balau
2011/01/24

The “include/asm” directory is a link to another directory. If you are using U-Boot 2010.03 then the link points to “u-boot-2010.03/include/asm-arm”. If you are using U-Boot 2010.09 then the link points to “u-boot-2010.09/arch/arm/include/asm”. The link is done during configuration (“make versatilepb_config …” in your case). Your output is strange because in my case the “Generating include/autoconf.mk” is done when I run the “make all …” command. Did you do a “make clean” or “make distclean” without specifying “ARCH” or “CROSS_COMPILE” ? because they tend to generate strange results. Try to do it again from a freshly unzipped source.

Adithya
2011/01/25

Thanks that worked !!!

Adithya
2011/01/25

In the command below:
“make versatilepb_config ARCH=arm CROSS_COMPILE=arm-none-eabi-”
where is the confg file “versatilepb_config” located ?
Can u plz tell me whether u-boot has a config file for realview-pbx-a9 ?

Balau
2011/01/25

The configuration files are header files in “include/configs” directory.
You can also check the various configuration files with the command “grep _config Makefile”
U-Boot currently has no support for that board, I think the people who could someday be able to develop it are the people in Linaro Kernel Consolidation Working Group, since they are currently working to support platforms with multiple Cortex-A9.

Kris
2011/03/09

Hi Balau,
Thanks a lot for excellent tutorials.
There is an issue while trying to build latest sources of U-boot. Build fails because of couple of undefined symbols.
U-boot source version/date: u-boot-2010.12
QEMU emulator version: 0.14.0

```
———————————————————-
kris@kris-laptop:~/bld_uboot/u-boot-2010.12$ make versatilepb_config ARCH=arm CROSS_COMPILE=arm-none-eabi-
Generating include/autoconf.mk
Generating include/autoconf.mk.dep
Configuring for versatile board…
Variant:: PB926EJ-S

kris@kris-laptop:~/bld_uboot/u-boot-2010.12$ make all ARCH=arm CROSS_COMPILE=arm-none-eabi-

[…]

make[1]: Entering directory `/home/kris/bld_uboot/u-boot-2010.12/arch/arm/lib’
arm-none-eabi-gcc -g -Os -fno-common -ffixed-r8 -msoft-float -D__KERNEL__ -DCONFIG_SYS_TEXT_BASE=0×01000000 -I/home/kris/bld_uboot/u-boot-2010.12/include -fno-builtin -ffreestanding -nostdinc -isystem /home/kris/CodeSourcery/Sourcery_G++_Lite/bin/../lib/gcc/arm-none-eabi/4.5.1/include -pipe -DCONFIG_ARM -D__ARM__ -marm -mabi=aapcs-linux -mno-thumb-interwork -march=armv5te -Wall -Wstrict-prototypes -fno-stack-protector \
-o board.o board.c -c
board.c: In function ‘__dram_init_banksize’:
board.c:233:29: error: ‘CONFIG_SYS_SDRAM_BASE’ undeclared (first use in this function)
board.c:233:29: note: each undeclared identifier is reported only once for each function it appears in
board.c: In function ‘board_init_f’:
board.c:279:18: error: ‘CONFIG_SYS_INIT_SP_ADDR’ undeclared (first use in this function)
————————————————————-
```

As a quick fix, by defining those two symbols to be a dummy value of zero in the header file,
~/bld_uboot/u-boot-2010.12/arch/arm/include/asm/config.h, and including it board.c source file , I was able to build uboot.bin. But,
qemu-system-arm -M versatilepb -m 128M -nographic -kernel u-boot.bin

didn’t show any U-boot prompt, there was no output of any kind, and qemu had to be terminated.

I would appreciate your comments on correct settings for those symbols, and any inputs to create a proper u-boot.bin with above said versions of sources and tools. Thanks.

Kris

Balau
2011/03/10

U-Boot developers recently changed the booting procedure for ARM cores, maybe they haven’t cleaned up the versatilepb build.
regarding your two symbols, I think CONFIG_SYS_SDRAM_BASE should work at 0, but CONFIG_SYS_INIT_SP_ADDR surely has the wrong value! This is because CONFIG_SYS_INIT_SP_ADDR is the initial address of the stack pointer, and the stack grows from top to bottom. Try to use 0×8000000, which is 128 MegaBytes. In this way you should place the stack at the end of the available memory.

Kris
2011/03/11

Thanks for your inputs. When CONFIG_SYS_INIT_SP_ADDR was set to 0×8000000, there was a different outcome as seen below.

```
————————————————
kris@kris-laptop:~/bld_uboot/u-boot-2010.12$ qemu-system-arm -M versatilepb -m 128M -nographic -kernel u-boot.bin
qemu: fatal: Trying to execute code outside RAM or ROM at 0xff000700
```

```
R00=fffcbf70 R01=ffff0000 R02=00000000 R03=01000000
R04=ffff0000 R05=fffcbf70 R06=ffff0000 R07=00000000
R08=08000000 R09=feff0000 R10=0101b1d4 R11=00000000
R12=fffcbfe8 R13=fffcbf68 R14=ff000700 R15=ff000700
PSR=600001d3 -ZC- A svc32
Aborted
——————————————————
```

While looking for some pointers about "outside RAM or ROM" error case, learnt that "success (of U-boot binary) is dependent on the exact combination of emulated machine and version of QEMU." from the discussion on the following page.
http://www.mail-archive.com/u-boot@lists.denx.de/msg42387.html

Instead of trying with different source versions of U-boot, got the U-boot binary for Versatile PB from the following site
http://arm.com/community/software-enablement/linux.php

As seen in the following log, the downloaded version (U-Boot 2010.09-rc2 (Sep 27 2010 – 07:21:34)) is working file on QEMU emulator version: 0.14.0, that I have.

```
—————————————————
kris@kris-laptop:~/bld_uboot/u-boot-2010.12$ qemu-system-arm -M
versatilepb -m 128M -nographic -kernel u-boot_bin_u-boot_versatilepb.axf

U-Boot 2010.09-rc2 (Sep 27 2010 – 07:21:34)
Code cloned from branch 090728_armdevCS of git://linux-arm.org/u-boot-
armdev.git
Release AEL-5.0
Remote commit cloned UNKNOWN
Latest commit locally UNKNOWN
git state UNKNOWN
DRAM: 0 Bytes
## Unknown FLASH on Bank 1 – Size = 0×00000000 = 0 MB
Flash: 0 Bytes
*** Warning – bad CRC, using default environment
In: serial
Out: serial
Err: serial
Net: SMC91111-0
Hit any key to stop autoboot: 0
Wrong Image Format for bootm command
ERROR: can't get kernel image!
VersatilePB #
——————————————————-
```

Now, Having a working combination of U-boot, QEMU for Versatile PB, I tried to run the "Hello World" program as per the current tutorial. But, not yet successful, because of the error "Unknown image format", as shown in the following log. Even tried with a u-boot.uimg, derived from uboot.bin, instead of test.uimg to creat flash.bin, but the error was same.

I am trying to understand why U-boot was looking for an image at 0x00007fc0 by default.

```
———————————————————
kris@kris-laptop:~/bld_uboot/u-boot-2010.12$ qemu-system-arm -M
versatilepb -m 128M -nographic -kernel flash.bin

U-Boot 2010.09-rc2 (Sep 27 2010 – 07:21:34)
[…]
DRAM: 0 Bytes
## Unknown FLASH on Bank 1 – Size = 0×00000000 = 0 MB
Flash: 0 Bytes
*** Warning – bad CRC, using default environment
In: serial
Out: serial
Err: serial
Net: SMC91111-0
Hit any key to stop autoboot: 0
Wrong Image Format for bootm command
ERROR: can't get kernel image!

VersatilePB # iminfo
## Checking Image at 00007fc0 …
Unknown image format!

VersatilePB # iminfo 0x262f4
## Checking Image at 000262f4 …
Unknown image format!
VersatilePB # QEMU: Terminated
————————————————————
```

Kris

Balau

2011/03/13

I think that U-boot is looking for an image at 0x00007fc0 by default because the "arm.com" version has been compiled with that configuration. Also, I see that you are trying to use a "axf" file, which is an executable file generated from ARM RVDS toolchain, and not a real binary file. I suppose the U-Boot source that arm.com is using to compile is heavily patched because the GCC toolchain for which U-Boot is designed to be built is very different. For these reasons I don't suggest using the pre-compiled images but build one yourself.

In my blog post I'm using the 2010.03 version that you can compile yourself to have full control of the configuration. When you see that the version works, you can go up one version at a time (2010.06, then 2010.09, then 2010.12) and see if it still works. My tutorial is valid for version 2010.03, and if it doesn't work for other versions it means that the U-Boot developers have changed something; unfortunately I have little time to keep myself updated on all the U-Boot development.

Kris

2011/03/15

Hi Balau, thanks for taking time out to suggest on the possible issues with pre-compiled binaries, and those arising out of version incompatibles. I was able to run U-boot on QEMU successfully by sticking to the respective versions mentioned in the tutorial.

By the way, in my experiment with pre-compiled version of U-boot, I'd coverted .axf to .bin the following way, and used it to create flash.bin
kris@kris-laptop:~/bld_uboot/u-boot-2010.12$ arm-none-eabi-objcopy -O binary u-boot_bin_u-boot_versatilepb.axf uboot.bin

Kris

Adithya

2011/04/07

Hello balau,

In the belo statement:

printf "bootm 0x%X\n" $(expr $(stat -c%s u-boot.bin) + 65536)

What does the value 65536 indicate ???

Regards
B. Adithya

Adithya

2011/04/07

Hello balau,

In the below statement:

arm-none-eabi-ld -T test.ld -Map=test.map test.o startup.o -o test.elf

1) What if -Map=test.Map option used? I guess it was not specified during the bare metal arm program.
2) Where is this file test.Map located.

Regards
B. Adithya

Balau

2011/04/07

In the statement:
`printf "bootm 0x%X\n" $(expr $(stat -c%s u-boot.bin) + 65536)`
the value 65536 (`0x10000` hexadecimal) is used as an offset because QEMU, when you pass a binary with the "`-kernel`" option, puts the binary at address `0x10000` that marks the absolute address of the beginning of `u-boot.bin`. The absolute address of the beginning of `test.uimg` is 0×10000 plus the size of `u-boot.bin`, because it is placed exactly after by the "`cat`" command.

The `test.map` file is an optional output of the linking step. It could be useful to understand where the various functions and variables are put, but it is not necessary to generate it, and the output file name can be anything. It should be generated in the same directory where the "`arm-none-eabi-ld`" command is executed, so in this case it should be the same directory as test.ld, test.elf, …

A.Ramachandran

2011/04/22

Thank you for your work. when i try to make using following command
make all ARCH=arm CROSS_COMPILE=arm-none-linux-gnueabi-
i got error
Fatal error: Invalid -march= option: `armv4'
i am using fedora 14, complete report
[root@localhost u-boot-1.1.6]# make all ARCH=arm CROSS_COMPILE=arm-none-linux-gnueabi-
for dir in tools examples post post/cpu ; do make -C $dir _depend ; done
make[1]: Entering directory `/home/Raman/s3c2440/u-boot-1.1.6/tools'

```
make[1]: Nothing to be done for `_depend'.
make[1]: Leaving directory `/home/Raman/s3c2440/u-boot-1.1.6/tools'
make[1]: Entering directory `/home/Raman/s3c2440/u-boot-1.1.6/examples'
make[1]: Nothing to be done for `_depend'.
make[1]: Leaving directory `/home/Raman/s3c2440/u-boot-1.1.6/examples'
make[1]: Entering directory `/home/Raman/s3c2440/u-boot-1.1.6/post'
make[1]: Nothing to be done for `_depend'.
make[1]: Leaving directory `/home/Raman/s3c2440/u-boot-1.1.6/post'
make[1]: Entering directory `/home/Raman/s3c2440/u-boot-1.1.6/post/cpu'
make[1]: Nothing to be done for `_depend'.
make[1]: Leaving directory `/home/Raman/s3c2440/u-boot-1.1.6/post/cpu'
make -C tools all
make[1]: Entering directory `/home/Raman/s3c2440/u-boot-1.1.6/tools'
make[1]: Leaving directory `/home/Raman/s3c2440/u-boot-1.1.6/tools'
make -C examples all
make[1]: Entering directory `/home/Raman/s3c2440/u-boot-1.1.6/examples'
arm-none-linux-gnueabi-gcc -g -Os -fno-strict-aliasing -fno-common -ffixed-
r8 -msoft-float -D__KERNEL__ -DTEXT_BASE=0×01000000 -
I/home/Raman/s3c2440/u-boot-1.1.6/include -fno-builtin -ffreestanding -
nostdinc -isystem include -pipe -DCONFIG_ARM -D__ARM__ -march=armv4
-mabi=apcs-gnu -Wall -Wstrict-prototypes -c -o hello_world.o hello_world.c
Assembler messages:
Fatal error: Invalid -march= option: `armv4'
hello_world.c:1:0: warning: target CPU does not support interworking
make[1]: *** [hello_world.o] Error 2
make[1]: Leaving directory `/home/Raman/s3c2440/u-boot-1.1.6/examples'
make: *** [examples] Error 2
[root@localhost u-boot-1.1.6]#
```
please tell me how to clear this error

william estrada
2011/04/22

A.Ramachandr
I am running Fedora 13 and I use "arm-gp2x-linux-".

Balau
2011/04/23

I tried to run the command that compiles hello_world.c, and it gets compiled with the "CPU does not support interworking" warning but without errors.
It seems that your error does not come from the compiler but from the assembler.
What's your compiler version? mine is the following:

```
$ arm-none-linux-gnueabi-gcc --version
arm-none-linux-gnueabi-gcc (Sourcery G++ Lite 2010q1-202)
4.4.1
...
$ arm-none-linux-gnueabi-as --version
GNU assembler (Sourcery G++ Lite 2010q1-202)
2.19.51.20090709
...
```

A.Ramachandran
2011/04/23

For assembler

```
[root@localhost u-boot-1.1.6]# arm-none-linux-gnueabi-as –version

GNU assembler (Sourcery G++ Lite 2010.09-50) 2.20.51.20100809
Copyright 2010 Free Software Foundation, Inc.
This program is free software; you may redistribute it under the terms of
the GNU General Public License version 3 or later.
This program has absolutely no warranty.
This assembler was configured for a target of `arm-none-linux-gnueabi'.
[root@localhost u-boot-1.1.6]#
```

A.Ramachandran
2011/04/25

Compiler version is 4.5.1

```
arm-none-linux-gnueabi-gcc (Sourcery G++ Lite 2010.09-50) 4.5.1
Copyright (C) 2010 Free Software Foundation, Inc.
This is free software; see the source for copying conditions. There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A
PARTICULAR PURPOSE.
```

After redefined the g cc path and compile, i got the following error

```
arm-none-linux-gnueabi-ld: error: Source object /usr/local/arm/arm-
2010.09/bin/../lib/gcc/arm-none-linux-gnueabi/4.5.1/libgcc.a(_bswapsi2.o)
has EABI version 5, but target u-boot has EABI version 0
arm-none-linux-gnueabi-ld: failed to merge target specific data of file
/usr/local/arm/arm-2010.09/bin/../lib/gcc/arm-none-linux-
```

gnueabi/4.5.1/libgcc.a(_bswapsi2.o)
/usr/local/arm/arm-2010.09/bin/../lib/gcc/arm-none-linux-gnueabi/4.5.1/libgcc.a(_bswapsi2.o):(.ARM.exidx+0x0): undefined reference to `__aeabi_unwind_cpp_pr0'
make: *** [u-boot] Error 1

can you tell me how to resolve it.

Balau
2011/04/26

I found this link: http://www.mail-archive.com/u-boot@lists.denx.de/msg32245.html
It seems that the old version of U-Boot could generate your linker error.
Is there a particular reason for you to use the U-Boot 1.1.6? If you can use a newer version, I suggest downloading something like version 2010.09 from U-Boot FTP archive. If you need to use version 1.1.6, you can add an empty function like the patch that you can find in the previous link. The "unwind" functions implement "exceptions" in case for example of division by zero, so they are not necessary to make U-Boot work, they are called only in special errors.

Stan Wu
2011/06/30

It's worked for me 😛

VersatilePB # bootm 0x250E0
## Booting kernel from Legacy Image at 000250e0 ...
Image Name:
Image Type: ARM Linux Kernel Image (uncompressed)
Data Size: 144 Bytes = 0.1 kB
Load Address: 00100000
Entry Point: 00100000
Loading Kernel Image ... OK
OK

Starting kernel ...

Hello world!

Ref
2011/07/01

$ mkimage -A arm -C none -O linux -T kernel -d test.bin -a 0×00100000 -e 0×00100000 test.uimg
mkimage: invalid load address 0×00100000

——————
test.ld:

$ cat test.ld
ENTRY(_Reset)
SECTIONS
{
. = 0×100000;
.startup . : { startup.o }
.text : { *(.text) }
.rodata : { *(.rodata) }
.data : { *(.data) }
.bss : { *(.bss) }
. = . + 0×1000; /* 4kB of stack memory */
stack_top = .;
}

Ref
2011/07/01

Nevermind. Worked. As usual, typo.

gangadhar
2011/10/11

hi
how we can findout uboot source am using ltib.i installed uboot,but i unable to find source file uboot

Balau
2011/10/11

I am sorry but I have no experience of LTIB. I have quickly read this manual and it seems u-boot is probably downloaded in `/opt/freescale/pkgs/`

Hi ... I ve been reading articles in ur blog ... And they are very nice !!

Thought i should mention .... had to use the following instead of given !!

Rajesh G

2011/10/21

```
make versatilepb_config ARCH=arm
CROSS_COMPILE=~/CodeSourcery/Sourcery_G++_Lite/bin/arm-none-
eabi-
```

Because CodeSourcery installed there by default !!

Balau

2011/10/21

Yeah, in my case when I installed CodeSourcery as a binary executable it modified the .bashrc script to update the PATH variable. If it is not done, then you need to supply the full path to the CROSS_COMPILE as you did.

Thanks for the clarification.

Rajesh G

2011/10/22

Hi .. thanks for the reply … that was fast !!

Plz forgive my noobness … !!
I could get the first two steps (ie) make and make all correctly in u-boot version 2011.09 !!
But it says can't load into RAM or something in the QEMU step (third step) !!
I understand from the comments that various u-boot versions doesn't work !!
But just wanted to ask is that the only reason this is happening to me ??
If i try u-boot 2010.03, will the qemu boot ??

Balau

2011/10/22

At some point U-Boot developers changed the way ARM architectures are managed. But now some old architectures do not work well because U-Boot developers don't have the hardware available, and the Versatile is one of them. So if you use a version of U-Boot before that change (2010.03 or 2010.09), it should work.

jim zhang

2011/10/30

Balau:
I was following the steps in this tutorial, and had problem with following commad:

```
linuxplayer@ubuntu:~/u-boot-2011.09$ qemu-system-arm -M versatilepb -m
128M -nographic -kernel u-boot.bin
qemu: fatal: Trying to execute code outside RAM or ROM at 0xffff0700

R00=fffcbf70 R01=ffff0000 R02=00000000 R03=0101b9ac
R04=ffff0000 R05=fffcbf70 R06=ffff0000 R07=00000000
R08=008fff78 R09=feff0000 R10=0101b9ac R11=00000000
R12=fffcbfe8 R13=fffcbf60 R14=ffff0700 R15=ffff0700
PSR=600001d3 -ZC- A svc32
Aborted
```

My question is: where is the kernel image, and how qemu load kernel image?

```
linuxplayer@ubuntu:~/u-boot-2011.09$ ls
api config.mk drivers MAINTAINERS nand_spl rules.mk u-boot
arch COPYING examples MAKEALL net snapshot.commit u-boot.bin
board CREDITS fs Makefile onenand_ipl spl u-boot.lds
boards.cfg disk include mkconfig post System.map u-boot.map
common doc lib mmc_spl README tools u-boot.srec
linuxplayer@ubuntu:~/u-boot-2011.09$
```

I am using qemu-0.15.1.
Thank you very much for your help!

Balau

2011/10/31

As I wrote earlier, the problem of new versions of U-Boot is that they don't work anymore with "old" hardware. In my opinion your "problem" is that you simply are using a version of U-Boot that is too new.

About the kernel image question, if you are asking about the Linux kernel, well there's no Linux. Instead, there's a small "Hello world" program described in a previous blog post. If you want to boot Linux you can follow this post.

linux player

2011/11/03

if I am using i.mx35 hardware, what board can I use in u-boot and qemu to emulate it? Will a Realview board work? I am using latest u-boot in my project, and qemu has to match my real project. Your opinion is highly appreciated.

Balau

2011/11/03

Unfortunately QEMU does not support that hardware. You can see the list of ARM hardware that can be emulated by using "`qemu-system-arm -M ?`"
If you want to emulate an ARM11 processor like the one inside the i.mx35, then QEMU can do it with the "`-cpu arm1136`" option, but be aware that the peripherals and the memory map can be completely different.

chandan

2012/02/09

getting error with
make all ARCH=arm CROSS_COMPILE=arm-none-eabi-

Result:

```
for dir in tools examples/standalone examples/api arch/arm/cpu/arm926ejs
/root/uboot/arch/arm/cpu/arm926ejs/ ; do \
make -C $dir _depend ; done
make[1]: ./CodeSourcery/Sourcery_G++_Lite/bin/arm-none-linux-gnueabi-
gcc: Command not found
make[1]: Entering directory `/root/uboot/tools'
make[1]: Nothing to be done for `_depend'.
make[1]: Leaving directory `/root/uboot/tools'
make[1]: ./CodeSourcery/Sourcery_G++_Lite/bin/arm-none-linux-gnueabi-
gcc: Command not found
make[1]: Entering directory `/root/uboot/examples/standalone'
/bin/sh: ./CodeSourcery/Sourcery_G++_Lite/bin/arm-none-linux-gnueabi-
gcc: No such file or directory
dirname: missing operand
Try `dirname –help' for more information.
make[1]: Nothing to be done for `_depend'.
make[1]: Leaving directory `/root/uboot/examples/standalone'
make[1]: ./CodeSourcery/Sourcery_G++_Lite/bin/arm-none-linux-gnueabi-
gcc: Command not found
make[1]: Entering directory `/root/uboot/examples/api'
/bin/sh: ./CodeSourcery/Sourcery_G++_Lite/bin/arm-none-linux-gnueabi-
gcc: No such file or directory
dirname: missing operand
Try `dirname –help' for more information.
make[1]: Nothing to be done for `_depend'.
make[1]: Leaving directory `/root/uboot/examples/api'
make[1]: ./CodeSourcery/Sourcery_G++_Lite/bin/arm-none-linux-gnueabi-
gcc: Command not found
make[1]: Entering directory `/root/uboot/arch/arm/cpu/arm926ejs'
make[1]: Nothing to be done for `_depend'.
make[1]: Leaving directory `/root/uboot/arch/arm/cpu/arm926ejs'
make[1]: ./CodeSourcery/Sourcery_G++_Lite/bin/arm-none-linux-gnueabi-
gcc: Command not found
make[1]: Entering directory `/root/uboot/arch/arm/cpu/arm926ejs'
make[1]: Nothing to be done for `_depend'.
make[1]: Leaving directory `/root/uboot/arch/arm/cpu/arm926ejs'
make -C tools all
make[1]: ./CodeSourcery/Sourcery_G++_Lite/bin/arm-none-linux-gnueabi-
gcc: Command not found
make[1]: Entering directory `/root/uboot/tools'
make[1]: Leaving directory `/root/uboot/tools'
make -C examples/standalone all
make[1]: ./CodeSourcery/Sourcery_G++_Lite/bin/arm-none-linux-gnueabi-
gcc: Command not found
make[1]: Entering directory `/root/uboot/examples/standalone'
/bin/sh: ./CodeSourcery/Sourcery_G++_Lite/bin/arm-none-linux-gnueabi-
gcc: No such file or directory
dirname: missing operand
Try `dirname –help' for more information.
./CodeSourcery/Sourcery_G++_Lite/bin/arm-none-linux-gnueabi-gcc -g -Os
-fno-common -ffixed-r8 -msoft-float -D__KERNEL__ -
DCONFIG_SYS_TEXT_BASE=0x01000000 -I/root/uboot/include -fno-builtin
-ffreestanding -nostdinc -isystem -pipe -DCONFIG_ARM -D__ARM__ -
march=armv5te -Wall -Wstrict-prototypes -o hello_world.o hello_world.c -c
make[1]: ./CodeSourcery/Sourcery_G++_Lite/bin/arm-none-linux-gnueabi-
gcc: Command not found
make[1]: *** [hello_world.o] Error 127
make[1]: Leaving directory `/root/uboot/examples/standalone'
make: *** [examples/standalone] Error 2
```

Balau

2012/02/09

Check that you have in the `PATH` environment variable the directory containing `arm-none-eabi-gcc` or the toolchain you are using; usually when you install CodeSourcery it creates the toolchain executables in directory "`~/CodeSourcery/Sourcery_G++_Lite/bin/`".
It's very strange because even if you call `make` with "`CROSS_COMPILE=arm-none-eabi-`" it tries to compile with `arm-none-linux-gnueabi-gcc` instead.

getting error with following command:

chandan
2012/02/12

make all ARCH=arm CROSS_COMPILE=./CodeSourcery/Sourcery_G++_Lite/bin/arm-none-linux-gnueabi-

Result:

```
Generating include/autoconf.mk
Generating include/autoconf.mk.dep
./CodeSourcery/Sourcery_G++_Lite/bin/arm-none-linux-gnueabi-gcc -DDO_DEPS_ONLY \
-g -Os -fno-common -ffixed-r8 -msoft-float -D__KERNEL__ -DCONFIG_SYS_TEXT_BASE=0×01000000 -I/root/uboot/include -fno-builtin -ffreestanding -nostdinc -isystem /root/uboot/CodeSourcery/Sourcery_G++_Lite/bin/../lib/gcc/arm-none-linux-gnueabi/4.5.1/include -pipe -DCONFIG_ARM -D__ARM__ -marm -mabi=aapcs-linux -mno-thumb-interwork -march=armv5te -Wall -Wstrict-prototypes -fno-stack-protector \
-o lib/asm-offsets.s lib/asm-offsets.c -c -S
Generating include/generated/generic-asm-offsets.h
tools/scripts/make-asm-offsets lib/asm-offsets.s include/generated/generic-asm-offsets.h
for dir in tools examples/standalone examples/api arch/arm/cpu/arm926ejs /root/uboot/arch/arm/cpu/arm926ejs/ ; do \
make -C $dir _depend ; done
make[1]: ./CodeSourcery/Sourcery_G++_Lite/bin/arm-none-linux-gnueabi-gcc: Command not found
make[1]: Entering directory `/root/uboot/tools'
make[1]: Nothing to be done for `_depend'.
make[1]: Leaving directory `/root/uboot/tools'
make[1]: ./CodeSourcery/Sourcery_G++_Lite/bin/arm-none-linux-gnueabi-gcc: Command not found
make[1]: Entering directory `/root/uboot/examples/standalone'
/bin/sh: ./CodeSourcery/Sourcery_G++_Lite/bin/arm-none-linux-gnueabi-gcc: No such file or directory
dirname: missing operand
Try `dirname –help' for more information.
make[1]: Nothing to be done for `_depend'.
make[1]: Leaving directory `/root/uboot/examples/standalone'
make[1]: ./CodeSourcery/Sourcery_G++_Lite/bin/arm-none-linux-gnueabi-gcc: Command not found
make[1]: Entering directory `/root/uboot/examples/api'
/bin/sh: ./CodeSourcery/Sourcery_G++_Lite/bin/arm-none-linux-gnueabi-gcc: No such file or directory
dirname: missing operand
Try `dirname –help' for more information.
make[1]: Nothing to be done for `_depend'.
make[1]: Leaving directory `/root/uboot/examples/api'
make[1]: ./CodeSourcery/Sourcery_G++_Lite/bin/arm-none-linux-gnueabi-gcc: Command not found
make[1]: Entering directory `/root/uboot/arch/arm/cpu/arm926ejs'
make[1]: Nothing to be done for `_depend'.
make[1]: Leaving directory `/root/uboot/arch/arm/cpu/arm926ejs'
make[1]: ./CodeSourcery/Sourcery_G++_Lite/bin/arm-none-linux-gnueabi-gcc: Command not found
make[1]: Entering directory `/root/uboot/arch/arm/cpu/arm926ejs'
make[1]: Nothing to be done for `_depend'.
make[1]: Leaving directory `/root/uboot/arch/arm/cpu/arm926ejs'
make -C tools all
make[1]: ./CodeSourcery/Sourcery_G++_Lite/bin/arm-none-linux-gnueabi-gcc: Command not found
make[1]: Entering directory `/root/uboot/tools'
gcc -g -Wall -Wstrict-prototypes -O2 -fomit-frame-pointer -idirafter /root/uboot/include -idirafter /root/uboot/include2 -idirafter /root/uboot/include -I /root/uboot/lib/libfdt -I /root/uboot/tools -DCONFIG_SYS_TEXT_BASE=0×01000000 -DUSE_HOSTCC -D__KERNEL_STRICT_NAMES -c -o env_embedded.o /root/uboot/common/env_embedded.c
gcc -Wall -Wstrict-prototypes -O2 -fomit-frame-pointer -idirafter /root/uboot/include -idirafter /root/uboot/include2 -idirafter /root/uboot/include -I /root/uboot/lib/libfdt -I /root/uboot/tools -DCONFIG_SYS_TEXT_BASE=0×01000000 -DUSE_HOSTCC -D__KERNEL_STRICT_NAMES -pedantic -o envcrc.o envcrc.c -c
gcc -Wall -Wstrict-prototypes -O2 -fomit-frame-pointer -idirafter /root/uboot/include -idirafter /root/uboot/include2 -idirafter /root/uboot/include -I /root/uboot/lib/libfdt -I /root/uboot/tools -DCONFIG_SYS_TEXT_BASE=0×01000000 -DUSE_HOSTCC -D__KERNEL_STRICT_NAMES -pedantic -o envcrc crc32.o env_embedded.o envcrc.o sha1.o
make[1]: Leaving directory `/root/uboot/tools'
make -C examples/standalone all
make[1]: ./CodeSourcery/Sourcery_G++_Lite/bin/arm-none-linux-gnueabi-
```

```
gcc: Command not found
make[1]: Entering directory `/root/uboot/examples/standalone'
/bin/sh: ./CodeSourcery/Sourcery_G++_Lite/bin/arm-none-linux-gnueabi-
gcc: No such file or directory
dirname: missing operand
Try `dirname –help' for more information.
./CodeSourcery/Sourcery_G++_Lite/bin/arm-none-linux-gnueabi-gcc -g -Os
-fno-common -ffixed-r8 -msoft-float -D__KERNEL__ -
DCONFIG_SYS_TEXT_BASE=0×01000000 -I/root/uboot/include -fno-builtin
-ffreestanding -nostdinc -isystem -pipe -DCONFIG_ARM -D__ARM__ -
march=armv5te -Wall -Wstrict-prototypes -o hello_world.o hello_world.c -c
make[1]: ./CodeSourcery/Sourcery_G++_Lite/bin/arm-none-linux-gnueabi-
gcc: Command not found
make[1]: *** [hello_world.o] Error 127
make[1]: Leaving directory `/root/uboot/examples/standalone'
make: *** [examples/standalone] Error 2
```

Please help

chandan
2012/02/12

so sorry to post again the same query.

chandan
2012/02/12

my ./CodeSourcery/Sourcery_G++_Lite/bin/ directory contains arm-none-linux-gnueabi-gcc,
arm-none-linux-gnueabi-gcc-4.5.1 and arm-none-linux-gnueabi-g++ tools.
But the result is showing arm-none-linux-gnueabi-gcc command not found.

Balau
2012/02/12

Actually it's not the same query: now you explicitly wrote that the command has "CROSS_COMPILE=./CodeSourcery/Sourcery_G++_Lite/bin/arm-none-linux-gnueabi-" and that's exactly the problem. During the building of u-boot, "make" changes directory, so the relative path to "arm-none-linux-gnueabi-gcc" changes, and it can't find it anymore.
You have to add the **absolute** path to "arm-none-linux-gnueabi-gcc" inside the PATH environment variable, for example:

```
export
PATH="$PATH:/home/chandan/CodeSourcery/Sourcery_G++_Lite/bin/
```

And then run the building with:

```
make all ARCH=arm CROSS_COMPILE=arm-none-linux-gnueabi-
```

eng trojan
2012/02/14

while trying to simulate the u-boot.bin on qemu i have this error

qemu: fatal: Trying to execute code outside RAM or ROM at 0x33f801f4

```
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
```

also this error appears while simulating the flash.bin

Balau
2012/02/14

If you run QEMU adding the following options: "-d in_asm,cpu -D qemu.log -singlestep", it will dump a log file (qemu.log) containing state of the CPU and the instructions during guest execution.
It can be useful to understand where does it crash.
A thing that I notice is that R14 is 0x000100fc. R14 is the link register, which is the register that is used to return from a function. It is an indicator that the function that causes the problem is called from address 0x000100fc.
If you run "arm-none-eabi-objdump -dxS u-boot > u-boot.code" after building u-boot, you can inspect the disassembled code. Keep in mind that QEMU puts u-boot.bin to an offset of 0×10000, and u-boot itself is compiled to run at another offset, 0×01000000, so the call you should seek is probably indicated to be at address 0x010000fc.

chandan
2012/02/19

great tutorial !!!

I got "hello world" printed with the help of this article.

Thank you so much.

gowtham

2012/03/25

hi,
txs for the good tutorials…
i've got the problem at the end of the process…. after the printf "bootm 0x%X\n" $(expr $(stat -c%s u-boot.bin) + 65536) statement i get the address as bootm 0×22110… and when i run this in uboot prompt i get this "bad magic number" error… the o/p in the terminal is as foolows:

```
U-Boot 1.1.6 (Mar 24 2012 – 21:52:16)

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

In: serial
Out: serial
Err: serial
Versatile # bootm 0×22110
## Booting image at 00022110 …
Bad Magic Number
```

Balau

2012/03/25

You could use iminfo and md to search for different memory locations:

```
VersatilePB # iminfo 0x250E0

## Checking Image at 000250e0 …
Legacy image found
Image Name:
Image Type: ARM Linux Kernel Image (uncompressed)
Data Size: 392 Bytes = 0.4 kB
Load Address: 00100000
Entry Point: 00100000
Verifying Checksum … OK
VersatilePB # md 0x250E0 16
000250e0: 56190527 65377844 ddde6e4f 88010000 '..VDx7eOn……
000250f0: 00001000 00001000 83a6fc05 00020205 …………….
00025100: 00000000 00000000 00000000 00000000 …………….
00025110: 00000000 00000000 00000000 00000000 …………….
00025120: e59fd004 eb000054 eafffffe 00101188 ….T……….
00025130: 00002341 61656100 A#…aea
```

The Magic Number is the first four bits of the test.uimg file (be aware of the endianess):

```
$ hexdump -C test.uimg
00000000 27 05 19 56 44 78 37 65 4f 6e de dd 00 00 01 88 |'..VDx7eOn……|
00000010 00 10 00 00 00 10 00 00 05 fc a6 83 05 02 02 00 |…………….|
00000020 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |…………….|
*
00000040 04 d0 9f e5 54 00 00 eb fe ff ff ea 88 11 10 00 |….T……….|
00000050 41 23 00 00 00 61 65 61 62 69 00 01 19 00 00 00 |A#…aeabi……|
…
```

And you should find them in the flash.bin image at a different offset (you should subtract 0×10000 from the bootm address):

```
$ hexdump -C flash.bin
…
00015100 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |…………….|
*
00015120 04 d0 9f e5 54 00 00 eb fe ff ff ea 88 11 10 00 |….T……….|
00015130 41 23 00 00 00 61 65 61 62 69 00 01 19 00 00 00 |A#…aeabi……|
…
```

Anup

2012/05/04

Hi Balau, I am getting below error when i 'bootm' the address where image is stored. Any idea why??

```
U-Boot 2010.03 (May 02 2012 – 18:20:05)

DRAM: 0 kB
## Unknown FLASH on Bank 1 – Size = 0×00000000 = 0 MB
Flash: 0 kB
*** Warning – bad CRC, using default environment

In: serial
Out: serial
Err: serial
Net: SMC91111-0
VersatilePB # bootm 0x25B38
## Booting kernel from Legacy Image at 00025b38 …
Image Name:
Image Type: ARM Linux Kernel Image (uncompressed)
Data Size: 136 Bytes = 0.1 kB
Load Address: 00100000
```

```
Entry Point: 00100000
Loading Kernel Image ... OK
OK

Starting kernel ...

arm_sysctl_read: Bad register offset 0×1
arm_sysctl_read: Bad register offset 0×2
arm_sysctl_read: Bad register offset 0×3
arm_sysctl_read: Bad register offset 0×5
arm_sysctl_read: Bad register offset 0×6
arm_sysctl_read: Bad register offset 0×7
arm_sysctl_read: Bad register offset 0×9
arm_sysctl_read: Bad register offset 0xa
arm_sysctl_read: Bad register offset 0xb
arm_sysctl_read: Bad register offset 0xd
arm_sysctl_read: Bad register offset 0xe
arm_sysctl_read: Bad register offset 0xf
goes on till 0xd2 and qemu aborts
```

Balau
2012/05/04

I never encountered this error.
It seems that the execution tries to access the address of the system controller sequentially byte by byte.
Have you tried launching QEMU with "`-kernel zImage -append `" directly instead of passing the flash binary that contains U-Boot and the kernel?
If it gives the same error, then you have a problem in the kernel compilation. If not, then it's something about U-Boot and the memory map that it expects, which is somehow different than the memory map of the emulated system.

Anup
2012/05/04

Thanks for the reply Balau.

I will try out launching QEMU with zImage. By the way is QEMU version something to do with this?
I am using: QEMU PC emulator version 0.12.3 (qemu-kvm-0.12.3),

Balau
2012/05/05

It could be, my current QEMU version is 1.0.

Rishi Agrawal
2012/06/13

It will be good if with every post you can mention the version of the software you are using as most of the issues are caused by the different versions.

Rishi Agrawal
2012/06/14

Hi Balau, I have become fan of your post .. really … the best part is they are correct to every bit.

I would just like to suggest you that you should post a PREREQUISITE section on the top of every blog which will mention the packages you will be using along with the the exact versions. Like:
UBOOT 1.1.6
Linux Kernel 3.2.0
etc etc

sourav
2012/06/27

I have been following your post and its really useful.Thanks for such good and clear articles.I have few questions regarding u-boot.Does U-boot support versatile express ARM cortex a-15?I didn't see any such file under include/configs in u-boot source.I want to compile u-boot for ARM cortex a-15.Can you please help?

Balau
2012/06/27

It seems to me that the last U-Boot version contains omap5_evm which includes OMAP5430 support.
Other than that I have neither the experience nor the time to help you port U-Boot to the architecture you want.

Terence

Hey Balau,

Great guides first of all. I'm getting a "Segmentation fault (core dumped)" error after running qemu-system-arm -M versatilepb -m 128M -nographic -kernel u-

2012/07/16

boot.bin. Any ideas? I'm on Ubuntu 12.04

Balau

2012/07/16

It seems a problem of QEMU. You can also try with different binaries to pass to the "kernel" option, to make sure the behavior is independent from the file you pass.
Then you can submit a bug to https://bugs.launchpad.net/qemu-linaro describing your problem and how to replicate.

chinlin

2012/07/20

Hi Bailau,

Is it possible to set up a video address for qemu, and have sample code to fill color rgba data?

It would be great if you can give some sample.

Thanks

Balau

2012/07/20

I don't know if I understand the question, but I think what you want to do is speak directly to the video controller with bare metal code.
If it's so, the code depends on the peripheral that controls the screen, for example the VersatilePB uses a (slightly modified) PL110 CLCD controller. You should find information on how to use it in the PL110 manual and in the VersatilePB documentation.

I can't give you some sample because I never did something like that. If in the future I do some tests with bare metal code using the LCD controller inside QEMU, then I will surely write a blog post about it. I can't promise I'll do it in the near future.

Chad Colgur

2012/09/20

Thanks for the article. Your link to VersatilePB is dead. I found the following: http://www.arm.com/products/tools/development-boards/versatile/index.php.

Balau

2012/09/23

Thanks for the indication! The link you provided shows only the ARM11 version of the Versatile Platform Baseboard, while for ARM926 there's only the CoreTile + Emulation Baseboard coupling. It seems that ARM dropped support for the VersatilePB that QEMU is emulation. I'm going to link directly to the manual which seems to be still in place.
Thanks again!

shabbir

2012/10/17

Hi Balau,
I am trying to analyse the uboot code of samsung exynos 4210,in which i found enable_mmu in lowlevel_init.S.I want to know what is the need of MMU in uboot level.

Balau

2012/10/17

I really don't know.
I suppose it could be to increase security and robustness during U-Boot execution, or for launching bare metal programs in a constricted environment, acting as an "hypervisor".
You could try asking in an irc channel (for example on freenode), or on U-Boot mailing list.

buiquanghuyen

2012/10/23

Hi Balau ,
I was run uboot but No ethernet found.

```
"U-Boot 2009.11 (Oct 23 2012 – 03:48:31)

DRAM: 0 kB
## Unknown FLASH on Bank 1 – Size = 0×00000000 = 0 MB
Flash: 0 kB
*** Warning – bad CRC, using default environment

In: serial
Out: serial
Err: serial
Net: No ethernet found. "
```

Please help my fix problem .Thank You!

Balau

2012/10/23

In my example U-Boot does not use the Ethernet so it's not a problem, it can boot Linux correctly.
Ethernet is needed when you only have U-Boot on your target and you need to fetch the operating system from the net with something like tftp.
If you need something like that, then the Ethernet controller (NIC) must be supported by U-Boot, and I don't know if it's the case for my example with VersatilePB or with whatever architecture you are trying. You could probably find more support from the U-Boot mailing list or freenode.org IRC channels.

buiquanghuyen

2012/10/24

Hi Balau,
I load tftp file uImage and bootm. I don't known it is not run.

```
“U-Boot 2010.03 (Oct 23 2012 – 08:42:49)
DRAM: 0 kB
## Unknown FLASH on Bank 1 – Size = 0×00000000 = 0 MB
Flash: 0 kB
*** Warning – bad CRC, using default environment
In: serial
Out: serial
Err: serial
Net: SMC91111-0
VersatilePB # setenv ipaddr 10.0.2.15
VersatilePB # dhcp
SMC91111: PHY auto-negotiate timed out
SMC91111: MAC 52:54:00:12:34:56
BOOTP broadcast 1
DHCP client bound to address 10.0.2.15
Using SMC91111-0 device
TFTP from server 10.0.2.2; our IP address is 10.0.2.15
Filename '/tftpboot/uImage'.
Load address: 0x7fc0
Loading:
#############################################################
#############################################################
#############################################################
#############################################################
#############################################################
#########################
done
Bytes transferred = 1788440 (1b4a18 hex)
VersatilePB # bootm 0x7fc0
## Booting kernel from Legacy Image at 00007fc0 ...
Image Name: Linux-3.5.0
Image Type: ARM Linux Kernel Image (uncompressed)
Data Size: 1788376 Bytes = 1.7 MB
Load Address: 00008000
Entry Point: 00008000
Loading Kernel Image ... OK
OK

Starting kernel ...

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

Please see full log bootm at http://canhdongvang.x10.mx/DTVT/balau.png
Thank You !
hqbui.

Balau

2012/10/24

It seems U-Boot does its job, and then the kernel hangs or does not display anything.
Maybe the kernel is booting but it's printing the messages somewhere else. try to add "`console=/dev/ttyAMA0`" to the kernel parameters, or even using the `earlyprintk` parameter.
See also my blog post "Booting Linux with U-Boot on QEMU ARM" if you have not seen it.

balaji

2013/05/09

Hi Balau,

I have one doubt relating to u-boot code execution.

1) I generated u-boot.bin by using arm-none-eabi toolchain with some Text_BASE
0x41e00000.
2) I tried to run the u-boot.bin from the DDR2 with Load_ADDR as 0×40000000 with go
command.
3) How u-boot.bin gets executed with the Load_ADDR. What ever the labels

present in the u-boot.bin are with the TEXT_BASE addresses.
4) I have seen the disassembly of u-boot where i found the labels are generated with the start of TEXT_BASE.

5) Stand alone application gets hanged when run it with LOAD_ADDR other than with actual TEXT_BASE.

6)But the above case is not happened with u-boot.

Can you clarify me if it is possible. The above are all doing for understanding relocation concept in u-boot.

Thanks
balaji

Balau

2013/05/09

I don't know if I understand correctly, but from what I understood you have two u-boot programs, one (let's call it u-boot-A) is used as the initial boot, and with the "go" command you launch the second (u-boot-B).
Then you have a standalone application, that you launch with u-boot-A using the go command in the same way.
You are seeing that u-boot-B gets executed correctly independently on the address to which it is copied, while your standalone application works only if you copy it on a particular address.
I think probably the reason is because u-boot-B contains position-independent code. When you disassemble, the disassembly is probably decorated with label addresses, but the code itself doesn't contain absolute addresses but only relative addresses to the current program counter.

Then there's relocation, but it's a different thing. I suspect that u-boot-A relocates itself to a high RAM address, then when you run the "go" command u-boot-B starts, then relocates itself at the same high RAM address as u-boot-A, thus overwriting u-boot-A. My hypothesis is that u-boot-B relocates itself to an address that is the same independently the load address that you decided.

You could try to compile your standalone program as position-independent and see if it works. With C code you have the -fPIC option in GCC. With assembly code you need to code it yourself without absolute addresses.

Hope this helps.

balaji

2013/05/10

Your understand is correct. I try with -fpic option.

Thank you very much

balaji

2013/05/14

Hi Balau,

Is there any possibility to see the relative addresses of the labels from u-boot.elf. Even relocation flag is enabled iam only able to see the absolute addresses only. During runtime the addresses are not as absolute address after relocation when i checked with DS-5 Debugger. How absolute addresses are modified with relative addresses. I know it was happened with relative symbol table but who will do the conversion(absolute to relative) linker, compiler or loader responsibility.

For example In disassembly of u-boot i found the following instruction.

bl 3f00200 // Here 3f00200 is the absolute address.

I relocated the code to 0×7000000 and i enabled relocation flag then what would be the transformation of above instruction .

Please help me in this.

Thanks
balaji.

Balau

2013/05/15

I want to stress that I'm not sure that u-boot uses position-independent code, I was suggesting an hypothesis. Maybe u-boot is not completely compiled to be position-independent, and your investigation seems to prove it. In light of this information another hypothesis is that only the initial part of u-boot can be run anywhere, and u-boot relocates itself always at the same address, so that the rest of the execution can contain absolute addresses, and they remain the same because the instructions are placed always on the same addresses.

balaji

2013/05/31

Hi Balau,

During internal development of bootloader, I got MMU precise abort when "MCR p15, #0, r0, c1, c0, #0" is executed. I read the TTB0 register its address is same as my page table address. When i commented above mentioned instruction i am able to go to next instruction. But i need to enable the MMU. It is needed for testing the L2 cache. I didn't enable the caches before enabling the MMU.

Can you tell me is there any possibility to avoid the precise abort.

Thanks
balaji

Balau
2013/06/02

Unfortunately I have no experience with ARM MMU, I am not able to give you pointers with this.

manju
2013/07/04

hello sir ,
how to overcome this error..
:~/Desktop/embedded/uboot/u-boot-2013.01$ qemu-system-arm -M versatilepb -m 128M -nographic -kernel flash.bin

```
pulseaudio: set_sink_input_volume() failed
pulseaudio: Reason: Invalid argument
pulseaudio: set_sink_input_mute() failed
pulseaudio: Reason: Invalid argument
qemu: fatal: Trying to execute code outside RAM or ROM at 0×08000000

R00=fffcbf60 R01=00000000 R02=00000000 R03=00000000
R04=00000000 R05=fffcbf60 R06=00000000 R07=00000000
R08=fffcbf60 R09=00000000 R10=01000020 R11=00000000
R12=fffcbfe0 R13=fffcbf50 R14=00000710 R15=08000000
PSR=600001d3 -ZC- A svc32
s00=00000000 s01=00000000 d00=0000000000000000
s02=00000000 s03=00000000 d01=0000000000000000
s04=00000000 s05=00000000 d02=0000000000000000
s06=00000000 s07=00000000 d03=0000000000000000
s08=00000000 s09=00000000 d04=0000000000000000
s10=00000000 s11=00000000 d05=0000000000000000
s12=00000000 s13=00000000 d06=0000000000000000
s14=00000000 s15=00000000 d07=0000000000000000
s16=00000000 s17=00000000 d08=0000000000000000
s18=00000000 s19=00000000 d09=0000000000000000
s20=00000000 s21=00000000 d10=0000000000000000
s22=00000000 s23=00000000 d11=0000000000000000
s24=00000000 s25=00000000 d12=0000000000000000
s26=00000000 s27=00000000 d13=0000000000000000
s28=00000000 s29=00000000 d14=0000000000000000
s30=00000000 s31=00000000 d15=0000000000000000
FPSCR: 00000000
Aborted (core dumped)
```

Balau
2013/07/04

0×08000000 is 128MiB, so it's the first address not mapped as memory.
You could have a problem in your configuration of U-Boot, where the amount of memory is, and for this reason U-Boot is trying to relocate itself to an address that is too big.
Or you could have a problem where the execution jumps somewhere and then continues to execute empty memory until it reaches the end of the memory. In this case I suggest using the "-s -S" options and attaching to QEMU with a debugger.

manju
2013/07/05

sir i'm new to linux ….will u elaborate a little bit about using the "-s -S" options and attaching to QEMU with a debugger.

Balau
2013/07/07

Be aware that what you are trying to do is in my opinion very difficult if you are not familiar to Linux.

Another thing that you could check before debugging is the addresses in file "u-boot.map" that is created during u-boot build.
If the addresses are similar to these, then there's no problem:

```
Memory Configuration

Name Origin Length Attributes
*default* 0x00000000 0xffffffff

Linker script and memory map

0x00000000 . = 0x0
```

```
0x00000000 . = ALIGN (0x4)
```

```
.text 0x01000000 0x13360
arch/arm/cpu/arm926ejs/start.o(.text)
.text 0x01000000 0x460 arch/arm/cpu/arm926ejs/start.o
0x01000000 _start
0x01000040 _TEXT_BASE
0x01000044 _bss_start_ofs
0x01000048 _bss_end_ofs
0x0100004c _end_ofs
0x01000050 IRQ_STACK_START_IN
0x01000078 relocate_code
```

but if they start from 0x08000000, that's your problem. You should check CONFIG_SYS_TEXT_BASE because that's probably what's wrong with your configuration. I don't know why it could be wrong because I don't know how you built u-boot.

About my previous suggestion:
From "man qemy-system-arm" you can find:

```
...
-S Do not start CPU at startup (you must type 'c' in the
monitor).
```

```
-gdb dev
Wait for gdb connection on device dev. Typical connections
will
likely be TCP-based, but also UDP, pseudo TTY, or even
stdio are
reasonable use case. The latter is allowing to start QEMU
from
within gdb and establish the connection via a pipe:
```

```
(gdb) target remote | exec qemu-system-i386 -gdb stdio ...
```

```
-s Shorthand for -gdb tcp::1234, i.e. open a gdbserver on
TCP port
1234.
...
```

So you run:

```
$ qemu-system-arm -M versatilepb -m 128M -nographic -kernel
flash.bin -s -S
```

This will open up QEMU without doing anything, waiting for a gdb connection.
Then, assuming your toolchain is something like "arm-linux-gnueabi-*" you run in another terminal:

```
$ arm-linux-gnueabi-gdb
```

Then inside gdb, assuming you compiled u-boot in the same directory you launched gdb:

```
(gdb) file u-boot
(gdb) target remote localhost:1234
(gdb) stepi
(gdb) stepi
...
```

There are many gdb tutorials online.

8 Trackbacks For This Post

Links 18/3/2010: Steam and Linux; Red Hat's CEO Talks | Boycott Novell →
March 19th, 2010 → 03:33
[...] U-boot for ARM on QEMU [...]

Compiling Linux kernel for QEMU ARM emulator « Balau →
March 27th, 2010 → 19:06
[...] Posts Hello world for bare metal ARM using QEMUU-boot for ARM on QEMUCompiling Linux kernel for QEMU ARM emulatorSimplest bare metal program for ARMSimplest serial port [...]

Booting Linux with U-Boot on QEMU ARM « Balau →
April 12th, 2010 → 19:55
[...] with QEMU emulation of an ARM Versatile Platform Board, making it run bare metal programs, the U-Boot boot-loader and a Linux kernel complete with a Busybox-based file system. I tried to put everything [...]

Blog stats for 2010 « Balau →
January 2nd, 2011 → 17:25
[...] U-boot for ARM on QEMU March 2010 14 comments [...]

Linux ARM emulation articles →
January 10th, 2011 → 15:31
[...] u-boot-for-arm-on-qemu [...]

QEMU for ARM, revisited | d-code →
February 7th, 2013 → 12:52
[...] Configuration (board_name) for U-Boot is : versatileqemu, as described here : http://comments.gmane.org/gmane.comp.boot-loaders.u-boot/101980. (For info, QEMU places the beginning of the U-Boot binary at 0x10000 : http://balau82.wordpress.com/2010/03/10/u-boot-for-arm-on-qemu/) [...]

QEMU 1.5.0 released, a backward compatibility warning | Balau →
May 21st, 2013 → 20:51
[…] U-boot for ARM on QEMU […]

Linux Kernel Online and Book Resources collection | Kaiwan's Tech Blog →
August 1st, 2013 → 12:04
[…] U-Boot for ARM on QEMU […]

Leave a Reply

Enter your comment here...

Blog at WordPress.com. *The Inuit Types Theme.*

Balau
Freedom Embedded

Booting Linux with U-Boot on QEMU ARM

Posted on 2010/04/12 111

In recent months I played with QEMU emulation of an ARM Versatile Platform Board, making it run bare metal programs, the U-Boot boot-loader and a Linux kernel complete with a Busybox-based file system. 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 page
- `zImage`: the Linux kernel created in my previous post here
- `rootfs.img.gz`: the Busybox-based file system created in my previous post here

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 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).

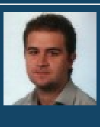
Understanding memory usage during the boot process is important because there is the risk of overwriting something during memory copy and relocation. One feature of U-Boot is self-relocation, which means that on execution the code copies itself into another address, which by default is `0x1000000` (16MiB). 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:

Search

Subscribe

Posts
Comments

Author



Top Posts & Pages

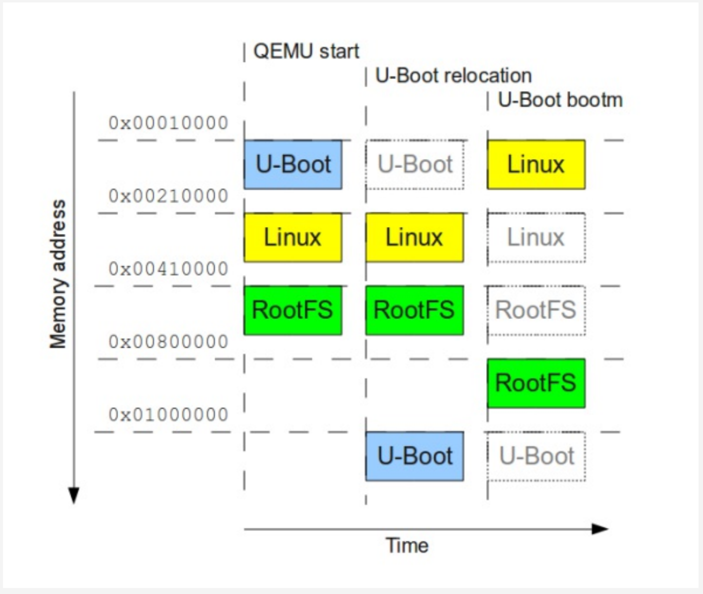
Programming Arduino Uno in pure C
Compile Linux kernel 3.2 for ARM and emulate with QEMU
Booting Linux with U-Boot on QEMU ARM
U-boot for ARM on QEMU
Hello world for bare metal ARM using QEMU
Simplest serial port terminal in C#
Simplest bare metal program for ARM
Trace and profile function calls with GCC
Busybox for ARM on QEMU
Using Newlib in ARM bare metal programs

Categories

Hardware
Links
Software
- Embedded
- Internet
- Security

Working

Tags



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/image.c
--- u-boot-2010.03.orig/common/image.c  2010-03-31 23:54:39.000000000 +0200
+++ 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_ARMADILLO)
+#if defined(CONFIG_B2) || defined(CONFIG_EVB4510) || defined(CONFIG_ARMADILLO
    /*
     * 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-2010.03/include/configs
--- u-boot-2010.03.orig/include/configs/versatile.h 2010-03-31 23:54:39.000000000 +02
+++ u-boot-2010.03/include/configs/versatile.h  2010-04-12 15:43:01.514733000 +020
@@ -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/init"
+#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 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`"

android ansic arduino ARM avr backup bare metal boot busybox codesourcery cortex-m3 csharp debian debug debugging eda electronics embedded encryption facebook firefox fpga free electrons ftdi gcc gdb gnu google hacking Hardware hdl howto icarus ip-xact java jtag kernel launchpad linux microcontrollers myhdl olimex opencores openocd openrisc open source open source hardware open source software privacy programming python qemu qemu-system-arm security self management serial serial port soc social soft work stm32 stmicroelectronics ted Texas Instruments truecrypt tutorial twitter u-boot uart ubuntu usb verilog vhdl wifi wordpress

License

After applying the patch, U-Boot can be built as seen in my previous post:

```
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 0x00010000 zI
mkimage -A arm -C none -O linux -T ramdisk -d rootfs.img.gz -a 0x00800000 -e 0x0080(
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.gz -append
```

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.

Google Twitter 4 More

Like this:

Tagged: *ARM*, *busybox*, *codesourcery*, *debian*, *linux*, *open source*, *qemu*, *u-boot*, *uboot*, *ubuntu*

Posted in: *Embedded*, *Hardware*

← *Browser plugin for Google Social Search* *Linux NFS Root under QEMU ARM emulator* →

111 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
http://ca9.upanh.com/19.0.24188231.OIR0/newbitmapimage.jpg

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

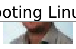
If your problem is the "-pflash" support, you can try these instructions : Using U-Boot and Flash emulation in Qemu
Other than that, you can try a newer u-boot version like the 2010.03, since you are using a version 1.1.6) of 5 years ago and maybe they fixed something.
You can also see that I have similar 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 0×0021000 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 0×0021000 0×0041000 .. didnt solve the problem .
Could you please help me out ?

I think you miss some "zeros" at the end of the address you are using. they


Balau

2011/03/13

should be "0×210000″ and "0×410000″
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 0×210000″ and "iminfo 0×410000″?
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 = 0×00000000 = 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 = 0×00000000 = 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 0×70023160 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 0×10000 (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 0×10000, 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 0×210000 0×410000″ 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:
http://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+0×8): undefined reference to `env_init'
lib_arm/libarm.a(board.o):(.data+0×10): 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.

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

Patrick

2012/03/14

QemV1. 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 0×90800000 uImage; bootm 0×90800000'
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 0×10000 (64KiB) and the ramdisk binary at address 0×800000 (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 0×210000"
in UBOOT versatile.h file, and prepared flash.bin
mkimage -A arm -C none -O linux -T kernel -d zImage -a 0×00010000 -e

0×00010000 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 0×800000 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 0×800000?

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+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
2012/06/24

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 0×800
uramdisk -load and entry address are 0×800000

$ tftp 0×100 uImage
$ tftp 0×4000000 uramdisk.img
$ setenv bootargs 'console=ttyAMA0,115200 root=/dev/ram rw'
$ bootm 0×100 0×4000000

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)

Let me clear my complete opinion.
I am convinced that if you try "`setenv bootargs`

Balau

2012/06/28

`'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:

Ask U-Boot mailing list
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?

In my environment, it is clear that it does not understand the "arrow" keys as "go up in history of commands".

Balau

2012/07/03

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 0×00010000 -e
0×00010000 uImage
$ mkimage -A arm -C gzip -O linux -T ramdisk -d rootfs.cpio.gz -a
0×00800000 -e 0×00800000 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 0×210000 0×410000
## 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 0×00800000 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" (0×08000000), but in my post I talk about address 0×00800000. 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.

Hi Balau,

I am using zynq_zc702 based U-Boot 2011.03 source for running on zynq's balaji

2013/04/16

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 0×210000 0×410000″
#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 0×1
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 0×210000 gave the following information.

```
## Checking Image at 00210000 …
Unknown image format!.
```

I am not able to see any content at 0×210000 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:

U-boot relocation overwrote your image
The "-kernel" option does not place the binary at 0×00010000

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

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`.

2013/07/27

You can increase the value of `DEBUG` to print also “`debugX(level, ...)`” messages.
These macros are defined in “`include/common.h`“.

7 Trackbacks For This Post

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 […]

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

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

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

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

【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 […]

Leave a Reply

Enter your comment here...

Blog at WordPress.com. *The Inuit Types Theme.*