Introduction to U-Boot bootloader

Marek Vašut <marek.vasut@gmail.com>

March 8th, 2018

Marek Vasut

- Software engineer
- Versatile Linux kernel hacker
- Custodian at U-Boot bootloader
- OE-core contributor (Yocto...)
- FPGA enthusiast

Structure of the talk

- What is U-Boot bootloader
- U-Boot walkthrough
- Conclusion

Booting a computer

- Multi-stage bootloader
- First stage on reset vector
- Inits HW, loads next stage
- OS kernel
- Userspace

U-Boot bootloader

- Boot loader
  - First[1]-ish code that runs on a system
  - Responsible for some HW initialization and starting OS
- Boot monitor
- Debug tool

[1] There are exceptions, ie. Boot ROMs

U-Boot example

```
U-Boot SPL 2018.01-00002-g9aa111a004 (Jan 20 2018 - 12:45:29)
Trying to boot from MMC1


U-Boot 2018.01-00002-g9aa111a004 (Jan 20 2018 - 12:45:29 -0600)

CPU  : AM335X-GP rev 2.1
I2C:   ready
DRAM:  512 MiB
Reset Source: Global warm SW reset has occurred.
Reset Source: Power-on reset has occurred.
MMC:   OMAP SD/MMC: 0, OMAP SD/MMC: 1

Model: BeagleBoard.org PocketBeagle
Net:   usb_ether
Press SPACE to abort autoboot in 2 seconds
=>
```

What can U-Boot do?

```
=> echo hello world
hello world
=> help
?       - alias for 'help'
bdinfo  - print Board Info structure
bootm   - boot application image from memory
cmp     - memory compare
coninfo - print console devices and information
crc32   - checksum calculation
dfu     - Device Firmware Upgrade
dhcp    - boot image via network using DHCP/TFTP protocol
echo    - echo args to console
go      - start application at address 'addr'
gpio    - query and control gpio pins
help    - print command description/usage
i2c     - I2C sub-system
load    - load binary file from a filesystem
usb     - USB sub-system
```

Getting further help

```
=> help mmc
mmc - MMC sub system

Usage:
mmc info - display info of the current MMC device
mmc read addr blk# cnt
mmc write addr blk# cnt
mmc erase blk# cnt
mmc rescan
mmc part - lists available partition on current mmc device
mmc dev [dev] [part] - show or set current mmc device [partition
mmc list - lists available devices
```

- Source, documentation in doc/
  `http://git.denx.de/?p=u-boot.git;a=tree;f=doc`
- IRC: `irc.freenode.net` #u-boot
- ML: `u-boot@lists.denx.de`

Probing system info

```
=> bdinfo
arch_number = 0x00000E05
boot_params = 0x80000100
DRAM bank   = 0x00000000
-> start    = 0x80000000
-> size     = 0x20000000
eth0name    = usb_ether
ethaddr     = 60:64:05:f4:79:7f
current eth = usb_ether
ip_addr     = 192.168.1.2
baudrate    = 115200 bps
TLB addr    = 0x9FFF0000
relocaddr   = 0x9FF44000
reloc off   = 0x1F744000
irq_sp      = 0x9DF23EC0
sp start    = 0x9DF23EB0
Early malloc usage: 2a8 / 400
```

Poking the memory

```
=> md 0x9FF44000
9ff44000: ea0000b8 e59ff014 e59ff014 e59ff014 ................
9ff44010: e59ff014 e59ff014 e59ff014 e59ff014 ................
=> mw 0x81000000 0x1234abcd 0x10
=> mw 0x82000000 0x1234abcd 0x10
=> cmp.b 0x81000000 0x82000000 0x40
Total of 64 byte(s) were the same

=> echo "Try toggling GPIOs the hard way"
=> md 0x4804c130 4
4804c130: 00000002 ffffffff f0000300 00000000 ................
=> mw 0x4804c134 0xfe1fffff
=> mw 0x4804c13c 0x00a00000
=> mw 0x4804c13c 0x01400000
=> md 0x4804c130 4
4804c130: 00000002 fe1fffff f1400300 01400000 ..........@...@.
```

U-Boot shell

- There are two – HUSH and the old no-name
- Persistent environment support
- Scripting support
- Similar to bourne shell

Environment access

```
1  => printenv
2  arch=arm
3  ...
4  Environment size: 26907/131068 bytes
5  =>
6  => printenv arch
7  arm
8  => echo "$arch"
9  arm
10
11 => setenv foo bar
12 => printenv foo
13 bar
14
15 => env ask quux "Set quux to"
16 Set quux to 1
17 => printenv quux
18 quux=1
```

Environment persistency

```
1  => setenv foo bar
2  => printenv foo
3  bar
4  => reset
5  => printenv foo
6  ## Error: "foo" not defined
7
8  => setenv foo bar
9  => saveenv
10 => reset
11 => printenv foo
12 bar
```

The run command

```
1 => setenv foo 'echo hello'
2 => run foo
3 hello
4
5 => setenv foo 'echo hello ; echo world'
6 => run foo
7 hello
8 world
```

Conditional expressions

```
=> true ; echo $?
0
=> false ; echo $?
1
=> if true ; do echo "hello" ; else echo "bye" ; fi
hello
=> false || echo "false!"
false!

=> setenv foo 'true && "true!"'
=> run foo
true!
```

Variables in environment

```
1  => setenv foo bar
2  => setenv quux echo $foo
3  => setenv foo baz
4  => run quux
5  bar
6  => printenv quux
7  quux=echo bar
8
9  => setenv quux echo \$foo
10 => printenv quux
11 => setenv quux 'echo $foo'
12 => printenv quux
```

Advanced manipulation of environment

```
=> md 0x9ff4e000 1
9ff4e000: ea0000b8
=> setexpr foo *0x9ff4e000
=> pri foo
foo=ea0000b8

=> setexpr foo gsub ab+ x "aabbcc"
foo=axcc
```

Loading from storage

```
=> mmc rescan
=> mmc part

Partition Map for MMC device 0  --   Partition Type: DOS

Part    Start Sector    Num Sectors     UUID            Type
  1     8192            6955008         1147c091-01     83 Boot
=> ls mmc 0:1
<DIR>         4096 .
<DIR>         4096 ..
<DIR>        16384 lost+found
              1359 bbb-uEnv.txt
...
=> load mmc 0:1 $loadaddr ID.txt
=> md.b $loadaddr $filesize
82000000: 42 65 61 67 6c 65 42 6f 61 72 ... BeagleBoard.org
82000010: 44 65 62 69 61 6e 20 49 6d 61 ... Debian Image 201
82000020: 38 2d 30 31 2d 32 38 0a           8-01-28.
```

Loading from network

```
=> setenv ethaddr 00:aa:bb:cc:dd:ee # optional!
=> setenv ipaddr 192.168.1.300
=> setenv netmask 255.255.255.0
=> setenv serverip 192.168.1.1
=> ping $serverip
=> tftp $loadaddr $serverip:somefile
=> dhcp $loadaddr $serverip:somefile
```

Loading over serial port

```
=> loady
<send file over ymodem protocol>
```

Booting the kernel

There are many image formats

- (z)Image (with separate DT)
- uImage , legacy since forever
- fitImage , multi-component image

Booting kernel image

```
=> printenv bootcmd
bootcmd=if test ${boot_fit} -eq 1; then
run update_to_fit; fi; run findfdt;
run init_console; run envboot; run distro_bootcmd
=> boot # equals run bootcmd

=> help bootz # or bootm for uImage/fitImage
bootz - boot Linux zImage image from memory

Usage:
bootz [addr [initrd[:size]] [fdt]]
    - boot Linux zImage stored in memory
        The argument 'initrd' is optional... The optional arg
        ':size' allows specifying the size of RAW initrd.

        When booting a Linux kernel which requires a flat
        device-tree a third argument is required which is
        the address of the device-tree blob.
```

U-Boot sources

- Git master at:
  `http://git.denx.de/?p=u-boot.git;a=summary`
- Custodian subtrees at:
  `http://git.denx.de/?p=u-boot.git;a=forks`
- Available via Git and HTTP protocols

Building the sources

```
$ git clone git://git.denx.de/u-boot.git
$ cd u-boot
$ export ARCH=plat                  # optional, set target architecture
$ export CROSS_COMPILE=plat-none-   # optional, set cross compiler
$ make board_defconfig              # ie. sandbox_defconfig
$ make
```

Practical part

- Task 1: Conveniently load custom environment
  HINT: loady and env import commands
- Task 2: Implement moving light using USR LEDs
  HINT: for and while commands
- Task 3: Barcode reader
  U-Boot queries ethernet MAC address from barcode reader, which does not necessarily use ASCII. Filter the MAC out and ignore the separators (ie. 00xaaxbbxccxddxee becomes 00:aa:bb:cc:dd:ee)
  HINT: askenv and setexpr

Practical part

- Task 4: Recovery system
  Check if USB stick is plugged in and contains kernel image and DT. If so, boot those, otherwise boot the images on SD card.
  HINT: usb and load commands
- Task 5: Bootdelay
  Clone U-Boot sources, configure them, adjust bootdelay to 30 seconds, compile U-Boot and install on the board.
  HINT:
  make am335x_pocketbeagle_defconfig
  make menuconfig # locate CONFIG_BOOTDELAY
  make
  dd if=u-boot.img of=/dev/sdg bs=384k seek=1 count=2

The End

Thank you for your attention!

Contact: Marek Vasut <marek.vasut@gmail.com>