

Embedded Systems Boot Loaders

- Design & Concepts -

Agenda

- **Motivation – Why do we need this lecture**
- **Booting – The boot sequence, commonly used open source boot loaders**
- **Linux boot process – The boot process in a nutshell**
- **Role of a Bootloader – What would we like from a boot loader**
- **Development – Developing with open source boot-loader**
- **Das U-Boot – to demonstrate the concept**

Motivation

- **Every system has a boot-loader.**
- **The boot-loader provides the foundation from which the other system software is spawned.**
- **In the past it was difficult to develop boot-loaders , it was out of reach for most engineers : no debug tools ,poor set of device drivers, “only HW guys understood”, no common base-code.**
- **Modern boot-loaders have expanded their functionality and can be used as a powerful tool for HW bring-up and diagnostics.**

We will present how boot loaders are developed these days, open source code base supporting most of the hardware in the market, and providing a rich set of features easing the product development process.

Booting

- **Booting – A bootstrapping process that starts Operating system when the user turns on a computer system.**
- **A boot sequence is the initial set of operations that the computer performs when it is switched on.**

Common Boot Loaders

- **Many of the open source boot loaders are available supporting wide variety of CPUs and boards.**
- **Embedded platforms :**
 - U-boot – evolved from PPC-boot project supports PowerPC, ARM, XScale, MIPS, Coldfire, NIOS, Microblaze, and x86.**
 - RedBoot – part of the ecos project supports ARM, FR-V, H8, M68K, IA32, MIPS, NEC V8xx, PowerPC, SPARC, SuperH .**

Common Boot Loaders - cont

uMon - MicroMonitor package supports ARM, XScale, MIPS, PowerPC, ColdFire/68K, and SH.

- **BIOS Replacement :**

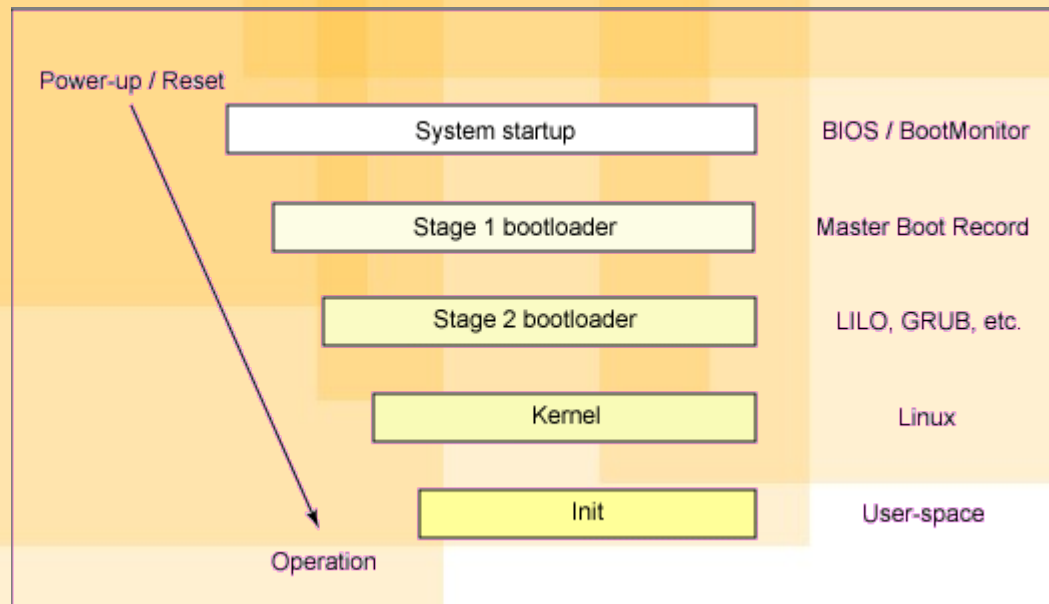
coreboot - formerly known as LinuxBIOS project aimed at replacing the proprietary BIOS.

Linux boot in a nutshell

- **The process of booting a Linux system consists of a number of stages. But whether you're booting a standard x86 desktop or a deeply embedded PowerPC target, much of the flow is surprisingly similar.**
- **When a system is first booted, or is reset, the processor executes code at a well-known location. In a PC, this location is in the basic input/output system (BIOS).The CPU in an embedded system invokes the reset vector to start a program at a known address in flash/ROM.**

Linux boot in a nutshell

- **Boot on PC :**



Role of a Bootloader

- **At a minimum an embedded loader provides the following features:**

Initializing the hardware, especially the memory controller.

Providing boot parameters for the application (Linux kernel).

Starting the application (Linux kernel).

- **So what else...**

Role of a Bootloader - cont.

- **"convenience" features that simplify development:**

Interact with user using CLI.

Reading and writing arbitrary memory locations (md ,mw)

Uploading new binary images to the board's RAM via a serial line or Ethernet.

Copying binary images from RAM to FLASH memory.

Role of a Bootloader - cont.

- **A set of helpful tools: CRC, compress / decompress, Flash utils, scripts handling, file system, and more**
- **Diagnostic : memory test, peripheral test,etc...**
- **Configurable – to tailor size and features set according to needs.**

Developing a Bootloader

- **Part of HW bring up in embedded systems**
- **Always a critical phase in product development – integration of the software with the target board.**
- **Today almost all chip makers provide basic boot loaders as open source code.**
- **Development**
 - **Download U-boot**
 - **Configure**
 - **Tweak to your hardware specifics**

Das U-Boot

-
- **Introduce the U-Boot program and list some of its main features.**
-

Das U-Boot

- **origins in the 8xxROM project, a boot loader for 8xx PowerPC systems by Magnus Damm.**
- **In 2000 project leader Wolfgang Denk brought the project to Sourceforge and renamed it to PPCBoot.**
- **PPCBoot became very popular driving developers to port it to new architectures.**
- **By September, 2002, PPCBoot supported four different ARM processors.**
- **In November, 2002, the PPCBoot team retired the project, which led directly to the surfacing of "Das U-Boot."**

Das U-Boot

- **GPL'ed cross-platform boot loader.**
- **Backed by an active developer and user community.**
- **Out-of-the-box support for hundreds of embedded boards and a wide variety of CPUs .**
- **Easily configure to strike the right balance between a rich feature set and a small binary footprint.**

U-Boot Source Code

- **Official U-boot Source Code is at:**

- <http://git.denx.de/>

The old project page at SourceForge is no longer used and dead.

- **U-Boot uses a 3 level version number containing a version, a sub-version, and a patchlevel "U-Boot-2.34.5" means:**

- **version "2",**
 - **sub-version "34", and**
 - **patchlevel "5".**

The patchlevel is used to indicate certain stages of development between released versions, i. e. officially released versions of U-Boot will always have a patchlevel of "0".

Supported Platforms

<u>Architecture</u>	<u>Processor</u>	<u>Number of Boa</u>	<u>Architecture</u>	<u>Processor</u>	<u>Number of Boa</u>
PPC	5xx	2	ARM (cont)	S3C44B0	1
	5xxx	6		AT91RM9200	1
	8xx	71		XScale	8
	824x	15	x86	SC520	2
	826x	26	m68k	Coldfire	2
	85xx	3	MIPS32	4Kc	2
	7xx/74xx	11		Au1x00	3
	4xx	38	MIPS64	5Kc	1
ARM	StrongARM	5	NIOS32		3
	ARM720T	3	Microblaze		1
	ARM92xT	11	Blackfin	BF533/BF535	3

Configuring & building the U-Boot program

- **Building U-Boot for a supported platform is very straight forward.**
- **To setup a default configuration for a particular board :**
 - `$ make distclean`
 - `$ make <platform>_config`
 - `$ make`
- **platform is one of the many platforms supported by U-Boot.**

Configuring & building the U-Boot program - Cont.

- **This should generate the following files:**

u-boot – ELF file.

u-boot.bin – BIN file.

u-boot.map – MAP file.

u-boot.srec – Motorola srec file.

Configuring & building the U-Boot program - Cont.

- **Fine tune the default configuration for your particular environment and board by editing the configuration file.**
- **"include/configs/<board>.h" file contains several C-preprocessor #define macros that you can modify for your needs.**

Configuring & building the U-Boot program - Cont.

- **Configuration *options* are selected using macros in the form of CONFIG_XXXX. (user configurable to enable specific operational features).**
- **Configuration *settings* are selected using macros in the form of CFG_XXXX.(hardware specific and require knowledge of the underlying hardware platform).**
- **#define CONFIG_BAUDRATE 9600**
- **#define CONFIG_COMMANDS (CONFIG_CMD_DFL | CFG_CMD_DHCP)**
- **#define CFG_LOAD_ADDR 0x100000 /* default load address */**

CLI

- **Support many sets of command.**
- **Since U-Boot is highly configurable, not all the commands may be available – configure according to your needs.**
- **Two different command interpreters are available:**
 - Simple CLI ·
 - Bourne compatible shell (HUSH shell from Busybox) ·
- **Configuration parameters and commands / command sequences (scripts !) can be stored in "environment variables" which can be saved to non-volatile storage (flash, EEPROM, NVRAM, etc.)**

U-Boot Basic Command Set (1/4)

Information Commands

- bdfinfo - print Board Info structure "
- coninfo - print console devices and informations
- flinfo - print FLASH memory information
- iminfo - print header information for application image
- imls - list all images found in flash
- help - print online help

Memory Commands

- base - print or set address offset
- crc32 - checksum calculation
- cmp - memory compare
- cp - memory copy
- md - memory display
- mm - memory modify (auto-incrementing)
- mtest - simple RAM test
- mw - memory write (fill)
- nm - memory modify (constant address)
- loop - infinite loop on address range

Flash Memory Commands

- cp - memory copy (program flash)
- flinfo - print FLASH memory information
- erase - erase FLASH memory
- protect - enable or disable FLASH write protection
-

Execution Control Commands

- autoscr - run script from memory
- bootm - boot application image from memory
- bootelf - Boot from an ELF image in memory
- bootvx - Boot vxWorks from an ELF image
- go - start application at address 'addr'

U-Boot Basic Command Set (2/4)

Network Commands

- bootp - boot image via network using BOOTP/TFTP protocol
- cdp - Perform Cisco Discovery Protocol network configuration
- dhcp - invoke DHCP client to obtain IP/boot params
- loadb - load binary file over serial line (kermit mode)
- loads - load S-Record file over serial line
- nfs - boot image via network using NFS protocol
- ping - send ICMP ECHO_REQUEST to network host
- rarpboot- boot image via network using RARP/TFTP protocol
- tftpboot- boot image via network using TFTP protocol

Environment Variables Commands

- printenv- print environment variables
- saveenv - save environment variables to persistent storage
- askenv - get environment variables from stdin
- setenv - set environment variables
- run - run commands in an environment variable
- bootd - boot default, i.e., run 'bootcmd'

U-Boot Basic Command Set (3/4)

Filesystem Support (FAT, cramfs, JFFS2, Reiser)

- `chpart` - change active partition
- `fsinfo` - print information about filesystems
- `fsload` - load binary file from a filesystem image
- `ls` - list files in a directory (default /)
- `fatinfo` - print information about filesystem
- `fatls` - list files in a directory (default /)
- `fatload` - load binary file from a dos filesystem
- `nand` - NAND flash sub-system
- `reiserls` - list files in a directory (default /)
- `reiserload` - load binary file from a Reiser filesystem

Special Commands

- `i2c` - I2C sub-system
- `doc` - Disk-On-Chip sub-system
- `dtm` - Digital Thermometer and Thermostat
- `eeeprom` - EEPROM sub-system
- `fpga` - FPGA sub-system
- `ide` - IDE sub-system
- `kgdb` - enter gdb remote debug mode
- `diskboot` - boot from IDE device
- `icache` - enable or disable instruction cache
- `dcache` - enable or disable data cache
- `diag` - perform board diagnostics (*POST* code)
- `log` - manipulate logbuffer
- `pci` - list and access PCI Configuration Space
- `regdump` - register dump commands
- `usb` - USB sub-system
- `sspi` - SPI utility commands

U-Boot Basic Command Set (4/4)

Miscellaneous Commands

- bmp - manipulate BMP image data ``
- date - get/set/reset date & time ``
- echo - echo args to console ``
- exit - exit script ``
- kbd - read keyboard status ``
- in - read data from an IO port ``
- out - write datum to IO port ``
- reset - Perform RESET of the CPU ``
- sleep - delay execution for some time ``
- test - minimal test like /bin/sh ``
- version - print monitor version ``
- wd - check and set watchdog ``
- ? - alias for 'help' ``

U-boot upgrade flow

- **Example using CLI commands to update the u-boot image on flash.**
- **Download command to bring an image to ram (using tftp).**

```
u-boot # tftp 8000 u-boot.bin
From server 10.0.0.1; our IP address is 10.0.0.11
Filename 'u-boot.bin'.
Load address: 0x8000
Loading: #####
done
Bytes transferred = 95032 (17338 hex)
```

U-boot upgrade flow

- **Flash memory commands to update the image.**

```
u-boot # protect off 1:0-4  
Un-Protect Flash Sectors 0-4 in Bank # 1
```

```
u-boot # erase off 1:0-4  
Erase Flash Sectors 0-4 in Bank # 1
```

```
u-boot # cp.b ${fileaddr} 1000000 ${filesize}  
Copy to Flash... ..... done
```

```
u-boot # protect on 1:0-4  
Protect Flash Sectors 0-5 in Bank # 1
```

U-boot -Advanced features...

- _____
- _____
- _____
- _____
- _____

U-boot features - jffs2

- How to define a jffs2 partition ?

where is the partition ?

```
#define CFG_JFFS2_FIRST_BANK  
#define CFG_JFFS2_NUM_BANKS  
#define CFG_JFFS2_FIRST_SECTOR  
#define CFG_JFFS2_LAST_SECTOR
```

U-boot features - jffs2

Enable jffs2 commands

- Add `CFG_CMD_JFFS2` to `CFG_BASE_CMD`
enables file commands: `ls`, `fsinfo` and `fsload` commands.
- `#define CONFIG_JFFS2_CMDLINE` (optional)
enable partition commands : `chpart`, `mtdparts`

U-boot features - jffs2

Advantages of using jffs2 from u-boot:

- **Pass partition info to kernel through boot command.**
- **Maintain a common set of configuration files for application and u-boot, for example network configuration.**
- **A way to communicate with u-boot (mark things for next load).**
- **Better flash space utilization – you can use images stored on the file system, instead of saving a predefined hard coded place on the flash.**

Summary

- **Motivation – Why do we need this lecture**
- **Booting – The boot sequence, commonly used open source boot loaders**
- **Linux boot process – The boot process in a nutshell**
- **Role of a Bootloader – What would we like from a boot loader**
- **Development - Developing with open source boot-loader**
- **Das U-Boot – to demonstrate the concept**

Questions?

- _____
- _____
- _____
- _____
- _____