07-1 Bootloaders

Bootloader Challenges

```
#include <stdio.h>
int main(int argc, char **argv) {
  printf("Hello, World!\n");
  return 0;
}
```

Challenges

- To do
  - DRAM Controller needs initialization
  - May need to copy from Flash to RAM
  - There is no stack
  - Libraries may be needed
  - A context needs to be established
- To where does the processor branch on power up?

u-boot/arch/arm/cpu/armv7/u-boot.lds

```
OUTPUT_FORMAT("elf32-littlearm", "elf32-
  littlearm", "elf32-littlearm")
OUTPUT_ARCH(arm)
ENTRY(_start)
SECTIONS
{
  . = 0x00000000;

  . = ALIGN(4);
  .text      :
  {
      arch/arm/cpu/armv7/start.o    (.text)
      *(.text)
  }
```

u-boot/arch/arm/cpu/armv7/start.S

```
.globl _start
_start:
        b       reset
        ldr     pc, _undefined_instruction
        ldr     pc, _software_interrupt
        ldr     pc, _prefetch_abort
        ldr     pc, _data_abort
        ldr     pc, _not_used
        ldr     pc, _irq
        ldr     pc, _fiq
_undefined_instruction: .word undefined_instruction
_software_interrupt:    .word software_interrupt
…
_pad:                   .word 0x12345678 /* now 16*4=64 */
.global _end_vect
_end_vect:
```

Figure 25-6. 32KB ROM Memory Map

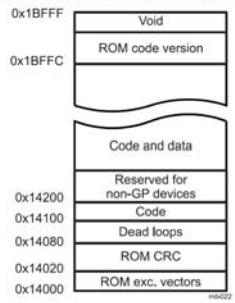


Figure 25-7. 64KB RAM Memory Map of GP Devices

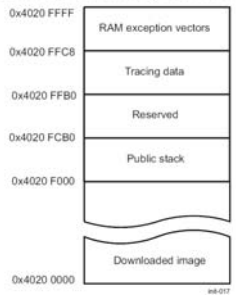


Table 25-7. ROM Exception Vectors

Address	Exception	Content
14000h	Reset	Branch to the public ROM code startup
14004h	Undefined	PC = 4020FFC8h
14008h	Software interrupt (SWI)	PC = 4020FFCCh
1400Ch	Prefetch abort	PC = 4020FFD0h
14010h	Data abort	PC = 4020FFD4h
14014h	Unused	PC = 4020FFD8h
14018h	IRQ	PC = 4020FFDCh
1401Ch	FIQ	PC = 4020FFE0h

System.map

```
80e80000 T _start
80e80020 t _undefined_instruction
80e80024 t _software_interrupt
80e80028 t _prefetch_abort
80e8002c t _data_abort
80e80030 t _not_used
80e80034 t _irq
80e80038 t _fiq
80e8003c t _pad
80e80040 T _end_vect
80e80040 t _TEXT_BASE
```

The Stack (start.S)

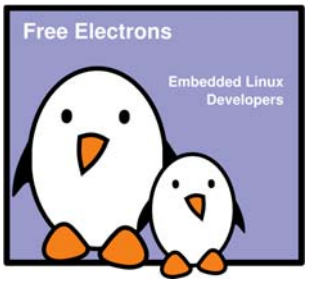
```
/* Set stackpointer in internal RAM to call board_init_f */
call_board_init_f:
  ldr   sp, =(CONFIG_SYS_INIT_SP_ADDR)
  bic   sp, sp, #7 /* 8-byte alignment for ABI compliance */
  ldr   r0,=0x00000000
  bl    board_init_f
```

- `board_init_f` is defined in `u-boot-arch/arm/lib/board.c`

The U-boot bootloader

The U-boot bootloader

Michael Opdenacker
Thomas Petazzoni
Free Electrons



U-Boot

U-Boot is a typical free software project

- Freely available at http://www.denx.de/wiki/U-Boot
- Documentation available at http://www.denx.de/wiki/U-Boot/Documentation
- The latest development source code is available in a Git repository: http://git.denx.de/cgi-bin/gitweb.cgi?p=u-boot.git;a=summary
- Development and discussions happen around an open mailing-list http://lists.denx.de/pipermail/u-boot/
- Since the end of 2008, it follows a fixed-interval release schedule. Every two months, a new version is released. Versions are named YYYY.MM.

Compiling/Installing U-Boot

- See http://elinux.org/EBC_Exercise_12_Cross-Compiling_and_Finding_the_Right_Kernel
- On Host

```
host$ scp u-boot.bin root@beagle:.
```

- On the Beagle

```
beagle$ ls -F /media/
card/  hdd/   mmcblk0p1/  mmcblk0p3/  ram/       union/
cf/    mmc1/  mmcblk0p2/  net/        realroot/
beagle$ ls -ls /media/mmcblk0p1
total 3430
  22 -rwxr-xr-x 1 root root   22232 May 12  2011 MLO
 279 -rwxr-xr-x 1 root root  284788 May 12  2011 U-BOOT.BIN
3129 -rwxr-xr-x 1 root root 3203088 May 12  2011 UIMAGE
   1 drwxr-xr-x 2 root root     512 Jun 20  2011 uEnv
   1 -rwxr-xr-x 1 root root     191 Sep 23 18:27 uEnv.txt
beagle$ cp u-boot.bin /media/mmcblk0p1
beagle$ shutdown -r now
```

U-boot prompt

- Power-up the board. On the serial console, you will see something like:

```
U-Boot 2011.03-rc1-00000-g9a3cc57-dirty (Apr 01 2011 - 17:41:42)
OMAP3630/3730-GP ES2.1, CPU-OPP2, L3-165MHz, Max CPU Clock 1 Ghz
OMAP3 Beagle board + LPDDR/NAND
I2C:   ready
DRAM:  512 MiB
NAND:  0 MiB
MMC:   OMAP SD/MMC: 0
*** Warning - readenv() failed, using default environment
In:    serial
Out:   serial
Err:   serial
Beagle xM Rev C
Die ID #397600029ff80000015f26ad0f01a010
Hit any key to stop autoboot:  0
OMAP3 beagleboard.org #
```

The U-Boot shell offers a set of commands. We will study the most important ones, see the documentation for a complete reference or the help command.

Information commands

Board information / Flash information

```
OMAP3 beagleboard.org # bdinfo
arch_number = 0x0000060A
boot_params = 0x80000100
DRAM bank   = 0x00000000
-> start    = 0x80000000
-> size     = 0x10000000
DRAM bank   = 0x00000001
-> start    = 0x90000000
-> size     = 0x10000000
baudrate    = 115200 bps
TLB addr    = 0x9FFF0000
relocaddr   = 0x9FF7E000
reloc off   = 0x1FF76000
irq_sp      = 0x9FF1DF68
sp start    = 0x9FF1DF60
FB base     = 0x00000000
```

```
u-boot # flinfo
```

Bank # 1: AMD Am29LV160DB 16KB,2x8KB,32KB,31x64KB
Size: 2048 KB in 35 Sectors
Sector Start Addresses:
S00 @ 0x01000000 | S01 @ 0x01004000 |
S02 @ 0x01006000 | S03 @ 0x01008000 |
S04 @ 0x01010000 | S05 @ 0x01020000 |
S06 @ 0x01030000 S07 @ 0x01040000

Can't find

Environment variables (1)

- U-Boot can be configured through environment variables, which affect the behavior of the different commands
- See the documentation for the complete list of environment variables
- The **printenv** command also to display all variables or one :

```
baudrate=115200
beaglerev=xMC
bootcmd=if mmc rescan ${mmcdev}; then if userbutton; then setenv bootenv
user.txt;fi;echo SD/MMC found on device ${mmcdev};if run loadbootenv; then echo
Loaded environment from ${bootenv};run importbootenv;fi;if test -n $uenvcmd; then
echo Running uenvcmd ...;run uenvcmd;fi;if run loaduimage; then run
mmcboot;fi;fi;run nandboot;
bootdelay=3
bootenv=uEnv.txt
buddy=trainer
camera=lbcm3m1
console=ttyS2,115200n8
defaultdisplay=dvi
dieid#=397600029ff80000015f26ad0f01a010
dvimode=640x480MR-16@60
```

Environment variables

```
nandargs=setenv bootargs console=${console} ${optargs}
   mpurate=${mpurate} buddy=${buddy} camera=${camera} vram=${vram}
   omapfb.mode=dvi:${dvimode} omapdss.def_disp=${defaultdisplay}
   root=${nandroot} rootfstype=${nandrootfstype}
nandboot=echo Booting from nand ...; run nandargs; nand read
   ${loadaddr} 280000 400000; bootm ${loadaddr}
nandroot=/dev/mtdblock4 rw
nandrootfstype=jffs2
ramargs=setenv bootargs console=${console} ${optargs}
   mpurate=${mpurate} buddy=${buddy} camera=${camera} vram=${vram}
   omapfb.mode=dvi:${dvimode} omapdss.def_disp=${defaultdisplay}
   root=${ramroot} rootfstype=${ramrootfstype}
ramboot=echo Booting from ramdisk ...; run ramargs; bootm ${loadaddr}
ramroot=/dev/ram0 rw ramdisk_size=65536 initrd=0x81000000,64M
ramrootfstype=ext2
rdaddr=0x81000000
usbtty=cdc_acm
vram=12M

Environment size: 2095/131068 bytes
```

Environment variables

- The value of the environment variables can be changed using the **setenv** command :
  u-boot # **setenv serverip 10.0.0.2**
- Environment variable changes can be stored to flash using the **saveenv** command. The location is defined at compile time in the U-Boot configuration file

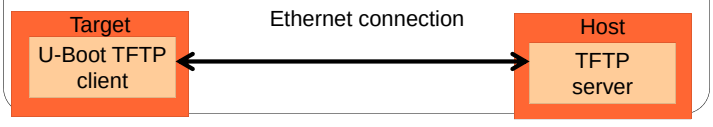
xM doesn't have flash

- You can even create small shell scripts stored in environment variables:
  setenv myscript 'tftp 0x21400000 uImage ; bootm 0x21400000'
- You can then execute the script:
  run myscript

Transferring files to the target

- U-Boot is mostly used to load and boot a kernel image, but it also allows to change the kernel image and the root filesystem stored in flash.
- Files must be exchanged between the target and the development workstation. This is possible:
  - Through the network if the target has an Ethernet connection, and U-Boot contains a driver for the Ethernet chip. If so, the TFTP protocol can be used to exchange files
  - Through the serial line if no Ethernet connection is available.

Not on Beagle



U-boot mkimage

```
host$ file u-boot u-boot.bin
u-boot:     ELF 32-bit LSB executable,
            ARM, version 1 (SYSV),
            statically linked, not
            stripped
u-boot.bin: data
host$ ls -sh u-boot u-boot.bin
1.4M u-boot
320K u-boot.bin
```

U-boot mkimage

- The kernel image that U-Boot loads and boots must be prepared, so that an U-Boot specific header is added in front of the image
- This is done with a tool that comes in U-Boot, mkimage
- Debian / Ubuntu: just install the uboot-mkimage package
- Or, compile it by yourself: simply configure U-Boot for any board of any architecture and compile it. Then install mkimage:
  `host$ cp uboot/tools/mkimage /usr/local/bin/`
- The special target uImage of the kernel Makefile can then be used to generate a kernel image suitable for U-Boot.

u-boot/include/configs/omap3_beagle.h

```
/*
 * High Level Configuration Options
 */
                                        /* This is an ARM V7 CPU core */
#define CONFIG_OMAP                1        /* in a TI OMAP core */
#define CONFIG_OMAP34XX            1        /* which is a 34XX */
#define CONFIG_OMAP3430            1        /* which is in a 3430 */
#define CONFIG_OMAP3_BEAGLE        1        /* working with BEAGLE */

#include <asm/arch/cpu.h>          /* get chip and board defs */
#include <asm/arch/omap3.h>

/*
 * Display CPU and Board information
 */
#define CONFIG_DISPLAY_CPUINFO     1
#define CONFIG_DISPLAY_BOARDINFO   1
```

.../include/configs/omap3_beagle.h

```
* commands to include */
#include <config_cmd_default.h>

#define CONFIG_CMD_CACHE
#define CONFIG_CMD_EXT2    /* EXT2 Support      */
#define CONFIG_CMD_FAT     /* FAT support       */
#define CONFIG_CMD_JFFS2   /* JFFS2 Support     */
…
#undef CONFIG_CMD_FLASH    /* flinfo, erase, protect  */
#undef CONFIG_CMD_FPGA     /* FPGA configuration Support */
#undef CONFIG_CMD_IMI      /* iminfo                  */
#undef CONFIG_CMD_IMLS     /* List all found images   */
```

.../include/configs/omap3_beagle.h

```
#define CONFIG_SYS_NO_FLASH
#define CONFIG_HARD_I2C              1
#define CONFIG_SYS_I2C_SPEED         100000
#define CONFIG_SYS_I2C_SLAVE         1
#define CONFIG_SYS_I2C_BUS           0
#define CONFIG_SYS_I2C_BUS_SELECT    1
#define CONFIG_I2C_MULTI_BUS         1
#define CONFIG_DRIVER_OMAP34XX_I2C 1
#define CONFIG_VIDEO_OMAP3      /* DSS Support */
```

U-Boot Monitor Commands

- U-Boot supports >70 standard command sets
- More than 150 unique commands
- Enable with CONFIG_CMD_* macros.

Command Set	Commands
CONFIG_CMD_FLASH	Flash memory commands
CONFIG_CMD_MEMORY	Memory dump, fill, copy, compare, and so on
CONFIG_CMD_DHCP	DHCP Support
CONFIG_CMD_PING	Ping support
CONFIG_CMD_EXT2	EXT2 File system support

U-Boot Monitor Commands

- To enable a specific command, define the macro
- Macros are defined in your board-specific configuration file
- Instead of typing out each individual macro start from the full set of commands defined in

u-boot/include/config_cmd_all.h.

- List of useful default commands sets

u-boot/include/config_cmd_default.h

```
$ wc config_cmd_*
  92  567 4181 config_cmd_all.h
  43  237 1673 config_cmd_default.h
  18   45  366 config_cmd_defaults.h
 153  849 6220 total
```