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

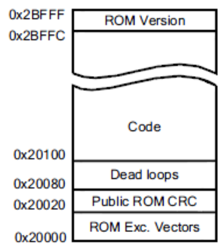
```
OUTPUT_FORMAT("elf32-littlearm", "elf32-
  littlearm", "elf32-littlearm")
OUTPUT_ARCH(arm)
ENTRY(_start)
SECTIONS
{
  . = 0x00000000;

  . = 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:
```

bone

Figure 26-3. ROM Memory Map



Page 4098

Figure 26-4. Public RAM Memory Map

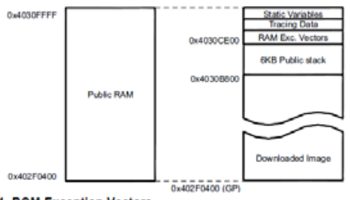
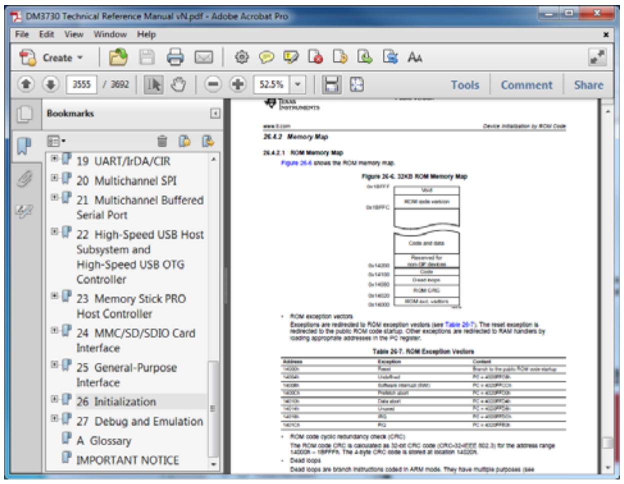


Table 26-1. ROM Exception Vectors

Address	Exception	Content
20000h	Reset	Branch to the Public ROM Code startup
20004h	Undefined	PC = 4030CE04h
20008h	SWI	PC = 4030CE08h
2000Ch	Pre-fetch abort	PC = 4030CE0Ch
20010h	Data abort	PC = 4030CE10h
20014h	Unused	PC = 4030CE14h
20018h	IRQ	PC = 4030CE18h
2001Ch	FIQ	PC = 4030CE1Ch


DM3730 Technical Reference Manual vN.pdf - Adobe Acrobat Pro
File Edit View Window Help
Create
3555 / 3692
52.5%
Tools Comment Share
Bookmarks
19 UART/IrDA/CIR
20 Multichannel SPI
21 Multichannel Buffered Serial Port
22 High-Speed USB Host Subsystem and High-Speed USB OTG Controller
23 Memory Stick PRO Host Controller
24 MMC/SD/SDIO Card Interface
25 General-Purpose Interface
26 Initialization
27 Debug and Emulation
A Glossary
IMPORTANT NOTICE
0x1BFFF
Void
ROM code version
0x1BFFC
Code and data
Reserved for non-GP devices
0x14200
Code
0x14100
Dead loops
0x14080
ROM CRC
0x14020
ROM exc. vectors
0x14000

TEXAS INSTRUMENTS

www.ti.com — Device Initialization by ROM Code

26.4.2 Memory Map

26.4.2.1 ROM Memory Map

Figure 26-6 shows the ROM memory map.

Figure 26-6. 32KB ROM Memory Map

- ROM exception vectors
  Exceptions are redirected to ROM exception vectors (see Table 26-7). The reset exception is redirected to the public ROM code startup. Other exceptions are redirected to RAM handlers by loading appropriate addresses in the PC register.

Table 26-7. ROM Exception Vectors

Address	Exception	Content
14000h	Reset	Branch to the public ROM code startup
14004h	Undefined	PC = 4020FFC4h
14008h	Software interrupt (SWI)	PC = 4020FFC8h
1400Ch	Prefetch abort	PC = 4020FFCCh
14010h	Data abort	PC = 4020FFD0h
14014h	Unused	PC = 4020FFD4h
14018h	IRQ	PC = 4020FFD8h
1401Ch	FIQ	PC = 4020FFDCh

- ROM code cyclic redundancy check (CRC)
  The ROM code CRC is calculated as 32-bit CRC code (CRC-32/IEEE 802.3) for the address range 14000h – 1BFFFh. The 4-byte CRC code is stored at location 14020h.
- Dead loops
  Dead loops are branch instructions coded in ARM mode. They have multiple purposes (see

u-boot/System.map - bone

```
80100000 T _start
80100020 t _undefined_instruction
80100024 t _software_interrupt
80100028 t _prefetch_abort
8010002c t _data_abort
80100030 t _not_used
80100034 t _irq
80100038 t _fiq
8010003c t _pad
80100040 T _TEXT_BASE
```

The Stack (u-boot/arch/arm/cpu/armv7/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
```

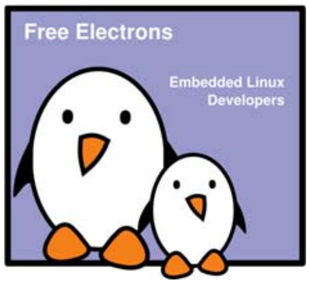
- **board_init_f** is defined in **u-boot-arch/arm/lib/board.c**
- **(From include/configs/omap3_beagle.h)**

```
#define CONFIG_SYS_INIT_RAM_ADDR      0x4020f800
#define CONFIG_SYS_INIT_RAM_SIZE      0x800
#define CONFIG_SYS_INIT_SP_ADDR       (CONFIG_SYS_INIT_RAM_ADDR + \
                                       CONFIG_SYS_INIT_RAM_SIZE - \
                                       GENERATED_GBL_DATA_SIZE)
```

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_22_Cross-Compiling_and_Finding_the_Right_Kernel#Installing_a_New_U-boot
- On Host

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

Compiling/Installing U-Boot

- On the Beagle

```
beagle$ cd /media/
beagle$ mkdir mmcblk0p1
beagle$ mount /dev/mmcblk0p1 mmcblk0p1/
beagle$ ls -ls mmcblk0p1/
total 354
  2 drwxr-xr-x 4 root root   2048 May 16 15:29 Docs
  2 drwxr-xr-x 5 root root   2048 May 16 15:29 Drivers
  6 -rwxr-xr-x 1 root root   5829 Aug 14 10:10 LICENSE.txt
 84 -rwxr-xr-x 1 root root  85058 Aug 14 08:19 MLO
 14 -rwxr-xr-x 1 root root  13976 Aug 14 10:10 README.htm
  2 -rwxr-xr-x 1 root root     27 Aug 14 10:23 Uenv.txt
  2 -rwxr-xr-x 1 root root    178 Aug 14 10:10 autorun.inf
  2 -rwxr-xr-x 1 root root    171 Aug 14 10:10 info.txt
238 -rwxr-xr-x 1 root root 241948 Aug 14 08:19 u-boot.img
  2 -rwxr-xr-x 1 root root     33 Aug 14 10:23 uEnv.txt.orig
beagle$ cp u-boot.bin /media/mmcblk0p1
beagle$ shutdown -r now
```

U-boot prompt

- Power-up the board.

```
U-Boot SPL 2011.09-00053-gb423c52 (Aug 10 2012 - 11:26:55)
Texas Instruments Revision detection unimplemented
No daughter card present
No AC power, disabling frequency switch
OMAP SD/MMC: 0
reading u-boot.img
reading u-boot.img

U-Boot 2011.09-00053-gb423c52 (Aug 10 2012 - 11:26:55)
I2C:   ready
DRAM:  256 MiB
WARNING: Caches not enabled
No daughter card present
NAND:  HW ECC Hamming Code selected
No NAND device found!!!
0 MiB
MMC:   OMAP SD/MMC: 0, OMAP SD/MMC: 1
*** Warning - readenv() failed, using default environment

Net:   cpsw
Hit any key to stop autoboot:  0
U-Boot#
```

U-boot prompt

- 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 :

```
autoload=yes
baudrate=115200
bootargs_defaults=setenv bootargs console=${console} ${optargs}
bootcmd=if mmc rescan; then echo SD/MMC found on device ${mmc_dev};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 mmc_load_uimage_ext4; then run mmc_args;bootm
${kloadaddr};fi;fi;run nand_boot;
bootdelay=1
bootenv=uEnv.txt
bootfile=uImage
console=ttyO0,115200n8
ethact=cpsw
ethaddr=d4:94:a1:39:ed:0c
```

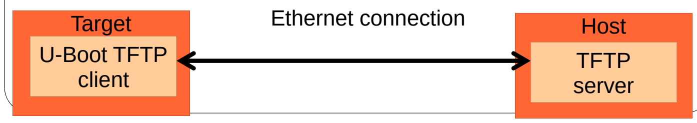
Environment variables

```
nor_src_addr=0x08080000
rootpath=/export/rootfs
script_addr=0x81900000
spi_args=run bootargs_defaults;setenv bootargs ${bootargs} root=${spi_root}
   rootfstype=${spi_root_fs_type}
 ip=${ip_method}
spi_boot=echo Booting from spi ...; run spi_args; sf probe ${spi_bus_no}:0; sf
   read ${kloadaddr}
 ${spi_src_addr} ${spi_img_siz}; bootm ${kloadaddr}
spi_bus_no=0
spi_img_siz=0x380000
spi_root=/dev/mtdblock4 rw
spi_root_fs_type=jffs2
spi_src_addr=0x62000
static_ip=${ipaddr}:${serverip}:${gatewayip}:${netmask}:${hostname}::off
stderr=serial
stdin=serial
stdout=serial
Environment size: 2755/8188 bytes
```

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$ cd u-boot
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
332K 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
```