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?

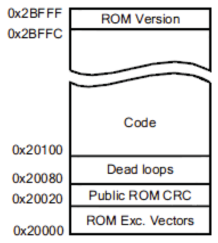
u-boot/arch/arm/cpu/u-boot.lds

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

  . = ALIGN(4);
  .text :
  {
        *(.__image_copy_start)
        *(.vectors)
        CPUDIR/start.o (.text*)
        *(.text*)
  }
```

u-boot/arch/arm/lib/vectors.S

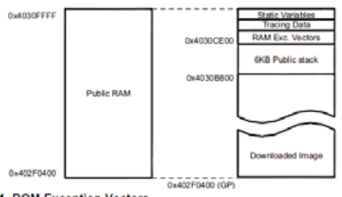
```
.globl _start
_start:
        b       reset
        ldr     pc, _undefined_instruction
        ldr     pc, _software_interrupt
        ldr     pc, _prefetch_abort                undefined_instruction:
        ldr     pc, _data_abort                            get_bad_stack
        ldr     pc, _not_used                              bad_save_user_regs
        ldr     pc, _irq                                   bl
        ldr     pc, _fiq                           do_undefined_instruction

_reset:                  .word reset
_undefined_instruction:  .word undefined_instruction
_software_interrupt:     .word software_interrupt
_prefetch_abort:         .word prefetch_abort
_data_abort:             .word data_abort
_not_used:               .word not_used
_irq:                    .word irq
_fiq:                    .word fiq
```

Figure 26-3. ROM Memory Map



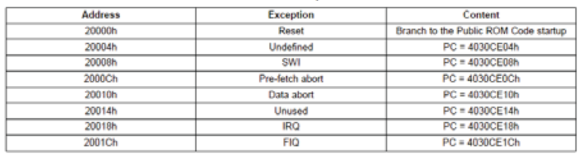
bone

Figure 26-4. Public RAM Memory Map

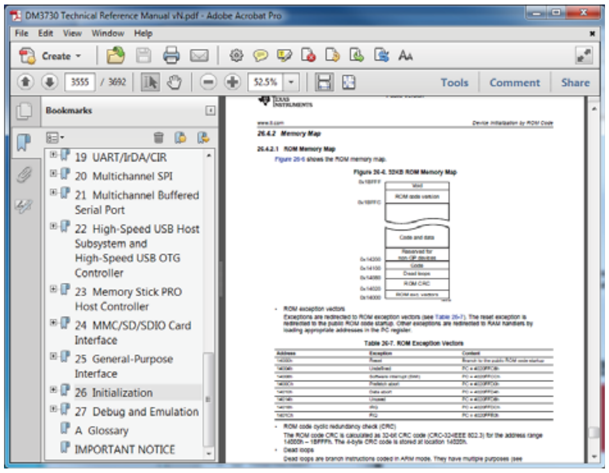


Page 4098

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

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 = 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

- 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

```
80800000 T __image_copy_start
80800000 T _start
80800020 t _reset
80800024 T _undefined_instruction
80800028 T _software_interrupt
8080002c T _prefetch_abort
80800030 T _data_abort
80800034 T _not_used
80800038 T _irq
8080003c T _fiq
80800040 T IRQ_STACK_START_IN
80800060 t undefined_instruction
808000c0 t software_interrupt
```

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
```

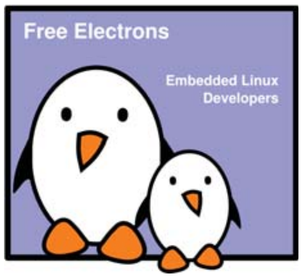
- `board_init_f` is defined in `u-boot-arch/arm/lib/board.c`
- `(From  include/configs/omap3_beagle.h)`

```
#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_Installing_U-Boot_Source

```
 host$ scp MLO u-boot.img root@bone:.
```

- Install on the device you booted from.

```
bone$ dd if=MLO of=/dev/mmcblk0p0 count=1
                          seek=1 conv=notrunc bs=128k
bone$ dd if=u-boot.img of=/dev/mmcblk0p0 count=2
                          seek=1 conv=notrunc bs=384k
```

- or install on the other device. Here if you are running from the SD card this will install on the eMMC.

```
bone$ dd if=MLO of=/dev/mmcblk1p0 count=1
                          seek=1 conv=notrunc bs=128k
bone$ dd if=u-boot.img of=/dev/mmcblk1p0 count=2
                          seek=1 conv=notrunc bs=384k
```

U-boot prompt

- Power-up the board.

```
U-Boot SPL 2014.07-00016-g329fca9 (Jul 28 2014 - 12:35:02)


U-Boot 2014.07-00016-g329fca9 (Jul 28 2014 - 12:35:02), Build: jenkins-
github_Bootloader-Builder-375

I2C:   ready
DRAM:  512 MiB
NAND:  0 MiB
MMC:   OMAP SD/MMC: 0, OMAP SD/MMC: 1
*** Warning - readenv() failed, using default environment

Net:   <ethaddr> not set. Validating first E-fuse MAC
Phy not found
cpsw, usb_ether
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

```
U-Boot# bdinfo
arch_number = 0x00000E05
boot_params = 0x80000100
DRAM bank   = 0x00000000
-> start    = 0x80000000
-> size     = 0x20000000
eth0name    = cpsw
ethaddr     = d0:39:72:1a:5e:ca
eth1name    = usb_ether
eth1addr    = d0:39:72:1a:5e:cc
current eth = cpsw
ip_addr     = <NULL>
baudrate    = 115200 bps
TLB addr    = 0x9FFF0000
relocaddr   = 0x9F741000
reloc off   = 0x1EF41000
irq_sp      = 0x9E720EE0
sp start    = 0x9E720ED0
```

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 :

```
U-Boot# printenv
arch=arm
autoconf=off
baudrate=115200
board=am335x
board_name=A335BNLT
board_rev=000C
boot_fdt=try
bootcmd=gpio set 53; i2c mw 0x24 1 0x3e; run findfdt; setenv mmcdev 0; setenv
bootpart 0:1; run mmcboot;gpio clear 56; gpio clear 55; gpio clear 54; setenv
mmcdev 1; setenv bootpart 1:1; run mmcboot;run nandboot;
bootcount=2
bootdelay=1
bootenv=uEnv.txt
bootfile=zImage
…
```

Environment variables

```
bootm_size=0x10000000
bootpart=0:2
bootscript=echo Running bootscript from mmc ...; source
  ${loadaddr}
console=ttyO0,115200n8
cpu=armv7
device=eth0
dfu_alt_info_emmc=rawemmc mmc 0 3751936
…
usbnet_devaddr=d0:39:72:1a:5e:cc
vendor=ti
ver=U-Boot 2014.07-00016-g329fca9 (Jul 28 2014 -
  12:35:02)

Environment size: 8540/131068 bytes
```

U-boot mkimage

```
host$ cd u-boot
host$ file u-boot u-boot.bin
u-boot:     ELF 32-bit LSB  shared
            object, ARM, EABI5 version 1
            (SYSV), dynamically linked
            (uses shared libs), not
            stripped
u-boot.bin: data
host$ ls -sh u-boot u-boot.bin
2.3M u-boot
436K 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/am335x_evm.h

```
#include <configs/ti_am335x_common.h>
#define CONFIG_SUPPORT_RAW_INITRD

#ifndef CONFIG_SPL_BUILD
# define CONFIG_FIT
# define CONFIG_TIMESTAMP
# define CONFIG_LZO
# ifdef CONFIG_ENABLE_VBOOT
#  define CONFIG_OF_CONTROL
#  define CONFIG_OF_SEPARATE
#  define CONFIG_DEFAULT_DEVICE_TREE am335x-boneblack
#  define CONFIG_FIT_SIGNATURE
#  define CONFIG_RSA
# endif
#endif

#define CONFIG_SYS_BOOTM_LEN            (16 << 20)
```

u-boot/include/configs/ti_am335x_common.h

```
/*
 * When building U-Boot such that there is no previous loader
 * we need to call board_early_init_f.  This is taken care of in
 * s_init when we have SPL used.
 */
#if !defined(CONFIG_SKIP_LOWLEVEL_INIT) && !defined(CONFIG_SPL)
#define CONFIG_BOARD_EARLY_INIT_F
#endif

#ifdef CONFIG_NAND
#define CONFIG_SPL_NAND_AM33XX_BCH      /* ELM support */
#endif

/* Now bring in the rest of the common code. */
#include <configs/ti_armv7_common.h>

#endif  /* __CONFIG_TI_AM335X_COMMON_H__ */
```

u-boot/include/configs/ti_armv7_common.h

```
/* I2C IP block */
#define CONFIG_I2C
#define CONFIG_CMD_I2C
#define CONFIG_SYS_I2C
#define CONFIG_SYS_OMAP24_I2C_SPEED     100000
#define CONFIG_SYS_OMAP24_I2C_SLAVE     1
#define CONFIG_SYS_I2C_OMAP24XX
…
/* McSPI IP block */
#define CONFIG_SPI
#define CONFIG_OMAP3_SPI
#define CONFIG_CMD_SPI

/* GPIO block */
#define CONFIG_OMAP_GPIO
#define CONFIG_CMD_GPIO
```

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_*
 100   620 4558 config_cmd_all.h
  43   236 1667 config_cmd_default.h
  18    45  366 config_cmd_defaults.h
 161   901 6591 total
```