
부트로더

- Chapter 06 -

Contents

- I. 부트로더 개요
- II. U-BOOT의 동작 이해
- III. Booting Sequence
- IV. Bootloader Compile
- V. Bootloader 설치
- VI. Bootloader 명령

부팅과 BIOS

▶ 부팅(Booting)

- ▶ 시스템에 전원이 공급된 후에 운영체제가 실행되기 전까지 수행되는 일련의 작업 과정
- ▶ 부팅 과정에는 프로세스가 초기화되고 메모리와 외부 디바이스를 검사한 후 초기화

▶ BIOS(Basic Input Output System)

▶ 메인보드에 포함된 펌웨어(Firmware)

- ▶ 펌웨어 : HW에 포함된 SW로서 주로 ROM에 저장되며 처리의 고속화와 회로의 단순화에 활용
- ▶ PC에 전원이 공급되면 BIOS는 메인보드의 **CMOS**에 저장된 설정 값을 읽어서 하드웨어를 초기화하고 그래픽 카드 정보 및 바이오스 정보를 출력
- ▶ 이와 같은 POST과정의 수행하고 운영체제를 메모리에 로드
 - ▶ POST(Power On Self Test)
 - BIOS에서 수행하는 각종 테스트나 초기화 작업
- ▶ 전원의 인가와 동시에 del키나 f2키를 누르면 시간, 날짜를 비롯하여 부팅 디바이스, 하드디스크, 메모리 등의 시스템 정보를 수정하고 저장할 수 있는 기능도 포함

부트로더 개요

▶ 부트로더(Bootloader)

- ▶ 부트스트랩 로더(bootstrap loader)의 줄임말
- ▶ 일반적으로 사용자가 컴퓨터를 사용할 수 있도록 OS를 읽어서 주기억장치에 적재하는 프로그램
- ▶ 컴퓨터를 켜올 때 가장 먼저 실행되는 프로그램으로 주로 하드디스크의 MBR에 저장
 - ▷ MBR(Master boot Record)
 - 컴퓨터를 부팅할 때 하드디스크에서 처음으로 읽히는 영역
 - 기억장치의 첫 **섹터**(섹터 0)인 512byte 시동 섹터

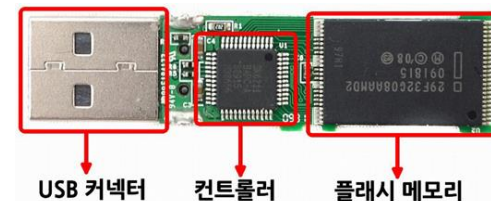
부트로더 개요

▶ 부트로더(Bootloader)

- ▶ 임베디드 시스템은 하드디스크가 없기 때문에 플래시 메모리 같은 곳에 부트로더를 적재하고 전원이 공급되면 타겟 시스템이 부팅되도록 함

▶ 플래시 메모리(Flash Memory)

- USB메모리, SD카드, Micro SD카드, SSD 등
- 전기적으로 데이터를 지우고 다시 기록할 수 있는 비휘발성 기억 장치
- 데이터 저장이 필요한 대부분의 전자 제품에 탑재
- 작고 가벼우며 기계적인 충격과 직사광선, 고온, 습기에 강함(전기 충격에는 약함)



▶ 부팅이 시작되면 타겟 시스템을 초기화하고 커널을 주메모리(DRAM)에 적재

- 커널(kernel)
 - 시스템의 각 장치들(CPU, 메모리, I/O, 디스크 등)을 관리하고 제어하기 위한 소프트웨어
 - 부트로더에 의해 메모리로 로딩되어 시스템이 종료될 때까지 메모리에 상주

부트로더 개요

▶ 부트로더(Bootloader)

- ▶ 임베디드 리눅스가 부팅되기까지 전 과정을 진행하는 부팅 프로그램

- ▶ 부트로더 존재의 이유

 - ▷ PC와 달리 임베디드 환경은 CMOS를 이용할 수 없는 특수한 환경이기 때문에 부트로더를 이용

- ▶ 부트로더의 위치

 - ▷ 일반적으로 시스템 메모리의 물리 주소 '0'번지에 위치

- ▶ 부트로더의 기능 : **Startup**, Monitoring mode, OS Boot

 - ▷ 하드웨어 초기화

 - Clock 설정, memory 초기화, 시리얼포트 초기화, 네트워크 초기화 등

 - ▷ 커널과 Root File System 적재

 - 커널과 root file system을 메모리에 적재하여 Linux를 실행 시킴

부트로더 개요

▶ 부트로더의 종류(1)

- ▶ 부트로더는 **아키텍처**마다 다르고 운영체제마다 다양한 종류가 존재
- ▶ 부트로더를 처음부터 개발하면 시간이 오래 걸리므로 가능하면 오픈 소스의 부트로더 사용을 권장
- ▶ 리눅스 계열의 부트로더
 - ▷ LILO, GRUB을 주로 사용
- ▶ 윈도우 계열
 - ▷ NTLDR
- ▶ 임베디드 리눅스
 - ▷ ARM계열의 BLOB, ARMBOOT
 - ▷ MPC계열의 PPCBOOT
- ▶ 맥OS 계열
 - ▷ BOOTX

부트로더 개요

▶ 부트로더의 종류(2)

▶ LILO(Linux LOader)

- ▷ 작고 가벼워서 x86계열에 널리 사용
- ▷ 호환성을 위해 대부분의 리눅스 배포판을 내장하고 있으며 사용자가 많음
- ▷ 멀티 부팅을 지원

▶ GRUB(GRand Unified Boot loader)

- ▷ x86환경에서 LILO를 대체하기 위하여 확장성과 기능들이 향상된 부트로더
- ▷ 다양한 파일시스템을 인식
- ▷ 최근 리눅스 배포판에는 대부분 LILO보다 GRUB을 기본적으로 사용

부트로더 개요

▶ 부트로더의 종류(3)

▶ ARMBOOT

- ▷ ARM과 StrongARM을 지원하기 위한 오픈 소스 부트로더
- ▷ 지원하는 보드의 종류가 적기때문에 ARMBOOT를 사용하기 위해서는 타켓 시스템에 직접 포팅해야 함

▶ PPCBOOT

- ▷ TCP/IP 네트워크를 사용한 부팅 기능을 제공

부트로더 개요

▶ U-BOOT(Universal BOOTloader)

- ▶ ARMBOOT와 PPCBOOT를 기반으로 개발
- ▶ PowerPC, ARM, MIPS, x86 등과 같은 대부분의 프로세서를 지원하고 주로 임베디드 시스템에서 동작 하도록 작성된 부트로더
- ▶ Open Source
- ▶ 환경 설정이 쉽고 깔끔한 코드로 구성되어 있어서 많은 임베디드 개발자들이 사용
- ▶ Achro-5250은 U-BOOT를 사용

부트로더 개요

▶ U-BOOT 특징

▶ Linux와 유사한 구조











- ▶ U-Boot는 Linux와 유사한 구조
- ▶ 일부 소스는 Linux용을 사용
- ▶ Linux 이미지를 부팅하기 쉬움

▶ 확장이 용이

- ▶ 새로운 Command 추가가 용이
- ▶ Configuration에 기능 추가/확장 제공

▶ U-BOOT 최신 소스 코드 다운로드

`ftp://ftp.denx.de/pub/u-boot`

 u-boot-2016.03-rc1.tar.bz2	10.5 MB	16. 2. 3. 오전 9:00:00
 u-boot-2016.03-rc2.tar.bz2	10.5 MB	16. 2. 16. 오전 9:00:00
 u-boot-2016.03-rc3.tar.bz2	10.6 MB	16. 3. 1. 오전 9:00:00
 u-boot-2016.03.tar.bz2	10.6 MB	16. 3. 14. 오전 9:00:00
 u-boot-2016.05-rc1.tar.bz2	10.8 MB	16. 4. 12. 오전 9:00:00
 u-boot-2016.05-rc2.tar.bz2	10.8 MB	16. 4. 26. 오전 9:00:00
 u-boot-2016.05-rc3.tar.bz2	10.8 MB	16. 4. 26. 오전 9:00:00
 u-boot-2016.05.tar.bz2	10.8 MB	16. 5. 16. 오전 9:00:00
 u-boot-2016.07-rc1.tar.bz2	10.9 MB	16. 6. 7. 오전 9:00:00
 u-boot-2016.07-rc2.tar.bz2	11.0 MB	16. 7. 4. 오전 9:00:00

U-BOOT의 동작 이해

▶ 일반적인 부트로더의 실행 순서

1) start.S

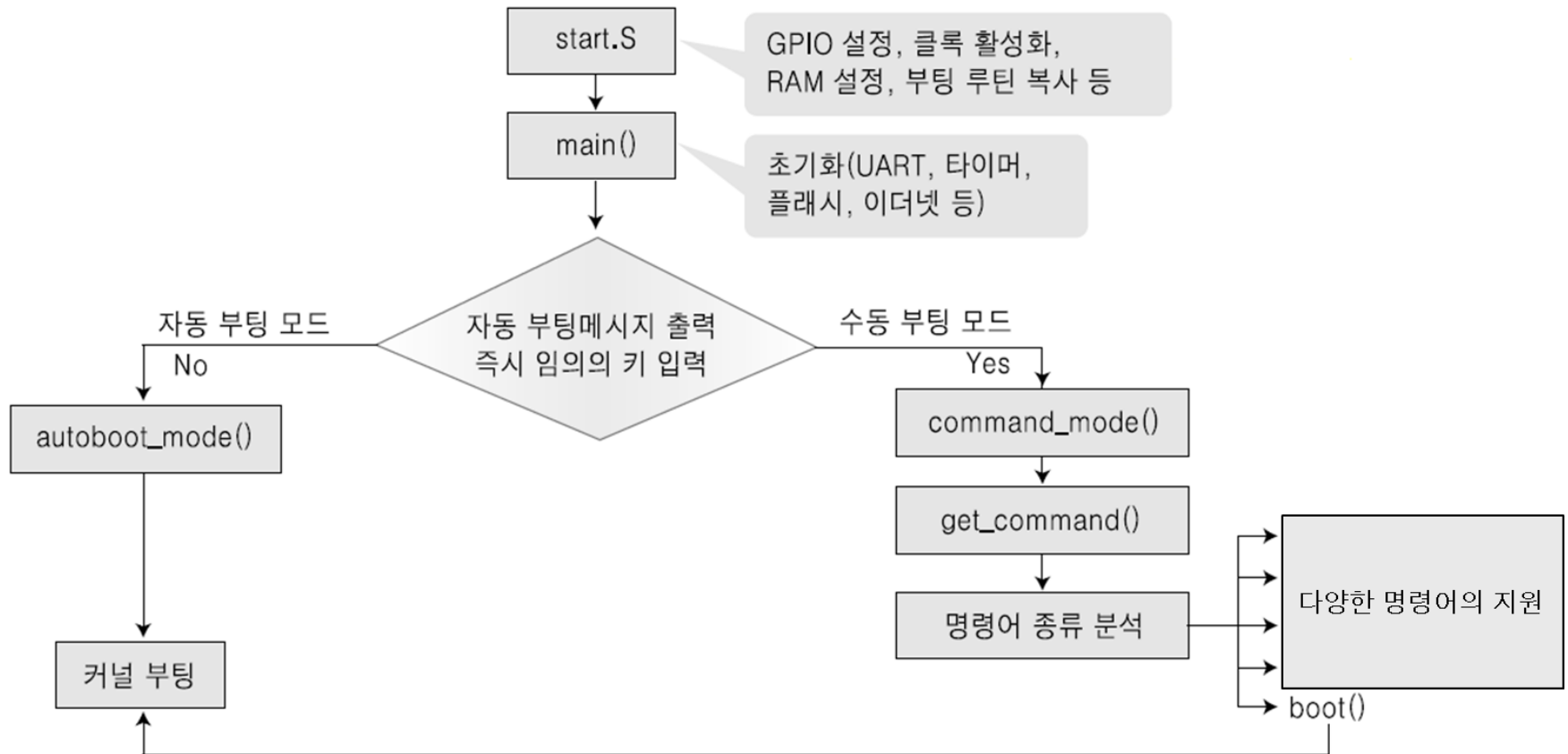
- ▶ 어셈블리어로 작성되었으며 전원 공급 시 최초로 실행되는 코드를 포함하고 있으며 모든 기능을 수행한 후 main()에 제어권을 넘김
 - 어셈블리로 작성되어 아키텍처에 의존적
- ▶ GPIO 설정, 클록 활성화, SRAM과 DRAM 설정, 부팅 루틴의 RAM복사, 스택 포인터 설정 등을 수행

2) main()

- ▶ main.c파일에 정의된 main()을 실행
- ▶ UART, 타이머, 이더넷 등을 초기화 후 부팅 방식을 묻는 메시지를 출력
 - 임의의 키를 입력하여 수동 부팅 모드로 진입하면 부트로더 프롬프트가 출력

U-BOOT의 동작 이해

▶ 일반적인 부트로더의 실행 순서



U-BOOT의 동작 이해

- ▶ Linux를 타겟 시스템에 부팅하기 위하여 커널과 파일시스템이 메모리에 load되어야 함
 - ▶ Linux Kernel
 - ▷ FLASH/ROM에서 동작하기 위해 Kernel을 구성
 - ▶ Root Filesystem(RFS)
 - ▷ Jffs2 image, RAMDISK
- ▶ 대부분의 부트로더는 다음의 두 가지 모드가 지원
 - ▶ Bootloading
 - ▷ 독립적으로 Kernel과 RFS를 내부 device(Flash)로 부터 load(autoboot)
 - ▶ Downloading
 - ▷ Kernel과 RFS를 외부 device(Host PC)로 부터 download
 - ▷ 초기에는 Kernel과 RFS를 install하기 위해 다운로드 모드를 사용하고 이후에는 updates를 위해 사용

Booting Sequence

▶ Booting 상세 과정

A. iROM code (BL0)

- ▶ platform에 무관한 작고 간단한 코드, CPU 내부 메모리(ROM)에 저장
 - CPU의 제조사가 해당 프로세서를 만들 때 미리 프로그램되어 내장됨
- ▶ Clock, Stack, Heap 같은 기본적 system function들을 초기화
- ▶ 정해진 booting 장치로부터 내부 SRAM에 first boot loader(BL1) image를 load
 - BL0에는 타겟보드에 대한 내용이 포함되어 있지 않기 때문에 하드웨어를 초기화 시킬 수 없음
- ▶ DIP 스위치 등을 읽어서 어디에서 부팅할 지를 결정
 - 본 교과에서는 SD 부팅 또는 eMMC 부팅 선택



Booting Sequence

▶ Booting 상세 과정

B. First boot loader (BL1)

- ▶ platform에 무관한 작고 간단한 코드. 외부 메모리 장치에 저장되어 있음
- ▶ BL0는 부트로더를 RAM으로 적재할 수 없기 때문에 BL1을 프로세서 내부의 램인 iRAM에 적재
 - BL1에는 CPU가 시스템을 초기화하고 부팅하기 위한 하드웨어 정보와 초기화를 할 수 있는 코드가 탑재
 - iRAM은 저장공간이 작기 때문에 BL2를 바로 올리지 못하고 BL1을 먼저 적재하여 BL2를 실행하기 위한 선행작업을 진행
- ▶ ROM 또는 SD카드에 있는 Second boot loader(BL2)를 내부 SRAM에 load
 - BL2의 영역에는 부트로더 전체 바이너리(u-boot.bin)가 포함되어 있음

Booting Sequence

▶ Booting 상세 과정

C. Second boot loader (BL2)

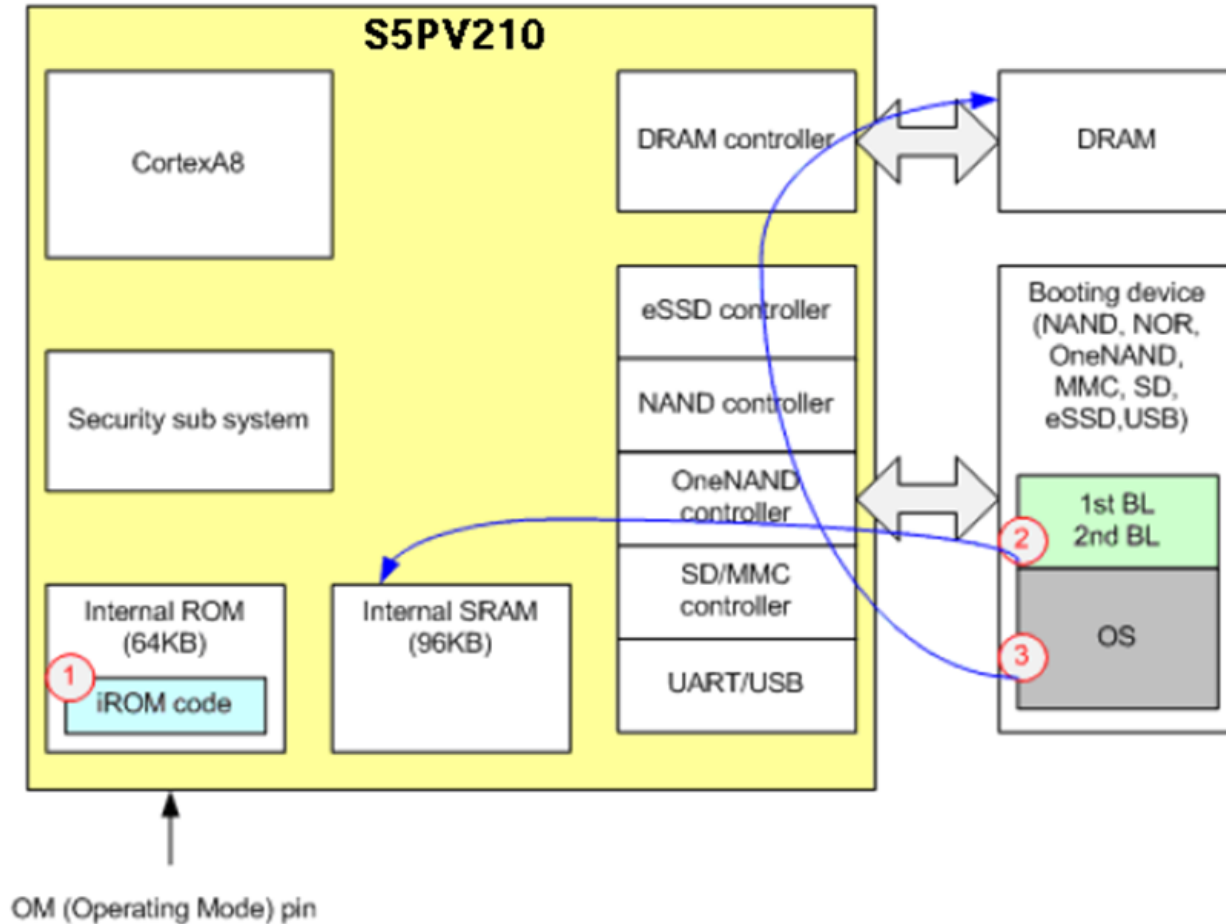
- ▶ platform에 관련성이 있는 복잡한 코드로 외부 메모리 장치에 저장
- ▶ CPU는 iRAM에 로딩된 부트로더 코드를 읽어서 U-BOOT를 실행
- ▶ System Clock, UART, DRAM controller를 초기화
- ▶ DRAM controller를 초기화한 후에, OS image를 booting 장치로부터 DRAM으로 load

D. Booting이 완료된 후에, second boot loader는 operating system으로 jump

- C. 부트로더에서 사용자 입력이 없으면 마지막 단계로 커널 이미지를 RAM으로 적재한 후 커널로 제어권을 넘김

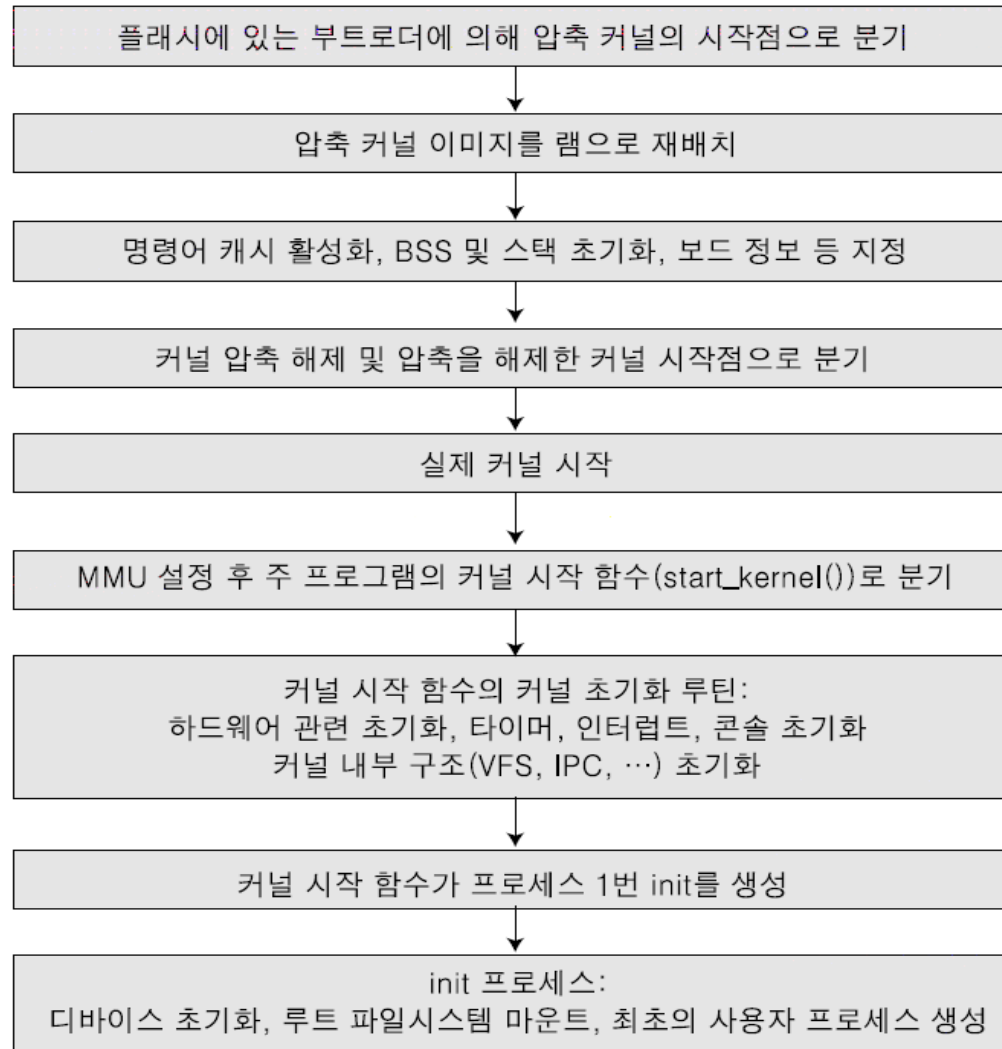
Booting Sequence

▶ Booting 과정



Booting Sequence

▶ 일반적인 임베디드 리눅스의 부팅 과정



Bootloader Compile

▶ 부트로더 컴파일

A. 소스 준비

```
# cd /work/achro5250
# cp -a /media/[achro5250 Disc]/bootloader/u-boot-140806.tar.gz .
# tar zxvf u-boot-140806.tar.gz
```

점 주의



B. BL2 툴 및 부트로더 컴파일

```
# cd /work/achro5250/u-boot/sd_fuse
# cp -a mkbl2 /work/achro5250/u-boot
# make distclean // 이전 부트로더 설정환경 초기화
# make clean // 이전 make 과정에서 생긴 파일들 제거 *.bak, *.o, *.a 등
# make achro5250_config // 타겟보드용 환경 설정
Configuring for achro5250 board...
# make // 타겟보드용 부트로더 바이너리 파일 생성
...
# ls -l u-boot.bin
-rwxr-xr-x 1 root root 154568 9월 22 11:10 u-boot.bin
```

Bootloader Compile

▶ 부트로더 컴파일

3) 컴파일 되는 과정을 통해 생성되는 파일

- ▶ 앞에서 `make achro5250_config`를 실행하면 `mkconfig` 스크립트가 호출되어
“`include/config.h`”, “`include/config.mk`” 파일을 생성
- ▶ 이 파일들을 참고로 조건에 맞추어 실제의 부트로더(`u-boot.bin`) 컴파일 작업이 진행

- “`include/config.h`”

```
/* Automatically generated - do not edit */  
#include <config_defaults.h>  
#include <configs/achro5250_android.h>  
#include <asm/config.h>
```

- “`include/config.mk`”

```
ARCH = arm  
CPU = arm_cortexa9  
BOARD = achro5250  
VENDOR = samsung  
SOC = s5pc210
```

Bootloader Compile

▶ config.mk 파일 내용 분석

1) arm

아키텍처가 무엇인지를 나타낸다.

어떤 크로스 컴파일러를 사용할 것인지 결정된다. (arm-none-linux-gnueabi-)

어떤 라이브러리를 사용할 것인지 결정된다. (lib_arm/*)

2) arm_cortexa9

CPU가 무엇인지를 나타낸다.(ARM Cortex A9 코어를 사용)

어떤 라이브러리를 사용할 것인지 결정된다. (cpu/arm_cortexa9/*)

3) achro5250

위의 CPU를 적용한 개발보드가 무엇인지를 나타낸다.

4) samsung

어떤 라이브러리를 사용할 것인지 결정된다. (board/samsung/achro5250/*)

5) s5pc210

SOC가 무엇인지를 나타낸다.

어떤 라이브러리를 사용할 것인지 결정된다. (cpu/arm_cortexa9/s5pc210/*)

Bootloader 설치

▶ 부트로더 기록 방법

1) 미니컴 실행

```
# minicom
```

```
root@ubuntu: /work/achro210/u-boot-110630
File Edit View Terminal Help

Welcome to minicom 2.4

OPTIONS: I18n
Compiled on Jan 25 2010, 06:49:09.
Port /dev/ttyS1

Press CTRL-A Z for help on special keys
```

2) 실습장비의 시리얼(Console) 포트와 PC를 USB-Serial 케이블로 연결



Bootloader 설치

▶ 부트로더 기록 방법

3) USB-Serial 케이블 연결의 확인

- ▶ USB-Serial 연결을 하면 **아이콘** 생성
- ▶ 필요시 아이콘을 우클릭 해서 Enable 시킴
- ▶ `# ls /dev/ttyU*` 명령어로 설치 확인 가능

Ubuntu-osc - VMware Workstation 12 Player (Non-commercial use only)

터미널

```
root@osc-vm: ~  
crw-rw---- 1 root dialout 4, 84 8월 31 13:46 /dev/ttyS20  
crw-rw---- 1 root dialout 4, 85 8월 31 13:46 /dev/ttyS21  
crw-rw---- 1 root dialout 4, 86 8월 31 13:46 /dev/ttyS22  
crw-rw---- 1 root dialout 4, 87 8월 31 13:46 /dev/ttyS23  
crw-rw---- 1 root dialout 4, 88 8월 31 13:46 /dev/ttyS24  
crw-rw---- 1 root dialout 4, 89 8월 31 13:46 /dev/ttyS25  
crw-rw---- 1 root dialout 4, 90 8월 31 13:46 /dev/ttyS26  
crw-rw---- 1 root dialout 4, 91 8월 31 13:46 /dev/ttyS27  
crw-rw---- 1 root dialout 4, 92 8월 31 13:46 /dev/ttyS28  
crw-rw---- 1 root dialout 4, 93 8월 31 13:46 /dev/ttyS29  
crw-rw---- 1 root dialout 4, 67 8월 31 13:46 /dev/ttyS3  
crw-rw---- 1 root dialout 4, 94 8월 31 13:46 /dev/ttyS30  
crw-rw---- 1 root dialout 4, 95 8월 31 13:46 /dev/ttyS31  
crw-rw---- 1 root dialout 4, 68 8월 31 13:46 /dev/ttyS4  
crw-rw---- 1 root dialout 4, 69 8월 31 13:46 /dev/ttyS5  
crw-rw---- 1 root dialout 4, 70 8월 31 13:46 /dev/ttyS6  
crw-rw---- 1 root dialout 4, 71 8월 31 13:46 /dev/ttyS7  
crw-rw---- 1 root dialout 4, 72 8월 31 13:46 /dev/ttyS8  
crw-rw---- 1 root dialout 4, 73 8월 31 13:46 /dev/ttyS9  
crw-rw---- 1 root dialout 188, 0 8월 31 15:45 /dev/ttyUSB0  
crw-rw---- 1 root root 5, 3 8월 31 13:46 /dev/ttyprintk  
시스템 설정 # ls -l /dev/ttyU*  
crw-rw---- 1 root dialout 188, 0 8월 31 15:45 /dev/ttyUSB0  
root@osc-vm:~#
```


Bootloader 설치

▶ 부트로더 기록 방법

4) MicroSD 부팅 설정

- ▶ 모바일 보드의 내측면의 OM핀 중 3번을 On으로 두고 나머지를 모두 Off로 함.
- ▶ 리눅스 실습시에는 항상 이 위치에 고정

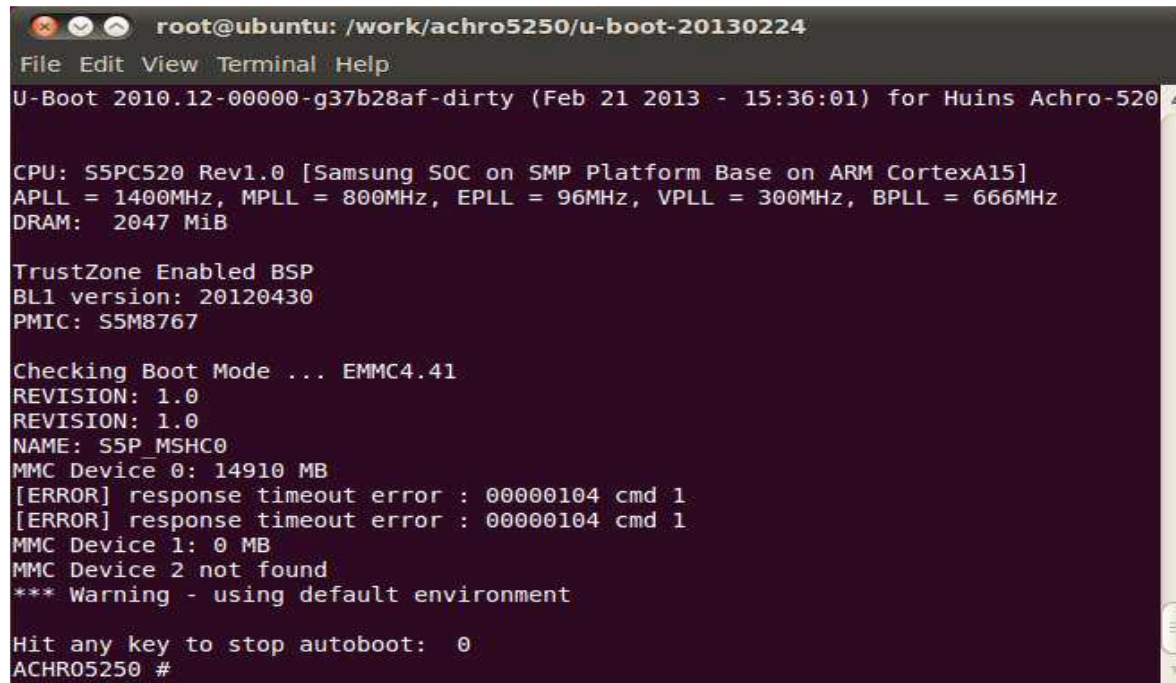


Bootloader 설치

▶ 부트로더 기록 방법

5) 보드 전원 인가

- ▶ 카운트가 0이 되기 전에 아무 키나 눌러 기존의 부트로더 상태 진입 → 시간 초과 시 리눅스 부팅을 실행
- ▶ 아래와 같은 화면이 나오지 않는 경우는 마이크로 SD 카드의 기존 부트로더가 깨지거나 새로운 SD 카드인 경우
- ▶ 본 교과에서 학습을 위해 마이크로 SD 카드에 새롭게 부트로더를 설치



```
root@ubuntu: /work/achro5250/u-boot-20130224
File Edit View Terminal Help
U-Boot 2010.12-00000-g37b28af-dirty (Feb 21 2013 - 15:36:01) for Huins Achro-520

CPU: S5PC520 Rev1.0 [Samsung SOC on SMP Platform Base on ARM CortexA15]
APLL = 1400MHz, MPLL = 800MHz, EPLL = 96MHz, VPLL = 300MHz, BPLL = 666MHz
DRAM: 2047 MiB

TrustZone Enabled BSP
BL1 version: 20120430
PMIC: S5M8767

Checking Boot Mode ... EMMC4.41
REVISION: 1.0
REVISION: 1.0
NAME: S5P_MSHC0
MMC Device 0: 14910 MB
[ERROR] response timeout error : 00000104 cmd 1
[ERROR] response timeout error : 00000104 cmd 1
MMC Device 1: 0 MB
MMC Device 2 not found
*** Warning - using default environment

Hit any key to stop autoboot: 0
ACHRO5250 #
```

Bootloader 설치

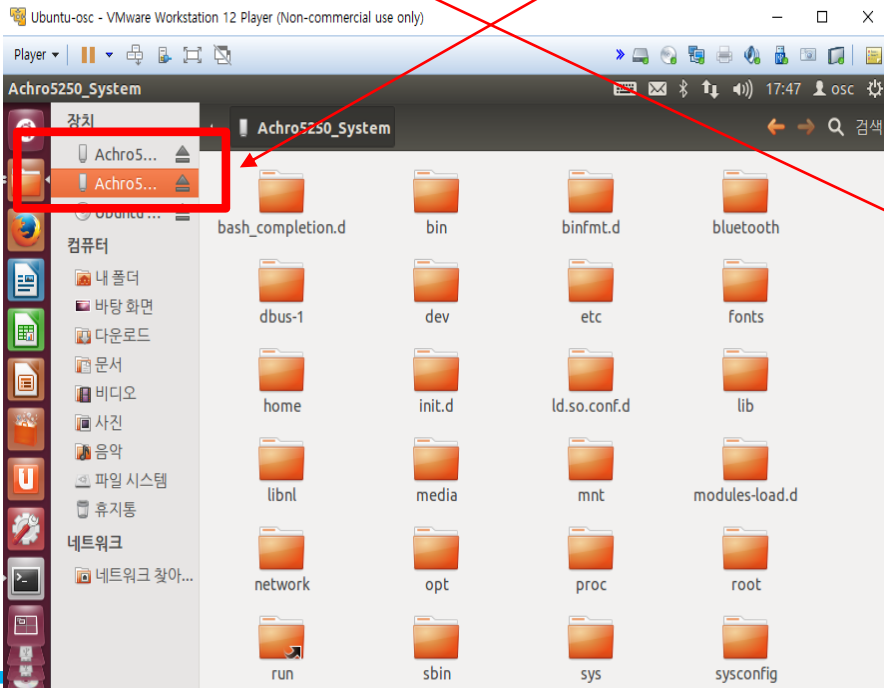
▶ 부트로더 기록 방법

6) 마이크로 SD카드를 시스템(PC)에 연결하고 초기화

▶ 기존의 부트로더가 있었을 경우 **2개의 장치**가 보인다.

▶ # ls /dev/sd* 명령어로 디바이스명 반드시 확인

■ **sdb** 인지 확인 → 이 디바이스 명으로 SD 카드를 초기화. 잘못 입력하면 우분투가 죽을 수도 있다



```
root@osc-vm: ~  
[21011.834984] usb 1-1: new high-speed USB device number 4 using ehci-pci  
[21012.109349] usb 1-1: New USB device found, idVendor=14cd, idProduct=125b  
[21012.109354] usb 1-1: New USB device strings: Mfr=1, Product=3, Serial=1  
[21012.109376] usb 1-1: Product: AutoRUN/Partition  
[21012.109378] usb 1-1: Manufacturer: Generic  
[21012.109380] usb 1-1: SerialNumber: 125B20100804  
[21012.114498] usb-storage 1-1:1.0: USB Mass Storage device detected  
[21012.114642] scsi35 : usb-storage 1-1:1.0  
[21013.503021] scsi 35:0:0:0: Direct-Access      Mass      Storage Device  
[21013.504402] sd 35:0:0:0: Attached scsi generic sg2 type 0  
[21013.656792] sd 35:0:0:0: [sdb] 15523839 512-byte logical blocks: (7.9  
[21013.656792] sd 35:0:0:0: [sdb] write protect is off  
[21013.660711] sd 35:0:0:0: [sdb] Mode Sense: 03 00 00 00  
[21013.663721] sd 35:0:0:0: [sdb] No Caching mode page found  
[21013.663752] sd 35:0:0:0: [sdb] Assuming drive cache: write through  
[21013.682579] sd 35:0:0:0: [sdb] No Caching mode page found  
[21013.682592] sd 35:0:0:0: [sdb] Assuming drive cache: write through  
[21013.706720] sdb: sdb1 sdb2  
[21013.723547] sd 35:0:0:0: [sdb] No Caching mode page found  
[21013.723561] sd 35:0:0:0: [sdb] Assuming drive cache: write through  
[21013.723570] sd 35:0:0:0: [sdb] Attached SCSI removable disk  
[21015.012669] EXT4-fs (sdb1): mounting ext3 file system using the ext4 s  
[21015.057480] EXT4-fs (sdb2): mounted filesystem with ordered data mode  
(null)  
[21015.154270] EXT4-fs (sdb1): mounted filesystem with ordered data mode  
(null)  
root@osc-vm: ~#
```

Bootloader 설치

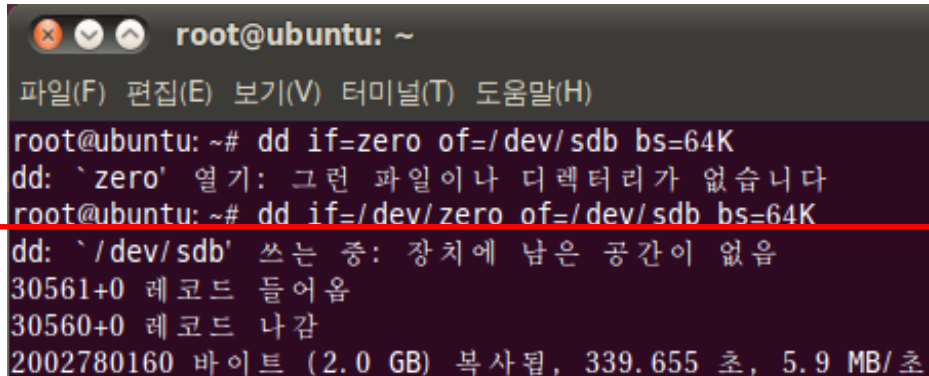
▶ 부트로더 기록 방법

6) 마이크로 SD카드를 시스템(PC)에 연결하고 초기화 - 계속

▶ 다음 명령으로 SD 카드를 초기화하되 **sdb** 확인 필수

▶ 많은 시간이 소요된다 (MicroSD 용량에 따라 다름 20~30분)

```
dd if=/dev/zero of=/dev/sdb bs=64K
```



A terminal window titled 'root@ubuntu: ~' showing the execution of the 'dd' command. The command is 'dd if=zero of=/dev/sdb bs=64K'. The output shows the progress of the write operation, including the number of records read and written, and the total bytes written. A red box highlights the progress output.

```
root@ubuntu: ~# dd if=zero of=/dev/sdb bs=64K
dd: `zero' 열기: 그런 파일이나 디렉터리가 없습니다
root@ubuntu: ~# dd if=/dev/zero of=/dev/sdb bs=64K
dd: `/dev/sdb' 쓰는 중: 장치에 남은 공간이 없음
30561+0 레코드 들어옴
30560+0 레코드 나감
2002780160 바이트 (2.0 GB) 복사됨, 339.655 초, 5.9 MB/초
```

▶ 진행시간을 알고 싶을 때는 호스트에서 다른 터미널을 열고 다음 명령을 실행한다. 30초 마다 상태가 보여진다

```
$ watch -n30 'sudo kill -USR1 $(pgrep ^dd)'
```

▶ 완료 후 SD카드를 PC에 재연결(SD카드를 제거했다가 다시 삽입) 한다.

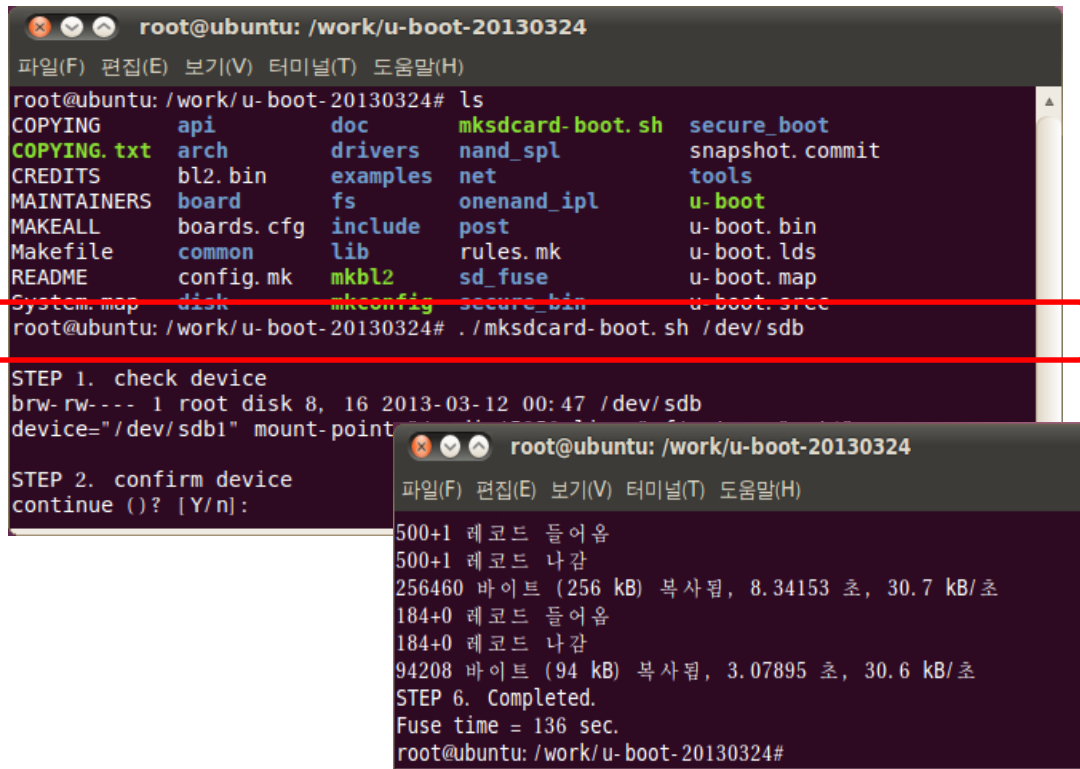
Bootloader 설치

▶ 부트로더 기록 방법

7) 다음의 부트로더 및 관련 파일을 SD 카드로 복사

▶ `./mksdcard-boot.sh /dev/sdb` (부트로더 소스 디렉토리 “/work/achro5250/u-boot”에서 실행)

- Confirm을 물어볼 경우 엔터키(Yes) 입력



```
root@ubuntu: /work/u-boot-20130324
파일(F) 편집(E) 보기(V) 터미널(T) 도움말(H)
root@ubuntu: /work/u-boot-20130324# ls
COPYING      api          doc          mksdcard-boot.sh  secure_boot
COPYING.txt  arch         drivers      nand_spl          snapshot.commit
CREDITS      bl2.bin     examples    net               tools
MAINTAINERS  board       fs           onenand_ipl       u-boot
MAKEALL      boards.cfg  include     post              u-boot.bin
Makefile     common     lib         rules.mk          u-boot.lds
README      config.mk  mkbl2       sd_fuse           u-boot.map
System.map   disk       mkconfig    secure_bin        u-boot.srec
root@ubuntu: /work/u-boot-20130324# ./mksdcard-boot.sh /dev/sdb

STEP 1. check device
brw-rw---- 1 root disk 8, 16 2013-03-12 00:47 /dev/sdb
device="/dev/sdb1" mount-point=""

STEP 2. confirm device
continue (?)? [Y/n]:

500+1 레코드 들어옴
500+1 레코드 나감
256460 바이트 (256 kB) 복사됨, 8.34153 초, 30.7 kB/초
184+0 레코드 들어옴
184+0 레코드 나감
94208 바이트 (94 kB) 복사됨, 3.07895 초, 30.6 kB/초
STEP 6. Completed.
Fuse time = 136 sec.
root@ubuntu: /work/u-boot-20130324#
```

기록되는 내용

- ① *.bl1
- ② bl2.bin
- ③ u-boot.bin
- ④ *.tzw (보안 모듈)

Bootloader 설치

▶ 부트로더 기록 방법

8) SD 카드 파티션 설정

- ▶ RAW 영역 : MBR, 부트로더 및 커널이 기록되는 영역
- ▶ 첫번째 파티션 : achro5250_System
- ▶ 두번째 파티션 : achro5250_Data

```
# fdisk /dev/sdb
Command (m for help) : n
Partition type:
   P   primary(0 primary, 0 extended, 4 free)
   e   extended
Select (default p) : p
Partition number (1-4, default 1) : 1
First sector (2048-(SD 카드마다 다를 수 있음), default 2048) : 1050624
Last sector, +sectors or +size{K,M,G} (1050624-(SD 카드 끝), default (SD 카드 끝)): +512M
Command (m for help) : p

.....
Device Boot      Start         End      Block      .....
/dev/sdb1    1050624    2099199               .....
.....
Command (m for help) : n
Partition type:
   P   primary(0 primary, 0 extended, 4 free)
   e   extended
Select (default p) : p
Partition number (1-4, default 2) : 2
First sector (2048-(SD 카드마다 다를 수 있음)), default 2048) : 2099200
( SD 카드마다 다를 수 있음. End+1 입력.)
Last sector, +sectors or +size{K,M,G} (2099200-(SD 카드 끝), default (SD 카드 끝)): Enter
Command (m for help) : w
```


Bootloader 설치

▶ 부트로더 기록 방법

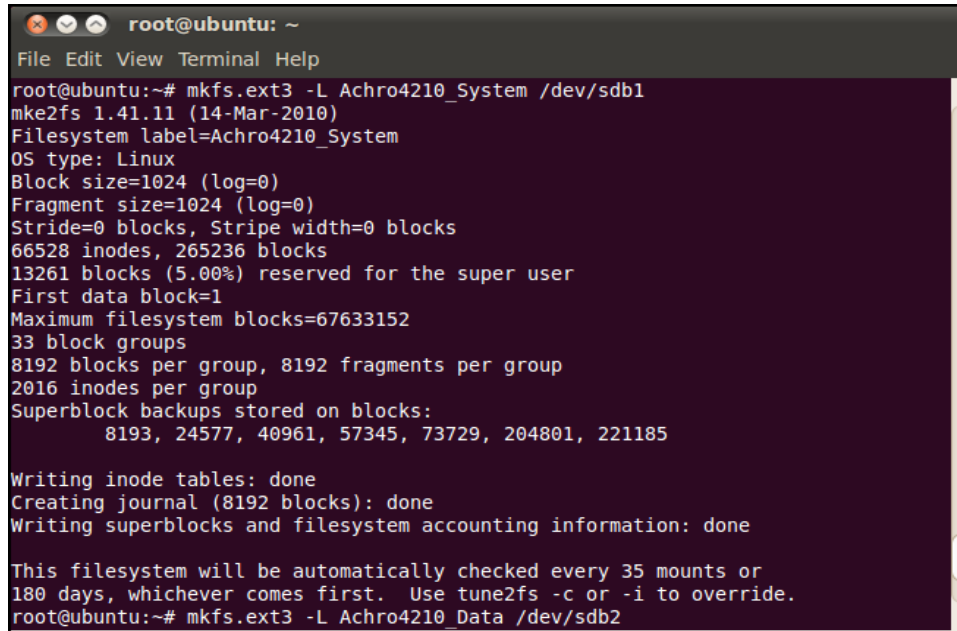
9) 파일 시스템 생성

▶ 리눅스 파일 시스템을 위한 영역 초기화

```
# mkfs.ext3 -L Achro5250_System /dev/sdb1
```

▶ 데이터 저장을 위한 파일 시스템 영역 초기화

```
# mkfs.ext3 -L Achro5250_Data /dev/sdb2
```

A terminal window titled 'root@ubuntu: ~' showing the output of the command 'mkfs.ext3 -L Achro4210_System /dev/sdb1'. The output includes details about the filesystem creation, such as block size, fragment size, and the number of inodes and blocks. It also shows the journal creation and the final state of the filesystem.

```
root@ubuntu: ~  
File Edit View Terminal Help  
root@ubuntu:~# mkfs.ext3 -L Achro4210_System /dev/sdb1  
mke2fs 1.41.11 (14-Mar-2010)  
Filesystem label=Achro4210_System  
OS type: Linux  
Block size=1024 (log=0)  
Fragment size=1024 (log=0)  
Stride=0 blocks, Stripe width=0 blocks  
66528 inodes, 265236 blocks  
13261 blocks (5.00%) reserved for the super user  
First data block=1  
Maximum filesystem blocks=67633152  
33 block groups  
8192 blocks per group, 8192 fragments per group  
2016 inodes per group  
Superblock backups stored on blocks:  
    8193, 24577, 40961, 57345, 73729, 204801, 221185  
  
Writing inode tables: done  
Creating journal (8192 blocks): done  
Writing superblocks and filesystem accounting information: done  
  
This filesystem will be automatically checked every 35 mounts or  
180 days, whichever comes first. Use tune2fs -c or -i to override.  
root@ubuntu:~# mkfs.ext3 -L Achro4210_Data /dev/sdb2
```

▶ 완료되면 호스트의 MicroSD를 제거해서 보드에 넣고 전원 인가

Bootloader 설치

▶ 부트로더 기록 방법

10) 부트로더 실행 확인

- ▶ 완료되면 호스트의 MicroSD를 제거해서 보드에 넣고 전원 인가하고 minicom으로 확인
- ▶ 시간이 초과되어도 나면 커널을 찾지 못한다는 메시지가 나오면 성공

```
root@osc-vm: /work/achro5250/u-boot
-      rfs:   33831 to   66598 ( 32768 sectors, 16777216 Bytes)
-      bl2:    30 to    61 (    32 sectors,   16384 Bytes)
[DEBUG] Partition Map ----
MMC Device 1: 3728 MB
MMC Device 2 not found
*** Warning - using default environment

Hit any key to stop autoboot:  0
[DEBUG] Partition Map ----
-      fwbl1:    1 to    30 (    30 sectors,   15360 Bytes)
- u-boot parted:  31 to    62 (    32 sectors,   16384 Bytes)
- u-boot:       63 to   718 (   656 sectors,  335872 Bytes)
- TrustZone S/W: 719 to  1030 (   312 sectors,  159744 Bytes)
- environment: 1031 to  1062 (    32 sectors,   16384 Bytes)
- kernel:      1063 to 33830 ( 32768 sectors, 16777216 Bytes)
- rfs:        33831 to 66598 ( 32768 sectors, 16777216 Bytes)
- bl2:        31 to    62 (    32 sectors,   16384 Bytes)
[DEBUG] Partition Map ----
reading kernel..device 0 Start 1063, Count 32768
MMC read: dev # 0, block # 1063, count 32768 ... 32768 blocks read: OK
completed
Wrong Image Format for bootm command
ERROR: can't get kernel image!
ACHRO5250 #
```


Bootloader 명령

▶ 부트로더 명령

▶ 도움말

- ▶ Help 명령을 통하여 사용할 수 있는 커맨드를 확인
- ▶ 세부적인 명령을 확인하고 싶다면 help 다음에 해당 명령을 입력

```
achro5250 # help
... (내용생략) ...
achro5250 # help bdfinfo
```

▶ 개발보드 정보 출력

```
achro5250 # bdfinfo
arch_number = 0x00000B16
env_t = 0x00000000
boot_params = 0x40000100
DRAM bank = 0x00000000
-> start = 0x00000000
-> size = 0x10000000
ethaddr = 00:40:5C:26:0A:5B
ip_addr = 192.168.1.121
baudrate = 115200 bps
```

Bootloader 명령

▶ 부트로더 명령

▶ 메모리 관련 명령

▷ cp: 메모리를 복사하는 명령

```
achro5250 # help cp
cp - memory copy
Usage:
cp [.b, .w, .l] source target count
achro5250 # cp 40000000 40008100 100
```

▷ md: 메모리 값을 표시

```
achro5250 # help md
md - memory display
Usage:
md [.b, .w, .l] address [# of objects]
achro5250 # md 40000000
40008000: 00002000 00000000 00000000 00000000 . .....
40008000: ea000013 e59ff014 e59ff014 e59ff014 .....
40008000: e59ff014 e59ff014 e59ff014 e59ff014 .....
40008000: 2fe001e0 2fe00240 2fe002a0 2fe00300 .../@../.../
40008000: 2fe00360 2fe003c0 2fe00420 12345678 `../.../ ../xV4.
(이하 생략)
```

Bootloader 명령

▶ 부트로더 명령

▶ 메모리 관련 명령

▷ mm: 메모리를 수정 (자동증가)

```
achro5250 # help mm
mm - memory modify (auto-incrementing address)
Usage:
mm [.b, .w, .l] address
achro5250 # mm 40000000
40008000: 00002000 ? 11112222
40008000: 00000000 ? 33334444
40008000: 00000000 ? q (mm 명령을 종료할 때는 16진수가 아닌 값을 됨)
```

▷ mw: 메모리 기록

```
achro5250 # help mw
mw - memory write (fill)
Usage:
mw [.b, .w, .l] address value [count]
achro5250 # mw 40000000 00002000
achro5250 # md 40000000 4
40008000: 00002000 33334444 00000000 00000000 . ...DD33.....
```

Bootloader 명령

▶ 부트로더 명령

▶ Flash memory(microSD) Command

▶ microSD(NAND) 관련 명령들

achro5250 # help movi

```
movi init - Initialize moviNAND and show card info
movi read {u-boot | kernel} {addr} - Read data from sd/mmc
movi write {fwbl1 | u-boot | kernel} {addr} - Write data to sd/mmc
movi read rootfs {addr} [bytes(hex)] - Read rootfs data from sd/mmc by size
movi write rootfs {addr} [bytes(hex)] - Write rootfs data to sd/mmc by size
movi read {sector#} {bytes(hex)} {addr} - instead of this, you can use "mmc read"
movi write {sector#} {bytes(hex)} {addr} - instead of this, you can use "mmc write"
```

▶ movi init : 초기화 명령, SD 정보 출력

achro5250 # movi init

```
Device: S5P_HSHC2
Manufacturer ID: 1b
OEM: 534d
Name: 00000
Tran Speed: 0
Rd Block Len: 512
SD version 2.0
High Capacity: No
Size: 1910MB (block: 3911680)
Bus Width: 2-bit
```

Bootloader 명령

▶ 부트로더 명령

▷ `movi read {u-boot | kernel} {device_number} {addr}`

SD의 u-boot, kernel 이미지를 읽어 SDRAM 번지에 로드

```
achro5250 # movi read u-boot S5P_MSHC2 40000000
reading bootloader.. 49, 1024
MMC read: dev # 0, block # 49, count 1024 ...1024 blocks read: OK
completed
achro5250 # movi read kernel S5P_MSHC2 40000000
reading kernel.. 1073, 8192
MMC read: dev # 0, block # 1073, count 8192 ...8192 blocks read: OK
Completed
```

▷ `movi read rootfs {device_number} {addr} [bytes(hex)]`

SD의 rootfs 이미지를 특정 크기만큼 읽어 SDRAM으로 로드

```
achro5250 # movi read rootfs S5P_MSHC2 40000000 100000
reading RFS.. 9265, 2048
MMC read: dev # 0, block # 9265, count 2048 ...2048 blocks read: OK
completed
```

Bootloader 명령

▶ 부트로더 명령

▷ `movi write {u-boot | kernel} {device_number} {addr}`

memory의 u-boot, kernel 데이터를 읽어 SD에 기록

```
achro5250 # movi read u-boot S5P_MSHC2 40000000
reading bootloader.. 49, 1024
MMC read: dev # 0, block # 49, count 1024 ...1024 blocks read: OK
completed
achro5250 # movi read kernel S5P_MSHC2 40000000
reading kernel.. 1073, 8192
MMC read: dev # 0, block # 1073, count 8192 ...8192 blocks read: OK
completed
```

▷ `movi write rootfs {device_number} {addr} [bytes(hex)]`

SDRAM의 영역에 있는 이미지를 플래시 영역에 기록

```
achro5250 # movi write rootfs S5P_MSHC2 40000000 100000
writing RFS.. 9265, 2048
MMC write: dev # 0, block # 9265, count 2048 ... 2048 blocks written: OK
completed
```

Q & A
