1. 커널 설치 및 패치

EZ-X5보드용 리눅스 커널을 구하여야 하는 이유를 정리하면 다음과 같다.

첫째, C 프로그램 작성을 위해서 ARM에 관련된 헤더 파일을 작성해야 하므로
둘째, 부트로더인 이지부트를 작성하기 위해서
셋째, 최종적으로 리눅스 커널을 올리기 위하여

이 문서는 ARM 용 리눅스 커널을 설치하는 법에 대한 것을 기술한다.
EZ-X5보드는 리눅스 커널 2.4.19로 설치되어 있다.

주의 사항)
EZ-X5보드용 커널을 만들고 컴파일하기 전에 XScale용 **크로스 컴파일러 환경**을 먼저 구축하여야 한다.

1.1. ARM Linux Kernel을 어디서 구하는가?

ARM Linux Kernel은 I386용 (PC 에서 사용되는) Linux와 전혀 다르게 설계되고 코딩된 것은 아니다. 리눅스 커널 자체가 상당히 이식하기에 좋은 구조로 되어 있기 때문에 이를 버리고 다른 커널을 디자인 할 필요는 없다. 이미 많은 사람들이 ARM용 리눅스를 만드는 데 공헌을 하였으며 다음 사이트가 공식적인 ARM 리눅스 사이트이다.

http://www.arm.linux.org.uk/

이 사이트를 찾아가보면 여러분은 많은 정보를 얻을 수 있을 것이다.

하지만 이 사이트는 너무 방대하고 EZ-X5보드를 위한 리눅스 커널을 구해야 하므로 이를 위해서 어떻게 구하는 가에 대한 것을 이 문서에서 기술하겠다.

EZ-X5보드를 위한 리눅스 커널을 구성하는 방법은 다음과 같은 순서를 따라야 한다.

첫째, 리눅스 커널을 구한다.
둘째, 암 패치를 수행한다.
셋째, XScale용 패치를 수행한다.
넷째, EZ 보드를 위한 패치를 수행한다.

FALINUX.COM

i386 용 리눅스 커널 구하기

가장 먼저 하여야 할 것은 i386용 리눅스 커널을 구해야 한다. 커널을 구할 때는 ARM용으로 패치 가능한 리눅스 커널을 구하여야 한다. 왜냐하면 커널 패치가 불가능한 것을 구하게 되면 여러분 스스로 ARM용 패치 파일을 만들어야 하는데 이런 것을 할 줄 아시는 분이라면 굳이 이 문서를 읽을 필요가 없기 때문이다.

리눅스 커널을 구할 수 있는 곳은

ftp://ftp.kernel.org/ 이다

우리가 구하려고 하는 커널이 있는 정확한 위치는 아래와 같다.

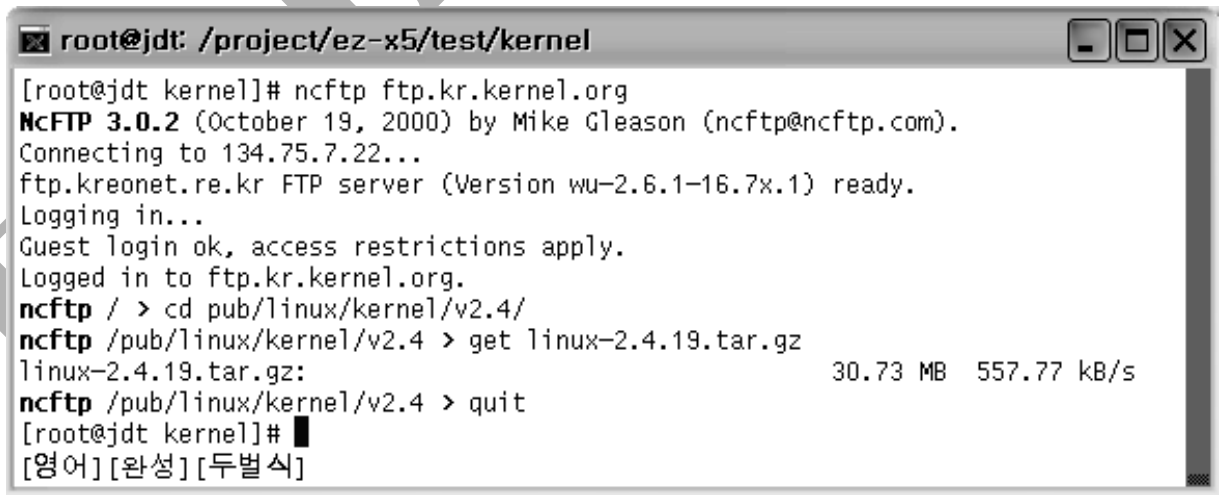
ftp://ftp.kernel.org/pub/linux/kernel/v2.4/linux-2.4.19.tar.gz

여기서 받으면 느리므로 국내 미러사이트인

ftp://ftp.kr.kernel.org/pub/linux/kernel/v2.4/linux-2.4.19.tar.gz

커널을 다운로딩 한다.

다음은 리눅스에서 tfp를 통하여 커널을 다운로드 하는 것을 그랩하였다.



제공된 CD에도 동일한 파일이 존재하므로 위 과정을 생략하고 싶은 사람은 CD에서 가져온다. 이하 패치 파일도 마찬 가지다.

LINUX.COM

ARM 용 패치 구하기

ARM용 패치는 다음의 위치에서 구할 수 있다.

ftp://ftp.arm.uk.linux.org/pub/linux/arm/kernel/v2.4/patch-2.4.19-rmk7.gz

이 ARM용 패치는 마지막에 rmk란 약어가 붙는데 이것은 Russell King이라는 사람의 이니셜이다.
일반적으로 rmk 패치는 ac 패치 즉, 알랜 콕스 패치에 대한 적용을 하는데 이때는 –ac?-rmk?. 이런식으로 붙는다.
그러나 위의 파일명을 보면 알겠지만 이 rmk 패치는 ac패치가 생략되어 있다.
그러므로 이곳에서는 바로 linux-2.4.19에 패치를 적용해도 된다는 것이다.

XScale PXA255 용 패치 구하기

EZ-X5보드는 XScale PXA255를 사용하는 보드이다.
EZ-X5보드는 XScale PXA255 회로도를 일부 수정하고 최소화 한 보드이다.

그러므로 우리는 XScale PXA255용을 구해야 한다.

ftp://ftp.arm.uk.linux.org/pub/linux/arm/people/nico/diff-2.4.19-rmk7-pxa1.gz

리코패치는 rmk에 대한 패치를 한다.
pxa는 XScale의 PXA255용이라는 의미이다. 그 뒷 번호는 버전 이다.

커널 패치를 위한 파일 및 이지보드 패치 파일 정리

위 과정을 살펴 보면 결론적으로 2.4.19 커널 패치에 관련된 파일은 3개이다.

- **linux-2.4.19.tar.gz**
- **patch-2.4.19-rmk7.gz**
- **diff-2.4.19-rmk7-pxa1.gz**

EZ-X5보드에 맞는 커널을 패치하기 위한 파일은 1개이다. 이는 제공한 CD에 있다.

- **diff-2.4.19-rmk7-pxa1-ez-x5.gz**

FALINUX.COM

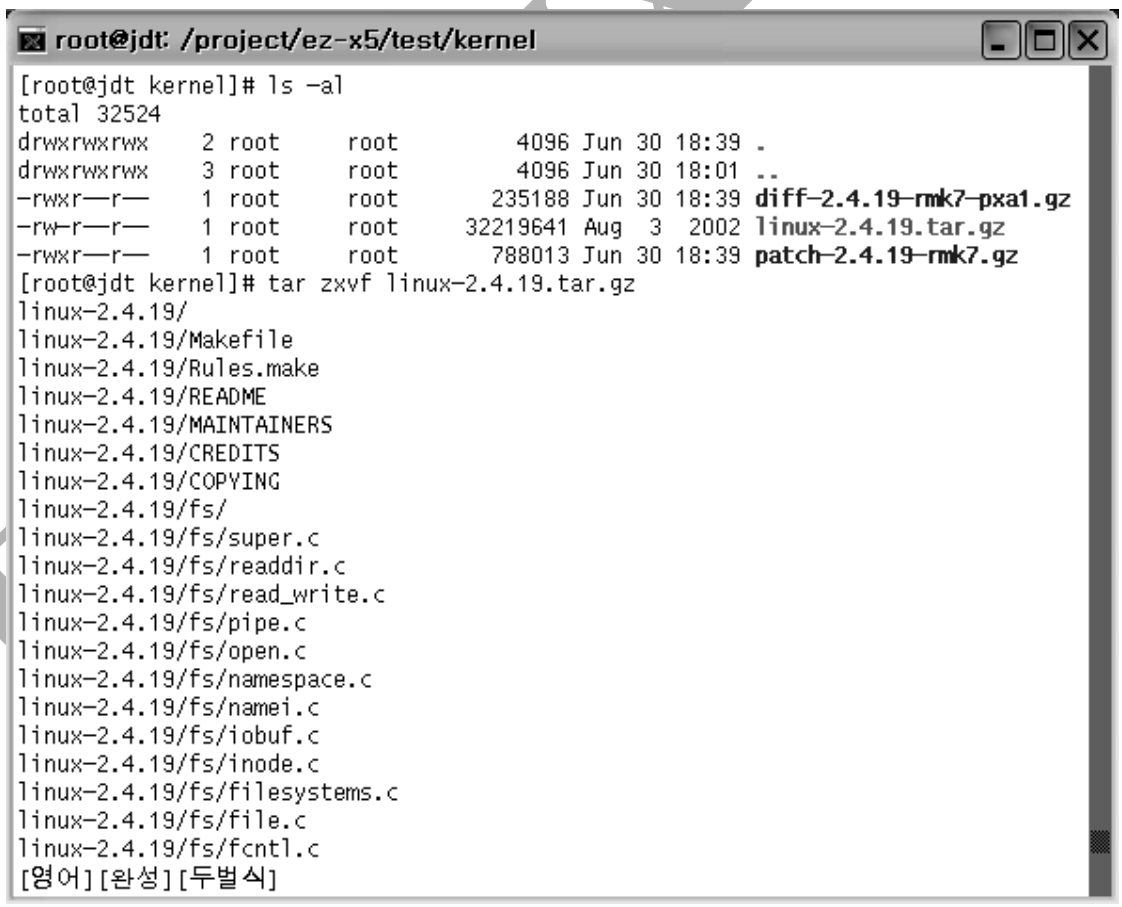
1.2. 커널 설치 및 패치

리눅스 커널을 설치하는 원래 위치는 /usr/src/linux 이다.
하지만 우리는 이곳에 리눅스를 설치하면 문제가 발생될 수 있다. 여러분의 linux에 여러 MPU의 개발 환경이 설치될 수도 있고 여기에서는 ARM용으로 패치 하여야 하기 때문이다.

그렇다면 어디에 설치하는 것이 좋을까?

여기서는 **/project/ez-x5/test/kernel** 이란 디렉토리에 설치하는 것을 가정하여 설명한다.

mkdir 명령을 이용하여 **/project/ez-x5/test/kernel** 란 디렉토리를 만든다.
이 디렉토리로 이동한다.
cd **/project/ez-x5/test/kernel**
이곳에 구한 파일을 모두 넣는다.
그리고 커널의 압축을 푼다.
 # tar zxvf linux-2.4.19.tar.gz



설치가 끝나면 linux 란 디렉토리가 생성된다.

다음은 ARM용 패치를 한다.

커널 압축을 풀면 linux-2.4.19 디렉토리가 생성되고, 여기에 패치 파일을 적용한다.

 # cd linux

 # gzip -cd ../patch-2.4.19-rmk7.gz | patch –p1

```
root@jdt: /project/ez-x5/test/kernel/linux-2.4.19
[root@jdt kernel]# ls -al
total 32528
drwxrwxrwx    3 root     root         4096 Jun 30 18:42 .
drwxrwxrwx    3 root     root         4096 Jun 30 18:01 ..
-rwxr--r--    1 root     root       235188 Jun 30 18:39 diff-2.4.19-rmk7-pxa1.gz
drwxr-xr-x   14 573      573          4096 Aug  3  2002 linux-2.4.19
-rw-r--r--    1 root     root     32219641 Aug  3  2002 linux-2.4.19.tar.gz
-rwxr--r--    1 root     root       788013 Jun 30 18:39 patch-2.4.19-rmk7.gz
[root@jdt kernel]# cd linux-2.4.19
[root@jdt linux-2.4.19]# gzip -cd ../patch-2.4.19-rmk7.gz | patch -p1
patching file CREDITS
patching file Documentation/Configure.help
patching file Documentation/arm/Booting
patching file Documentation/arm/ConfigVars
patching file Documentation/arm/MEMC
patching file Documentation/arm/SA1100/SA1100_USB
patching file Documentation/arm/SA1100/nanoEngine
patching file Documentation/cpufreq
patching file Documentation/cpufreq-old
patching file Documentation/l3/structure
patching file Documentation/serial/driver
patching file MAINTAINERS
patching file Makefile
patching file Rules.make
patching file arch/alpha/config.in
[영어][완성][두벌식]
```

다음은 XScale 용 패치를 한다.

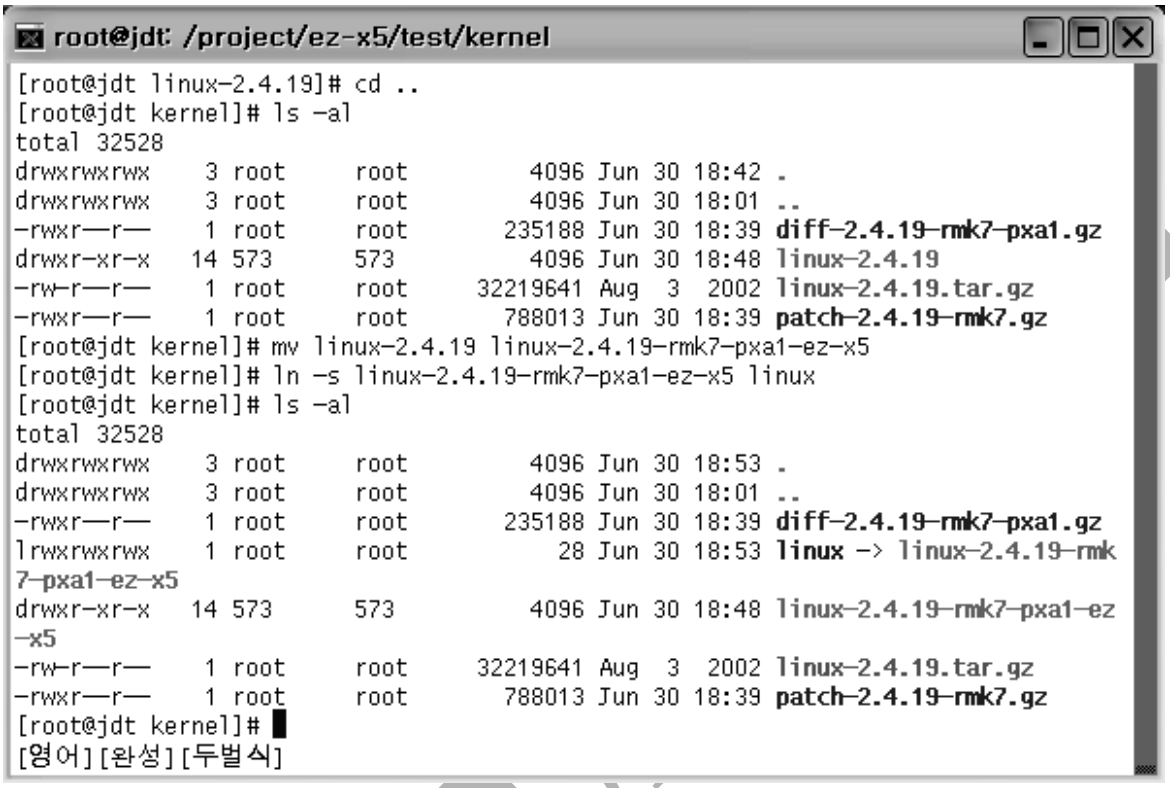
 # gzip -cd ../diff-2.4.19-rmk7-pxa1.gz | patch -p1

```
root@jdt: /project/ez-x5/test/kernel/linux-2.4.19
[root@jdt linux-2.4.19]# gzip -cd ../diff-2.4.19-rmk7-pxa1.gz | patch -p1
patching file Documentation/Configure.help
patching file Documentation/arm/XScale/PXA/USB-client
patching file Makefile
patching file arch/arm/Makefile
patching file arch/arm/boot/Makefile
patching file arch/arm/boot/compressed/Makefile
patching file arch/arm/boot/compressed/head-xscale.S
patching file arch/arm/boot/compressed/head.S
patching file arch/arm/config.in
patching file arch/arm/def-configs/cerfboard_pxa
patching file arch/arm/def-configs/cerfpda_pxa
patching file arch/arm/def-configs/csb226
[영어][완성][두벌식]
```

아마도 패치에 대부분 성공했을 것이다.

****[Falinux가 사용하는 kernel 관리법 소개]*****

현재 당사는 이렇게 패치한 커널을 버전별로 파일 이름을 바꾸고, 이를 링크를 걸어 사용하고 있다



앞으로의 진행 역시 이렇게 링크된 것을 전제로 하여 진행한다

FALINUX

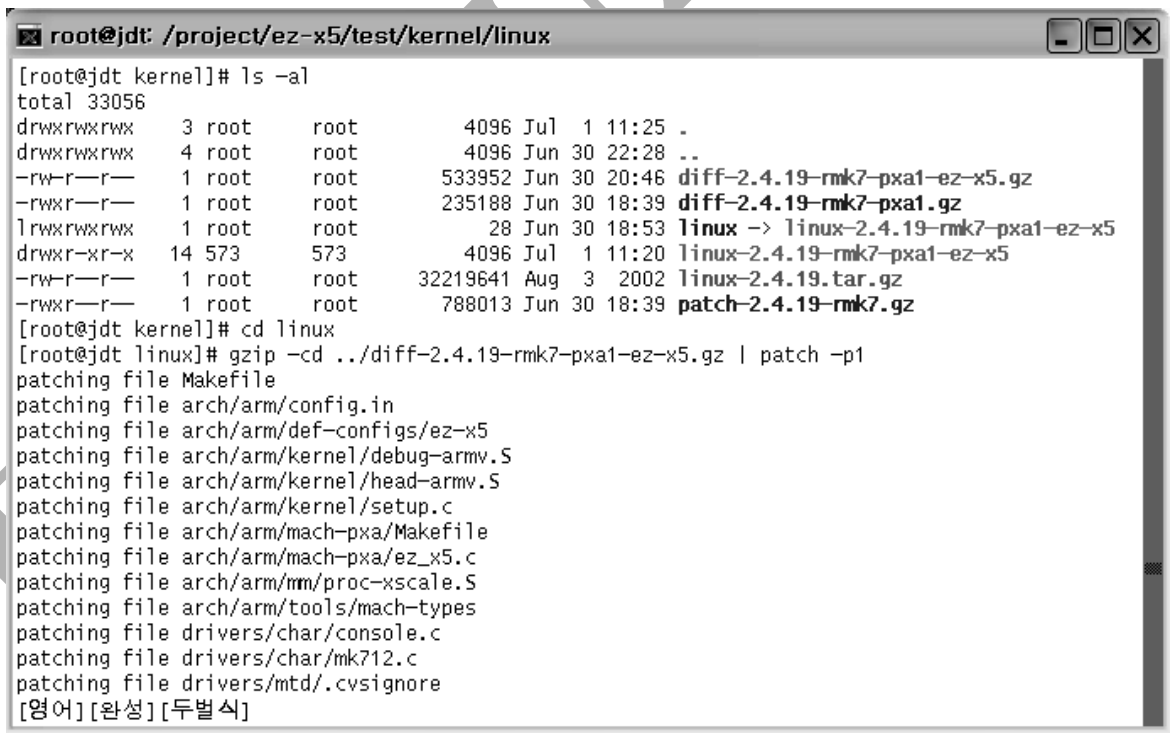
EZ-X5 보드 패치

여기까지의 패치는 일반적으로 배포되는 리눅스 커널에 ARM과 XScale 패치를 적용한 것이다. 마지막으로 **보드 패치**를 적용시킨다.

보드 패치라고 하는 것은 이미 배포되고 있는 커널의 보드와 직접 제작한 보드에 따라 디바이스와 배선이 다르기 때문에 이러한 디바이스를 제대로 인식시키기 위하여, 커널을 보드에 맞게 수정해준 것을 의미하며 기존 패치된 커널에 보드 패치를 적용해 주어야 한다.

자신이 직접 보드 패치를 하기 위하여서는 보드에 관련된 회로도와 CPU 및 보드에 사용되는 부품들의 스펙을 다 조사하여 보드에 맞게 패치 파일을 만들어 준다.
보드 패치를 한 후에는 일반적인 커널 컴파일 과정과 유사하다.
EX-X5보드는 패치 파일이 이미 제공되므로 CD에서 복사해 온다.

```
# cp –a /mnt/cdrom/sw/kernel/diff-2.4.19-rmk7-pxa1-ez-x5.gz diff-2.4.19-rmk7-pxa1-ez-x5.gz
# cd linux
# gzip -cd ../diff-2.4.19-rmk7-pxa1-ez-x5.gz | patch –p1
```

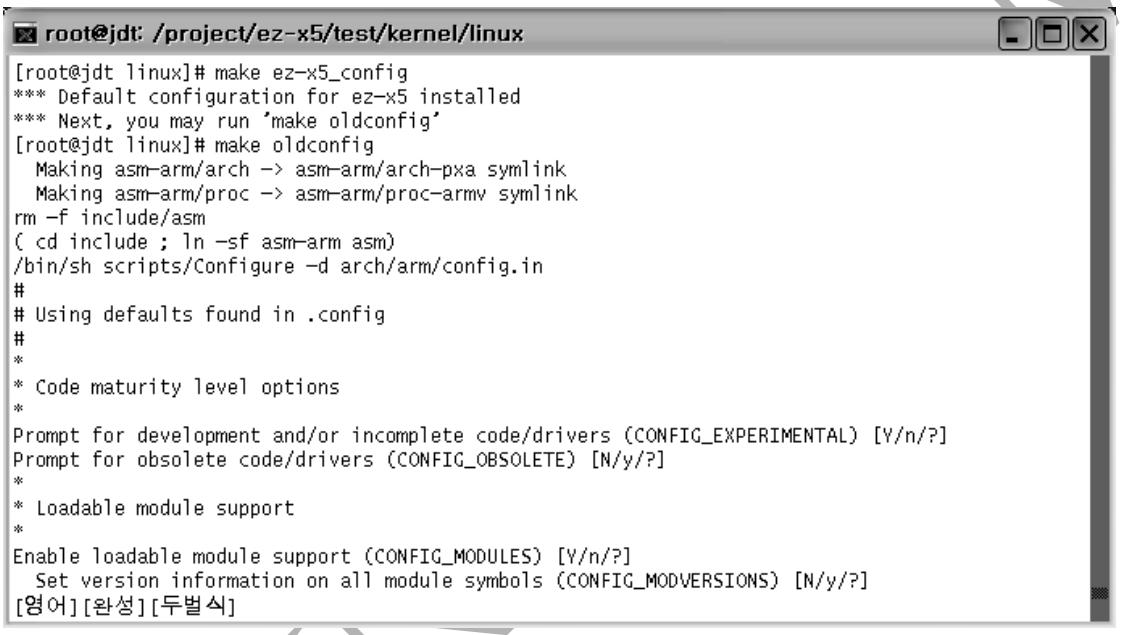


1.3. 커널을 컴파일

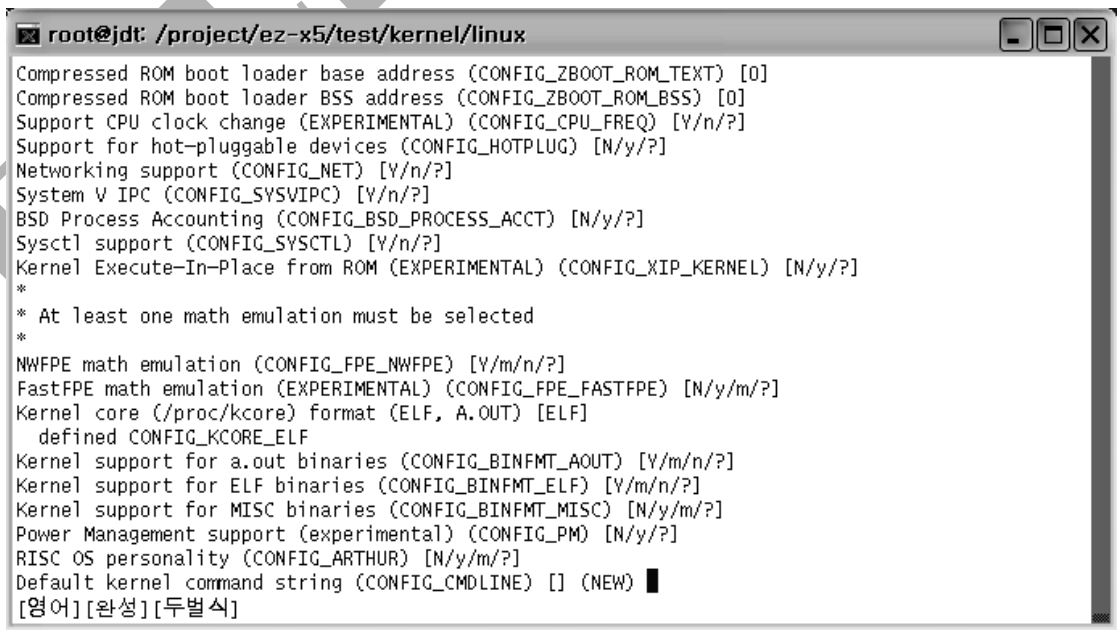
커널 패치를 모두 정상적으로 수행하였다면 커널을 컴파일 한다.
우선은 커널 컴파일 환경을 설정하여야 한다.
이후의 커널 컴파일 과정은 x86에서의 과정과 동일하다.
약간 다른 것은 config 설정하는 것이 다를 뿐이다.

 # make ez-x5_config

 # make oldconfig



중간에 커널 커맨드 문자열을 넣으라는 메시지가 나오는데 그냥 엔터를 친다.
EZ-X5에서는 부트로더에서 커널 커맨드 문자열을 써 넣을 수 있게 했기 때문이다.

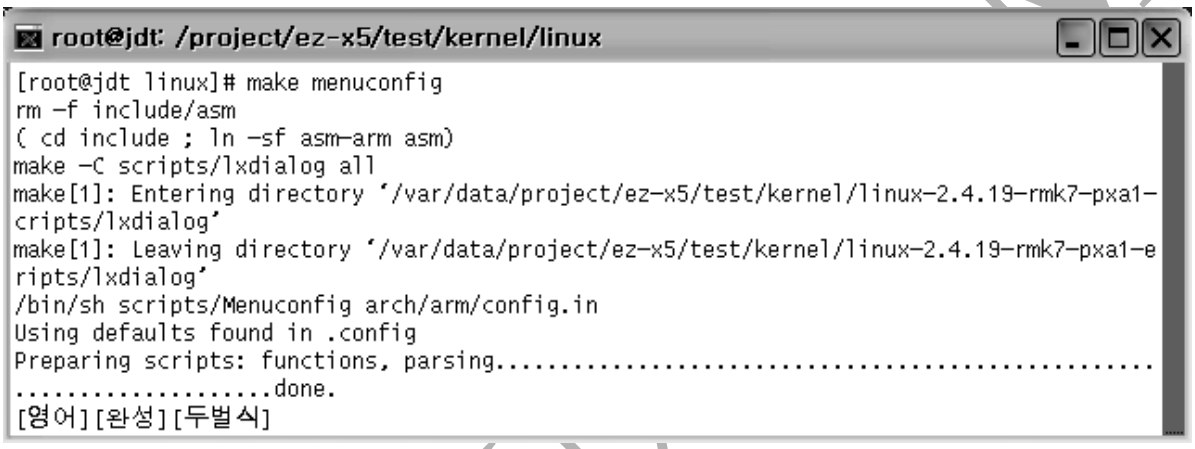


1.4. 커널 컴파일 옵션

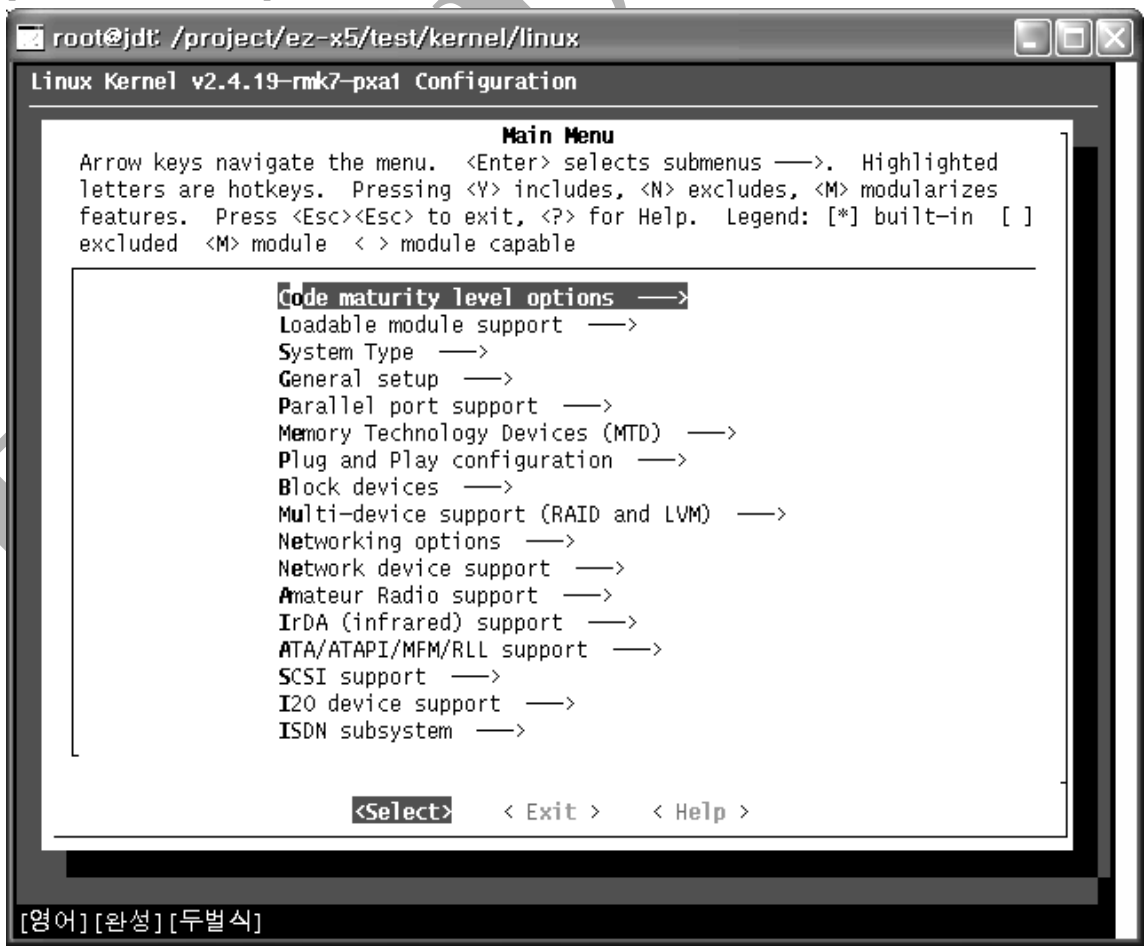
이 부분에 대한 설명은 이 장 마지막 환경을 참조할 것 제품 출하 시에 설정된 커널 컴파일 옵션이 있다.

커널 컴파일 환경 설정은 make menuconfig를 수행한다.

수정된 내용이 없더라도 여기서는 저장할 것인가를 물어 볼 때 Yes를 입력한다.
[참고 자료용으로 보기 바란다.]



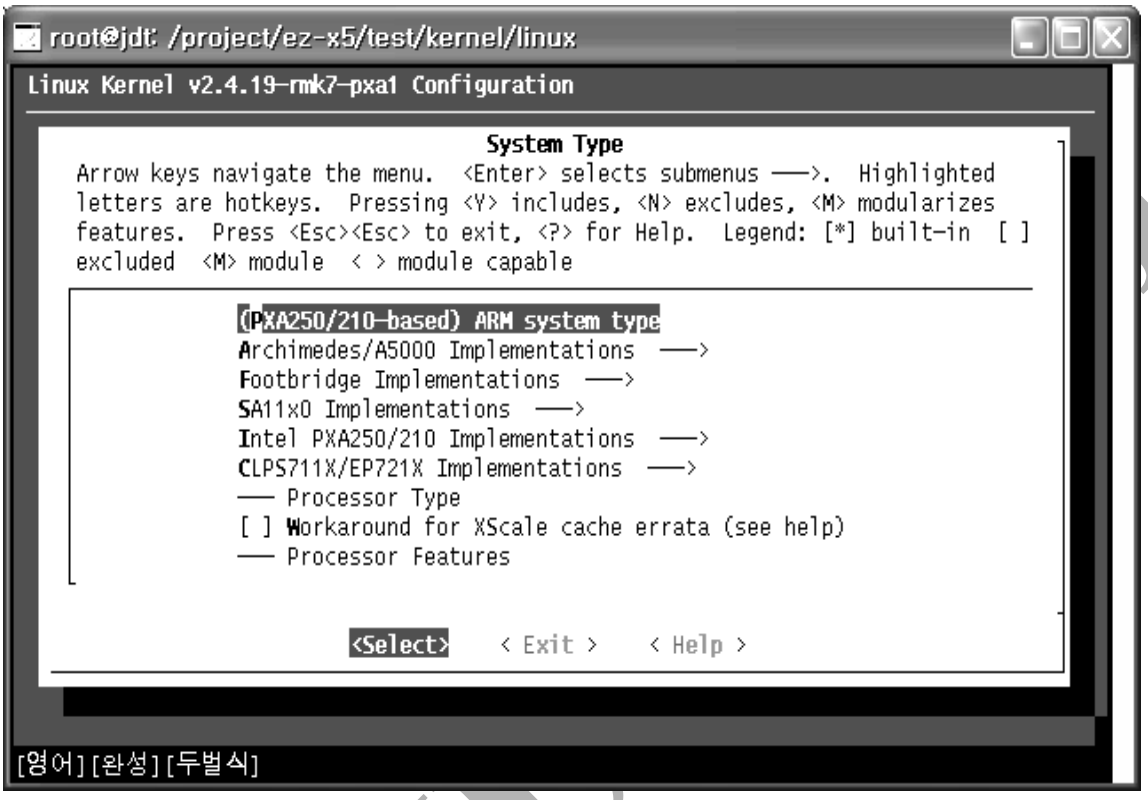
[Main Menu 화면]



[System Type 화면]

커널 패치, make ezboard_config, make oldconfig를 모두 하였다면 아래의 System Type 화면에서 (PXA250/210-based)) ARM system type 라는 것을 볼 수 있다.

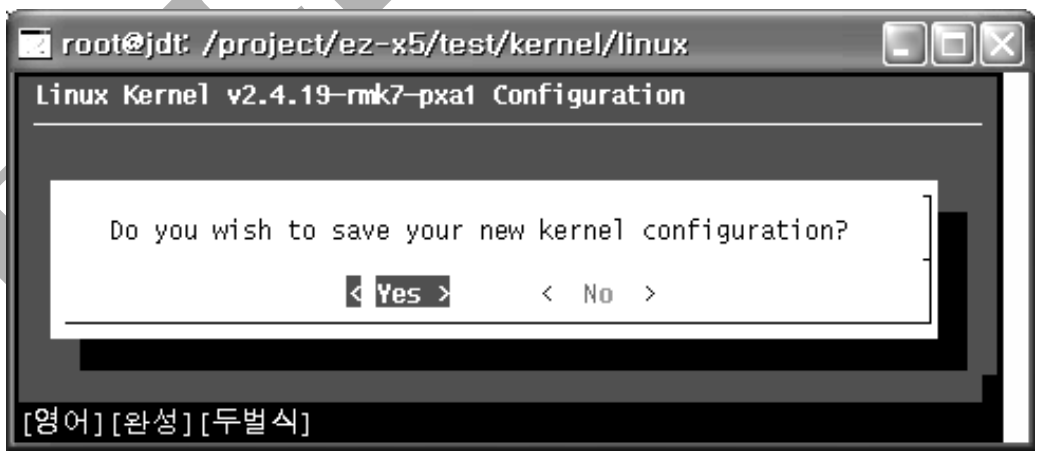
만약, 이와 같이 않다면 패치를 다시 실행하기 바란다.



[Exit 화면]

<Exit> 로 menuconfig 화면을 빠져 나올 때 저장을 할 것인지를 물어본다.

여기서 Yes를 선택하고 menuconfig 화면을 빠져 나온다.



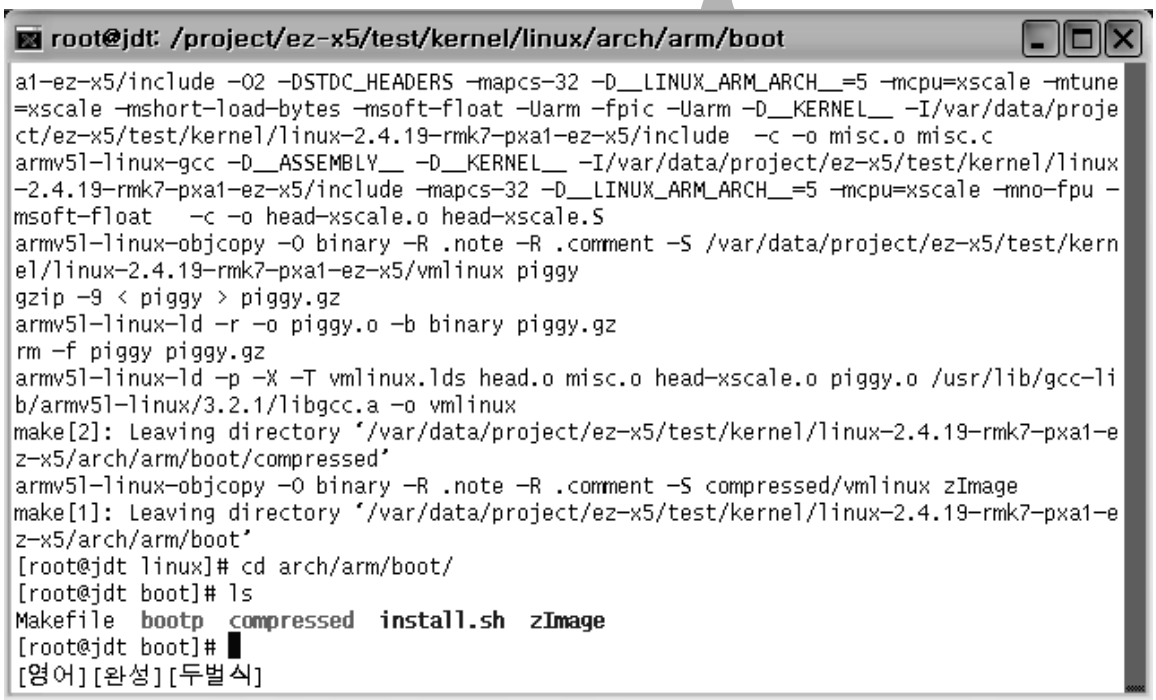
컴파일

그리고 다음과 같은 과정을 수행한다.

 # make dep
 # make clean
 # make zImage

정상적으로 컴파일이 끝났다면 커널 이미지가 생성이 된다.
타겟보드에서 사용하는 커널 이미지는 커널의 압축을 해제한 곳에 있는 vmlinuz가 아니다. **타겟보드에서 사용하는 커널 이미지는 ./arch/arm/boot/zImage의 형태로 존재한다.**

linux/arch/arm/boot 디렉토리 밑에 zImage라는 파일이 생긴다.



이 zImage 파일을 EZ-X5 보드에 다운로드하여 사용하면 된다.

FA COM

2. EZ-X5 커널 패치 파일 제작.

패치화일을 보아야하는 이유

자신의 보드를 만든 후에 커널을 어떻게 수정하여야 하는지는 매우 막막해진다. 이때 이미 있는 평가보드의 패치 파일을 보면 어떤 부분을 수정해야 하는지를 알 수 있다.
EZ-X5보드를 구매하신 분들은 EZ-X5보드 패치 파일을 보면서 자신의 보드에 맞는 커널을 패치 할 수가 있는 것이다.
패치 파일은 두개의 파일의 차이를 기록한 것이기 때문에 패치파일 문법만 알면 자동적으로 어떤 부분을 수정해야 하는지를 손 쉽게 알 수 있다.
이 문서에서는 패치 파일에 대한 문법은 표현하지 않겠다. KLDP.ORG와 같은 사이트를 방문하면 필요한 문서를 찾을 수 있을 것이다.

위의 커널 설치 및 패치를 수행하면서 EZ-X5보드에서 사용할 수 있는 커널이 만들어졌다. 이것은 단지 제공한 diff-2.4.19-rmk7-pxa1-ez-x5.gz 패치를 적용함으로써 가능하다.
만일 이것이 제공되지 않는다면 여러분은 어떻게 할 것인가?
다음의 내용들은 EZ-X5보드에 맞는 패치 커널을 만드는 방법에 대한 설명이다.

EZ-X5 보드 커널 패치 파일 작성

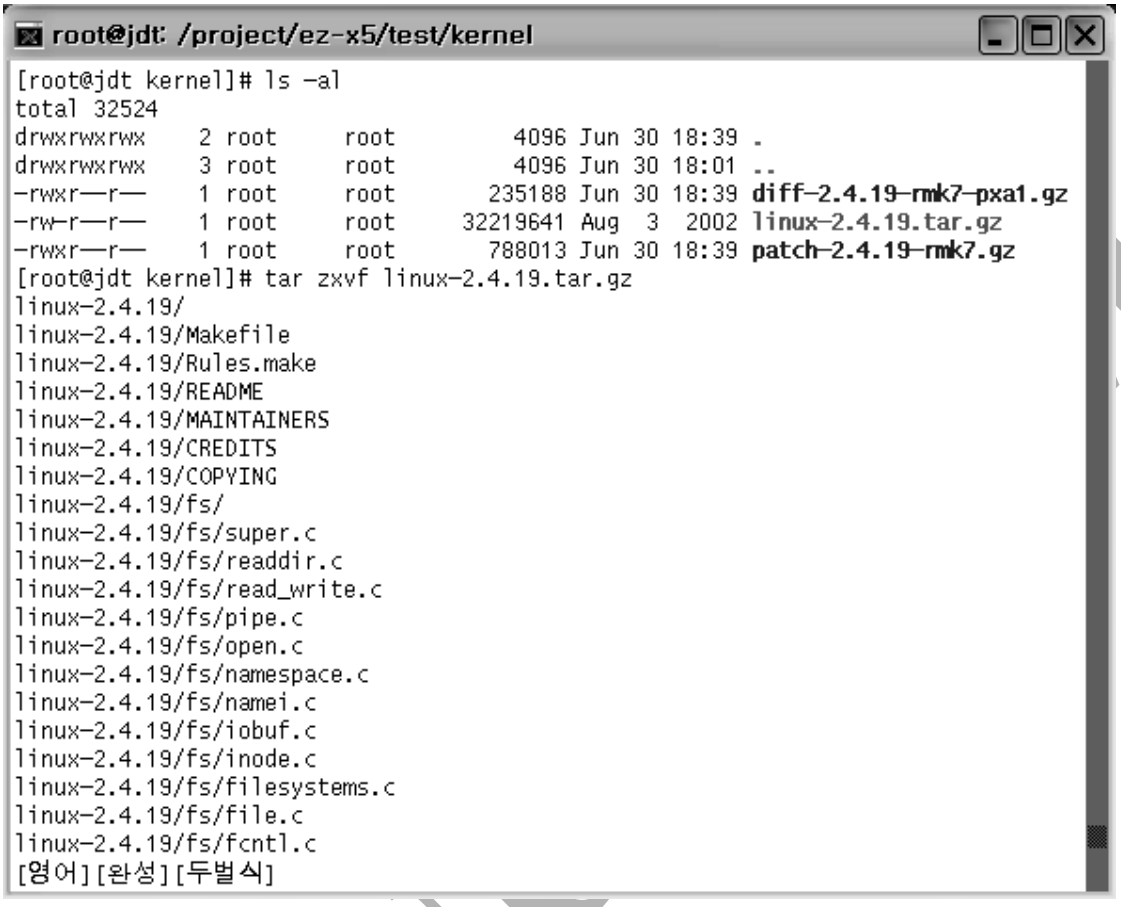
EZ-X5보드 커널 패치파일은 다음 순서에 입각하여 만든다. 이미 커널과 ARM용 패치파일을 구하는 방법에 대해서는 앞에서 설명 하였다. 다시 한번 확인 하자면 아래의 세가지 파일만 구하면 된다.

- **linux-2.4.19.tar.gz**
  ftp://ftp.kr.kernel.org/pub/linux/kernel/v2.4/linux-2.4.19.tar.gz
- **patch-2.4.19-rmk7.gz**
  ftp://ftp.arm.uk.linux.org/pub/linux/arm/kernel/v2.4/patch-2.4.19-rmk7.gz
- **diff-2.4.19-rmk7-pxa1.gz**
  ftp://ftp.arm.uk.linux.org/pub/linux/arm/people/nico/diff-2.4.19-rmk7-pxa1.gz

FALINUX.COM

- 받아온 커널의 압축을 풀자.

  # tar zxvf linux-2.4.19.tar.gz



  위의 명령어를 이용하여 커널의 압축을 풀게 되면 linux-2.4.19 라는 디렉토리가 생기면서 압축이 풀리게 된다.

- 커널에 패치를 적용시키자.

  커널에 패치를 적용시켜 보자. 패치를 적용하기 위하여 커널 압축을 풀어 놓은 linux-2.4.19 디렉토리로 이동 한 후에 커널 패치를 한다.

  패치할 것은 두가지 이다. 하나는 arm 패치이고 또 하나의 XScale 패치이다.

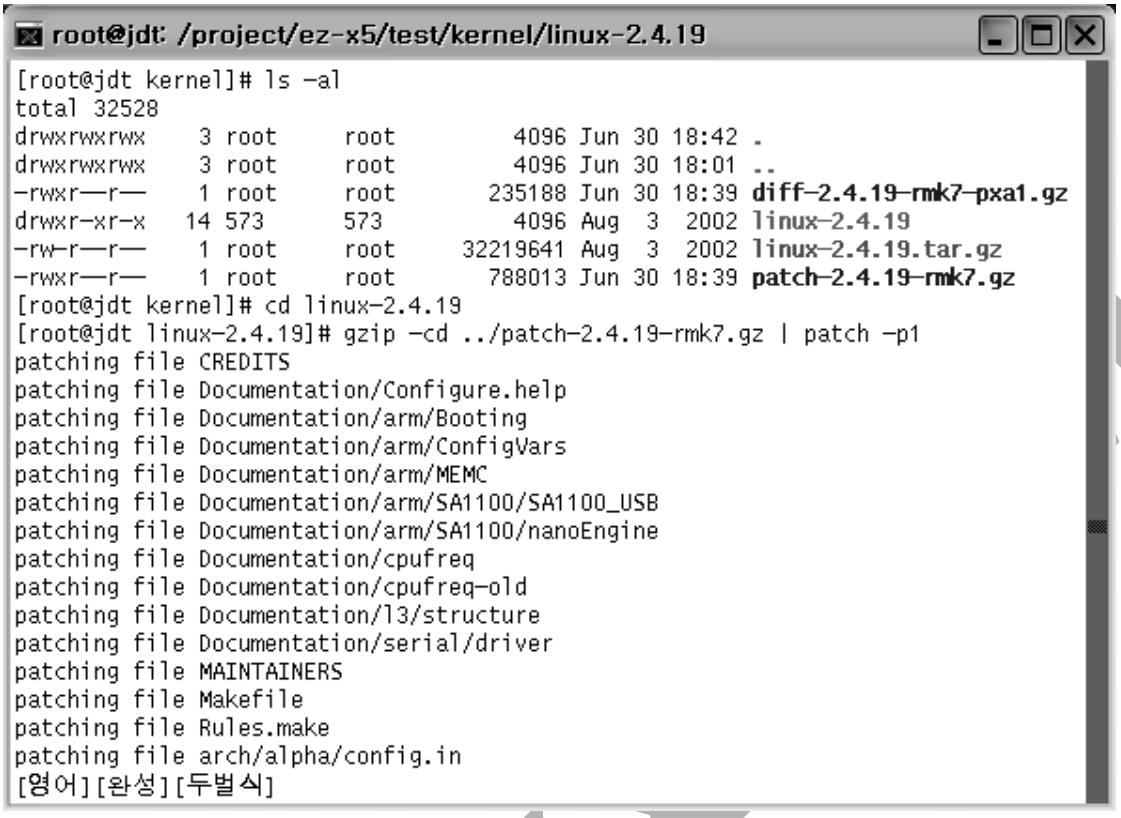
  # cd linux-2.4.19

  # gzip -cd ../patch-2.4.19-rmk7.gz | patch –p1

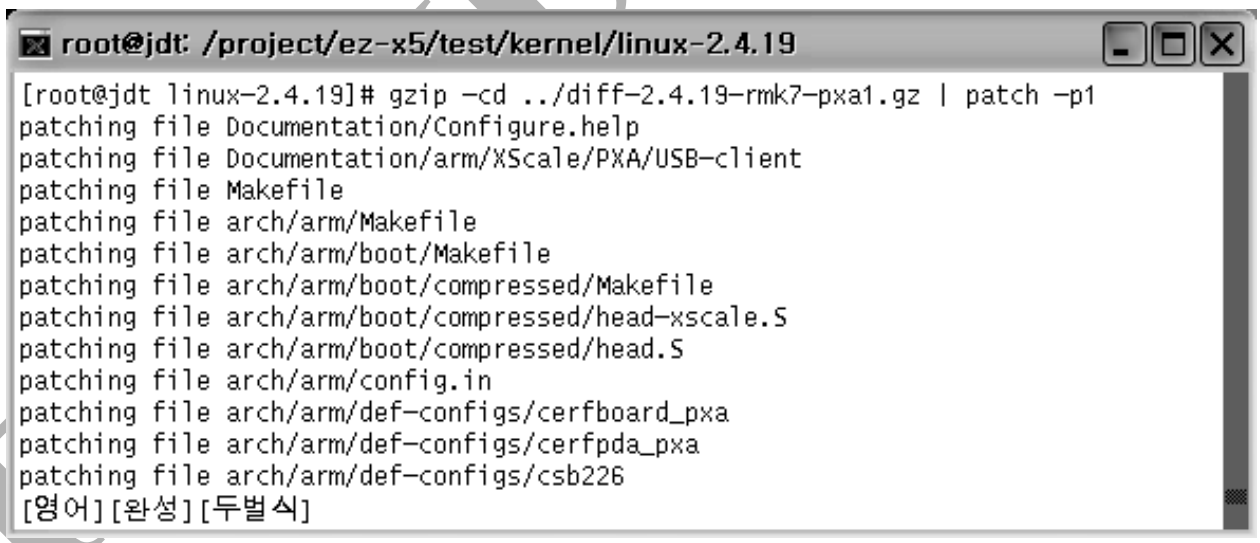
  # gzip -cd ../diff-2.4.19-rmk7-pxa1.gz | patch -p1

FALINUX

 # gzip -cd ../patch-2.4.19-rmk7.gz | patch –p1



 # gzip -cd ../diff-2.4.19-rmk7-pxa1.gz | patch -p1

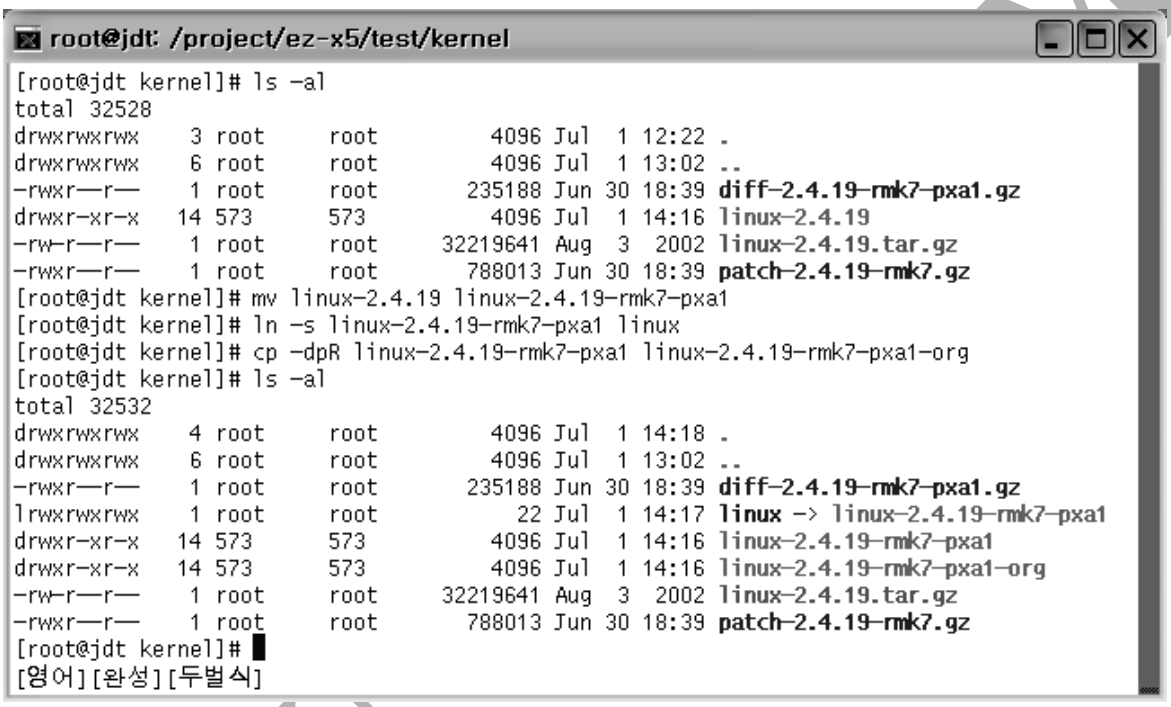


커널 압축을 푼 linux-2.4.19 디렉토리를 linux-2.4.18-rmk7-pxa1 이라는 디렉토리로 파일명을 바꾸고, 이 디렉토리를 linux로 링크하였다.
또한 패치 파일을 만들기 위해 이렇게 만들어 놓은 원본 커널을 하나 복사해둔다.

 # mv linux-2.4.19 linux-2.4.19-rmk7-pxa1

 # ln –s linux-2.4.18-arm linux

 # cp –dpR linux-2.4.18-arm linux-2.4.18-arm-org



- 작업 위치
  모든 작업은 링크가 되어 있는 linux 안에서 이루어 지는 작업이다.
  [현재 여기서 linux는 **linux-2.4.19-rmk7-pxa1** 디렉토리를 링크 걸어 놓았다.]

 # cd linux

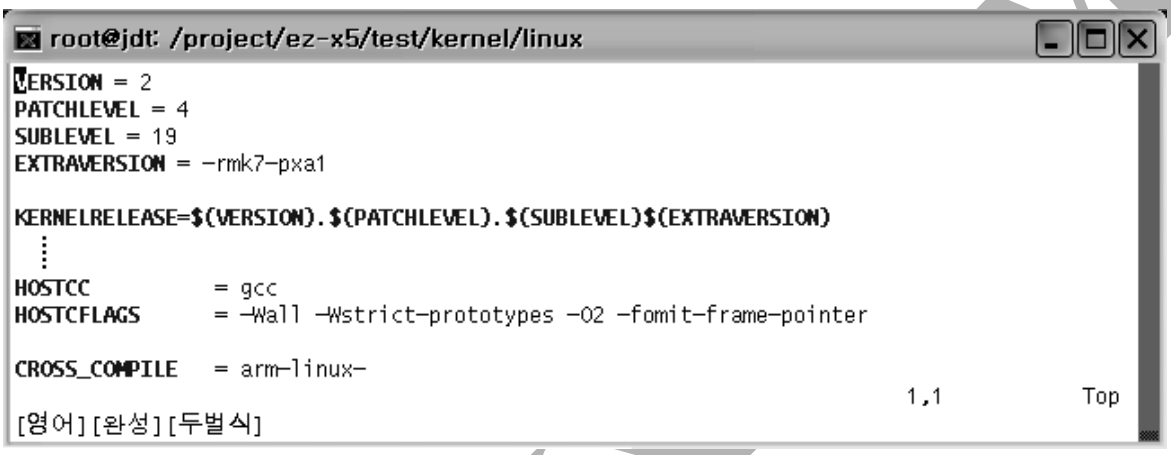
FALINUX.COM

EZ-X5 에 맞는 보드 패치

Makefile 수정

커널 메인디렉토리(linux)안에 있는 Makefile에서 버전을 고쳐 주는 것이다.
수정할 내용은 버전에 대한 수정과 크로스 컴파일러의 프리픽스 문자열을 수정한다.

[수정 전]



[수정 후]

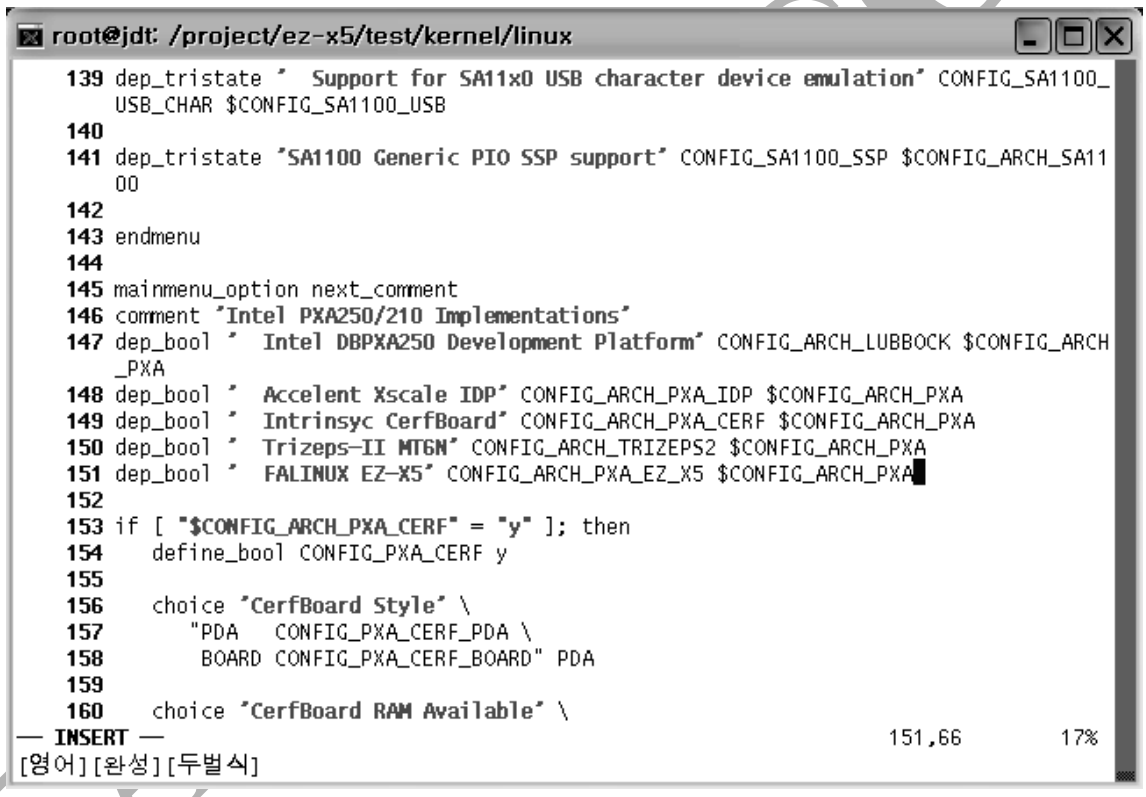
EXTRAVERSION = –rmk7 를 EXTRAVERSION = –rmk7-ez_x5 으로 수정한다.
CROSS_COMPILE = armv5l-linux- 로 수정한다.

./arch/arm/config.in 수정

파일에서 보드에 관련된 부분을 추가한다.
여기에 추가를 해주어야 configuration시에 EZ-X5에 관련된 부분을 선택 할 수가 있으며, 앞으로 CONFIG_ARCH_PXA_EZ_X5 사용하여 CONFIG시에 사용 할 수 있기 때문이다.

./arch/arm/config.in 파일 150라인쯤에 아래와 같이 소스파일에 추가해 준다.
[vi 명령에서 :set nu 하면 라인 번호가 나온다.]

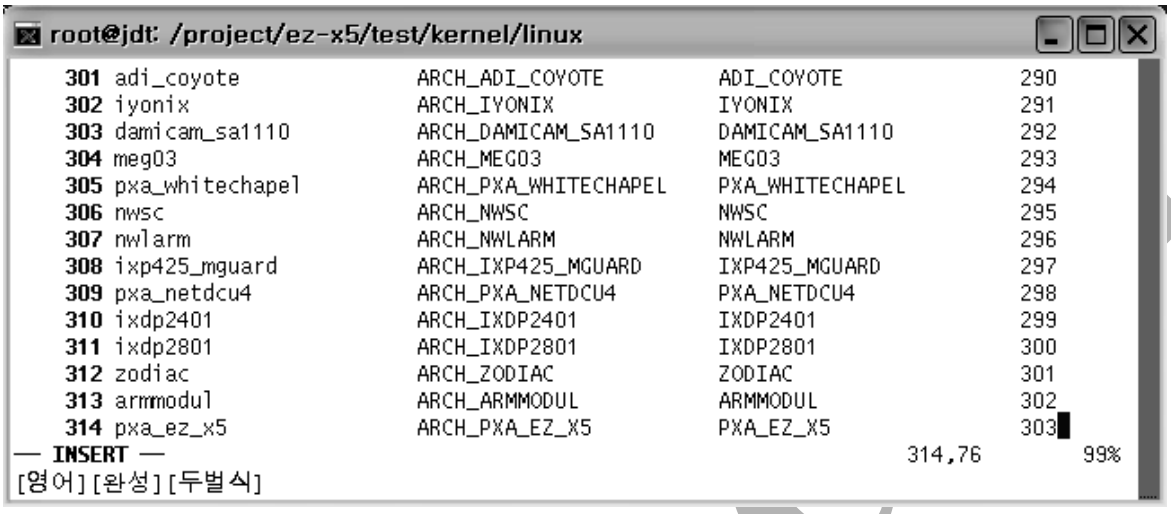
dep_bool ' FALINUX EZ-X5' CONFIG_ARCH_PXA_EZ_X5 $CONFIG_ARCH_PXA



./arch/arm/tools/mach-types 수정

./arch/arm/tools/mach-types 파일에 보드 타입을 추가해 준다.

pxa_ez_x5 ARCH_PXA_EZ_X5 PXA_EZ_X5 303



이 보드 타입은 원래 ARM사이트에 등록을 해서 번호를 부여 받아야 하는데, 등록하지 않았기 때문에 임의의 번호를 할당했다.

주의)

현재 작업을 하고 있는 커널에서는 이렇게 설정을 할 수 있을 지라도, 높은 버전의 커널이라면 이 번호가 할당 되어 있을 수가 있으므로, 또 다른 번호를 하던지, 이 번호의 보드를 덮어 쓰는 수 밖에 없을 것이다.

FALINUX.COM

./arch/arm/kernel/setup.c 수정

./arch/arm/kernel/setup.c 파일에서 메모리에 대한 것을 설정을 해준다.
33 라인에 정의 되어 있다.

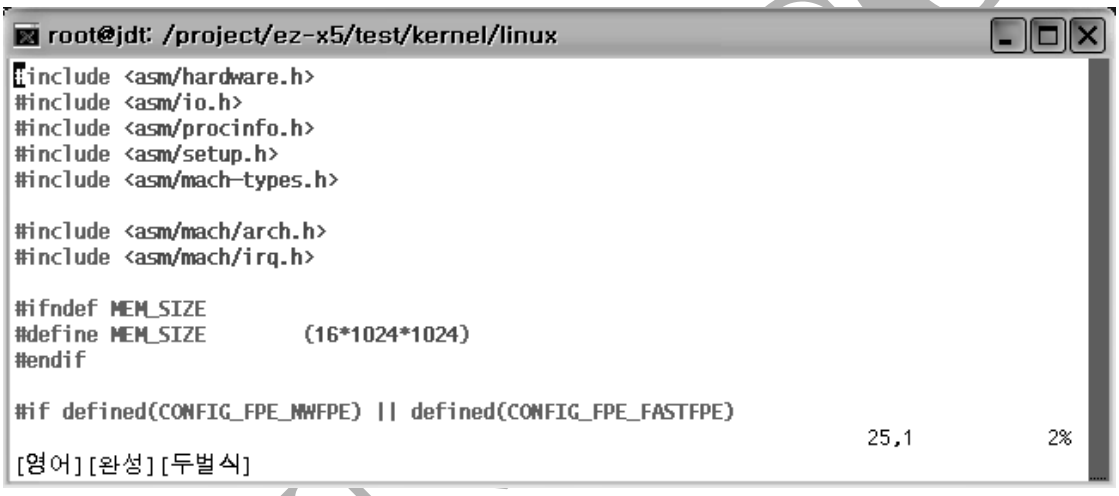
EZ-X5는 SDRAM이 64M이므로 16을 64로 바꾸어 준다.

#define MEM_SIZE (16*1024*1024)

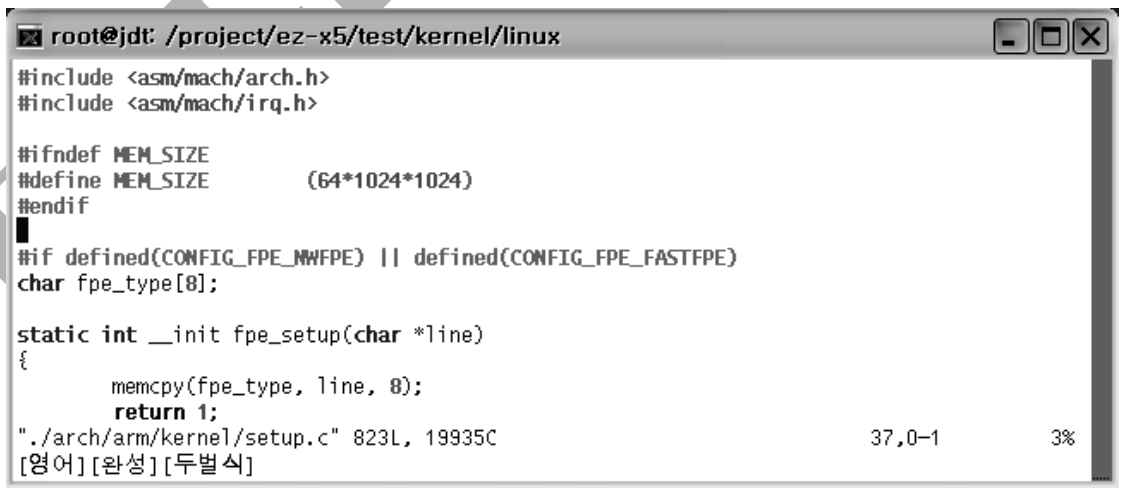
위의 것을 아래와 같이 수정한다.

#define MEM_SIZE (32*1024*1024)

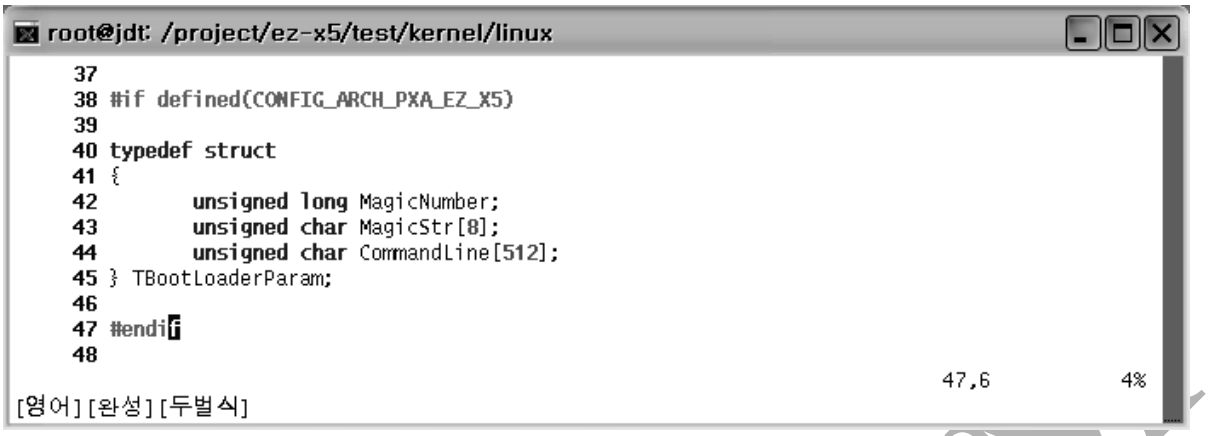
[수정 전]



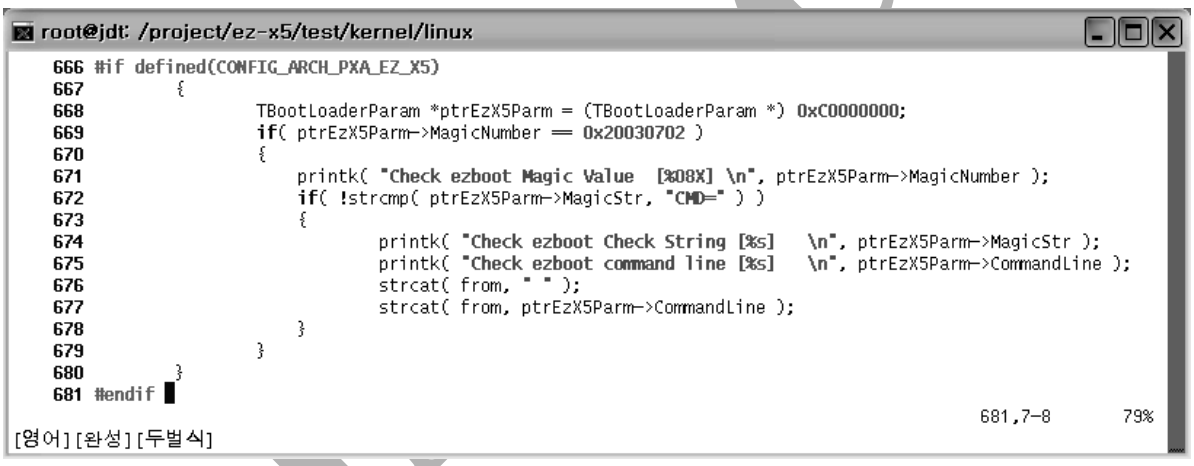
[수정 후]



이지부트에서 전달된 커널 컨맨드 문자열을 커널에 전달하기 위해서 다음 타입을 선언한다.



다음은 전달된 문자열을 전달하기 위한 루틴으로 **setup_arch** 함수에서 memcpy(saved_command_line, from, COMMAND_LINE_SIZE); 줄 앞에 다음을 추가 한다.



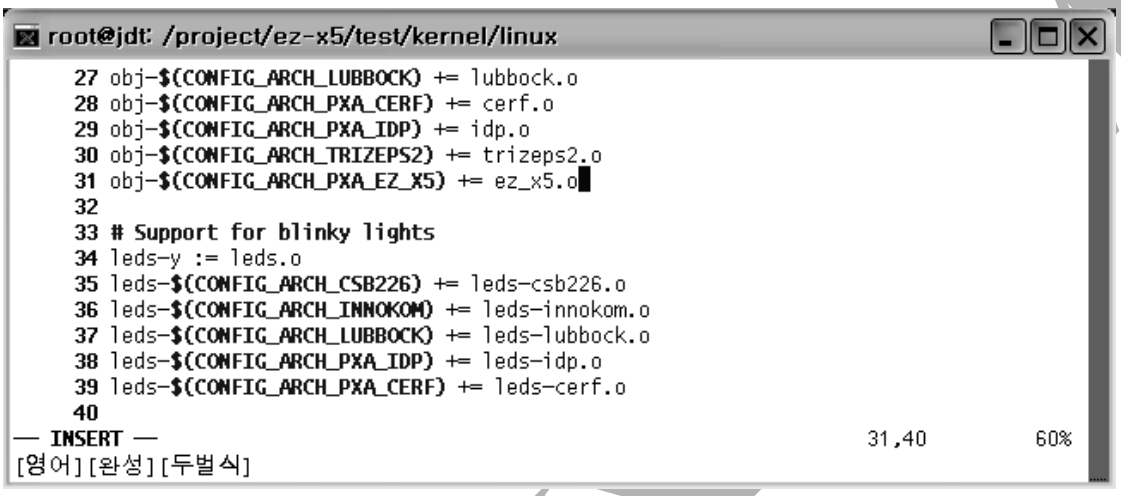
FALL COM

./ arch/arm/mach-pxa/Makefile 수정

./arch/arm/mach-pxa/Makefile 파일에서 EZ-X5보드에 관련된 부분을 처리해 준다.

31라인에 커널 컴파일 환경설정에서 이지보드를 선택했을 때 ez_x5.o를 생성 할 수 있게 하는 루틴을 추가 한다.

obj-$(CONFIG_ARCH_PXA_EZ_X5) += ez_x5.o



"./arch/arm/mach-pxa/ez_x5.c " 작성

이 파일에서 하는 일은 가상 메모리에 대한 할당 및 처리를 하는 부분과 시리얼에 관련된 부분 등 기본적으로 커널이 돌아 갈수 있는 부분을 설정해 주는 부분이다.

FAINUX

다음은 작성해야 할 **ez_x5.c** 의 소스 코드이다.

root@jdt: /project/ez-x5/test/kernel/linux/arch/arm/mach-pxa

```
 1 /*
 2  *  linux/arch/arm/mach-pxa/ez_x5.c
 3  *
 4  *  This program is free software; you can redistribute it and/or modify
 5  *  it under the terms of the GNU General Public License version 2 as
 6  *  published by the Free Software Foundation.
 7  */
 8 #include <linux/init.h>
 9 #include <linux/major.h>
10 #include <linux/fs.h>
11 #include <linux/interrupt.h>
12 #include <linux/sched.h>
13
14 #include <asm/types.h>
15 #include <asm/setup.h>
16 #include <asm/memory.h>
17 #include <asm/mach-types.h>
18 #include <asm/hardware.h>
19 #include <asm/irq.h>
20
21 #include <asm/mach/arch.h>
22 #include <asm/mach/map.h>
23 #include <asm/mach/irq.h>
24
25 #include <asm/io.h>
26 #include <asm/arch/irq.h>
27
28 #include "generic.h"
29
30 static void __init ez_x5_init_irq(void)
31 {
32         pxa_init_irq();
33 }
34
35 static int __init ez_x5_init(void)
36 {
37         /*
38          * All of the code that was here was SA1111 init code
39          * which we do not have.
40          */
41         return 0;
42 }
43
44 _initcall(ez_x5_init);
                                                    44,1          Top
[영어][완성][두벌식]
```

root@jdt: /project/ez-x5/test/kernel/linux/arch/arm/mach-pxa

```
 45
 46 static void __init
 47 fixup_ez_x5(struct machine_desc *desc, struct param_struct *params,
 48                 char **cmdline, struct meminfo *mi)
 49 {
 50         // SDRAM 을 선언하는 부분이다.    -- 64Mbyte
 51         SET_BANK (0, 0xa0000000, 64*1024*1024);
 52         mi->nr_banks = 1;
 53
 54         // 램디스크를 만든다.
 55         setup_ramdisk (1, 0, 0, 8192);
 56         // 램디스크의 이미지를 램디스크에 푼다.
 57         setup_initrd (__phys_to_virt(0xa0800000), 4*1024*1024);
 58         // 루트 디렉토리에 마운트 될 장치를 램 디스크로 설정한다.
 59         ROOT_DEV = MKDEV(RAMDISK_MAJOR,0);
 60 }
 61
 62 /*
 63  * IO map for the devices.
 64  */
 65 static struct map_desc ez_x5_io_desc[] __initdata = {
 66  /*   virtual     physical                length      domain     r  w  c  b */
 67         { 0xf1000000, 0x00000000  +0x400000, 0x00100000, DOMAIN_IO, 0, 1, 0, 0 },  // nCS0 CS8900 영역  -- slow RD/WR
 68         { 0xf1200000, PXA_CS1_PHYS+0x000000, 0x00100000, DOMAIN_IO, 0, 1, 0, 0 },  // nCS1 NAND-Flash   -- fast RD/WR
 69         { 0xf1300000, PXA_CS1_PHYS+0x400000, 0x00100000, DOMAIN_IO, 0, 1, 0, 0 },  // nCS1 MK712        -- fast RD/WR
 70
 71         { 0xf2000000, PXA_CS2_PHYS         , 0x01000000, DOMAIN_IO, 0, 1, 0, 0 },  // nCS2 영역  16MByte Area
 72         { 0xf3000000, PXA_CS3_PHYS         , 0x01000000, DOMAIN_IO, 0, 1, 0, 0 },  // nCS3 영역  16MByte
 73         { 0xf4000000, PXA_CS4_PHYS         , 0x01000000, DOMAIN_IO, 0, 1, 0, 0 },  // nCS4 영역  16Mbyte
 74         { 0xf5000000, PXA_CS5_PHYS         , 0x01000000, DOMAIN_IO, 0, 1, 0, 0 },  // nCS5 영역  16Mbute
 75         LAST_DESC
 76 };
 77
 78 static void __init ez_x5_map_io(void)
 79 {
 80         pxa_map_io();
 81         iotable_init(ez_x5_io_desc);
 82
 83         // Hardware Init ==============
 84
 85         // GPSR0 = ( GPIO_bit(2) );
 86         // GPCR0 = ( GPIO_bit(2) );
 87 }
 88
                                                                88,0-1        84%
[영어][완성][두벌식]
```

root@jdt: /project/ez-x5/test/kernel/linux/arch/arm/mach-pxa

```
 89 MACHINE_START(PXA_EZ_X5, "WWW.FALINUX.COM EZ-X5 for PXA255 Board")
 90         MAINTAINER("J.D&T Co.,Ltd.")
 91         BOOT_MEM(0xa0000000, 0x40000000, io_p2v(0x40000000))
 92         FIXUP(fixup_ez_x5)
 93         MAPIO(ez_x5_map_io)
 94         INITIRQ(ez_x5_init_irq)
 95 MACHINE_END
 96
                                                                96,0-1        Bot
[영어][완성][두벌식]
```

이 **ez_x5.c** 소스는 제공한 CD의 /sw/kernel/ez_x5.c 에 있으며, 위의 내용을 작성하지 않고 복사하여 사용해도 된다.

[주의]

CD에 제공되는 ez_x5.c 에는 ez_x5_map_io 함수에 다음이 추가 되었다.

```
// Hardware Init ==================
set_GPIO_mode(GPIO34_FFRXD_MD );
set_GPIO_mode(GPIO35_FFCTS_MD );
set_GPIO_mode(GPIO36_FFDCD_MD );
set_GPIO_mode(GPIO37_FFDSR_MD );
set_GPIO_mode(GPIO38_FFRI_MD    );
set_GPIO_mode(GPIO39_FFTXD_MD );
set_GPIO_mode(GPIO40_FFDTR_MD );
set_GPIO_mode(GPIO41_FFRTS_MD );
set_GPIO_mode(GPIO42_BTRXD_MD );
set_GPIO_mode(GPIO43_BTTXD_MD );
set_GPIO_mode(GPIO44_BTCTS_MD );
set_GPIO_mode(GPIO45_BTRTS_MD );
```

이것은 EZ-X5보드가 FFUART, BTUART 시리얼 포트를 사용 가능하게 만들었기 때문입니다. 하지만 이 패치 과정의 소스에는 굳이 포함시키지 않는 이유는 주 목적이 기본 사용에 대한 것을 설명하기 위한 것입니다.

만약 원래 보드 사양 대로 사용 하시려면 저희 회사에서 제공한 패치 화일을 이용하실 것을 권유합니다.

FALINUX.COM

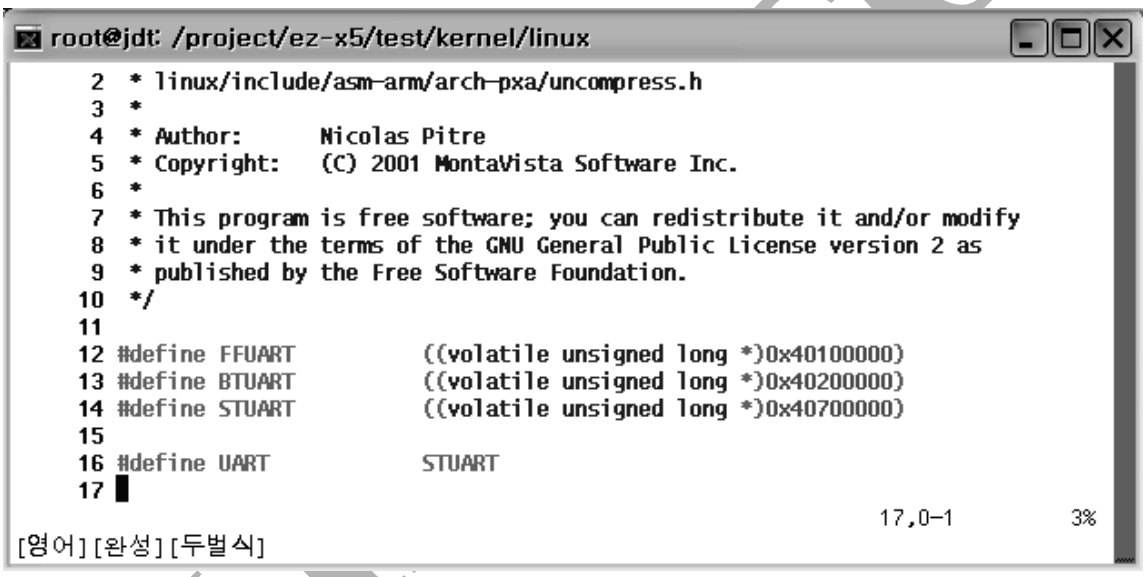
./include/asm-arm/arch-pxa/uncompress.h 수정

기존의 XScale 커널은 FFUART 를 콘솔용으로 내정하고 있다. 이에 반해서 EX-X5 보드는 콘솔을 STUART를 콘솔로 내정하고 있다.

커널 압축 해제 시 표출되는 메세지를 STUART로 보내려면

include/asm-arm/arch-pxa/uncompress.h 수정를 수정하여야 한다.

#define UART FFUART 로 선언된 부분을
#define UART STUART 로 바꾸어야 한다.



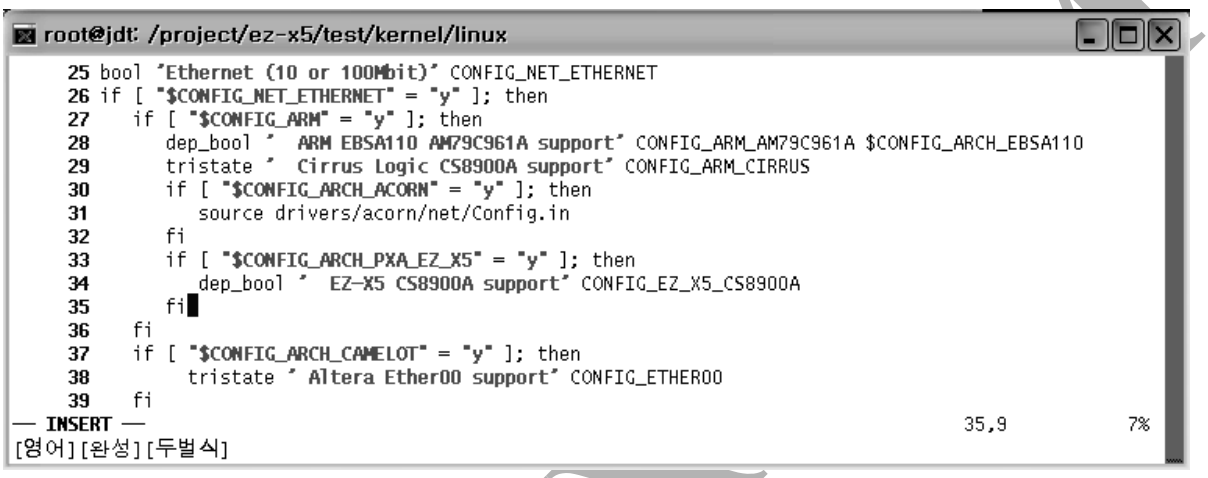
이후에 더 설명이 나오겠지만 커널의 기본 콘솔역시 STUART로 바꾸어야 하는데 이 부분은 커널 커맨드를 설정함으로써 가능하다. 커널 커맨드는 커널 옵션을 설정해서 바꾸거나 이지부트에서 설정하여야 한다.

FALINUX.COM

./drivers/net/Config.in 수정

EZ-X5보드에 CS8900A(이더넷 콘트롤러)를 사용하기 위해 Config부분에 다음을 추가해 준다.

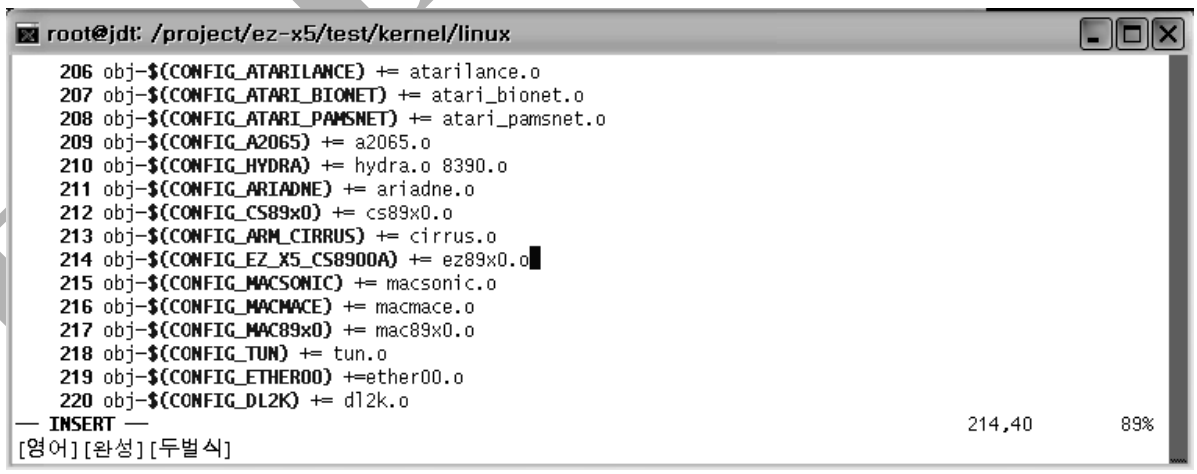
```
if [ "$CONFIG_ARCH_PXA_EZ_X5" = "y" ]; then
        dep_bool '    EZ-X5 CS8900A support' CONFIG_EZ_X5_CS8900A
fi
```



./drivers/net/Makefile 수정

./drivers/net/Makefile 파일에서도 다음을 추가를 해주어야 컴파일을 할 수 있다.

obj-$(CONFIG_EZ_X5_CS8900A) += ez89x0.o

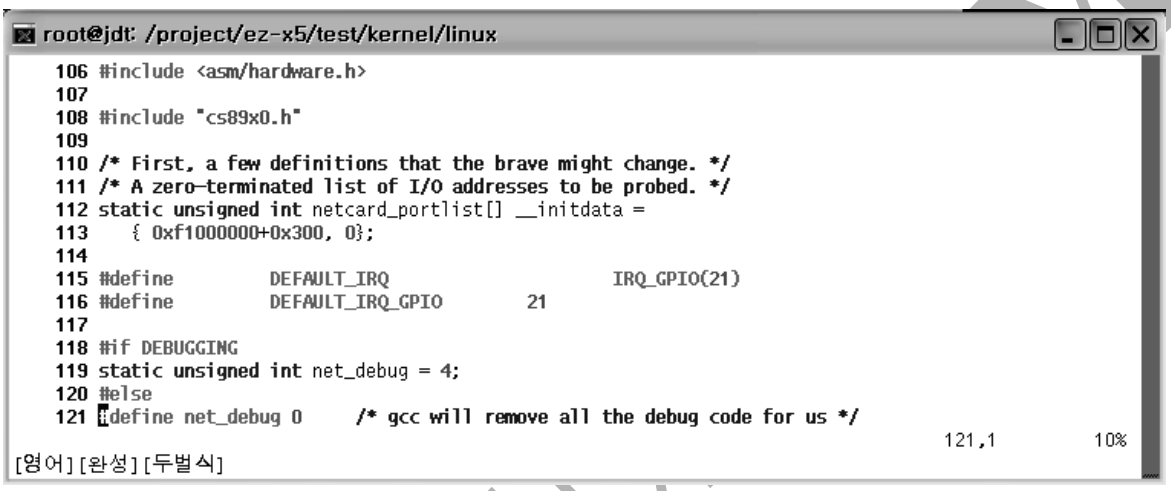


./drivers/net/ez89x0.c 작성

이 파일을 직접 작성하기에는 많은 시간이 소요된다. 따라서 제공한 CD에서 이 파일을 복사하여 사용한다.

CD의 디렉토리 /sw/kernel/ez89x0.c 을 가져온다.

다음은 하드웨어와 매칭되는 부분에 대한 소스 설명이다.



이 부분에서 DEFAULT_IRQ 와 DEFAULT_GPIO는 EZ-X5 의 하드웨어적인 IRQ와 연결되어 있는 것을 맞추는 부분이다.

또한 __initdata 에서 표현하고 있는 0xf1000000는 물리적인 주소 0x00400000을 가상주소로 맵핑한 주소이다.

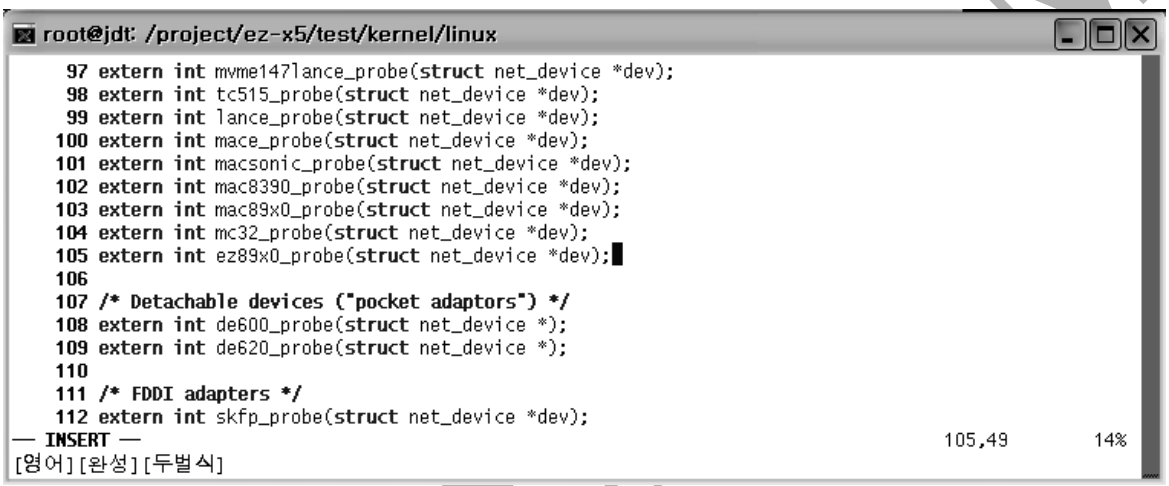
이 맵핑 정보는 arch/arm/mach-pxa/ez_x5.c 에 정의되어 있다.

FALINUX.COM

./drivers/net/Space.c 수정

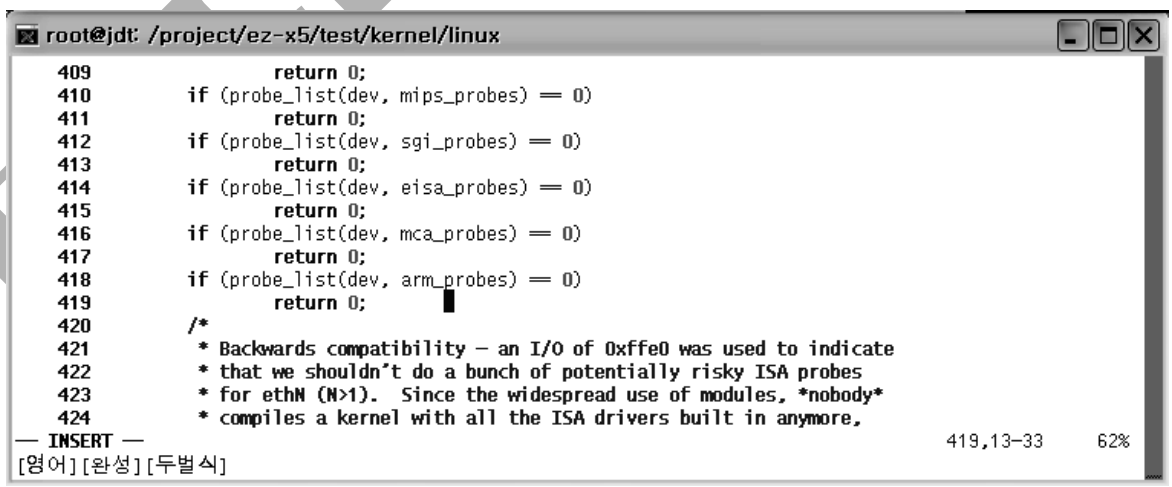
이제 ez89x0의 함수를 커널에 포함 시킨다고 했기 때문에 ez89x0_probe 함수를 호출해 주는 부분이 있어야 할 것이다. 모듈로 할 경우 insmod가 이 역할을 대신해 주는데, 커널에 포함시키기 위해서는 이 역할을 해줄 부분을 추가해 주어야 한다. 아래의 그림과 같이 추가를 한다.

extern int ez89x0_probe(struct net_device *dev); 선언 추가



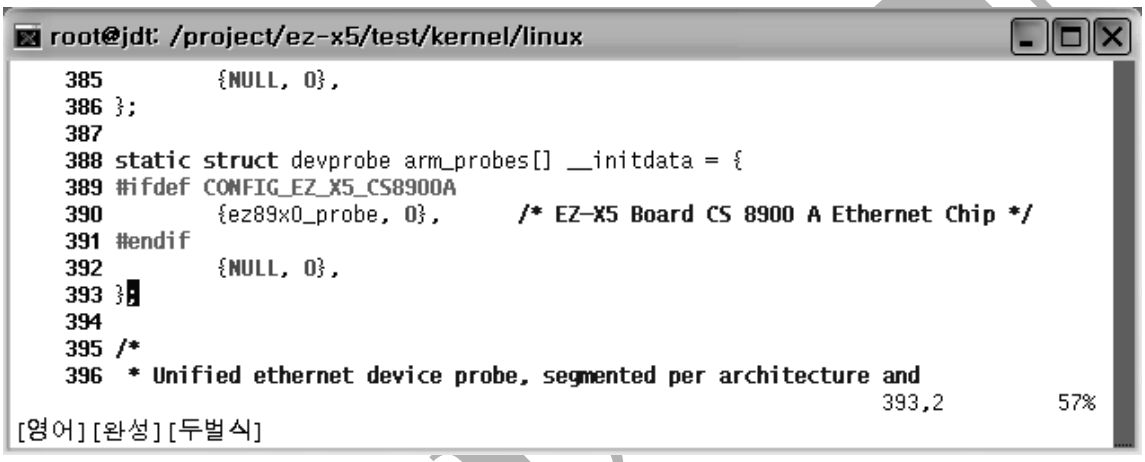
ethif_probe 함수내에

if (probe_list(dev, arm_probes) == 0) return 0; 을 추가한다.



static int __init **ethif_probe**(struct net_device *dev) 함수위에 __initdata 함수를 선언하는 마지막 부분에 다음과 같이 추가 한다.

```
static struct devprobe arm_probes[] __initdata = {
#ifdef CONFIG_EZ_X5_CS8900A
        {ez89x0_probe, 0},          /* EZ-X5 Board CS 8900 A Ethernet Chip */
#endif
        {NULL, 0},
};
```



./drivers/char/mk712.c 수정

EZ-X5는 터치 시스템으로 MK712칩을 내장하고 있다. 이 칩용 드라이버 소스는 크루소 용으로 이미 만들어진 것이 있는데 이를 수정해서 사용한다. 그러나 너무 많은 것을 수정해야 하므로 CD에 제공된 것을 이용한다.

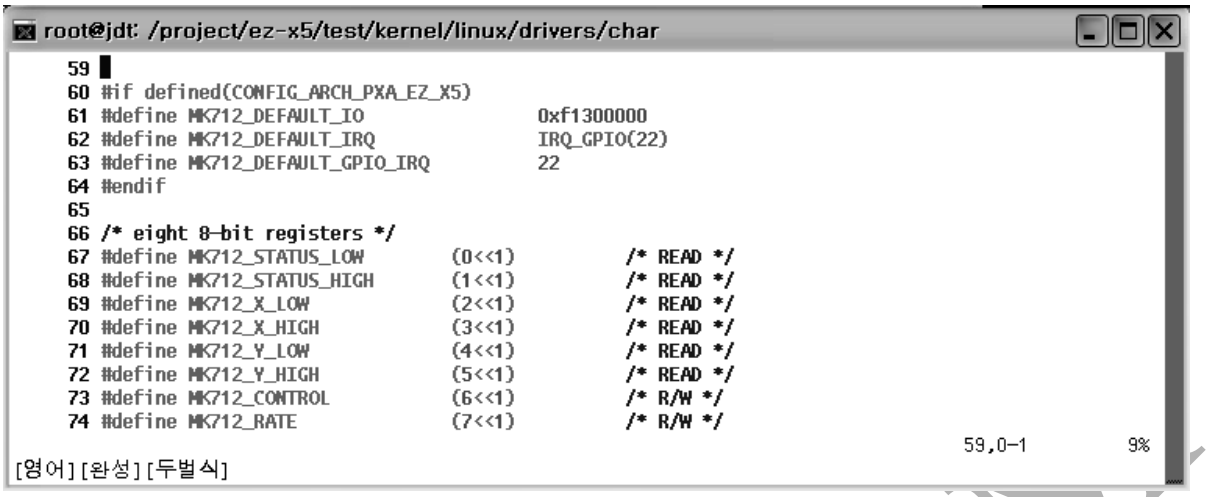
CD의 디렉토리 /sw/kernel/mk712.c 을 driver/char/에 복사해 넣는다.

MK712 터치 컨트롤러는 nCS1 번인 물리적인 주소에 연결되어 있다. 또한 인터럽트는 GPIO 22 번 핀에 연결되어 있다.

PXA_CS1_PHYS+0x400000의 주소는 가상주소 0xf1300000로 맵핑시키고 있다.
이에 대한 정보는 arch/arm/mach-pxa/ez_x5.c 에 있다.

이 소스는 이에 대한 주소와 인터럽트를 설정하고 있는 부분이다.

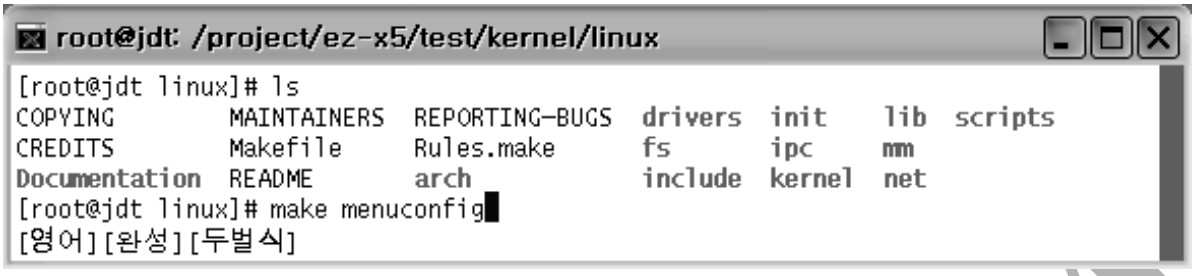
FALINUX.COM


root@jdt: /project/ez-x5/test/kernel/linux/drivers/char
59
60 #if defined(CONFIG_ARCH_PXA_EZ_X5)
61 #define MK712_DEFAULT_IO 0xf1300000
62 #define MK712_DEFAULT_IRQ IRQ_GPIO(22)
63 #define MK712_DEFAULT_GPIO_IRQ 22
64 #endif
65
66 /* eight 8-bit registers */
67 #define MK712_STATUS_LOW (0<<1) /* READ */
68 #define MK712_STATUS_HIGH (1<<1) /* READ */
69 #define MK712_X_LOW (2<<1) /* READ */
70 #define MK712_X_HIGH (3<<1) /* READ */
71 #define MK712_Y_LOW (4<<1) /* READ */
72 #define MK712_Y_HIGH (5<<1) /* READ */
73 #define MK712_CONTROL (6<<1) /* R/W */
74 #define MK712_RATE (7<<1) /* R/W */
59,0-1 9%
[영어][완성][두벌식]

FALINUX.COM

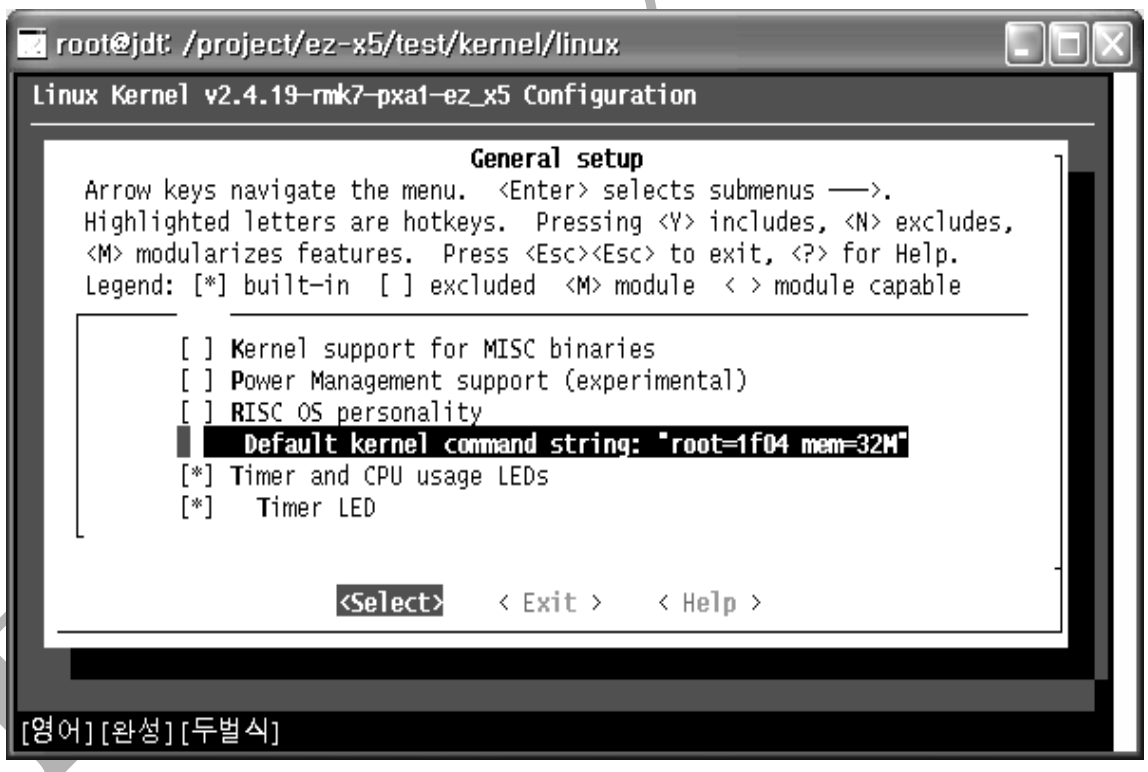
모든 내용을 모두 수정 및 추가 하였다면 커널 컴파일 옵션을 설정한다.

 # make menuconfig



를 수행하고, 제 7장의 커널 컴파일 옵션을 참조하여, 환경 설정을 한다.
만약, 커널 옵션을 잘못 설정하면 자신이 바꾼 부분이 활성화가 되지 않는 경우가 있으니 커널 컴파일 옵션 부분을 면밀히 검토하기 바란다.

[참고]
커널 컴파일 옵션 중 다음을 제대로 설정하지 않아서 커널 부팅 시 에러가 발생하는 경우가 있다.



만약 부트로더에서 커널 커맨드를 주지 않는 경우라면 다음과 같이 바꾸어 주어야 한다.

General Setup 에서

Default kernel command string: “root=1f04 mem=32M” 를

Default kernel command string: **“keepinitrd console=ttyS02,115200 root=/dev/ramdisk,rw”**

만약 부트로더에서 커널 커맨드를 주는 경우라면 꼭 스페이스를 넣는다.

General Setup 에서

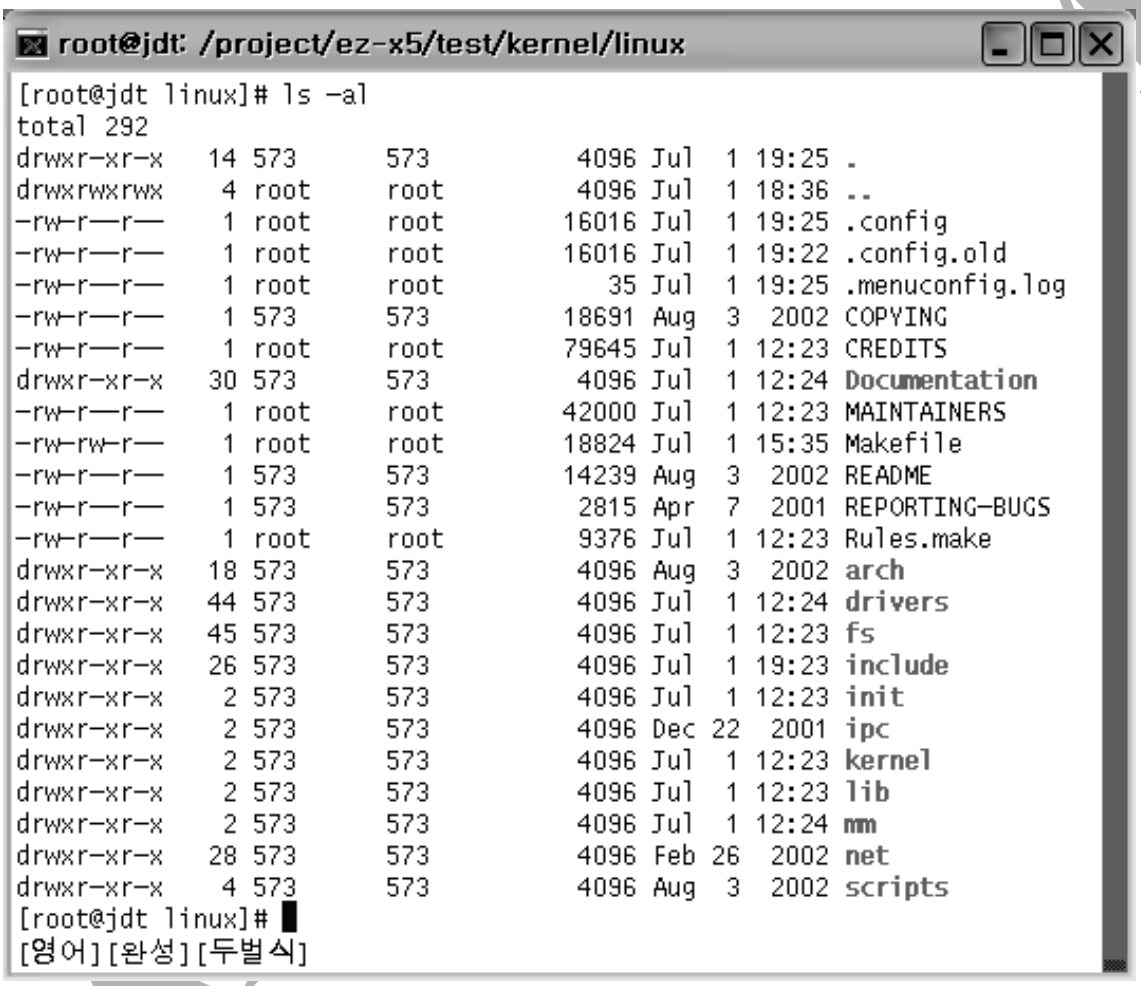
Default kernel command string: “root=1f04 mem=32M” 를

Default kernel command string: **“ ”**

이렇게 만들어진 환경설정은 다음의 파일을 생성한다.

 # ls –al

명령으로 보면 .config 라는 파일이 생성된 것을 볼 수 있다.



FA

이제 이 .config 파일을 ./arch/arm/def_config/ez-x5 로 복사하고 난 후에 커널을 컴파일 한다.

 # cp .config arch/arm/def-config/ez-x5

 # make ez-x5_config

 # make oldconfig

 # make dep

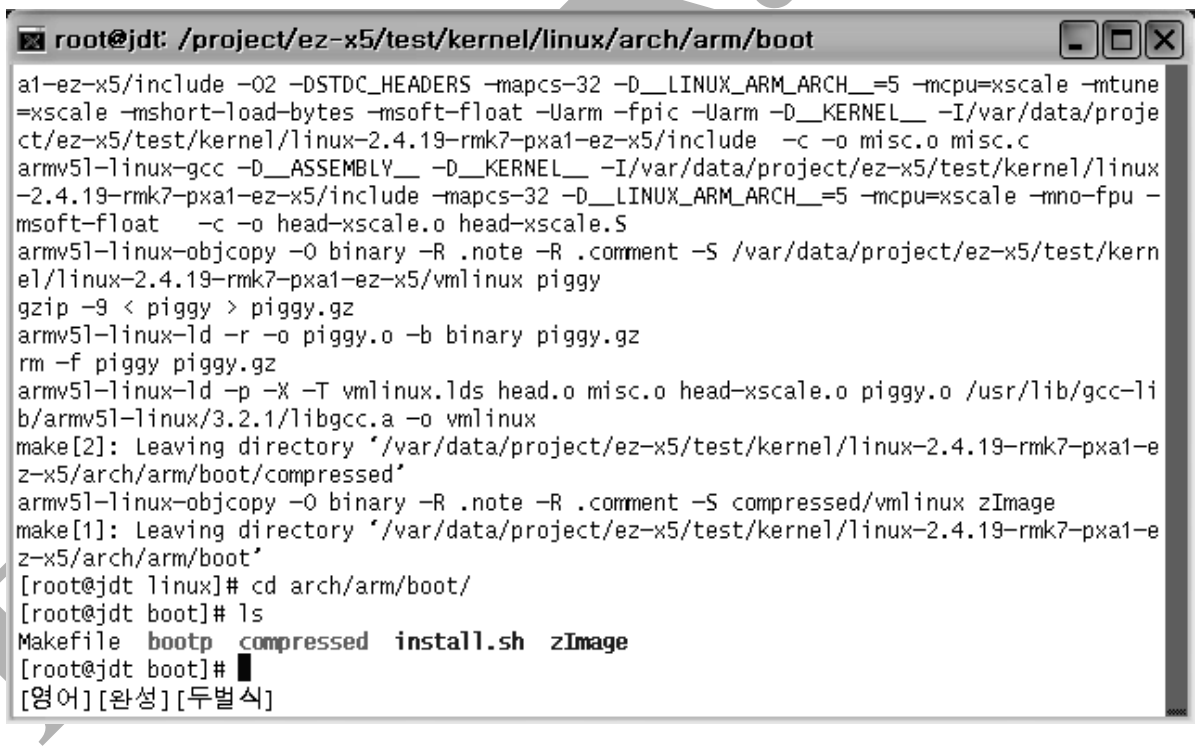
 # make clean

 # make zImage

모듈작성

커널 옵션에서 모듈(M)로 선택한 것을 컴파일 하기 위해 다음과 같은 명령을 한다. 이 모듈은 램디스크의 /lib/modules에 들어갈 내용이다. 따라서 이 모듈작성 부분은 제 8장의 램디스크 이미지 만들기에서 /lib/modules에 복사를 하면된다.

모든 작업이 정상적으로 수행되었다면 **./arch/arm/boot/zImage** 파일이 생성된 것을 볼 수 있다. 이 커널 이미지 파일을 이지보드로 다운로드하면 된다.



[참고]
EZ-X보드 패치파일을 만드는 방법과 모듈작성에 대하여 알아보자.

EZ-X5*보드 패치파일 만들기*

diff 명령을 사용하여 EZ-X5보드용 패치파일을 만든다.

만드는 방법은 다음과 같다.

패치 파일을 만들기 위해 복사해둔 원본 커널로 들어간다.

```
# cd linux-2.4.19-rmk7-pxa1-org
# make distclean
```

패치한 커널로 들어간다.

```
# cd linux-2.4.19-rmk7-pxa1
# make distclean
```

이제 패치파일을 만들자.

```
# diff –urN linux-2.4.19-rmk7-pxa1-org/ linux-2.4.19-rmk7-pxa1/ > diff-2.4.19-rmk7-pxa1-ez-x5
# gzip diff-2.4.19-rmk7-pxa1-ez-x5
```

패치파일을 다 만들었으면 패치파일을 만들기 위해 지웠던 내용을 다시 커널 컴파일을 한다.

```
# cd linux
# make ez-x5_config
# make oldconfig
# make menuconfig
# make dep
# make clean
# make zImage
```

FALINUX.COM

모듈작성

make modules
INSTALL_MOD_PATH=<자신이 만든 램디스크 디렉토리> make modules_install

<예>
INSTALL_MOD_PATH=/ez/ramdisk/ make modules_install

이렇게 해주면 자신이 만든 커널 모듈을 램디스크에 넣을 수 있다.

FALINUX.COM