```
#
#  ezboot 메이크 화일
#
#  이 메이크 화일은 등록된 디렉토리에 있는 모든 것을 make 수행한다.
#

#
# 이 화일에 선언된 모든 변수를 서브 메이크 화일에서 사용할수 있게 한다.
#
.EXPORT_ALL_VARIABLES:

#
# 컴파일 할 대상이 있는 디렉토리를 적은다.
#

DIRS =  start main
TOPDIR  := $(shell if [ "$$PWD" != "" ]; then echo $$PWD; else pwd; fi)
TINY_DIR = eztiny

#
# 컴파일 하기 위한 기본 환경 내용이 들어 있다.
#

CC = armv5l-linux-gcc
LD = armv5l-linux-ld
OC = armv5l-linux-objcopy

INCLUDES = -I. -I$(TOPDIR)/include

CFLAGS    = -nostdinc $(INCLUDES)
CFLAGS   += -Wall -Wstrict-prototypes -Wno-trigraphs -O2
CFLAGS   += -fno-strict-aliasing -fno-common -pipe -mapcs-32
CFLAGS   += -march=armv5 -Wa,-mxscale  -mtune=strongarm  -mshort-load-bytes -msoft-float -fno-builtin

START_LDFLAGS  = -p -X -T ./start-ld-script
MAIN_LDFLAGS   = -static -nostdlib -nostartfiles -nodefaultlibs -p -X -T ./main-ld-script

OCFLAGS = -O binary -R .note -R .comment -S

BOOT_IMAGE = ezboot.x5
TINY_IMAGE = eztiny.x5


#
# ezboot 이미지를 만든다.
#
all:
        for i in $(DIRS) ; do make -C $$i || exit $? ; done
        dd if=start/start_org of=image/$(BOOT_IMAGE) bs=1k conv=sync
        dd if=main/main_org of=image/$(BOOT_IMAGE) bs=1k seek=2

#       cp image/$(BOOT_IMAGE) /tftpboot/$(BOOT_IMAGE)
#       cp image/$(BOOT_IMAGE) /nfsfg/$(BOOT_IMAGE)
#       chmod 777 /nfsfg/$(BOOT_IMAGE)

tiny:
        make -C $(TINY_DIR)
        dd if=$(TINY_DIR)/$(TINY_IMAGE) of=image/$(TINY_IMAGE) bs=128 conv=sync
        cp image/$(TINY_IMAGE) /tftpboot/$(TINY_IMAGE)
#       cp image/$(TINY_IMAGE) /nfsfg/$(TINY_IMAGE)

#
# 쓸데없는 화일을 지운다.
#
clean:
        for i in $(DIRS) ; do make -C $$i clean; done
        rm -f image/$(BOOT_IMAGE)


tiny_clean:
        make -C $(TINY_DIR) clean
        rm -f image/$(TINY_IMAGE)
```

```
#
# 쏘스 참조를 자동으로 만든다.
#
dep:
	for i in $(DIRS) ; do make -C $$i; done

tiny_dep:
	make -C $(TINY_DIR) dep



--------------------------------------------------------------------------------




#
#  초기화 관련 ezBoot 메이크 화일
#


SRCS       = start.S gpio.S memory.S
OBJS       = $(SRCS:.S=.o)

TARGET     = start_org
PRE_TARGET = start-elf32

%.o:%.c
	@echo "Compiling $< ..."
	$(CC) -c $(CFLAGS) -o $@ $<

#
# Compilation target for C++ files
#
%.o:%.S
	@echo "Assembler compiling $< ..."
	$(CC) -c $(CFLAGS)  -o $@ $<


all : $(PRE_TARGET)
	$(OC) $(OCFLAGS)  $(PRE_TARGET) $(TARGET)


$(PRE_TARGET) : $(OBJS)
	$(LD) $(START_LDFLAGS) -o $@  $(OBJS)

dep :
	$(CC) -M $(INCLUDES) $(SRCS) > .depend

clean:
	rm -f *.o
	rm -f *.s
	rm -f $(PRE_TARGET)
	rm -f $(TARGET)

distclean: clean
	rm -rf .depend

new :
	$(MAKE) clean
	$(MAKE)

.depend: dep

ifeq (.depend,$(wildcard .depend))
include .depend
endif
```

```
-------------------------------------------------------------------------------

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

        . = ALIGN(4);
        .text : { *(.text) }

        . = ALIGN(4);
        .rodata : { *(.rodata) }

        . = ALIGN(4);
        .data : { *(.data) }

        . = ALIGN(4);
        .got : { *(.got) }

        . = ALIGN(4);
        .bss : { *(.bss) }
}

-------------------------------------------------------------------------------

/*=============================================================================================

  화일명 : start.S
  설  명 : 리셋시 호출되는 루틴

  작성자 : (주)제이닷디앤티  푸지
  작성일 : 2003년 4월 21일

  비  고 :

=============================================================================================*/

#include        <ez_x5.h>

.text

//-----------------------------------------------------------------------------------------------
//
//  인터럽트  벡터 테이블
//
//-----------------------------------------------------------------------------------------------
.globl          _start

_start:         b       reset                   // MCU가 리셋된후 실행된다.
                b       undefined_instruction   // 정의되지 않는 명령이 명령 인스트럭터에 패치되었을때 실행
된다.
                b       software_interrupt      // 소프트웨어적인 인터럽트 명령(SWI)가 수행되었을 때 실행된
다.
                b       prefetch_abort          // 데이타를 프리 패치 할때 버스 에러가 발생하면 실행된다.
                b       data_abort              // 데이타 에러가 발생하면 실행된다.
                b       not_used                // 사용되지 않는다.
                b       IRQ                     // IRQ 인터럽트가 발생되면 실행된다.
                b       FIQ                     // IRQ 인터럽트보다 우선권이 있는 인터럽트가 발생되면 실행
된다.


//-----------------------------------------------------------------------------------------------
//
//  캐쉬에 관련된 코프로세스 명령을 적용할때의 지연
//
//-----------------------------------------------------------------------------------------------
.macro          CPWAIT
                mrc     p15, 0, r0, c2, c0, 0
                mov     r0, r0
                sub     pc, pc, #4
.endm

//-----------------------------------------------------------------------------------------------
```

```
//
//  RESET이 발생하거나 전원이 들어 왔을때 맨 처음 수행되는 루틴이다.
//
//-------------------------------------------------------------------------------------------------
reset:
                //-----------------------------------------
                // change Supervisor mode & IRQ/FIQ Disable
                //-----------------------------------------

                mrs     r0, CPSR
                bic     r0, r0, #0x1f
                orr     r0, r0, #(0x13 | 0xc0)
                msr     CPSR, r0

                //-----------------------------------------
                // Change CPU SPEED
                //-----------------------------------------

                // CCCR 레지스터를 설정하여 CPU 속도를 맞춘다.
                ldr     r0, =PXA_REG_CCCR
                ldr     r1, =CPU_SPEED
                str     r1, [r0]

                // 동작속도를 변경한다.
                mov     r0, #PXA_COP_CCLKCFG_FCS
                mcr     p14, 0, r0, c6, c0, 0

                // 터보모드가 있는지 확인한다.
                and     r1, r1, #(0x7 << 7)     // CCCR_BF_N Mask
                cmp     r1, #CCCR_BF_N_RUN_X10
                beq     10f

                // 터보모드를 활성화 한다.
                mov     r0, #PXA_COP_CCLKCFG_TURBO
                mcr     p14, 0, r0, c6, c0, 0
10:
                //-----------------------------------------
                // Active I-Chache
                //-----------------------------------------

                // I-Cache 비활성화
                mcr     p15, 0, r1, c7, c5, 0
                CPWAIT

                // I-Cache 활성화
                mrc     p15, 0, r0, c1, c0, 0
                orr     r0, r0, #0x1000
                mcr     p15, 0, r0, c1, c0, 0
                CPWAIT

                        mov     r5, #DEBUG_START
                        bl      led_out

                //-----------------------------------------
                //  GPIO 설정
                //-----------------------------------------

                bl      gpio_init

                //-----------------------------------------
                //  SDRAM, Static MEM 설정
                //-----------------------------------------

                bl      mem_config

                        mov     r5, #DEBUG_READY_MEMTEST
                        bl      led_out

                //-----------------------------------------
                //  Memory Test
                //-----------------------------------------

                bl      mem_test

                        mov     r5, #DEBUG_MEM_OK
```

```
                        bl      led_out

                //-----------------------------------------
                //  부트로더를 메모리에 올린다.
                //-----------------------------------------

                bl      mem_clear
                bl      copy_loader_to_ram

                //-----------------------------------------
                //  STACK 포인터를 설정한다.
                //-----------------------------------------

                ldr     r0, =EZ_X5_RAM_BOOT_END
                sub     sp, r0, #0x04

                        mov     r5, #DEBUG_JUMP_C
                        bl      led_out

                //-----------------------------------------
                //  ezboot 임을 메모리에 표시한다.
                //-----------------------------------------

                mov     r1, #0xA0000000
                mov     r2, #0x0000
                str     r2, [r1]

                //-----------------------------------------
                //  main.c 로 이동한다.
                //-----------------------------------------

                ldr     r0, =EZ_X5_RAM_BOOT
                add     r0, r0, #EZ_X5_C_MAIN_OFFSET

                mov     pc, r0

                // 이곳으로는 진행되지 않는다.
                b       error_loop

//-------------------------------------------------------------------------------------------------
//
//  Reset 이외의 인터럽트 처리
//-------------------------------------------------------------------------------------------------

data_abort:
                mov     r5, #DEBUG_DATA_ABORT
                bl      led_out
                b       error_loop

undefined_instruction:
software_interrupt:
prefetch_abort:
not_used:
IRQ:
FIQ:
                mov     r5, #DEBUG_OTHER_EXCEPT
                bl      led_out
                b       error_loop
```

```
/*=====================================================================================================

  화일명 : memory.S
  설  명 : EZ BOOT의 리셋 후 램에 관려된 처리 루틴이다.

  작성자 : (주)제이닷디앤티  푸지
  작성일 : 2003년 4월 21일

  수  정 : 2003-09-24   오재경(푸지)
              pcmcia 레지스터 설정방법 수정

  비  고 :

=====================================================================================================*/

#include        <ez_x5.h>

.text

//-----------------------------------------------------------------------------------------------------
//
//  메모리 설정을 초기화 한다.
//
//-----------------------------------------------------------------------------------------------------
.globl mem_config
mem_config:
                // OSCC 설정
                ldr     r0, =PXA_REG_OSCC
                ldr     r1, =OSCC_VALUE
                str     r1, [r0]

                // MSC 0 설정
                ldr     r0, =PXA_REG_MSC0
                ldr     r1, =MSC0_VALUE
                str     r1, [r0]
                ldr     r1, [r0]        // must be read after it is written

                // MSC 1 설정
                ldr     r0, =PXA_REG_MSC1
                ldr     r1, =MSC1_VALUE
                str     r1, [r0]
                ldr     r1, [r0]        // must be read after it is written

                // MSC 2 설정
                ldr     r0, =PXA_REG_MSC2
                ldr     r1, =MSC2_VALUE
                str     r1, [r0]
                ldr     r1, [r0]        // must be read after it is written

                // PCMCIA reg MECR
                ldr     r0, =PXA_REG_MECR
                ldr     r1, =MECR_VALUE
                str     r1, [r0]
                ldr     r1, [r0]        // write and then read

                // PCMCIA 타이밍 설정
                ldr     r0, =PXA_REG_MCMEM_S0

                ldr     r1, =MCMEM_S0_VALUE
                str     r1, [r0, #(PXA_REG_MCMEM_S0-PXA_REG_MCMEM_S0)]
                ldr     r1, [r0, #(PXA_REG_MCMEM_S0-PXA_REG_MCMEM_S0)]  // write and then read

                ldr     r1, =MCMEM_S1_VALUE
                str     r1, [r0, #(PXA_REG_MCMEM_S1-PXA_REG_MCMEM_S0)]
                ldr     r1, [r0, #(PXA_REG_MCMEM_S1-PXA_REG_MCMEM_S0)]  // write and then read

                ldr     r1, =MCATT_S0_VALUE
                str     r1, [r0, #(PXA_REG_MCATT_S0-PXA_REG_MCMEM_S0)]
                ldr     r1, [r0, #(PXA_REG_MCATT_S0-PXA_REG_MCMEM_S0)]  // write and then read

                ldr     r1, =MCATT_S1_VALUE
                str     r1, [r0, #(PXA_REG_MCATT_S1-PXA_REG_MCMEM_S0)]
                ldr     r1, [r0, #(PXA_REG_MCATT_S1-PXA_REG_MCMEM_S0)]  // write and then read

                ldr     r1, =MCIO_S0_VALUE
```

```
                str     r1, [r0, #(PXA_REG_MCIO_S0 -PXA_REG_MCMEM_S0)]
                ldr     r1, [r0, #(PXA_REG_MCIO_S0 -PXA_REG_MCMEM_S0)]  // write and then read

                ldr     r1, =MCIO_S1_VALUE
                str     r1, [r0, #(PXA_REG_MCIO_S1 -PXA_REG_MCMEM_S0)]
                ldr     r1, [r0, #(PXA_REG_MCIO_S1 -PXA_REG_MCMEM_S0)]  // write and then read

                // CKEN 설정
                ldr     r0, =PXA_REG_CKEN
                ldr     r1, =CKEN_VALUE
                str     r1, [r0]

        // Syncronous Static Memory  설정 ==========================

                ldr     r0, =PXA_REG_SXCNFG
                ldr     r1, =SXCNFG_VALUE
                str     r1, [r0]

        // SDRAM Clock 설정 ======================================

                ldr     r0, =PXA_REG_MDREFR
                ldr     r1, =MDREFR_VALUE

                // diable E1PIN
                bic     r1, r1, #MDREFR_E1PIN
                str     r1, [r0]

                // enable E1PIN
                orr     r1, r1, #MDREFR_E1PIN
                str     r1, [r0]
                nop
                nop

        // SDRAM Config 설정 =====================================

                ldr     r0, =PXA_REG_MDCNFG
                ldr     r1, =MDCNFG_VALUE

                // SDRAM Partition 0/1 disable
                bic     r1, r1, #(MDCNFG_DE0 | MDCNFG_DE1)
                str     r1, [r0]

                // CBR refresh cycles 8
                ldr     r3, =EZ_X5_BASE_RAM
                .rept   8
                str     r3, [r3]
                .endr

                // SDRAM Partition 0/1 enable
                ldr     r1, [r0]
                orr     r1, r1, #(MDCNFG_DE0|MDCNFG_DE1)
                str     r1, [r0]

        // SDRAM Burst length 설정 ================================

                ldr     r0, =PXA_REG_MDMRS
                ldr     r1, =MDMRS_VALUE
                str     r1, [r0]

                mov     pc, lr


//-------------------------------------------------------------------------------------------------
//
//  부트 로더가 올려질 메모리 영역을 검사한다.
//
//-------------------------------------------------------------------------------------------------
.globl mem_test
mem_test:
                ldr     r2, =0x55555555
                ldr     r3, =0xAAAAAAAA

10:             ldr     r0, =EZ_X5_RAM_BOOT             // 메모리 베이스 어드레스
                ldr     r1, =EZ_X5_RAM_BOOT_END         // 128Kbyte 를 테스트 한다.
```

```
                // mem-write loop
20:             str     r2, [r0], #4
                str     r3, [r0], #4

                cmp     r0, r1
                bne     20b

                // mem-comp loop
                ldr     r0, =EZ_X5_RAM_BOOT             // 메모리 베이스 어드레스

30:             ldr     r4, [r0], #4
                ldr     r5, [r0], #4

                cmp     r2, r4
                bne     mem_error
                cmp     r3, r5
                bne     mem_error

                cmp     r0, r1
                bne     30b

                mov     r4, #0xAA
                and     r2, r2, #0xff
                cmp     r2, r4

                moveq   pc, lr          // return

                ldr     r2, =0xAAAAAAAA
                ldr     r3, =0x55555555
                b       10b

mem_error:
                // lr 을 무시하고 호출한다.
                mov     r5, #DEBUG_MEM_ERROR
                bl      led_out

                b       error_loop

//-------------------------------------------------------------------------------------------------
//
//  부트 로더가 올려질 메모리 영역을 클리어 한다.
//
//-------------------------------------------------------------------------------------------------
.globl mem_clear
mem_clear:
                ldr     r0, =EZ_X5_RAM_BOOT             // 메모리 베이스 어드레스
                ldr     r1, =EZ_X5_RAM_BOOT_END         // 128Kbyte
                mov     r2, #0x0                        // 메모리에 쓸 값
10:
                str     r2, [r0], #4
                cmp     r0, r1
                bne     10b

                mov     pc, lr

//-------------------------------------------------------------------------------------------------
//
//  롬영역의 부트 로더를 램에 올린다.
//
//-------------------------------------------------------------------------------------------------
.globl copy_loader_to_ram
copy_loader_to_ram:

                ldr     r0, =EZ_X5_RAM_BOOT             // 메모리 베이스 어드레스
                ldr     r1, =EZ_X5_RAM_BOOT_END         // 128kbyte
                mov     r2, #EZ_X5_BASE_ROM             // 롬 베이스 어드레스

10:             ldr     r3, [r2], #4
                str     r3, [r0], #4
                cmp     r0, r1
                bne     10b

                // 비교한다.
                ldr     r0, =EZ_X5_BASE_ROM
                ldr     r1, =EZ_X5_RAM_BOOT
```

```
                ldr     r4, =EZ_X5_RAM_BOOT_END

20:             ldr     r2, [r0], #4
                ldr     r3, [r1], #4
                cmp     r2, r3
                bne     comp_error

                cmp     r1, r4
                bne     20b

                mov     pc, lr


comp_error:     mov     r5, #DEBUG_MEM_ERROR
                bl      led_out

                b       error_loop




-------------------------------------------------------------------------------------------

/*=============================================================================================================

  화일명 : gpio.S
  설  명 : EZ-X5 의 GPIO를 제어 하는 루틴이다.

  작성자 : (주)제이닷디앤티  푸지
  작성일 : 2003년 4월 21일

  비  고 :

=============================================================================================================*/

#include        "ez_x5.h"

.text

//-------------------------------------------------------------------------------------------------------
//
//  gpio 초기화
//
//-------------------------------------------------------------------------------------------------------
.globl gpio_init
gpio_init:
                // GPIO Alternate function
                ldr     r0, =PXA_REG_GP_BASE

                ldr     r1, =GAFR0_L_VALUE
                str     r1, [r0, #PXA_REG_OFFSET_GAFR0_L]
                ldr     r1, =GAFR0_U_VALUE
                str     r1, [r0, #PXA_REG_OFFSET_GAFR0_U]

                ldr     r1, =GAFR1_L_VALUE
                str     r1, [r0, #PXA_REG_OFFSET_GAFR1_L]
                ldr     r1, =GAFR1_U_VALUE
                str     r1, [r0, #PXA_REG_OFFSET_GAFR1_U]

                ldr     r1, =GAFR2_L_VALUE
                str     r1, [r0, #PXA_REG_OFFSET_GAFR2_L]
                ldr     r1, =GAFR2_U_VALUE
                str     r1, [r0, #PXA_REG_OFFSET_GAFR2_U]

                ldr     r1, =GPDR0_VALUE
                str     r1, [r0, #PXA_REG_OFFSET_GPDR0]
                ldr     r1, =GPDR1_VALUE
                str     r1, [r0, #PXA_REG_OFFSET_GPDR1]
                ldr     r1, =GPDR2_VALUE
                str     r1, [r0, #PXA_REG_OFFSET_GPDR2]

                // PSSR 설정 - GPIO 입력 활성화
                ldr     r0, =PXA_REG_PSSR
                ldr     r1, =PSSR_VALUE
                str     r1, [r0]
```

```
                // debug led off
                ldr     r0, =PXA_REG_GP_BASE
                mov     r1, #(_LED_3|_LED_2|_LED_1|_LED_0)
                str     r1, [r0, #PXA_REG_OFFSET_GPSR0]

                mov     pc, lr

//------------------------------------------------------------------------------------------------
//
//   debug LED 를 켜거나 끈다
//
//      r5 : 상태값이 넘어온다
//
//------------------------------------------------------------------------------------------------
.globl led_out
led_out:
                ldr     r0, =PXA_REG_GP_BASE

                // led 0,1,2 off
                mov     r1, #(_LED_2|_LED_1|_LED_0)
                str     r1, [r0, #PXA_REG_OFFSET_GPSR0]

                // state on
                mov     r1, #0

                and     r4, r5, #0x4
                cmp     r4, #0
                beq     10f
                orr     r1, r1, #_LED_2

10:             and     r4, r5, #0x2
                cmp     r4, #0
                beq     20f
                orr     r1, r1, #_LED_1

20:             and     r4, r5, #0x1
                cmp     r4, #0
                beq     30f
                orr     r1, r1, #_LED_0

30:             str     r1, [r0, #PXA_REG_OFFSET_GPCR0]

                mov     pc, lr

//------------------------------------------------------------------------------------------------
//
//  LED를 점멸 시킨다. 이 루틴은 무한 루틴으로 빠진다.
//
//------------------------------------------------------------------------------------------------
.globl error_loop
error_loop:

                ldr     r0, =PXA_REG_GP_BASE
                mov     r1, #ERROR_LED

10:             str     r1, [r0, #PXA_REG_OFFSET_GPCR0]

                mov     r4, #0x800000
20:             nop
                nop
                subs    r4, r4, #1
                bne     20b

                str     r1, [r0, #PXA_REG_OFFSET_GPSR0]

                mov     r4, #0x800000
30:             nop
                nop
                subs    r4, r4, #1
                bne     30b

                b       10b
```