

u-boot 1.1.2 Porting Guide for PXA27x

v 0.01

2005.12.21

김기오 (gurugio@gmail_nospam.com)

<http://www.asmlove.co.kr>

<http://wiki.kldp.org/wiki.php/gurugio>

알림.

이 문서는 개인적인 참고 자료의 모음과 연구실 내부에서 신입생 교육에 사용될 개인적인 자료입니다. 혹시 저작권이나 기타 위반 사항 등이 발견되면 언제라도 연락 주시면 즉시 바로잡도록 하겠습니다. 이 문서는 개인 적인 참고 용도로만 사용될 수 있습니다.

김기오 (gurugio@gmail_nospam.com.nospam)

update.

2005년 12월 18일 kldp.wiki 등록

2005년 12월 20일 레지스터 설명, 관련 코드 추가 v. 0.01

플래시 프로그래머 포팅

먼저 인텔에서 jflashmm 프로그램을 다운받는다. 이 프로그램은 윈도우 전용이다. 먼저 윈도우에 설치해서 보드를 인식하는지 테스트하고 소스를 리눅스로 옮기고 Makefile을 만들어서 포팅한다.

JFlashmm 다운로드 주소

<http://www.intel.com/design/pca/applicationsprocessors/swsup/JFlashMM.htm>

인텔은 다운로드 주소를 수시로 바꾸므로 인텔 홈페이지에서 검색한다.

JFlash_MM_V5_01_007.exe 이라는 프로그램을 설치하면 디렉토리안에 Flash_88XX~~.dat 라는 설정 파일들과 Source.zip 소스 파일등이 있다. 프로그램을 설치한 후에 RelNode_JFlashmm.htm 을 읽으면서 giveio.inf 드라이버 파일을 윈도우에 설치한다. 그래야 프로그램이 정상적으로 작동된다. 그리고 보드를 만들 때 프로세서와 플래쉬, 램등 최소한의 장치만 조립하고 윈도우에 설치된 JFlashmm로 동작을 테스트한다. 그런데 한가지 빠진 파일이 있는데 bulbcx.dat 파일이다. <http://www.linuxgazette.com/node/9786> 문서는 2.4 커널을 mainstone 보드에 포팅하는 문서인데 bulbcx.dat 에 대해 다음과 같이 말하고 있다.

Connect the parallel port connector to MAIN JTAG (J4) and parallel port on your computer. Secondly copy the bulbcx.dat and blob-smc files to the folder where the jflash is installed.

Ex: C:\Program Files\Intel Corporation\JFlash_MM\

Bulbcx.dat: Has the information about the processor and the board. And it is provided by the board manufacturer.

이 파일을 인터넷에서 구해서 jflash 폴더에 복사한다. 그리고 다음과 같이 실행해본다.

```
JFlash Window Console

C:\JFlash_MM>JFlashmm.exe bulbcx testfile.bin i
JFLASH Version 5.01.007
COPYRIGHT (C) 2000 - 2003 Intel Corporation

PLATFORM SELECTION:
Processor=          PXA27x
Development System= Mainstone
Data Version=       1.00.002

PXA27x revision C0
Found flash type: 28F128K3

Program successful completion.
Processor and Flash Memory Identified.
No programming operation performed.

C:\JFlash_MM>JFlashmm.exe bulbcx testfile.bin v
JFLASH Version 5.01.007
COPYRIGHT (C) 2000 - 2003 Intel Corporation

PLATFORM SELECTION:
Processor=          PXA27x
Development System= Mainstone
Data Version=       1.00.002

PXA27x revision C0
Found flash type: 28F128K3

Starting Verify
verify error at address = 0 exp_dat = 11111111 act_dat = 11111111

C:\JFlash_MM>
```

i 옵션으로 실행해서 플래시와 프로세서를 인식하는지를 확인한다. 여기까지 된다면 프로세서와 플래시 메모리는 동작하는 것이다.

다음으로 Source.zip과 *.dat 설정 파일들을 리눅스로 복사한다.

* jflashmm을 포팅하는 것은 수정할 양이 많으므로 일단 하이버스에서 제공한 시디에 있는 jflashmm을 사용한다. 추후 참고 자료를 구하거나 시간이 날 때 다시 분석한다. 하이버스에서 제공하는 설정파일 pxa27x32.dat는 bulbcx.dat와 같은 파일이다. 따라서 bulbcx.dat를 사용한다.

** 리눅스로 복사한 소스 중 Global_variable.h에서 41줄에 있는 CABLE_TYPES를 Parallel_Jtag에서Insight_Jtag으로 수정한다. 원래 인텔에서 제공하는 jflashmm 소스를 보면 이 부분이 Insight_Jtag으로 되어있어서 이렇게 고친다. Parallel Jtag 타입으로 컴파일하면 인식을 못하고 JTag에서 나오는 신호가 다르다.

```

gurugio@gioserver:/home/SKKU/bulverde/jflashmm
34 char flash_data_filename[MAX_IN_LENGTH] = "Flash_18_2_32.dat"; // Global variable for the flash data file
35 char int_data_filename[MAX_IN_LENGTH] = "DBPXA250_INTEGRITY.DAT"; // Global integrity data file
36
37 char VERSION_LOCK[11] = "VL000000001";
38 char FLASH_VERSION_LOCK[11] = "VLF00000001";
39
40 /* fixed by HYBUS */
41 //CABLE_TYPES CableType = Parallel_Jtag; // Global variable for specifying the Cable type
42
43 /* In original jflashmm sources, CABLE_TYPES is Insight_Jtag
44 * fixed by Gi-Oh Kim */
45 CABLE_TYPES CableType = Insight_Jtag;
46
47
48 DWORD ChipSelect0 = 0; // Global variable for chip select 0
49 DWORD ChipSelect1 = 0; // Global variable for chip select 1
50 DWORD ChipSelect2 = 0; // Global variable for chip select 2
51 DWORD ChipSelect3 = 0; // Global variable for chip select 3
52 DWORD ChipSelect4 = 0; // Global variable for chip select 4
53 DWORD ChipSelect5 = 0; // Global variable for chip select 5

```

Jflash.cpp에는 다음과 같이 jtag케이블 타입에 대해 처리하는 부분이 있다. HYBUS에서 사용하는 jtag은 패러럴 케이블을 바로 보드에 연결하는 형태인데 프린터 포트의 5번이 nTRST, 2번이 TCK, 3번이 TDI, 11번이 TDO, 4번이 TMS로 사용되고 있지만 일반적으로 현재 연구실에서 사용하는 jtag케이블과 다른 것 같다.

```

362 // Compare the selected cable name with all supported cables
363 if (!strcmp("INS", Cable_Type)){
364     CableType = Insight_Jtag;
365 } else if (!strcmp("PAR", Cable_Type)){
366     CableType = Parallel_Jtag;
367 }

```

케이블 타입에 따라서 출력하는 신호가 다르다.

```

938 if(CableType == Parallel_Jtag)
939 {
940     // TMS is D2, TDI is D1, and TCK is D0, so construct an output by creating a
941     // rising edge on TCK with TMS and TDI data set.
942     /* fixed */
943     _outp(lpt_address, tms*4+tdi*2+8); // TCK low
944     _outp(lpt_address, tms*4+tdi*2+1+8); // TCK high
945
946     // if we want to read the port, set TCK low because TDO is sampled on the
947     // TCK falling edge.
948     /* fixed */
949     if(rp == READ_PORT)
950         _outp(lpt_address, tms*4+tdi*2+8); // TCK low
951     if(rp == READ_PORT)
952         tdo = !((int)_inp(lpt_address + 1) >> 7); // get TDO data
953 }
954 else if (CableType == Insight_Jtag)
955 {
956     // There's some bit clearing here that isn't needed. It should make this
957     // code easier to understand.
958
959     //defines for the INSIGHT IIC-1 JTAG cable
960
961     /* the output port (lpt_address) */
962     #define INSIGHT_DIN 0x01

```

```

963     #define INSIGHT_CLK      0x02
964     #define INSIGHT_TMS_IN   0x04
965
966     /* Output Enable for the standard JTAG outputs
967     (not TDO since this is an output from the
968     chip we want to talk to */
969     #define nINSIGHT_CTRL     0x08
970     #define nINSIGHT_PROG     0x10 /* This causes the TDO line to be driven. We'll leave it high.*/
971
972
973     /*the input port (lpt_address + 1)*/
974     #define TDO_INPUT         0x10
975     #define TDO_INPUT_BITPOS  4
976
977     int lpt_data;
978
979     //form the data we want to write to the parallel port
980     lpt_data = nINSIGHT_PROG; //Output to TDO off
981     lpt_data &= ~nINSIGHT_CTRL; //Enable the outputs
982
983     if(tms == 1) lpt_data |= INSIGHT_TMS_IN;
984     if(tdi == 1) lpt_data |= INSIGHT_DIN;
985
986     // construct an output by creating a
987     // rising edge on TCK with TMS and TDI data set.
988     lpt_data &= ~INSIGHT_CLK;
989     _outp(lpt_address, lpt_data); // TCK low
990
991     lpt_data |= INSIGHT_CLK;
992     _outp(lpt_address, lpt_data); // TCK high
993
994     // if we want to read the port, set TCK low because TDO is sampled on the
995     // TCK falling edge.
996     if(rp == READ_PORT)
997     {
998         lpt_data &= ~INSIGHT_CLK;
999         _outp(lpt_address, lpt_data); // TCK high ??? Low?
1000         tdo = ((int)_inp(lpt_address + 1) & TDO_INPUT) >> TDO_INPUT_BITPOS; // get TDO data
1001     }
1002 }

```

다음과 유사하게 출력되야 JFlashmm과 보드의 프로세서와 플래시 메모리가 동작한다는 것을 알 수 있다.

```

gurugio@gioserver: /home/SKKU/bulverde/jflashmm
[gurugio@gioserver jflashmm] $
[gurugio@gioserver jflashmm] $ su
Password:
[root@gioserver jflashmm]# jflashmm_skku bulbcx testfile.bin i

JFLASH Version 5.01.007
COPYRIGHT (C) 2000 - 2003 Intel Corporation

PLATFORM SELECTION:
Processor=          PXA27x
Development System= Mainstone
Data Version=       1.00.002

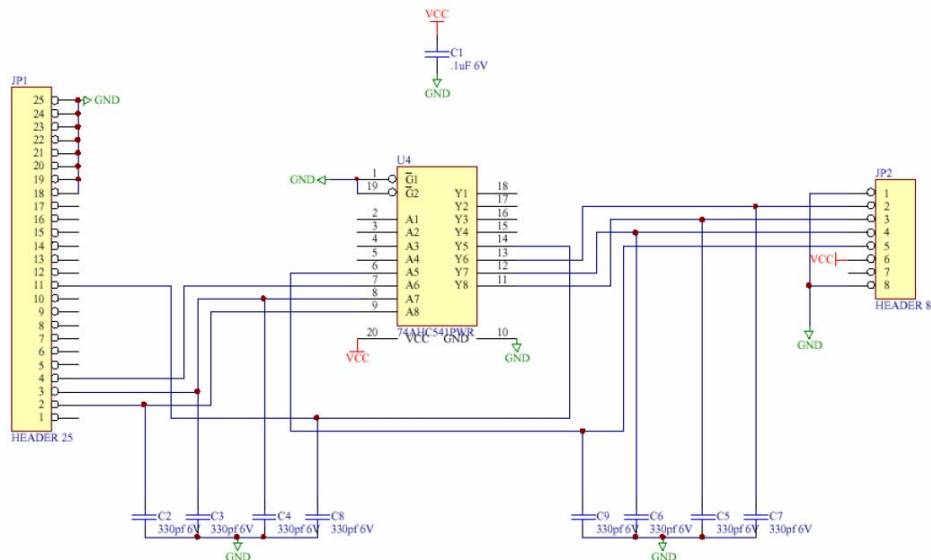
PXA27x revision C0
Found flash type: 28F128K3

Program successful completion.
Processor and Flash Memory Identified.
No programming operation performed.
[root@gioserver jflashmm]#

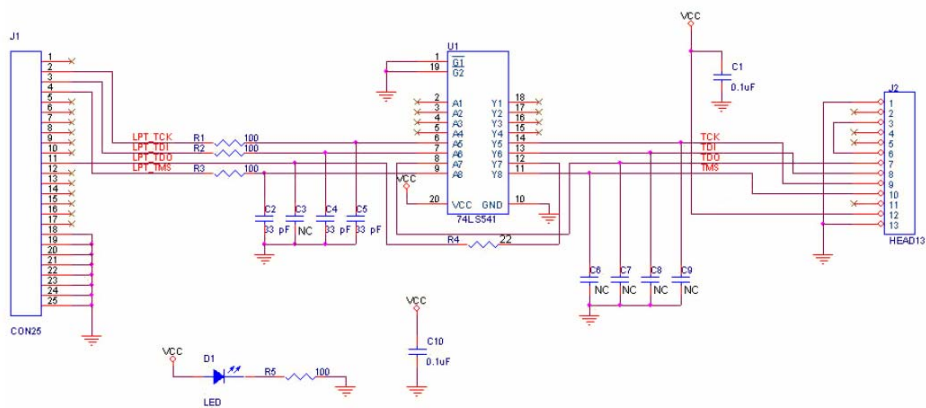
```


JTAG 동작을 테스트하는 방법

1. 프린터 포트를 이용하여 jtag을 사용할 때
프린터 포트는 5V로 동작한다. jtag은 3V에서 동작하기 때문에 이것을 변환하는 회로가 필요하다. 다음은 삼성에서 만든 Secjtag의 회로도이다.



Holly Gates' Schematics for the JTAG Dongle (<http://www.lart.tudelft.nl/projects/jtag/>)



Modified Schematic for the JTAG Dongle Circuit Using 74LS541

그리고 다음은 falinux의 강좌 일부를 복사한 것이다.

0.2 JTAG 회로

이회로를 보통은 동글이라고 말합니다.

그림 0.2.a

JTAG에 대해선 설명하지 않겠습니다. 유명창님 강좌에 넘 잘나왔으니까...

대부분의 칩들은 동작 전압이 3.3V 입니다. PC의 프린터 포트는 5V 이구요..
전압이 틀린것을 알수 있죠. 틀리 전압이 만나면 어떻게 될까요.. 썬님이 썰다고
5V가 3.3V를 밀어냅니다. 1.7V 의 차만큼 3.3V 입장에서 마이너스 전원이 걸리는
셈이지요. 하지만 칩들은 TR의 집합이므로 이런 전원이 걸리지 않습니다. 하지만
데이터 라인은 약 4.3V 정도 걸리지요.. 문제는 이런것이 아니고 데이터 버스가
흔들리는 것입니다. 이런 흔들림에 의해서 인근의 라일들까지 잡음이 끼게 되지요
(높은 물과 낮은 물이 갑자기 만나면 물이 흔들리는 것과 같은 이치이지요.)

스코프를 찍어 보면 이런 파형이 나옵니다.

그림 0.2.b

동글은 전압이 다른 두 칩의 연결시 이런 잡음등으로 칩을 보호하는 회로입니다.

보통은 74HC245, 74HC125 등의 단전원 칩을 사용하기도 하고 3.3V, 5V 의 양전원이
들어가는 74LV245 등을 쓰기도 합니다.

74HC245, 74HC125 는 3.3V 전원을 인가하는데 이것도 어차피 5V와 만나면서 잡음이
발생합니다.

여기서 잡음을 최소화 하기 위해 저항과 콘덴서를 연결합니다.

저항의 효과는 두전원의 전압차를 이 저항이 전압을 갖게되는 것이고 콘덴서는

항상 쓰는 방식대로 흔들림을 잡아주죠..

저항값을 200옴으로 쓰는 이유는 잘 모르구요.. 테스트한 결과 100~200 이 무난합니다.

다른 회로도 이값을 씁니다.

콘덴서 220pF 을 쓰는 이유는 프린터 포트의 속도가 최대 1.5usec 이므로 이 시간에서
큰값으로 설정한 것입니다.

어찌됐건 74HC 씨리즈는 CMOS 로 폭넓은 동작 전원을 가지고 있어 아주 유용합니다.

여담1:

저항을 그저 저항으로 생각하지 마시고...

전자라는 녀미 길을가다 거치른 저항을 만나면 힘이 빠집니다. 잡음의 폭이 그만큼 준다는

야그지요.. 왜 전압이 약간 떨어지니까..

글구 전자가 저항을 만나면 진행속도가 느려지죠.. 이걸 파형이 그만큼 늘어진다는 야그고요

여담2:

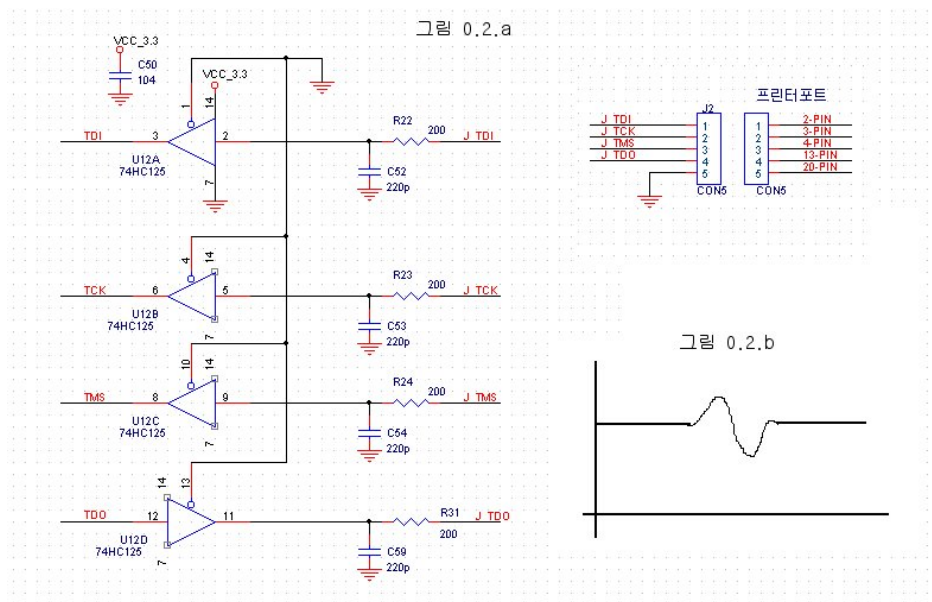
예전 MP3 플레이어중 프린트포트를 사용하는 제품은 이잡음때문에 무지 고생했습니당.. -.-

다음 강좌는 DC/DC 전원에 대해 알아 보겠습니다. 그럼 전 휘리릭 ~~~

[소유권]

이 문서 지적 소유권은 (주)제이닷디엔티에 있다.

문서에 관련된 문의 사항이 있다면 freefrug@falinux.com으로 연락 바란다



이렇게 살펴보고 jtag 케이블이 회로에 맞게 설계되었는지 전압이 올바르게 출력되는지를 확인한다.

2. TDI, TMS, TCLK의 신호가 발생하는가?
 예를 들어 보드를 켜지 않은 상태에서 jtag을 연결시키고 다음과 같은 명령을 실행한다면,

```
jflashmm bulbcx testfile.bin i
```

 오실로스코프를 1ms 단위 정도로 조정하고 TDI와 TCLK을 측정하면 아주 짧게라도 신호가 나올 것이다. 이정도 신호가 나온다면 일단 jtag 케이블은 동작이 된다고 판단해도 좋다.
3. Bulberde에서 nTRST에 high 신호가 인가되어 있는지 확인한다.
4. “jflashmm bulbcx testfile.bin i” 로 실행해서 identification이 되는지 확인한다.
5. 2005-11-25 현재 SKKU 보드를 이와 같은 방법으로 확인함

JTAG 동작 확인 후 하드웨어 테스트

JTAG이 동작한다는 것은 프로세서와 플래시 메모리가 동작한다는 것을 뜻한다. 여기서 보다 확실하게 보드가 동작하는 지 또 JTAG이 프로그램을 정상적으로 퓨징할 수 있는지 확인하기 위해서 간단하게 LED를 깜박이고 시리얼에 문자를 출력하는 프로그램을 만들어 본다. 꼭 LED가 있어야 하는건 아니고 GPIO 하나를 골라서 주기적으로 신호를 보내서 오실로스코프로 파형이 발생하는지 확인하면 된다. Falinux의 EZ-X5 보드는 4개의 LED를 달아서 실행 도중 오류에 따라 다르게 LED를 깜박거린다. 따라서 어디에서 오류가 발생했는지 짐작할 수 있게 하였다. 보드를 설계할 때 이렇게 디버깅을 위한 고려도 하면 개발 시간을 아낄 수 있다.

가장 먼저 해야 할 일은 적당한 GPIO를 고르는 것이다. SKKU보드에는 따로 보드 동작을 위한 LED가 없으므로 임시로 현재 사용하지 않는 오디오 칩에 관련된 GPIO<30>을 사용한다.

PXA270의 GPIO는 입력과 출력으로 나뉘지고 입력과 출력 각각에 따라 3가지 기능이 있다. 즉 하나의 GPIO 핀이 6가지의 기능을 가지고 있는 셈이다. GPIO를 사용하기 전에 어떤 기능으로 사용해야 할지를 결정해서 입출력을 결정하는 레지스터와 기능을 결정하는 레지스터를 설정해주면 된다. Intel PXA27x Processor Family Developer's Manual에 24장 General-Purpose I/O Controller에서 Table 24-2를 보면 핀 이름과 기능들이 나열되어 있다. 회로에 따라 여기서 기능을 선택하여 GPIO 설정 레지스터의 값들을 결정하면 된다.

GPIO를 설정하는 레지스터는 GPLR, GPDR, GPSR, GPCR, GRER, GFER, GEDR, GAFR이 있다. Intel PXA27x Processor Family Developer's Manual에서 테이블 24-41을 보면 끝부분에 96번 이상의 GPIO를 설정하는 레지스터가 따로 있는 것을 알 수 있다. 이는 PXA270에서 PXA250부터 사용된 GPIO 이외에 추가됐기 때문이다. 각 레지스터를 간략하게 설명하면 다음과 같다.

- GPLR : 읽기 전용 상태 레지스터, High/Low 상태를 알 수 있다.
- GPDR : 입출력 방향 설정
- GPSR : Set, 하이 신호 인가
- GPCR : Clear, 로우 신호 인가
- GRER/GFER : Rising Edge 디텍트, Falling Edge 디텍트. GEDR에 디텍트된 결과 값이 출력된다. 만약 Rising Edge 디텍트 기능을 사용한다면 해당 핀에 Rising edge가 발생할 때마다 GEDR의 해당 비트가 1로 셋팅되고 이 비트에 1을 쓰면 0으로 클리어 하게 된다.
- GEDR : GRER/GFER에 설정한대로 Edge를 디텍트하면 1로 설정되고 1을 써주어서 셋팅된 값을 클리어 한다.
- GAFR : 하나의 GPIO마다 2비트씩 설정할 수 있다. 0은 GPIO로 사용하는 것이고 1~3은 테이블 24-2에 나온 기능들을 사용하도록 설정하는 것이다. 회로도에 따라 어떤 기능을 사용하는지 세심하게 결정해서 이 값을 결정해야 한다.

예를 들어 GPIO<30>에 관련된 설명을 보면 입력으로는 따로 사용되는 기능이 없고 출력으로는 I2S_SDAT_OUT, AC97_SDAT_OUT, USB_P3_2에 사용된다고 써있다. GAFR0_L 레지스터에서 28,29번 비트의 설정 값을 00으로 해주면 이런 기능들이 아니라 GPIO로 사용된다. 그리고 GPDR0의 30번 비트를 1로 셋팅해서 출력으로 설정해주면 된다. GPSR0의 30번 비트에 1을 써서 하이 값을 출력하고 잠시 후에 GPCR0의 30번 비트에 1을 써서 로우 값을 출력하고 잠시 후에 다시 하이 값을 출력하는 일을 반복하면 된다. 현재 보드에는 오디오에 관련된 칩을 부착하지 않고 오직 프로세서와 메모리, 전원관련 칩들만 부착했으므로 신경쓰지 말고 오실로스코프로 오디오 칩의 AC97_SDAT_OUT 핀을 찍어보면서 프로그램 동작을 확인하면 된다.

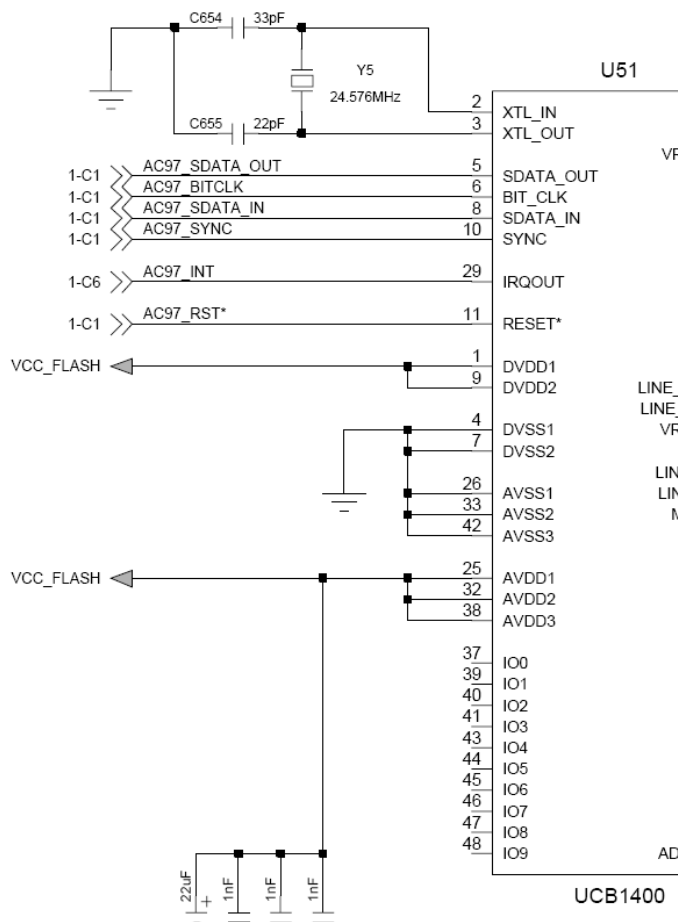
FFUART를 사용하기 위해서 테이블 24-2와 SKKU 보드의 회로도를 보면 GPIO<34> ~ GPIO<41>을 사용하는 것을 알 수 있다. GPIO<34> ~ GPIO<38>은 입력으로, GPIO<39> ~ GPIO<41>은 출력으로 사용된다. 각 핀의 사용법은 다음과 같다.

- GPIO<34> : FFRXD
- GPIO<35> : FFCTS
- GPIO<36> : FFDCD
- GPIO<37> : FFDSR
- GPIO<38> : FFRI
- GPIO<39> : FFTXD
- GPIO<40> : FFDTR
- GPIO<41> : FFRTS

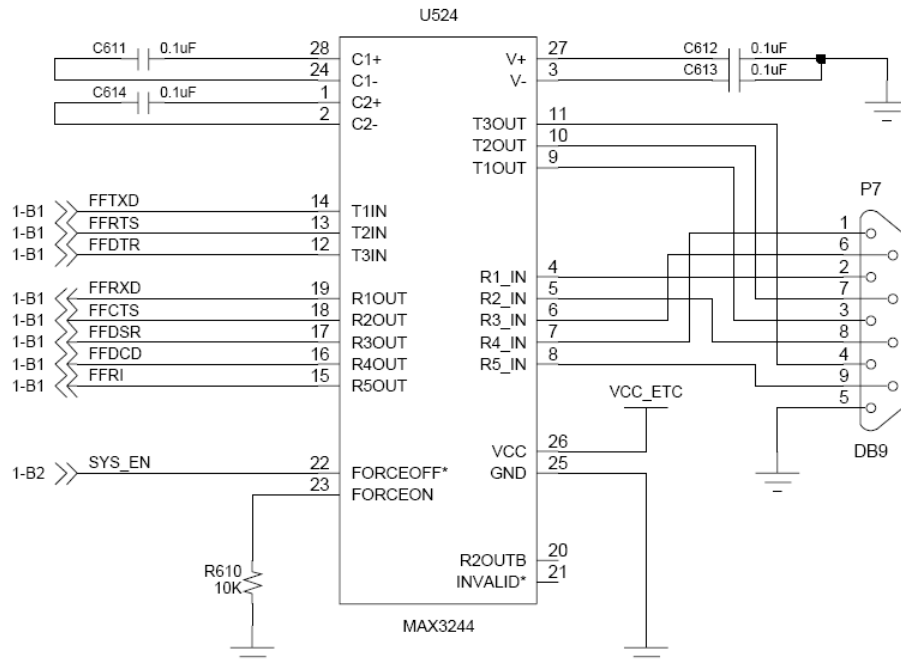
PXA270의 UART에는 SUART(Standard UART)와 FFUART(Full Function UART)가 있다. 이름에서 알 수 있듯이 SUART에는 TxD와 RxD만이 있다. Mainstone 보드에는 FFUART를 사용하므로 SKKU에도 동일하게 구성했다.

이후의 프로그램 소스와 컴파일 과정 등은 마이크로소프트 2003년 10월부터 유명창님께서 연재하신 ‘ARM 부트로더 제작기’를 참조하여 만들었다. gcc 버전에 따라 달라진 내용과 우리 보드에 맞게 수정한 내용을 주로 설명한다.

다음 그림은 오디오 칩 회로도의 일부를 나타낸 것이다. AC97_SDATA_OUT이 5번 핀으로 되어있다.



다음 그림은 UART 회로이다.



다음은 led.S 이다. 소스를 자세히 이해하기 위해서는 ARM 관련 서적을 참조해서 어셈블리 명령어를 익혀야 한다. 그리고 GPIO 설정 레지스터의 주소 값은 Intel PXA27x Processor Family Developer's Manual에 24장 General-Purpose I/O Controller에서 24.6 Register Summary를 참조한다. 레지스터들의 물리 메모리 주소를 알 수 있다. 또 UART의 설정에 대해서는 Intel PXA27x Processor Family Developer's Manual의 10장 UARTs 를 읽어본다. 현재 소스는 간단한 동작만을 하므로 u-boot 1.1.2에서 C로 작성된 초기화 함수와 한 문자 출력 함수를 보고 어셈블리로 변경만 해서 작성한 것이다.

GPIO관련 레지스터들의 시작 주소는 0x40e00000이다. Intel PXA27x Processor Family Developer's Manual에서 24.6 Register Summary에 있는 테이블에 보면 가장 먼저 GPLR0 레지스터가 있고 시작 주소가 0x40e00000인 것을 알 수 있다. 그리고 다른 레지스터들의 오프셋 값도 이 테이블을 참조하면 알 수 있다.

각 레지스터의 주소 값. GPIO<30>은 첫 번째 설정 레지스터에서 설정된다.

```
11 // GPIO Pin Direction register GPCR0/1/2/3
12 #define GPCR0 0x40e0000c
13 // GPIO Pin Output Set register GPSR0/1/2/3 => set high
14 #define GPSR0 0x40e00018
15 // GPIO Pin Output Clear register GPCR0/1/2/3 => set low
16 #define GPCR0 0x40e00024
17
```

GPIO<34> ~ GPIO<41>은 각기 두 번째 레지스터에서 설정된다.

```
18 #define GPCR1 0x40e00010
19 #define GPSR1 0x40e0001c
20 #define GPCR1 0x40e00028
21 #define GPCR1_L 0x40e0005c
22
```

GPIO 출력 값을 설정하고 루프를 돌면서 대기하는 횟수

```
23 #define WAIT_TIME_NOCACHE    0xf0000
```

GPIO 번호. 우리는 30번 GPIO를 이용하므로 GPDR0의 30번 비트를 사용한다. 따라서 $1 << 30$ 으로 값을 설정한다.

```
24 #define DEBUG_LED              (1<<30)
25
```

u-boot 1.1.2 소스에서 복사한 FFUART 설정 레지스터

```
26 /* Full Function UART (FFUART) */
27 #define FFUART    0x40100000
28 #define FFRBR 0x40100000 /* Receive Buffer Register (read only) */
29 #define FFTHR 0x40100000 /* Transmit Holding Register (write only) */
30 #define FFIER 0x40100004 /* Interrupt Enable Register (read/write) */
31 #define FFIIR 0x40100008 /* Interrupt ID Register (read only) */
32 #define FFFCR 0x40100008 /* FIFO Control Register (write only) */
33 #define FFLCR 0x4010000C /* Line Control Register (read/write) */
34 #define FFMCR 0x40100010 /* Modem Control Register (read/write) */
35 #define FFLSR 0x40100014 /* Line Status Register (read only) */
36 #define FFMSR 0x40100018 /* Modem Status Register (read only) */
37 #define FFSPR 0x4010001C /* Scratch Pad Register (read/write) */
38 #define FFISR 0x40100020 /* Infrared Selection Register (read/write) */
39 #define FFDLL 0x40100000 /* Divisor Latch Low Register (DLAB = 1) (read/write) */
40 #define FFDLH 0x40100004 /* Divisor Latch High Register (DLAB = 1) (read/write) */
41
```

클럭 설정 레지스터. FFUART에 들어가는 클럭을 활성화시킨다.

```
42 #define CKEN 0x41300004 /* Clock Enable Register */
43 #define CKEN6_FFUART (1<<6) /* CKEN[6] = FFUART unit clock enable
44
```

FFLCR의 설정 값

```
45 #define LCR_WLS0 (1) /*
46 #define LCR_WLS1 (1 << 1)
47 #define LCR_DLAB (1 << 7)
48
```

기타 FFUART 설정값

Baudrate는 38400bps로 하고 8-N-1로 설정한다.

```
49 #define QUOT 24 /* 38400 */
50 #define IER_UUE (1 << 6)
51
```

FFUART를 사용하기 위한 GPIO 설정 값. 반드시 한 비트씩 확인해서 이해할 것

```
52 #define CFG_GPDR1_VAL    0x00000380
53 #define CFG_GPSR1_VAL    0x00000000
54 #define CFG_GPCR1_VAL    0x00000380
55 #define CFG_GAFR1_L_VAL  0x0000a950
56
57 .text
58
59 .global _start
60
61 _start:
62
63     b reset
64     b undefined_instruction
65     b software_interrupt
66     b prefetch_abort
67     b data_abort
68     b not_used
69     b IRQ
70     b FIQ
71
72
73 reset:
74
```

GPIO<30>를 출력으로 지정

```
76     ldr r1, =DEBUG_LED
```

```

77
78 // set direction of 4th bit as output
79 ldr r0, =GPDR0 // r0 = PXA_REG_GP_BASE
80 str r1, [r0] // [r0+GPDR0] = r1
81

```

로우 신호 인가

```

82 // set 4th bit as low -> turn on LED
83 ldr r0, =GPCR0
84 str r1, [r0] // [r0+GPCR0] = r1
85
86

```

FFUART를 위한 GPIO 설정

```

87 GPIO_init:
88 /* GPIO<34-41> are used for FFUART */
89 /* GPIO<34-41> are controlled by GPLR1, GPSR1, GPCR1, CPDR1,
90    GRER1, GFER1, GEDR1, GAFR1_L */
91 /* GPIO<34-38> are used as INPUT, Function 1 */
92 /* GPIO<39-41> are used as OUTPUT, Function 2 */
93 /* GPDR1 = 0x380 */
94 /* GPSR1 = 0x0 */
95 /* GPCR1 = 0x380 */
96 /* GRER1,GFER1,GPLR1 = don't care */
97 /* GAFR1_L = 0xa950 */
98 ldr r0, =GPSR1
99 ldr r1, =CFG_GPSR1_VAL
100 str r1, [r0]
101 ldr r0, =GPCR1
102 ldr r1, =CFG_GPCR1_VAL
103 str r1, [r0]
104
105
106 ldr r0, =GPDR1
107 ldr r1, =CFG_GPDR1_VAL
108 str r1, [r0]
109
110 ldr r0, =GAFR1_L
111 ldr r1, =CFG_GAFR1_L_VAL
112 str r1, [r0]
113

```

FFUART를 위한 초기화 과정

순서를 이대로 지켜서 해야 함

```

114 serial_init:
115 /* CKEN |= CKEN6_FFUART */
116 /* Enable FFUART clock */
117 ldr r0, =CKEN
118 ldr r1, [r0]
119 ldr r2, =CKEN6_FFUART
120 orr r1, r1, r2
121 str r1, [r0]
122
123 /* FFIER = FFFCR = 0 */
124 /* Disable interrupts and FIFOs */
125 mov r0, #0
126 ldr r1, =FFIER
127 ldr r2, =FFFCR
128 str r0, [r1]
129 str r0, [r2]
130
131 /* FFLCR = LCR_WLS0 | LCR_WLS1 | LCR_DLAB */
132 /* 8bit character, Access Divisor Latch registers (DLL, DLH) */
133 mov r1, #LCR_WLS0
134 orr r1, r1, #LCR_WLS1
135 orr r1, r1, #LCR_DLAB
136 ldr r0, =FFLCR
137 str r1, [r0]

```

```

138
139 /*FFDLL = QUOT & 0xff, low byte of baudrate */
140 mov r1, #QUOT
141 ldr r0, =FFDLL
142 str r1, [r0]
143
144 /* FFDLH = QUOT >> 8, high byte of baudrate */
145 mov r1, #0
146 ldr r0, =FFDLH
147 str r1, [r0]
148
149 /* FFLCR = LCR_WLS0 | LCR_WLS1 */ 150 /* Clear DLAB bit */
151 mov r1, #LCR_WLS0
152 orr r1, r1, #LCR_WLS1
153 ldr r0, =FFLCR
154 str r1, [r0]
155
156

```

FFUART 동작 시작

```

157 /* FFIER = IER_UUE */
158 /* Enable UART unit */
159 mov r1, #IER_UUE
160 ldr r0, =FFIER
161 str r1, [r0]
162

```

원본 소스에는 `mov` 명령어를 사용하지만 30번 비트를 조작하므로 오퍼랜드 범위가 12비트를 넘는다. ARM에서 오퍼랜드의 크기는 12비트가 한계이므로 `mov` 명령어가 아니라 `ldr` 명령어를 사용한다.

```

163 ldr r1, =DEBUG_LED
164

```

일정 시간 딜레이

```

165 B10:
166 mov r4, #WAIT_TIME_NOCACHE // r4 = WAIT.
167 B20:
168 nop
169 nop
170 subs r4, r4, #1 // r4 = r4 - 1 & effect CPSR
171 bne B20 // if NE(not equal) is set (r4 != 0), branch to B20
172

```

GPIO<30>에 로우 신호 인가

```

173 // set 4th bit as high -> turn off LED
174 ldr r0, =GPSR0
175 str r1, [r0] // [r0+GPSR0] = r1
176
177 mov r4, #WAIT_TIME_NOCACHE // r4 = WAIT_TIME_NOCACHE
178
179 B30:
180 nop
181 nop
182 subs r4, r4, #1
183 bne B30
184

```

GPIO<30>에 하이 신호 인가

```

185 // turn on LED
186 ldr r0, =GPCR0
187 str r1, [r0]
188

```

시리얼에 '0' 출력

```

189 // serial_putc(0)
190 mov r2, #0x30
191 ldr r3, =FFTHR
192 str r2, [r3]

```



```

193
194     b B10
195
196 error_loop:
197
198 undefined_instruction:
199 software_interrupt:
200 prefetch_abort:
201 data_abort:
202 not_used;
203 IRQ:
204 FIQ:
205
206     b error_loop
207
208 // Here is end of program

```

다음은 로더를 위한 스크립트 파일 `led-ld-script`이다. 어셈블을 하고 코드의 시작 주소를 지정해 주기 위해서 필요하다. ARM은 리셋 직후 0x0번지의 명령어를 실행하므로 프로그램 코드의 시작 주소가 0x0이 되도록 로더를 설정해야 한다.

```

1 OUTPUT_FORMAT("elf32-littlearm", "elf32-littlearm", "elf32-littlearm")
2 OUTPUT_ARCH(arm)
3 ENTRY(_start)
4 SECTIONS
5 {
6     . = 0x00000000;
7
8     . = ALIGN(4);
9     .text : { *(.text) }
10
11     . = ALIGN(4);
12     .text : { *(.rodata) }
13
14     . = ALIGN(4);
15     .text : { *(.data) }
16
17     . = ALIGN(4);
18     .text : { *(.got) }
19
20     . = ALIGN(4);
21     .text : { *(.bss) }
22 }

```

다음은 Makefile이다.

```

1
2 CC = arm-linux-gcc
3 LD = arm-linux-ld
4 OC = arm-linux-objcopy
5
6 INCLUDE = -nostdinc -I. -I$(TOPDIR)/include
7
8 CFLAGS = $(INCLUDE)
9 CFLAGS += -Wall -Wstrict-prototypes -Wno-trigraphs -O2
10 CFLAGS += -fno-strict-aliasing -fno-common -pipe -mapcs-32

```

gcc 버전이 3이 넘어가면서 바뀐 옵션이 있다. `short-load-bytes` 옵션은 이름이 `alignment-traps`로 바뀌고 디폴트로 설정된다. 또 `-Wa,mxscale` 옵션은 필요없다. PXA27x는 armv5 명령어 셋을 사용하므로 다음과 같은 옵션들이 필요하다.

```

11 #CFLAGS += -march=armv5 -Wa,-mxscale -mtune=xscale -mshort-load-bytes -msoft-fl oat -fno-builtin
12 # in gcc-3.x, short-load-bytes is renamed to alignment-traps and seted as defau lt
13 # -Wa,-mxscale option should be removed for gcc 3.x

```

```

14 CFLAGS += -march=armv5 -mtune=xscale -msoft-float -fno-builtin
15
16 START_LDFLAGS = -p -X -T ./led-ld-script
17
18 OCFLAGS = -O binary -R .note -R .comment -S
19
20 BOOT_IMAGE = ledtest_pxa270
21
22 SRCS = led.S
23 OBJS = led.o
24
25 TARGET = ledtest_org
26 PRE_TARGET = ledtest-elf32
27
28 %.o:%.S
29     @echo "Assembler compiling $< ... "
30     $(CC) -c $(CFLAGS) -o $@ $<
31
32 #make image file
33
34 all:$(PRE_TARGET)
35     $(OC) $(OCFLAGS) $(PRE_TARGET) $(TARGET)
36     dd if=$(TARGET) of=$(BOOT_IMAGE) bs=1k conv=sync
37
38 $(PRE_TARGET):$(OBJS)
39     $(LD) $(START_LDFLAGS) -o $@ $(OBJS)
40
41 clean:
42     rm -f *.o
43     rm -f $(PRE_TARGET)
44     rm -f $(TARGET)
45

```

혹시 어셈블을 할 때 old and new-style option 이라는 에러가 나오면 새로운 버전의 gcc에 맞는 옵션이 아니라는 것이므로 옵션을 수정해야 한다. google에서 설정한 옵션들을 검색해보면 같은 에러에 관해 질문한 내용들이 많을 것이다. 찾아서 답변들을 읽고 고치거나 삭제하면 된다.

이제 다음과 같이 푸징을 한다.

```
gurutio@gioserver:/home/SKKU/bulverde/jflashmm
Flash_8803_2_32.dat Flash_8813_2_32.dat jflashmm25x32 pxa27x16.dat
Flash_8806_2_32.dat Flash_8814_2_32.dat jflashmm27x16 pxa27x32.dat
Flash_8807_2_32.dat Flash_8815_1_16.dat jflashmm27x32 RelNote_JFlashmm.htm
Flash_880B_2_32.dat Flash_8815_2_32.dat jflashmm.dsp SWLicense.pdf
Flash_880C_1_16.dat Flash_8816_2_32.dat jflashmm.dsw testfile.bin
Flash_880C_2_32.dat giveio.inf jflashmm.mak
Flash_880D_1_16.dat giveio.ini jflashmm.ncb
Flash_880D_2_32.dat giveio.sys jflashmm_skkcu
[root@gioserver jflashmm]# jflashmm bulbcx ledtest_pxa270 p

JFLASH Version 5.01.007
COPYRIGHT (C) 2000 - 2003 Intel Corporation

PLATFORM SELECTION:
Processor= PXA27x
Development System= Mainstone
Data Version= 1.00.002

PXA27x revision C0
Found flash type: 28F128K3
The last 90 percent of image file is all zeros
Would you like to save time by not programming that area? [y/n]: y

Unlocking block at address 0
Erasing block at address 0
Starting programming
Using BUFFER programming mode...
Programming done
Starting Verify
verify error at address = 0 exp_dat = ea000006 act_dat = a000006

[root@gioserver jflashmm]#
[root@gioserver jflashmm]#
```

테스트 프로그램을 다운로드 한 후 보드를 리셋하고 이 핀을 확인해서 신호가 출력되고 시리얼을 통해서 문자도 출력된다면 jflash도 정상적으로 동작하고 보드도 정상적이라는 것을 알 수 있다. 이제 최소한의 하드웨어 상태를 알았으니 부트 로더를 포팅할 준비가 끝났다.

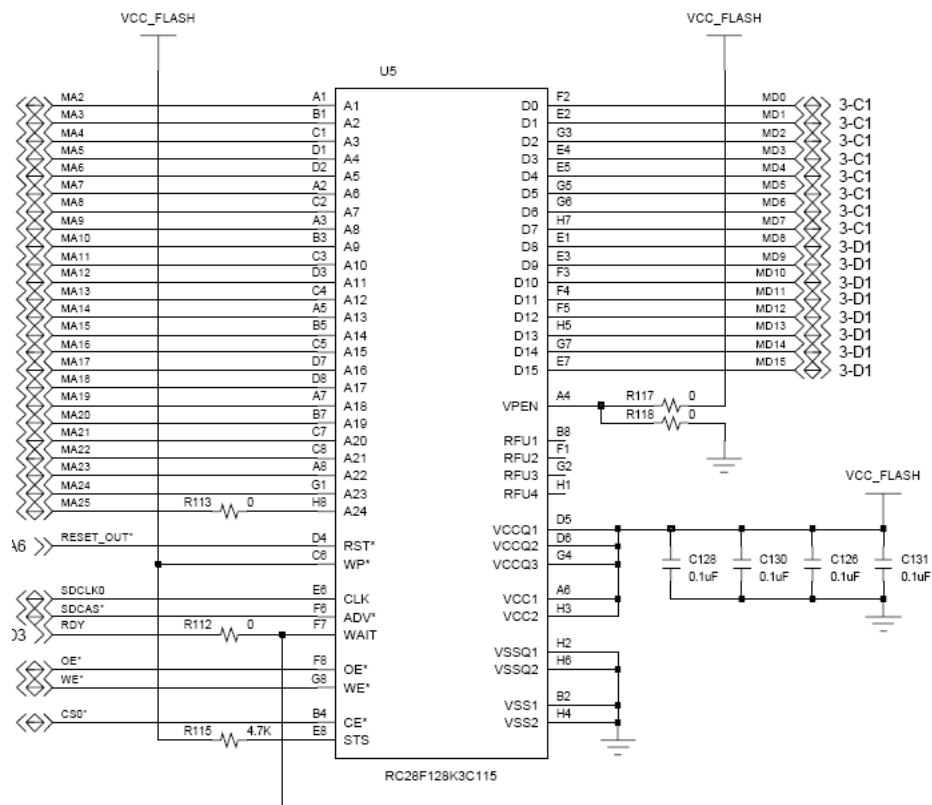
2005-11-29 현재 왜 verify 에러가 발생하는지 알아내지 못함.

메모리관련 회로 이해

SKKU의 보드에서 부트 로더에 필요한 사항들을 살펴본다.

먼저 메모리에 대해서 살펴보면 SKKU 보드에는 2개의 플래시 롬과 2개의 SDRAM이 있다. 메모리 컨트롤에 관한 사항은 Intel PXA27x Processor Family Developer's Manual에서 chap 6 Memory Controller를 참조한다.

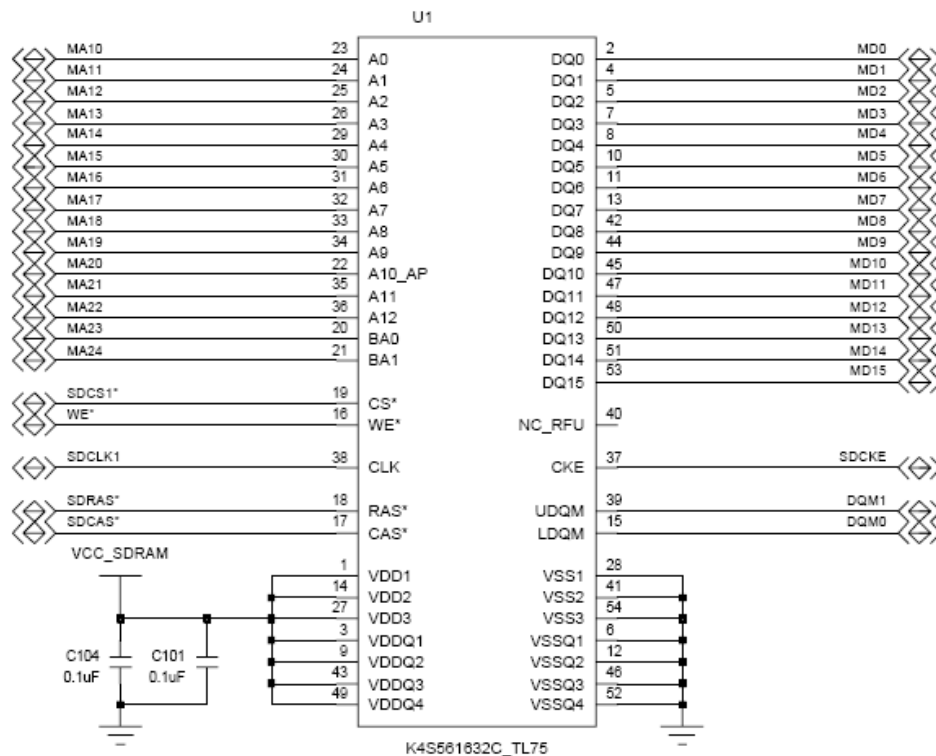
플래시 롬은 인텔의 Strata Flash 28F128K3를 사용한다. 이 플래시 메모리는 25ns 시간이 소모되는 비동기식 접근과 13ns 시간이 소모되는 동기식 접근이 지원된다. 흔히 사용하는 Strata Flash 28F128J3A는 비동기식 접근만 지원되는데 반해 이 플래시는 좀더 빠른 동기식 접근과 버스트 모드를 지원한다. 모델 번호에서 알 수 있듯이 128Mbit의 크기를 가지며 128블록으로 구성되어 있어서 한 섹터당 크기가 1Mbit, 즉 256Kbyte가 되고 NOR 플래시 이다. K3는 3volt 전원이 필요하다는 것을 나타낸다. 칩 하나의 크기가 128Mbit, 16Mbyte이고 두 개를 사용하므로 32Mbyte의 플래시 롬을 가지게 된다. 다음은 SKKU 보드의 플래시 회로이다.



하나의 28F128k3은 8M 개의 워드로 구성되므로 23개의 주소 핀이 필요하다. 그래서 A<24>의 연결을 선택할 수 있도록 했고 R113을 없애도 된다. 버스 폭이 32비트이므로 MA<1:0>은 주소 지정에 필요 없으므로 MA<24:2>를 사용한다. SCDLK0을 클럭으로 사용하고 nCS0으로 칩 셀렉트를 하므로 플래시 롬의 시작 주소는 0x00000000이 된다. 즉 롬을 위한 बैं크 중에서 첫번째 बैं크로 지정된다. SKKU 보드에서는 비동기로 사용하므로 WAIT을 연결하지 않고 STS에 하이 신호를 인가한다. 또 VPEN에 하이 신호를 인가해서 데이터를 프로그래밍할 수 있도록 한다.

SDRAM은 삼성의 K4S561632C-TL75를 사용한다. 하나의 칩이 4Mbit X 16bit X 4banks로 구성되어 있어서 크기가 256Mbit, 즉 32MByte가 된다. 하나의 칩의 데이터 폭이 16비트이고 4개의 बैं크로 이루어져있다. 매뉴얼을 확인하면 TL75가 최대 133MHz로 동작할 수 있다는 것을 나타낸

다고 알 수 있다. 매뉴얼에서 확인해야 할 사항들이 CAS latency가 2&3이고 64ms의 리프레시 주기를 가지며 CL=3라는 것을 알아두어야 한다. SDRAM에 관한 기본적인 사항들은 <http://kelp.or.kr/korweblog/stories.php?story=04/12/15/5293387&topic=30> 에 조형기님의 강좌를 참조한다. 다음은 SKKU 보드의 회로도중 SDRAM에 관한 부분이다.



칩 셀렉트 핀인 nCS에는 SDCS1을 연결했다. SDCS0이 아니라 SDCS1을 연결한 특별한 이유는 없다. 어쨌든 SDCS1을 연결했으므로 SDRAM의 시작 주소는 0xA4000000이 된다. PXA270의 SDCS0~3 핀은 SDRAM 파티션을 선택하기 위한 핀이다. SDCS0은 물리 주소 0xA0000000에서 시작하고 한 파티션의 크기는 64MByte이다. 따라서 다음 파티션 SDCS1은 0xA4000000이고 각각 0x04000000씩 떨어져있다(Intel PXA27x Processor Family Developer's Manual에서 Figure 6-2 참조). 클럭은 SDCLK1이 연결되어 있고 주소 라인이 10번 핀부터 24번 핀이 연결되어 있다. 주소 라인의 연결은 Intel PXA27x Processor Family Developer's Manual에서 테이블 6-3을 참조한다. 이 테이블은 일반적인 주소 지정 방식을 사용하고 SDCLK<0>과 SDCLK<3>이 플래시에 사용되지 않을 때, 즉 가장 일반적으로 사용되는 SDCLK가 SDRAM에만 사용되고 nCS가 플래시 롬에 사용되는 상황에서의 주소 라인 사용에 대해서 나열되어 있다. 보드에 사용된 SDRAM의 매뉴얼에서 A0~A12에 관한 내용을 찾아보면 ROW 비트가 RA0~RA12로 13개, COLUMN 비트가 CA0~CA8로 9개, 데이터 폭은 DQ0~DQ15로 16비트, 뱅크 선택은 BA0~BA1로 2비트로 구성되어 있다는 것을 알 수 있다. 테이블 6-3에서 이런 특성을 찾아보면 테이블 가장 위에 MA<24:10>이 사용되어야 한다는 것이 나와있고 MDCNFG라는 레지스터에서 STACK 항목의 값이 00이어야 한다고 나와있다. MDCNFG 레지스터에 대한 내용은 부트 로더 설정에서 살펴본다. 테이블에서 진하게 출력된 핀 번호 MA<24:23>이 뱅크 설정을 위한 BA0~BA1 핀이다. 즉 MA<24>를 BA1에 MA<32>를 BA0에 연결해야 한다. 자세한 주소 지정 방식은 Intel PXA27x Processor Family

Developer's Manual에서 6.4.2.3 SDRAM Memory Size Options을 참조한다. 현재 대부분의 PXA270, PXA25x보드가 삼성의 램을 사용하므로 비슷한 설정을 사용한다. 다른 보드들에 대한 자료를 찾아서 비교해보면 더 잘 이해할 수 있다. 그 외의 핀들에 대한 설명은 SDRAM의 매뉴얼과 Intel PXA27x Processor Family Developer's Manual의 chap 6을 참조한다. 반드시 각 핀에 대한 역할을 이해해야만 부트 로더 설정을 이해하고 필요한 값을 찾을 수 있다.

부트로더에 설정 파일 추가하기

u-boot를 사용하기 위해서는 u-boot와 u-boot를 컴파일하기 위한 크로스 컴파일러 패키지 ELDK를 준비해야 한다. ELDK에 대한 자세한 사항은 <http://www.denx.de/wiki/DULG/ELDK>를 참조한다.

u-boot : <http://sourceforge.net/projects/u-boot/>

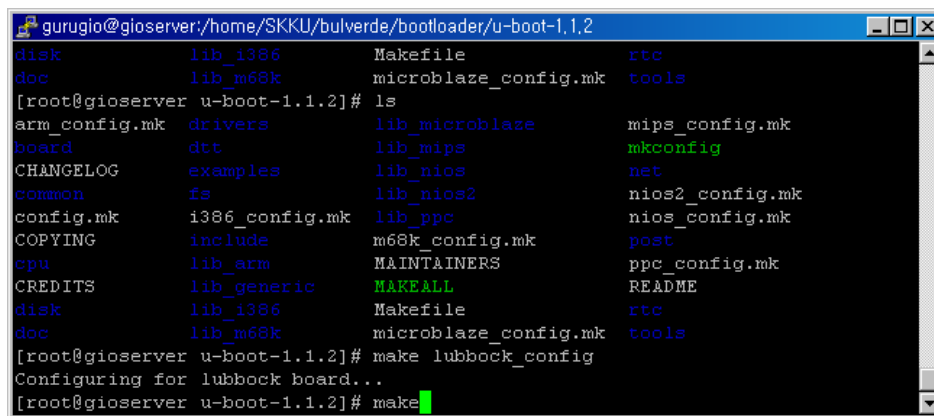
ELDK : <http://www.denx.de/wiki/view/DULG/ELDKAvailability>

하이버스에서 제공하는 arm-linux-ld를 사용하면 'libgcc.a uses VFP instructions, whereas u-boot does not'이라는 에러가 생기면서 컴파일을 할 수 없었다. ELDK 사이트에 들어가서 arm버전 시디 이미지를 다운받는다. mount -o loop arm-버전.iso /media/cdrom 와 같이 mount 명령어로 마운트하고 디렉토리에 들어가서 ./install -d /usr/local/ELDK 명령어로 설치한다. 간단하게 설치할 수 있어서 편하고 2.6 커널을 지원한다.

u-boot는 1.1.3 버전까지 개발됐고 우리는 1.1.2 버전을 사용한다.

먼저 컴파일이 잘 되는지 확인하기 위한 테스트로 lubbock를 위한 설정과 컴파일을 해본다. 다음 그림과 같이 먼저 원하는 보드 타입에 대한 설정을 해주고 그 다음에 컴파일을 실행하면 된다. 우리는 board 디렉토리 밑에 skku 라는 디렉토리를 만들고 PXA270 프로세서와 우리 보드에 맞는 코드들을 추가할 것이므로 다음과 같은 명령어를 사용하게 된다.

```
make skku_config
make
```



```
gurugio@gioserver:/home/SKKU/bulverde/bootloader/u-boot-1.1.2
disk      lib_i386      Makefile      rtc
doc       lib_m68k      microblaze_config.mk  tools
[root@gioserver u-boot-1.1.2]# ls
arm_config.mk  drivers      lib_microblaze  mips_config.mk
board          dtb          lib_mips        mkconfig
CHANGELOG     examples    lib_nios        net
common        fs           lib_nios2       nios2_config.mk
config.mk     i386_config.mk lib_ppc         nios_config.mk
COPYING       include     m68k_config.mk  post
cpu           lib_arm     MAINTAINERS     ppc_config.mk
CREDITS       lib_generic MAKEALL         README
disk          lib_i386    Makefile        rtc
doc           lib_m68k    microblaze_config.mk  tools
[root@gioserver u-boot-1.1.2]# make lubbock_config
Configuring for lubbock board...
[root@gioserver u-boot-1.1.2]# make
```

그 외에 make clobber가 있는데 이는 보드에 대한 설정 파일들까지 삭제하게 된다. make clean은 일반적으로 사용하는 것과 같이 컴파일된 목적 파일들을 지운다.

다음 그림과 같이 u-boot.bin 파일이 생성되면 컴파일이 정상적으로 완료된 것이다. 이 u-boot.bin 파일이 플래시 롬에 퓨징될 바이너리 코드이다. 이렇게 바이너리 코드가 이상없이 생성된다면 크로스 컴파일러가 정상적으로 설치된 것이므로 우리 보드에 대한 설정을 시작한다.

```
gurugio@gioserver:/home/SKKU/bulverde/bootloader/u-boot-1.1.2
ck.a cpu/pxa/libpxa.a lib_arm/libarm.a fs/cramfs/libcramfs.a fs/fat/libfat.a
fs/fdos/libfdos.a fs/jffs2/libjffs2.a fs/reiserfs/libreiserfs.a fs/ext2/libext
2fs.a net/libnet.a disk/libdisk.a rtc/librtc.a dtb/libdtb.a drivers/libdrivers
.a drivers/sk98lin/libsk98lin.a post/libpost.a post/cpu/libcpu.a common/libcom
mon.a |sed -n -e 's/.*\(_u_boot_cmd_.*\)/-u\1/p'|sort|uniq';\
arm-linux-ld -Bstatic -T /home/SKKU/bulverde/bootloader/u-boot-1.1.2/b
oard/lubbock/u-boot.lds -Ttext 0xa3080000 $UNDEF_SYM cpu/pxa/start.o \
--start-group lib_generic/libgeneric.a board/lubbock/liblubbo
ck.a cpu/pxa/libpxa.a lib_arm/libarm.a fs/cramfs/libcramfs.a fs/fat/libfat.a fs
/fdos/libfdos.a fs/jffs2/libjffs2.a fs/reiserfs/libreiserfs.a fs/ext2/libext2f
s.a net/libnet.a disk/libdisk.a rtc/librtc.a dtb/libdtb.a drivers/libdrivers.a
drivers/sk98lin/libsk98lin.a post/libpost.a post/cpu/libcpu.a common/libcommo
n.a --no-warn-mismatch -L /usr/local/ELDK/usr/bin/../../lib/gcc-lib/arm-linux/3.3
.3 -lgcc --end-group \
-Map u-boot.map -o u-boot
arm-linux-objcopy --gap-fill=0xff -O srec u-boot u-boot.srec
arm-linux-objcopy --gap-fill=0xff -O binary u-boot u-boot.bin
[root@gioserver u-boot-1.1.2]#
```

다음은 우리 보드에 맞는 설정을 추가해주는 것이다. 다음 과정을 순서대로 실행하면 된다.

1. Makefile 수정

Makefile을 열면 다음과 같은 부분이 있다.

```
gurugio@gioserver:/home/SKKU/bulverde/bootloader/u-boot-1.1.2
41
42 TOPDIR := $(shell if [ "$$PWD" != "" ]; then echo $$PWD; else pwd; fi)
43 export TOPDIR
44
45 ifeq (include/config.mk,$(wildcard include/config.mk))
46 # load ARCH, BOARD, and CPU configuration
47 include include/config.mk
48 export ARCH CPU BOARD VENDOR SOC
49 # load other configuration
50 include $(TOPDIR)/config.mk
51
52 ifndef CROSS_COMPILE
53 ifeq ($(HOSTARCH),ppc)
54 CROSS_COMPILE =
55 else
56 ifeq ($(ARCH),ppc)
57 CROSS_COMPILE = ppc_8xx-
58 endif
59 ifeq ($(ARCH),arm)
60 CROSS_COMPILE = arm-linux-
61 endif
62 ifeq ($(ARCH),i386)
63 ifeq ($(HOSTARCH),i386)
64 CROSS_COMPILE = i386-linux-
65 endif
66 endif
67 endif
68
69 #
70 #
71 #
72 #
73 #
74 #
75 #
76 #
77 #
78 #
79 #
80 #
81 #
82 #
83 #
84 #
85 #
86 #
87 #
88 #
89 #
90 #
91 #
92 #
93 #
94 #
95 #
96 #
97 #
98 #
99 #
100 #
101 #
102 #
103 #
104 #
105 #
106 #
107 #
108 #
109 #
110 #
111 #
112 #
113 #
114 #
115 #
116 #
117 #
118 #
119 #
120 #
121 #
122 #
123 #
124 #
125 #
126 #
127 #
128 #
129 #
130 #
131 #
132 #
133 #
134 #
135 #
136 #
137 #
138 #
139 #
140 #
141 #
142 #
143 #
144 #
145 #
146 #
147 #
148 #
149 #
150 #
151 #
152 #
153 #
154 #
155 #
156 #
157 #
158 #
159 #
160 #
161 #
162 #
163 #
164 #
165 #
166 #
167 #
168 #
169 #
170 #
171 #
172 #
173 #
174 #
175 #
176 #
177 #
178 #
179 #
180 #
181 #
182 #
183 #
184 #
185 #
186 #
187 #
188 #
189 #
190 #
191 #
192 #
193 #
194 #
195 #
196 #
197 #
198 #
199 #
200 #
201 #
202 #
203 #
204 #
205 #
206 #
207 #
208 #
209 #
210 #
211 #
212 #
213 #
214 #
215 #
216 #
217 #
218 #
219 #
220 #
221 #
222 #
223 #
224 #
225 #
226 #
227 #
228 #
229 #
230 #
231 #
232 #
233 #
234 #
235 #
236 #
237 #
238 #
239 #
240 #
241 #
242 #
243 #
244 #
245 #
246 #
247 #
248 #
249 #
250 #
251 #
252 #
253 #
254 #
255 #
256 #
257 #
258 #
259 #
260 #
261 #
262 #
263 #
264 #
265 #
266 #
267 #
268 #
269 #
270 #
271 #
272 #
273 #
274 #
275 #
276 #
277 #
278 #
279 #
280 #
281 #
282 #
283 #
284 #
285 #
286 #
287 #
288 #
289 #
290 #
291 #
292 #
293 #
294 #
295 #
296 #
297 #
298 #
299 #
300 #
301 #
302 #
303 #
304 #
305 #
306 #
307 #
308 #
309 #
310 #
311 #
312 #
313 #
314 #
315 #
316 #
317 #
318 #
319 #
320 #
321 #
322 #
323 #
324 #
325 #
326 #
327 #
328 #
329 #
330 #
331 #
332 #
333 #
334 #
335 #
336 #
337 #
338 #
339 #
340 #
341 #
342 #
343 #
344 #
345 #
346 #
347 #
348 #
349 #
350 #
351 #
352 #
353 #
354 #
355 #
356 #
357 #
358 #
359 #
360 #
361 #
362 #
363 #
364 #
365 #
366 #
367 #
368 #
369 #
370 #
371 #
372 #
373 #
374 #
375 #
376 #
377 #
378 #
379 #
380 #
381 #
382 #
383 #
384 #
385 #
386 #
387 #
388 #
389 #
390 #
391 #
392 #
393 #
394 #
395 #
396 #
397 #
398 #
399 #
400 #
401 #
402 #
403 #
404 #
405 #
406 #
407 #
408 #
409 #
410 #
411 #
412 #
413 #
414 #
415 #
416 #
417 #
418 #
419 #
420 #
421 #
422 #
423 #
424 #
425 #
426 #
427 #
428 #
429 #
430 #
431 #
432 #
433 #
434 #
435 #
436 #
437 #
438 #
439 #
440 #
441 #
442 #
443 #
444 #
445 #
446 #
447 #
448 #
449 #
450 #
451 #
452 #
453 #
454 #
455 #
456 #
457 #
458 #
459 #
460 #
461 #
462 #
463 #
464 #
465 #
466 #
467 #
468 #
469 #
470 #
471 #
472 #
473 #
474 #
475 #
476 #
477 #
478 #
479 #
480 #
481 #
482 #
483 #
484 #
485 #
486 #
487 #
488 #
489 #
490 #
491 #
492 #
493 #
494 #
495 #
496 #
497 #
498 #
499 #
500 #
501 #
502 #
503 #
504 #
505 #
506 #
507 #
508 #
509 #
510 #
511 #
512 #
513 #
514 #
515 #
516 #
517 #
518 #
519 #
520 #
521 #
522 #
523 #
524 #
525 #
526 #
527 #
528 #
529 #
530 #
531 #
532 #
533 #
534 #
535 #
536 #
537 #
538 #
539 #
540 #
541 #
542 #
543 #
544 #
545 #
546 #
547 #
548 #
549 #
550 #
551 #
552 #
553 #
554 #
555 #
556 #
557 #
558 #
559 #
560 #
561 #
562 #
563 #
564 #
565 #
566 #
567 #
568 #
569 #
570 #
571 #
572 #
573 #
574 #
575 #
576 #
577 #
578 #
579 #
580 #
581 #
582 #
583 #
584 #
585 #
586 #
587 #
588 #
589 #
590 #
591 #
592 #
593 #
594 #
595 #
596 #
597 #
598 #
599 #
600 #
601 #
602 #
603 #
604 #
605 #
606 #
607 #
608 #
609 #
610 #
611 #
612 #
613 #
614 #
615 #
616 #
617 #
618 #
619 #
620 #
621 #
622 #
623 #
624 #
625 #
626 #
627 #
628 #
629 #
630 #
631 #
632 #
633 #
634 #
635 #
636 #
637 #
638 #
639 #
640 #
641 #
642 #
643 #
644 #
645 #
646 #
647 #
648 #
649 #
650 #
651 #
652 #
653 #
654 #
655 #
656 #
657 #
658 #
659 #
660 #
661 #
662 #
663 #
664 #
665 #
666 #
667 #
668 #
669 #
670 #
671 #
672 #
673 #
674 #
675 #
676 #
677 #
678 #
679 #
680 #
681 #
682 #
683 #
684 #
685 #
686 #
687 #
688 #
689 #
690 #
691 #
692 #
693 #
694 #
695 #
696 #
697 #
698 #
699 #
700 #
701 #
702 #
703 #
704 #
705 #
706 #
707 #
708 #
709 #
710 #
711 #
712 #
713 #
714 #
715 #
716 #
717 #
718 #
719 #
720 #
721 #
722 #
723 #
724 #
725 #
726 #
727 #
728 #
729 #
730 #
731 #
732 #
733 #
734 #
735 #
736 #
737 #
738 #
739 #
740 #
741 #
742 #
743 #
744 #
745 #
746 #
747 #
748 #
749 #
750 #
751 #
752 #
753 #
754 #
755 #
756 #
757 #
758 #
759 #
760 #
761 #
762 #
763 #
764 #
765 #
766 #
767 #
768 #
769 #
770 #
771 #
772 #
773 #
774 #
775 #
776 #
777 #
778 #
779 #
780 #
781 #
782 #
783 #
784 #
785 #
786 #
787 #
788 #
789 #
790 #
791 #
792 #
793 #
794 #
795 #
796 #
797 #
798 #
799 #
800 #
801 #
802 #
803 #
804 #
805 #
806 #
807 #
808 #
809 #
810 #
811 #
812 #
813 #
814 #
815 #
816 #
817 #
818 #
819 #
820 #
821 #
822 #
823 #
824 #
825 #
826 #
827 #
828 #
829 #
830 #
831 #
832 #
833 #
834 #
835 #
836 #
837 #
838 #
839 #
840 #
841 #
842 #
843 #
844 #
845 #
846 #
847 #
848 #
849 #
850 #
851 #
852 #
853 #
854 #
855 #
856 #
857 #
858 #
859 #
860 #
861 #
862 #
863 #
864 #
865 #
866 #
867 #
868 #
869 #
870 #
871 #
872 #
873 #
874 #
875 #
876 #
877 #
878 #
879 #
880 #
881 #
882 #
883 #
884 #
885 #
886 #
887 #
888 #
889 #
890 #
891 #
892 #
893 #
894 #
895 #
896 #
897 #
898 #
899 #
900 #
901 #
902 #
903 #
904 #
905 #
906 #
907 #
908 #
909 #
910 #
911 #
912 #
913 #
914 #
915 #
916 #
917 #
918 #
919 #
920 #
921 #
922 #
923 #
924 #
925 #
926 #
927 #
928 #
929 #
930 #
931 #
932 #
933 #
934 #
935 #
936 #
937 #
938 #
939 #
940 #
941 #
942 #
943 #
944 #
945 #
946 #
947 #
948 #
949 #
950 #
951 #
952 #
953 #
954 #
955 #
956 #
957 #
958 #
959 #
960 #
961 #
962 #
963 #
964 #
965 #
966 #
967 #
968 #
969 #
970 #
971 #
972 #
973 #
974 #
975 #
976 #
977 #
978 #
979 #
980 #
981 #
982 #
983 #
984 #
985 #
986 #
987 #
988 #
989 #
990 #
991 #
992 #
993 #
994 #
995 #
996 #
997 #
998 #
999 #
1000 #
1001 #
1002 #
1003 #
1004 #
1005 #
1006 #
1007 #
1008 #
1009 #
1010 #
1011 #
1012 #
1013 #
1014 #
1015 #
1016 #
1017 #
1018 #
1019 #
1020 #
1021 #
1022 #
1023 #
1024 #
1025 #
1026 #
1027 #
1028 #
1029 #
1030 #
1031 #
1032 #
1033 #
1034 #
1035 #
1036 #
1037 #
1038 #
1039 #
1040 #
1041 #
1042 #
1043 #
1044 #
1045 #
1046 #
1047 #
1048 #
1049 #
1050 #
1051 #
1052 #
1053 #
1054 #
1055 #
1056 #
1057 #
1058 #
1059 #
1060 #
1061 #
1062 #
1063 #
1064 #
1065 #
1066 #
1067 #
1068 #
1069 #
1070 #
1071 #
1072 #
1073 #
1074 #
1075 #
1076 #
1077 #
1078 #
1079 #
1080 #
1081 #
1082 #
1083 #
1084 #
1085 #
1086 #
1087 #
1088 #
1089 #
1090 #
1091 #
1092 #
1093 #
1094 #
1095 #
1096 #
1097 #
1098 #
1099 #
1100 #
1101 #
1102 #
1103 #
1104 #
1105 #
1106 #
1107 #
1108 #
1109 #
1110 #
1111 #
1112 #
1113 #
1114 #
1115 #
1116 #
1117 #
1118 #
1119 #
1120 #
1121 #
1122 #
1123 #
1124 #
1125 #
1126 #
1127 #
1128 #
1129 #
1130 #
1131 #
1132 #
1133 #
1134 #
1135 #
1136 #
1137 #
1138 #
1139 #
1140 #
1141 #
1142 #
1143 #
1144 #
1145 #
1146 #
1147 #
1148 #
1149 #
1150 #
1151 #
1152 #
1153 #
1154 #
1155 #
1156 #
1157 #
1158 #
1159 #
1160 #
1161 #
1162 #
1163 #
1164 #
1165 #
1166 #
1167 #
1168 #
1169 #
1170 #
1171 #
1172 #
1173 #
1174 #
1175 #
1176 #
1177 #
1178 #
1179 #
1180 #
1181 #
1182 #
1183 #
1184 #
1185 #
1186 #
1187 #
1188 #
1189 #
1190 #
1191 #
1192 #
1193 #
1194 #
1195 #
1196 #
1197 #
1198 #
1199 #
1200 #
1201 #
1202 #
1203 #
1204 #
1205 #
1206 #
1207 #
1208 #
1209 #
1210 #
1211 #
1212 #
1213 #
1214 #
1215 #
1216 #
1217 #
1218 #
1219 #
1220 #
1221 #
1222 #
1223 #
1224 #
1225 #
1226 #
1227 #
1228 #
1229 #
1230 #
1231 #
1232 #
1233 #
1234 #
1235 #
1236 #
1237 #
1238 #
1239 #
1240 #
1241 #
1242 #
1243 #
1244 #
1245 #
1246 #
1247 #
1248 #
1249 #
1250 #
1251 #
1252 #
1253 #
1254 #
1255 #
1256 #
1257 #
1258 #
1259 #
1260 #
1261 #
1262 #
1263 #
1264 #
1265 #
1266 #
1267 #
1268 #
1269 #
1270 #
1271 #
1272 #
1273 #
1274 #
1275 #
1276 #
1277 #
1278 #
1279 #
1280 #
1281 #
1282 #
1283 #
1284 #
1285 #
1286 #
1287 #
1288 #
1289 #
1290 #
1291 #
1292 #
1293 #
1294 #
1295 #
1296 #
1297 #
1298 #
1299 #
1300 #
1301 #
1302 #
1303 #
1304 #
1305 #
1306 #
1307 #
1308 #
1309 #
1310 #
1311 #
1312 #
1313 #
1314 #
1315 #
1316 #
1317 #
1318 #
1319 #
1320 #
1321 #
1322 #
1323 #
1324 #
1325 #
1326 #
1327 #
1328 #
1329 #
1330 #
1331 #
1332 #
1333 #
1334 #
1335 #
1336 #
1337 #
1338 #
1339 #
1340 #
1341 #
1342 #
1343 #
1344 #
1345 #
1346 #
1347 #
1348 #
1349 #
1350 #
1351 #
1352 #
1353 #
1354 #
1355 #
1356 #
1357 #
1358 #
1359 #
1360 #
1361 #
1362 #
1363 #
1364 #
1365 #
1366 #
1367 #
1368 #
1369 #
1370 #
1371 #
1372 #
1373 #
1374 #
1375 #
1376 #
1377 #
1378 #
1379 #
1380 #
1381 #
1382 #
1383 #
1384 #
1385 #
1386 #
1387 #
1388 #
1389 #
1390 #
1391 #
1392 #
1393 #
1394 #
1395 #
1396 #
1397 #
1398 #
1399 #
1400 #
1401 #
1402 #
1403 #
1404 #
1405 #
1406 #
1407 #
1408 #
1409 #
1410 #
1411 #
1412 #
1413 #
1414 #
1415 #
1416 #
1417 #
1418 #
1419 #
1420 #
1421 #
1422 #
1423 #
1424 #
1425 #
1426 #
1427 #
1428 #
1429 #
1430 #
1431 #
1432 #
1433 #
1434 #
1435 #
1436 #
1437 #
1438 #
1439 #
1440 #
1441 #
1442 #
1443 #
1444 #
1445 #
1446 #
1447 #
1448 #
1449 #
1450 #
1451 #
1452 #
1453 #
1454 #
1455 #
1456 #
1457 #
1458 #
1459 #
1460 #
1461 #
1462 #
1463 #
1464 #
1465 #
1466 #
1467 #
1468 #
1469 #
1470 #
1471 #
1472 #
1473 #
1474 #
1475 #
1476 #
1477 #
1478 #
1479 #
1480 #
1481 #
1482 #
1483 #
1484 #
1485 #
1486 #
1487 #
1488 #
1489 #
1490 #
1491 #
1492 #
1493 #
1494 #
1495 #
1496 #
1497 #
1498 #
1499 #
1500 #
1501 #
1502 #
1503 #
1504 #
1505 #
1506 #
1507 #
1508 #
1509 #
1510 #
1511 #
1512 #
1513 #
1514 #
1515 #
1516 #
1517 #
1518 #
1519 #
1520 #
1521 #
1522 #
1523 #
1524 #
1525 #
1526 #
1527 #
1528 #
1529 #
1530 #
1531 #
1532 #
1533 #
1534 #
1535 #
1536 #
1537 #
1538 #
1539 #
1540 #
1541 #
1542 #
1543 #
1544 #
1545 #
1546 #
1547 #
1548 #
1549 #
1550 #
1551 #
1552 #
1553 #
1554 #
1555 #
1556 #
1557 #
1558 #
1559 #
1560 #
1561 #
1562 #
1563 #
1564 #
1565 #
1566 #
1567 #
1568 #
1569 #
1570 #
1571 #
1572 #
1573 #
1574 #
1575 #
1576 #
1577 #
1578 #
1579 #
1580 #
1581 #
1582 #
1583 #
1584 #
1585 #
1586 #
1587 #
1588 #
1589 #
1590 #
1591 #
1592 #
1593 #
1594 #
1595 #
1596 #
1597 #
1598 #
1599 #
1600 #
1601 #
1602 #
1603 #
1604 #
1605 #
1606 #
1607 #
1608 #
1609 #
1610 #
1611 #
1612 #
1613 #
1614 #
1615 #
1616 #
1617 #
1618 #
1619 #
1620 #
1621 #
1622 #
1623 #
1624 #
1625 #
1626 #
1627 #
1628 #
1629 #
1630 #
1631 #
1632 #
1633 #
1634 #
1635 #
1636 #
1637 #
1638 #
1639 #
1640 #
1641 #
1642 #
1643 #
1644 #
1645 #
1646 #
1647 #
1648 #
1649 #
1650 #
1651 #
1652 #
1653 #
1654 #
1655 #
1656 #
1657 #
1658 #
1659 #
1660 #
1661 #
1662 #
1663 #
1664 #
1665 #
1666 #
1667 #
1668 #
1669 #
1670 #
1671 #
1672 #
1673 #
1674 #
1675 #
1676 #
1677 #
1678 #
1679 #
1680 #
1681 #
1682 #
1683 #
1684 #
1685 #
1686 #
1687 #
1688 #
1689 #
1690 #
1691 #
1692 #
1693 #
1694 #
1695 #
1696 #
1697 #
1698 #
1699 #
1700 #
1701 #
1702 #
1703 #
1704 #
1705 #
1706 #
1707 #
1708 #
1709 #
1710 #
1711 #
1712 #
1713 #
1714 #
1715 #
1716 #
1717 #
1718 #
1719 #
1720 #
1721 #
1722 #
1723 #
1724 #
1725 #
1726 #
1727 #
1728 #
1729 #
1730 #
1731 #
1732 #
1733 #
1734 #
1735 #
1736 #
1737 #
1738 #
1739 #
1740 #
1741 #
1742 #
1743 #
1744 #
1745 #
1746 #
1747 #
1748 #
1749 #
1750 #
1751 #
1752 #
1753 #
1754 #
1755 #
1756 #
1757 #
1758 #
1759 #
1760 #
1761 #
1762 #
1763 #
1764 #
1765 #
1766 #
1767 #
1768 #
1769 #
1770 #
1771 #
1772 #
1773 #
1774 #
1775 #
1776 #
1777 #
1778 #
1779 #
1780 #
1781 #
1782 #
1783 #
1784 #
1785 #
1786 #
1787 #
1788 #
1789 #
1790 #
1791 #
1792 #
1793 #
1794 #
1795 #
1796 #
1797 #
1798 #
1799 #
1800 #
1801 #
1802 #
1803 #
1804 #
1805 #
1806 #
1807 #
1808 #
1809 #
1810 #
1811 #
1812 #
1813 #
1814 #
1815 #
1816 #
1817 #
1818 #
1819 #
1820 #
1821 #
1822 #
1823 #
1824 #
1825 #
1826 #
1827 #
1828 #
1829 #
1830 #
1831 #
1832 #
1833 #
1834 #
1835 #
1836 #
1837 #
1838 #
1839 #
1840 #
1841 #
1842 #
1843 #
1844 #
1845 #
1846 #
1847 #
1848 #
1849 #
1850 #
1851 #
1852 #
1853 #
1854 #
1855 #
1856 #
1857 #
1858 #
1859 #
1860 #
1861 #
1862 #
1863 #
1864 #
1865 #
1866 #
1867 #
1868 #
1869 #
1870 #
1871 #
1872 #
1873 #
1874 #
1875 #
1876 #
1877 #
1878 #
1879 #
1880 #
1881 #
1882 #
1883 #
1884 #
1885 #
1886 #
1887 #
1888 #
1889 #
1890 #
1891 #
1892 #
1893 #
1894 #
1895 #
1896 #
1897 #
1898 #
1899 #
1900 #
1901 #
1902 #
1903 #
1904 #
1905 #
1906 #
1907 #
1908 #
1909 #
1910 #
1911 #
1912 #
1913 #
1914 #
1915 #
1916 #
1917 #
1918 #
1919 #
1920 #
1921 #
1922 #
1923 #
1924 #
1925 #
1926 #
1927 #
1928 #
1929 #
1930 #
1931 #
1932 #
1933 #
1934 #
1935 #
1936 #
1937 #
1938 #
1939 #
1940 #
1941 #
1942 #
1943 #
1944 #
1945 #
1946 #
1947 #
1948 #
1949 #
1950 #
1951 #
1952 #
1953 #
1954 #
1955 #
1956 #
1957 #
1958 #
1959 #
1960 #
1961 #
1962 #
1963 #
1964 #
1965 #
1966 #
1967 #
1968 #
1969 #
1970 #
1971 #
1972 #
1973 #
1974 #
1975 #
1976 #
1977 #
1978 #
1979 #
1980 #
1981 #
1982 #
1983 #
1984 #
1985 #
1986 #
1987 #
1988 #
1989 #
1990 #
1991 #
1992 #
1993 #
1994 #
1995 #
1996 #
1997 #
1998 #
1999 #
2000 #
2001 #
2002 #
2003 #
2004 #
2005 #
2006 #
2007 #
2008 #
2009 #
2010 #
2011 #
2012 #
2013 #
2014 #
2015 #
2016 #
2017 #
2018 #
2019 #
2020 #
2021 #
2022 #
2023 #
2024 #
2025 #
2026 #
2027 #
2028 #
2029 #
2030 #
2031 #
2032 #
2033 #
2034 #
2035 #
2036 #
2037 #
2038 #
2039 #
2040 #
2041 #
2042 #
2043 #
2044 #
2045 #
2046 #
2047 #
2048 #
2049 #
2050 #
2051 #
2052 #
2053 #
2054 #
2055 #
2056 #
2057 #
2058 #
2059 #
2060 #
2061 #
2062 #
2063 #
2064 #
2065 #
2066 #
2067 #
2068 #
2069 #
2070 #
2071 #
2072 #
2073 #
2074 #
2075 #
2076 #
2077 #
2078 #
2079 #
2080 #
2081 #
2082 #
2083 #
2084 #
2085 #
2086 #
2087 #
2088 #
2089 #
2090 #
2091 #
2092 #
2093 #
2094 #
2095 #
2096 #
2097 #
2098 #
2099 #
2100 #
2101 #
2102 #
2103 #
2104 #
2105 #
2106 #
2107 #
2108 #
2109 #
2110 #
2111 #
2112 #
2113 #
2114 #
2115 #
2116 #
2117 #
2118 #
2119 #
2120 #
2121 #
2122 #
2123 #
2124 #
2125 #
2126 #
2127 #
2128 #
2129 #
2130 #
2131 #
2132 #
2133 #
2134 #
2135 #
2136
```



```
gurugio@gioserver:/home/SKKU/bulverde/bootloader/u-boot-1.1.2_skku
42 TOPDIR := ${shell if [ "$$PWD" != "" ]; then echo $$PWD; else pwd; fi}
43 export TOPDIR
44
45 ifeq (include/config.mk,${wildcard include/config.mk})
46 # load ARCH, BOARD, and CPU configuration
47 include include/config.mk
48 export ARCH CPU BOARD VENDOR SOC
49 # load other configuration
50 include $(TOPDIR)/config.mk
51
52 CROSS_COMPILE = arm-linux-
53
54 ifndef CROSS_COMPILE
55 ifeq ($(HOSTARCH),ppc)
56 CROSS_COMPILE =
57 else
58 ifeq ($(ARCH),ppc)
59 CROSS_COMPILE = ppc_8xx-
60 endif
61 ifeq ($(ARCH),arm)
62 CROSS_COMPILE = arm-linux-
63 endif
64 ifeq ($(ARCH),i386)
```

그리고 필요없는 example 부분을 삭제한다.

```
gurugio@gioserver:/home/SKKU/bulverde/bootloader/u-boot-1.1.2
125
126
127 # The "tools" are needed early, so put this first
128 # Don't include stuff already done in $(LIBS)
129 SUBDIRS = tools \
130     examples \
131     post \
132     post/cpu
133 .PHONY : $(SUBDIRS)
134
135 #####
136 #####
137 #####
```

```
gurugio@gioserver:/home/SKKU/bulverde/bootloader/u-boot-1.1.2
125
126
127 # The "tools" are needed early, so put this first
128 # Don't include stuff already done in $(LIBS)
129 SUBDIRS = tools \
130     post \
131     post/cpu
132 .PHONY : $(SUBDIRS)
133
134 #####
135 #####
136 #####
```

그 다음은 SKKU 보드에 대한 컴파일 설정을 추가한다. PXA250 프로세서를 사용하는 보드들에 대한 설정이 있는데 그 밑에 PXA270 프로세서를 사용하는 SKKU 보드에 대한 설정을 추가한

다. arm pxa270 skku라고 된 구문은 lib_arm/ 디렉토리를 컴파일하고 cpu/pxa270/ 을 컴파일 하고 skku_config를 생성하도록 하는 문장이다. cpu/pxa270 디렉토리는 직접 만들어줘야 한다. cpu/pxa 를 복사해서 만들고 수정해서 쓴다.

```

gurugio@gioserver:/home/SKKU/bulverde/bootloader/u-boot-1.1.2
1390  @./mkconfig ${@:_config=} arm pxa wep250
1391
1392  xaeliax_config : unconfig
1393  @./mkconfig ${@:_config=} arm pxa xaeliax
1394
1395  xm250_config : unconfig
1396  @./mkconfig ${@:_config=} arm pxa xm250
1397
1398  xsengine_config : unconfig
1399  @./mkconfig ${@:_config=} arm pxa xsengine
1400
1401  #####
1402  ## Bulverde Systems
1403  #####
1404
1405  skku_config : unconfig
1406  @./mkconfig ${@:_config=} arm pxa270 skku
1407
1408  #=====
1409  # i386
1410  #=====
1411  #####
1412  ## AMD SC520 CDP
1413  #####
1414  sc520_cdp_config : unconfig
1415  @./mkconfig ${@:_config=} i386 i386 sc520_cdp
1416
1417  sc520_spunk_config : unconfig
1418  @./mkconfig ${@:_config=} i386 i386 sc520_spunk
"Makefile" 1604L, 48886C written
1402,11 88%

```

2. cpu/pxa270/ 디렉토리 추가

cpu/pxa 디렉토리를 복사해서 pxa270 디렉토리를 만든다.

```

gurugio@gioserver:/home/SKKU/bulverde/bootloader/u-boot-1.1.2/cpu
[root@gioserver cpu]# ls
74xx_7xx  arm926ejs  lh7a40x  mpc5xx  mpc8260  nios2  sa1100
arm720t  at91rm9200  mcf52x2  mpc5xxx  mpc85xx  ppc4xx
arm920t  i386  microblaze  mpc8220  mpc8xx  pxa
arm925t  ixp  mips  mpc824x  nios  s3c44b0
[root@gioserver cpu]# cp -a pxa pxa270
[root@gioserver cpu]# ls
74xx_7xx  arm926ejs  lh7a40x  mpc5xx  mpc8260  nios2  s3c44b0
arm720t  at91rm9200  mcf52x2  mpc5xxx  mpc85xx  ppc4xx  sa1100
arm920t  i386  microblaze  mpc8220  mpc8xx  pxa
arm925t  ixp  mips  mpc824x  nios  pxa270
[root@gioserver cpu]#

```

3. board/skku/ 디렉토리 추가

Makefile을 수정한 다음에는 SKKU 보드에 맞는 설정들을 추가한다. 보드 타입에 따른 파일들은 /board 디렉토리에 각 보드 이름의 디렉토리를 만들어서 저장하면 된다. u-boot 1.1.2 버전은 Mainstone 보드를 지원하지 않으므로 PXA250을 사용하는 가장 비슷한 보드인 Lubbock 보드의 디렉토리를 복사해서 사용한다.

```
gurugio@gioserver:/home/SKKU/bulverde/bootloader/u-boot-1.1.2/board
[root@gioserver board]# cp -a lubbock/ skku
[root@gioserver board]# ls skku/ lubbock/
lubbock/:
config.mk flash.c lubbock.c Makefile memsetup.S u-boot.lds

skku/:
config.mk flash.c lubbock.c Makefile memsetup.S u-boot.lds
[root@gioserver board]#
```

그리고 /board/skku/에 있는 lubbock.c 파일을 skku.c 로 바꾸고 Makefile도 lubbock.o를 skku.o 로 바꾼다.

```
gurugio@gioserver:/home/SKKU/bulverde/bootloader/u-boot-1.1.2/board/skku
rm -f tools/gdb/astest tools/gdb/gdbcont tools/gdb/gdbsend
rm -f tools/env/fw_printenv tools/env/fw_setenv
rm -f board/cray/L1/bootscript.c board/cray/L1/bootscript.image
rm -f board/trab/trab_fkt
[root@gioserver u-boot-1.1.2]# cd board/skku/
[root@gioserver skku]# ls
config.mk flash.c Makefile memsetup.S skku.c u-boot.lds
[root@gioserver skku]#
```

```
gurugio@gioserver:/home/SKKU/bulverde/bootloader/u-boot-1.1.2/board/skku
19 # You should have received a copy of the GNU General Public License
20 # along with this program; if not, write to the Free Software
21 # Foundation, Inc., 59 Temple Place, Suite 330, Boston,
22 # MA 02111-1307 USA
23 #
24
25 include $(TOPDIR)/config.mk
26
27 LIB = lib$(BOARD).a
28
29 OBJS      := skku.o flash.o
30 SOBJS     := memsetup.o
31
32 $(LIB): $(OBJS) $(SOBJS)
33         ${AR} crv $@ $(OBJS) $(SOBJS)
34
35 clean:
```

skku.c를 열어보면 다음과 같이 보드 타입 번호를 지정하는 부분이 있다. LUBBOCK로 된 부분을 SKKU로 수정한다.

```
gurugio@gioserver:/home/SKKU/bulverde/bootloader/u-boot-1.1.2/board/skku
42 /* so we do nothing here */
43
44 /* arch number of SKKU-Board */
45 gd->bd->bi_arch_number = MACH_TYPE_SKKU;
46
47 /* adress of boot parameters */
48 gd->bd->bi_boot_params = 0xa0000100;
"skku.c" 75L, 1984C written
```

4. board/skku/u-boot.lds 수정
cpu/pxa/start.o를 참조하도록 된 부분을 cpu/pxa270/start.o로 고친다.

```
gurugio@gioserver:/home/SKKU/bulverde/bootloader/u-boot-1.1.2
23
24 OUTPUT_FORMAT("elf32-littlearm", "elf32-littlearm", "elf32-littlearm")
25 OUTPUT_ARCH(arm)
26 ENTRY(_start)
27 SECTIONS
28 {
29     . = 0x00000000;
30
31     . = ALIGN(4);
32     .text :
33     {
34         cpu/pxa270/start.o    (.text)
35         *(.text)
36     }
37
38     . = ALIGN(4);
39     .rodata : { *(.rodata) }
"board/skku/u-boot.lds" 55L, 1356C written                28,1        57%
```

5. include/asm-arm/mach-type.h 수정

include/asm-arm/mach-types.h 파일을 열어보면 각 보드들에 대한 타입 번호와 보드 타입을 확인하기 위한 매크로들이 들어있다. 여기에 SKKU 보드를 추가한다. 보드 번호는 HYBUS 보드 번호를 사용했다. 이 번호는 리눅스 커널에 넘기기 위한 것이다.

```
gurugio@gioserver:/home/SKKU/bulverde/bootloader/u-boot-1.1.2/include/asm-arm
625 #define MACH_TYPE_KB9200                612
626 #define MACH_TYPE_SX1                   613
627 #define MACH_TYPE_SKKU                   65280
628
629
630 #ifdef CONFIG_ARCH_SKKU
631 # ifdef machine_arch_type
632 #  undef machine_arch_type
633 #  define machine_arch_type __machine_arch_type
634 # else
635 #  define machine_arch_type MACH_TYPE_SKKU
636 # endif
637 # define machine_is_skku() (machine_arch_type == MACH_TYPE_SKKU)
638 #else
639 # define machine_is_skku() (0)
640 #endif
641
642
643 #ifdef CONFIG_ARCH_EBSA110
644 # ifdef machine_arch_type
645 #  undef machine_arch_type
"mach-types.h" 7973L, 215306C written                627,14        7%
```

6. include/configs/skku.h 생성

include/configs/lubbock.h 를 skku.h로 복사한다. 이 파일이 u-boot를 위한 설정이 들어있는 파일이다. 일단 SKKU 보드를 위한 파일들을 모두 추가하고 그 다음에 이 설정을 보드에 맞춰서 하나씩 맞춰나간다. 지금은 일단 다음 그림과 같이 보드 이름만 수정하고 #define CONFIG_LCD 1 설정을 #undef CONFIG_LCD로 고친다.

```

gurugio@gioserver:/home/SKKU/bulverde/bootloader/u-boot-1.1.2
42 */
43 #define CONFIG_PXA270      1  /* This is an PXA250 CPU */
44 #define CONFIG_SKKUK      1  /* on an SKKU board */
45 #undef CONFIG_LCD
46 #ifdef CONFIG_LCD
47 #define CONFIG_SHARP_LM8V31
48 #endif
"include/configs/skku.h" 241L, 7192C 45,17 17%

```

7. include/asm-arm/arch-pxa270 생성

include/asm-arm/arch-pxa를 복사해서 include/asm-arm/arch-pxa/arch-pxa270 디렉토리를 만든다.

```

gurugio@gioserver:/home/SKKU/bulverde/bootloader/u-boot-1.1.2/include/asm-arm
arch-arm925t arch-sa1100 hardware.h proc-armv types.h
arch-arm926ejs atomic.h io.h processor.h u-boot-arm.h
arch-at91rm9200 bitops.h mach-types.h ptrace.h u-boot.h
arch-ixp byteorder.h mach-types.h.bak setup.h
[root@gioserver asm-arm]# cp -a arch-pxa arch-pxa270
[root@gioserver asm-arm]# ls
arch-arm720t arch-pxa byteorder.h mach-types.h.bak setup.h
arch-arm920t arch-pxa270 errno.h memory.h sizes.h
arch-arm925t arch-s3c44b0 global_data.h posix_types.h string.h
arch-arm926ejs arch-sa1100 hardware.h proc-armv types.h
arch-at91rm9200 atomic.h io.h processor.h u-boot-arm.h
arch-ixp bitops.h mach-types.h ptrace.h u-boot.h
[root@gioserver asm-arm]#

```

8. 컴파일 옵션 수정

cpu/pxa270/config.mk 를 열어보면 PXA250을 위한 컴파일 설정이 들어있다. 이것을 gcc 3.3.3 버전에 맞고 PXA270에 맞도록 다음과 같이 수정한다.

```

gurugio@gioserver:/home/SKKU/bulverde/bootloader/u-boot-1.1.2
12 # the License, or (at your option) any later version.
13 #
14 # This program is distributed in the hope that it will be useful,
15 # but WITHOUT ANY WARRANTY; without even the implied warranty of
16 # MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
17 # GNU General Public License for more details.
18 #
19 # You should have received a copy of the GNU General Public License
20 # along with this program; if not, write to the Free Software
21 # Foundation, Inc., 59 Temple Place, Suite 330, Boston,
22 # MA 02111-1307 USA
23
24
25 PLATFORM_RELFLAGS += -fno-strict-aliasing -fno-common -ffixed-r8 \
26 -msoft-float
27
28 PLATFORM_CPPFLAGS += -mapcs-32 -march=armv5 -mtune=xscale
23,1 Bot

```

9. 이미지 생성

make skku_config 명령으로 SKKU 보드 관련 설정을 실행시키고 make 로 컴파일을 시작한다.

```

gurugio@gioserver:/home/SKKU/bulverde/bootloader/u-boot-1.1.2
[root@gioserver u-boot-1.1.2]# make skku_config
Configuring for skku board...
[root@gioserver u-boot-1.1.2]# make

```


보드 하드웨어에 맞게 부트 로더 설정 하기

Lubbock 보드의 파일들을 복사해서 사용하므로 Lubbock 보드의 사양에 맞게 설정되어 있다. 메모리 맵 설정과 메모리 컨트롤러 설정, 네트워크 디바이스 등을 주의해서 설정 파일들을 수정해야 한다. 또 프로세서 초기화 코드들이 PXA250에 맞게 작성되어 있다. PXA270 프로세서도 코어는 PXA250 코어를 사용하므로 크게 달라진 부분은 없지만 프로세서 속도가 빨라졌고 이에 따라 클럭 관련 코드를 수정해 주어야 한다. 결론적으로 다음 설정 파일들을 수정하게 된다.

```
board/skku/config.mk  
include/configs/skku.h
```

이 설정 파일들을 수정하고 PXA250에 맞게 작성된 프로세서 초기화 코드를 PXA270에 맞게 약간 수정하면 포팅이 끝난다.

이 파일들을 수정하기 전에 다음 사항들에 대해 이해해야 한다.

- SDRAM의 시작 주소와 크기
 - 시작 주소 : 0xA400 0000, nSDCS1으로 파티션을 선택하므로 0xA000 0000 이 아니라 그 다음 파티션인 0xA400 0000이 된다.
 - 크기 : 0x0400 0000 (64MByte = 32MB X 2개)
- SDRMA의 속도 설정
 - 68
- 플래시 메모리의 시작 주소와 크기
 - 시작 주소 : 0x0000 0000, nCS0을 사용해서 첫번째 파티션이 된다.
 - 크기 : 0x0200 0000 (32MByte = 16MB X 2개)
- 플래시 메모리의 속도 설정
 -
- 부트 로더가 플래시에 써지는 시작 주소와 크기
 - 0x0000 0000
- 부트 로더가 램으로 복사되는 시작 주소와 크기
 - 0xA408 0000 ~ 0xA40995F4, BSS ~0xA40C DF0C
- 부팅 파라미터가 리눅스 커널로 넘겨지는 주소
 - 0xa000 0100
- 커널이 복사되는 시작 주소
 - 0xA400 8000, 0x0004 0000
- 28F128K3 플래시의 CFI 인식 번호
 - 0x8802

프로세서의 속도에 대한 설정은 Intel PXA27x Processor Family Developer' s Manual에서 Table 3-7을 본다. Core Turbo Freq를 520MHz로 맞추기 위한 설정들이 나와있다. 이 테이블에 따라서 레지스터들을 설정해주면 된다.

가장 먼저 include/configs/skku.h 를 수정한다. 기본적으로 Lubbock 보드에 맞게 되어있지만 이것을 한줄씩 SKKU 보드에 맞게 수정해야 하고 PXA250용 레지스터 설정들을 PXA270에 맞게 고

쳐야 한다. 몇가지 새로 추가해야 하는 설정들은 뒤에서 소스들을 살펴보면서 이야기한다. 다음은 skku.h의 내용이다. lubbock.h와 비교하면서 보면 이해하기 더 쉽다.

```
/*
 * (C) Copyright 2006
 * SungKyunKwan Univ. Embedded LAb. gurugio@gmail.com
 *
 * (C) Copyright 2006
 *
 *
 * Configuration settings for the SKKU board.
 *
 * See file CREDITS for list of people who contributed to this
 * project.
 *
 * This program is free software; you can redistribute it and/or
 * modify it under the terms of the GNU General Public License as
 * published by the Free Software Foundation; either version 2 of
 * the License, or (at your option) any later version.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program; if not, write to the Free Software
 * Foundation, Inc., 59 Temple Place, Suite 330, Boston,
 * MA 02111-1307 USA
 */
```

```
#ifndef __CONFIG_H
#define __CONFIG_H
```

```
/*
 * If we are developing, we might want to start armboot from ram
 * so we MUST NOT initialize critical regs like mem-timing ...
 */
```

start.S 파일을 열어보면 CONFIG_INIT_CRITICAL가 선언되어 있다면 cpu_init_crit 함수를 호출하는 부분이 있다. 이 함수는 프로세서의 속도나 메모리 컨트롤러등을 설정하는 일을 하므로 반드시 호출해주어야 한다.

```
#define CONFIG_INIT_CRITICAL /* undef for developing */
```

```
/*
 * High Level Configuration Options
 * (easy to change)
 */
```

특별히 하는 일은 없다. 프로세서 종류와 보드 이름을 선언한다.

```
#define CONFIG_PXA270 1 /* This is an PXA270 CPU */
#define CONFIG_SKKU 1 /* on an SKKU Board */
#define BOARD_LATE_INIT 1
```

IRQ 설정 코드를 실행시키지 않는다. 부트 로더에서는 굳이 IRQ가 필요없다.

```
#undef CONFIG_USE_IRQ /* we don't need IRQ/FIQ stuff */
```

```
/*
 * Size of malloc() pool
 */
```

전역 변수를 저장하기 위해서 미리 약간의 메모리 영역을 지정해놓는다. start.S 에서 SDRAM을 위한 메모리 맵을 설정하는 부분이 있다. 부트 로더 자신을 SDRAM으로 복사하고 그 외에 환

경 설정이나 전역 변수등에 대한 영역, 스택 메모리등의 영역을 설정하는데 사용된다.

```
#define CFG_MALLOC_LEN (CFG_ENV_SIZE + 128*1024)
```

include/asm-arm/global_data.h에는 gd_t라는 구조체가 있다. 이 구조체는 부트 로더 초기 설정을 기록하기 위해 사용된다. 이 구조체가 저장되는 영역의 크기를 결정한다.

```
#define CFG_GBL_DATA_SIZE 128 /* size in bytes reserved for initial data */
```

```
/*  
 * Hardware drivers  
 */
```

SKKU 보드는 네트워크 장치로 CS8900를 사용한다. 칩 셀렉트 핀에 nCS1을 사용하므로 0x0400 0000 번지로 접근할 수 있다. drivers/ 를 보면 다양한 이데넷 칩의 드라이버가 있다.

```
#define CONFIG_DRIVER_CS8900 1  
#define CS8900_BASE 0x04000000  
#define CS8900_BUS16 1
```

```
/*  
 * select serial console configuration  
 */
```

시리얼 통신 설정

```
#define CONFIG_FFUART 1 /* we use FFUART on SKKU */  
#define CONFIG_BAUDRATE 115200
```

```
/* allow to overwrite serial and ethaddr */  
#define CONFIG_ENV_OVERWRITE
```

u-boot에서 지원하는 명령어 설정. README 파일을 읽어보면 가능한 명령어들에 대한 목록이 있다. NAND, DHCP등 다양한 명령들이 있다. CONFIG_CMD_DFL은 디폴트 명령들을 설정한다.

```
#define CONFIG_COMMANDS (CONFIG_CMD_DFL | CFG_CMD_FAT)
```

```
/* this must be included AFTER the definition of CONFIG_COMMANDS (if any) */  
#include <cmd_confdefs.h>
```

대기 시간

```
#define CONFIG_BOOTDELAY 5
```

네트워크 설정

```
#define CONFIG_ETHADDR 12:34:56:78:9a:bc  
#define CONFIG_NETMASK 255.255.255.0  
#define CONFIG_IPADDR 10.51.15.132  
#define CONFIG_SERVERIP 10.51.15.126
```

대기 시간이 지나면 실행하는 명령. 커널 주소를 지정해주면 자동으로 커널로 점프한다. 주석으로 남기면 자동으로 부팅하지 않고 프롬프트를 출력한다.

```
#define CONFIG_BOOTCOMMAND "bootm 40000"
```

부팅 옵션. 리눅스의 부팅 옵션과 동일하게 설정된다.

```
#define CONFIG_BOOTARGS "root=/dev/mtdblock2 rootfstype=jffs2 console=ttyS0,115200"
```

```
#define CONFIG_CMDLINE_TAG
```

```
#if (CONFIG_COMMANDS & CFG_CMD_KGDB)  
#define CONFIG_KGDB_BAUDRATE 230400 /* speed to run kgdb serial port */  
#define CONFIG_KGDB_SER_INDEX 2 /* which serial port to use */  
#endif
```

```
/*  
 * Miscellaneous configurable options  
 */
```

README 참조.

```
#define CFG_HUSH_PARSER 1  
#define CFG_PROMPT_HUSH_PS2 ">"
```

```
#define CFG_LONGHELP /* undef to save memory */  
#ifdef CFG_HUSH_PARSER
```

프롬프트 텍스트 설정

```
#define CFG_PROMPT "SKKU2005>" /* Monitor Command Prompt */
#else
#define CFG_PROMPT ">" /* Monitor Command Prompt */
#endif
#define CFG_CBSIZE 256 /* Console I/O Buffer Size */
#define CFG_PBSIZE (CFG_CBSIZE+sizeof(CFG_PROMPT)+16) /* Print Buffer Size */
#define CFG_MAXARGS 16 /* max number of command args */
#define CFG_BARGSIZE CFG_CBSIZE /* Boot Argument Buffer Size */
#define CFG_DEVICE_NULLDEV 1
```

몇가지 보드들은 메모리 테스트를 한다. Lubbock 보드는 실행하지 않음.

```
#define CFG_MEMTEST_START 0xa4000000 /* memtest works on */
#define CFG_MEMTEST_END 0xa4080000 /* 0 ~ 8 MB in DRAM */
```

```
#undef CFG_CLKS_IN_HZ /* everything, incl board info, in Hz */
```

리눅스 커널을 SDRAM으로 로드하는 주소

```
#define CFG_LOAD_ADDR 0xa4008000 /* default load address */
```

/* interrupt.c -> get_tclk(). number of time ticks per second */

cpu/pxa/interrupt.c에는 `udelay_masked`라는 함수가 있다. 이 함수는 `us` 단위로 대기하는 일을 한다. 이 함수에 `CFG_HZ` 값이 사용되는데 이 값을 `OSCR` 값과 비교해서 시간이 측정한다. `OSCR`은 Intel PXA27x Processor Family Developer's Manual의 22.5.5 OSCR0의 설명에서 보듯이 3.25MHz 클럭에 따라 값이 증가한다. 결국 1초에 3250000 증가하고 1us에는 3이 증가하므로 이 값과 비교하면 `us` 단위로 딜레이를 구현할 수 있다.

```
#define CFG_HZ 3250000 /* incrementer freq: 3.25 MHz */
```

cpu/pxa/start.S 파일에서 `CCCR` 레지스터를 이 값으로 설정한다. Intel PXA27x Processor Family Developer's Manual의 3.8.2.1 CCCR을 참조한다. 매뉴얼의 Table 3-7에 따라 프로세서를 520MHz로 동작시키기 위해서는 `CCCR[L] = 16`, `CCCR[2N] = 5`, `CCCR[A] = 1`이 되어야 한다.

```
#define CFG_CPUSPEED 0x08000290 /* set core clock to 520 MHz */
```

/* ADDED BY GIO for start.S */

start.S에 PXA270의 터보 모드 진입에 관한 코드를 추가해야 한다. PXA250에는 없는 코드이므로 기타 다른 부트 로더를 참조하여 코드를 추가한다. Table 3-7에 따라 레지스터 비트들을 셋팅하면 된다. `CLKCFG` 레지스터는 Coprocessor14에 속한 레지스터이므로 일반 레지스터와는 약간 다르게 다루어야 한다.

```
#define CFG_CLKCFG_VAL 0x0000000b /* B=1, HT=0, T=1 F=1 in CP14 */
```

/* valid baudrates */

```
#define CFG_BAUDRATE_TABLE { 9600, 19200, 38400, 57600, 115200 }
```

/*

* Stack sizes

*/

* The stack sizes are set up in start.S using the settings below

*/

C 코드에서 함수 호출등을 위해 사용될 스택의 크기 설정

```
#define CONFIG_STACKSIZE (128*1024) /* regular stack */
```

```
#ifdef CONFIG_USE_IRQ
```

```
#define CONFIG_STACKSIZE_IRQ (4*1024) /* IRQ stack */
```

```
#define CONFIG_STACKSIZE_FIQ (4*1024) /* FIQ stack */
```

```
#endif
```

/*

* Physical Memory Map

*/

SKKU 보드에서는 nSDCS1을 사용해서 SDRAM의 파티션을 선택하므로 2번째 뱅크가 사용된다.

```
#define CONFIG_NR_DRAM_BANKS 1 /* we have 1 banks of DRAM */
```

```
#define PHYS_SDRAM_1 0xa0000000 /* SDRAM Bank #1 */
#define PHYS_SDRAM_1_SIZE 0x00000000 /* No SDRAM at 1st partition */
#define PHYS_SDRAM_2 0xa4000000 /* SDRAM Bank #2 */
#define PHYS_SDRAM_2_SIZE 0x04000000 /* 64 MB */
#define PHYS_SDRAM_3 0xa8000000 /* SDRAM Bank #3 */
#define PHYS_SDRAM_3_SIZE 0x00000000 /* 0 MB */
#define PHYS_SDRAM_4 0xac000000 /* SDRAM Bank #4 */
#define PHYS_SDRAM_4_SIZE 0x00000000 /* 0 MB */
```

```
/*
 * SKKU uses two '28F128K3', 128Mbit = 16MB
 * 128Blocks X 1Mbit(128KB)
 */
```

28F128J3A와 28F128K3는 128KB의 섹터 크기를 가지며 128개의 블록으로 이루어져있다. ARM 계열의 프로세서들은 리셋후 메모리의 0번지를 참조하므로 반드시 0번지에 롬을 붙여야 한다.

```
#define PHYS_FLASH_1 0x00000000 /* Flash Bank #1 */
#define PHYS_FLASH_2 0x04000000 /* Flash Bank #2 */
#define PHYS_FLASH_SIZE 0x02000000 /* 32 MB */
#define PHYS_FLASH_BANK_SIZE 0x02000000 /* 32 MB Banks */
#define PHYS_FLASH_SECT_SIZE 0x00040000 /* 256 KB sectors (x2) */
```

```
#define CFG_DRAM_BASE PHYS_SDRAM_2
#define CFG_DRAM_SIZE PHYS_SDRAM_2_SIZE
```

```
#define CFG_FLASH_BASE PHYS_FLASH_1
```

```
/*
 * FLASH and environment organization
 */
```

```
#define CFG_MAX_FLASH_BANKS 1 /* max number of memory banks */
#define CFG_MAX_FLASH_SECT 128 /* max number of sectors on one chip */
```

```
#define FPGA_REGS_BASE_PHYSICAL 0x08000000
```

```
/*
 * GPIO settings
 */
```

GPIO 사용 설정. 부트로더에서는 FFUART와 최소한의 장치만 사용한다.

```
#define CFG_GPSR0_VAL 0x00008004
#define CFG_GPSR1_VAL 0x00020080
#define CFG_GPSR2_VAL 0x0001FFFF
#define CFG_GPCR0_VAL 0x00000000
#define CFG_GPCR1_VAL 0x00000380
#define CFG_GPCR2_VAL 0x00000000
#define CFG_GPDRO_VAL 0x0060A800
#define CFG_GPDR1_VAL 0x00000380
#define CFG_GPDR2_VAL 0x0001C000
#define CFG_GAFR0_L_VAL 0x98400000
#define CFG_GAFR0_U_VAL 0x00002950
#define CFG_GAFR1_L_VAL 0x0000a950
#define CFG_GAFR1_U_VAL 0x0005AAAA
#define CFG_GAFR2_L_VAL 0xA0000000
#define CFG_GAFR2_U_VAL 0x00000002
```

PSSR[RDH] = 0, PSSR[PH] = 0 GPIO를 활성화시킨다. PSSR[BFS] = 1 배터리 폴트가 발생하면 프로세서를 슬립 모드로 바꾼다.

```
#define CFG_PSSR_VAL 0x20
```

```
/*
 * Memory settings
 */
```

nCS<1:0>으로 연결된 플래시 메모리에 대한 설정

```
#define CFG_MSC0_VAL 0x788912b3 /* for Flash */
```

nCS<5:2>에는 아무것도 연결되지 않음

```
#define CFG_MSC1_VAL          0x00000000
#define CFG_MSC2_VAL          0x00000000

/*
 * SDRAM partition 1 enable,
 * 2 banks X 13 ROWs X 9 Column X 16 bit width X 2 chips
 * 4 internal banks per 1 chips
 * tRP=3, CL=3, tRCD=3, tRAS=7, tRC=11
 * Normal addressing mode
 * MA<24:10> are used
 */
#define CFG_MDCNFG_VAL        0x08000bca /* for SDRAM */

/*
 * SRAM uses SDCLK<1>
 * Flash uses SDCLK<0>
 * According to CCCR, CLK_MEM is 208MHz,
 * thus it should be divided by 2 to be used for SDRAM(104Mhz) -> K1DB2=1
 */
#define CFG_MDREFR_VAL        0x0103801e

/* These value is dummy value.
 * There value does not affect SDRAM setting */
#define CFG_MDMRS_VAL         0x00000000

/* Asynchronous flash memory for SKKU */
비동기 방식으로 플래시를 사용하므로 모두 0으로 설정한다.
#define CFG_SXCNFG_VAL 0x00000000

/*
 * PCMCIA and CF Interfaces
 */
#define CFG_MECR_VAL          0x00000000 /* No PC Card */
#define CFG_MCMEM0_VAL        0x00010504
#define CFG_MCMEM1_VAL        0x00010504
#define CFG_MCATT0_VAL        0x00010504
#define CFG_MCATT1_VAL        0x00010504
#define CFG_MCIO0_VAL         0x00004715
#define CFG_MCIO1_VAL         0x00004715

/* timeout values are in ticks */
#define CFG_FLASH_ERASE_TOUT   (25*CFG_HZ) /* Timeout for Flash Erase */
#define CFG_FLASH_WRITE_TOUT   (25*CFG_HZ) /* Timeout for Flash Write */

/* FIXME */
#define CFG_ENV_IS_IN_FLASH 1
#define CFG_ENV_ADDR           (PHYS_FLASH_1 + 0x1C000) /* Addr of Environment Sector */
#define CFG_ENV_SIZE           0x4000 /* Total Size of Environment Sector */

/* clock */
메모리 컨트롤러, 48MHz 클럭, OS Timer, FFUART 클럭 활성화
#define CFG_CKEN_VAL           0x00400a40

#endif /* __CONFIG_H */
```

board/skku/config.mk의 내용은 다음과 같다. 부트 로더가 SDRAM에 로드될 주소를 정해주면 된다. 0xA4008000에 커널이 로드되므로 이보다 충분히 큰 크기가 되어 한다.

```
gurugio@gio:/home/SKKU/bulverde/bootloader/u-boot-1.1.2
1 #EXT_BASE = 0xa4080000
~
~
~
~
~
~
~
~
1,1 All
```

프로세서와 메모리 초기화를 위한 레지스터 이해

프로세서와 메모리를 초기화하기 위한 코드들에서 설정한 레지스터들에 대해 간단하게 살펴본다. Intel PXA27x Processor Family Developer's Manual에서 다룬 순서를 따른다.

1. 클럭 설정 레지스터

- CCCR : 프로세서의 코어 클럭 설정
 - L : 런 모드 클럭과 오실레이터의 비율
 - 2N : 터보 모드의 클럭과 런 모드 클럭의 비율
 - A : 메모리 컨트롤러 클럭 설정
 - CLKCFG[B] = 1, CCCR[A] = 0, L = 16일 때, 메모리 컨트롤러는 $13\text{MHz} * L / 2 = 104\text{MHz}$ 로 동작한다.
- CKEN : 프로세서 내부 모듈의 클럭 활성화
 - <22> : 메모리 컨트롤러 클럭
 - <16> : LCD 컨트롤러 클럭
 - <11> : USB 클라이언트, 48MHz 클럭
 - <6> : FFUART 클럭
- CLKCFG : 클럭 모드 변경, Coprocessor 14에 들어있음.
 - T : 터보 모드 활성화
 - F : 클럭 주기 변경 시작
 - HT : 하프 터보 모드
 - B : 시스템 버스 클럭을 런 모드 클럭 주기와 같게 한다. 520MHz 터보 모드에서 런 모드 클럭은 208MHz이다.

2. 메모리 컨트롤러 레지스터

SDRAM을 컨트롤하는 레지스터

- MDCNFG : SDRAM의 파티션에 대한 설정을 한다. 0/1 파티션과 2/3 파티션으로 나뉘어서 각각 설정한다.
 - DE0 : SDRAM 파티션 0 활성화
 - DE1 : SDRAM 파티션 1 활성화
 - DWID0 : 0/1 파티션에서 데이터 버스 폭
 - DCAC0 : 0/1 파티션에서 주소 비트의 열 개수
 - DRAC0 : 0/1 파티션에서 주소 비트의 행 개수
 - DNB0 : 0/1 파티션에서 뱅크의 개수
 - DTC0 : SDRAM의 AC 타이밍을 설정. 삼성에서 출시하는 대부분의 32MB 크기의 SDRAM은 0b11이나 0b10 둘 다 사용될 수 있다.
 - DADDR0 : 주소 지정 모드
 - DSA1110_0 : SA-1110 주소 지정 방식 호환 모드
 - STACK0 : STACK1 비트와 함께 메모리에 사용될 주소 핀을 설정한다.
- MDMRS : MDCNFG 레지스터를 설정한 다음에 MDMRS 레지스터를 설정해야 한다. 대부분의 비트가 메모리의 동작 상태를 나타내는 읽기 전용 비트이므로 MDCNFG 레지스터에 대한 설정을 끝낸 직후에 MDMRS 레지스터에 0x00000000값을 써주는 것이 일반적인 사용법이다.
- MDREFR : SDRAM의 리프레시 시간과 동작 클럭을 설정한다. SDCLK, SDCKE를 설정한다.

- DRI : SDRAM은 한번에 한 행씩 리프레시를 해준다. 이 리프레시의 시간 간격을 지정한다. 16/32MB의 SDRAM은 리프레시 시간이 64ms가 되어 한다. 따라서 CLK_MEM이 104MHz일 때, 매뉴얼에 나온 공식대로 $(64ms / 13rows * 104MHz - 31) / 32 = 15$ 로 설정하면 된다.
- K0RUN : SDCLK<0>/<3>의 활성화. SDCLK<0>은 주로 동기 플래시 롬에 사용된다. 만약 비동기 플래시 롬만을 사용한다면 필요가 없다.
- K0DB2 : SDCLK<0>/<3>의 속도 설정
- E1PIN : SDCKE 활성화
- K1RUN : SDCLK<1> 활성화
- K1DB2 : SDCLK<1> 속도 설정
- K2RUN : SDCLK<2> 활성화
- K2DB2 : SDCLK<2> 속도 설정
- APD : Auto power down 활성화. 램이 없거나 사용하지 않을 때 램 관련 동작들을 멈춰서 전력 소모를 줄인다. 보통 사용하지 않는다.
- SLFRSH : SDRAM의 Self-refresh를 활성화한다.
- K0FREE : SDCLK<0>/<3>의 동작 제어. 1로 설정하면 동작함.

플래시 메모리 관련 설정

- SXCNFG : 동기 플래시 메모리를 사용하는 경우에 설정한다.
 - SXEN0 : 0번 파티션을 동기 플래시 메모리로 설정
 - SXEN1 : 1번 파티션을 동기 플래시 메모리로 설정
- MSCx : 비동기 플래시 메모리 설정. nCS<5:0>으로 플래시 파티션을 선택하는데 MSCx 레지스터 한 개가 두개의 nCS를 설정한다. 즉 MSC0이 nCS<1:0>을 설정하게 된다. MSC0을 중심으로 설명한다.
 - RT0 : nCS0의 디바이스 타입.
 - RBW0 : nCS0의 버스 폭
 - RDF0 : nOE, nWE의 신호가 Assert 되는 길이.
 - RDN0 : 버스트 모드일 때 사용된다. nOE, new가 Deassert된 후 다시 Assert될 때까지의 클럭 수. 보통 RDF와 같다.
 - RRR0 : 논버스트 모드에서 nCS0이 액티브되고 다음에 액티브될 수 있는 최소 클럭 수
 - RBUFF0 : 플래시를 사용할 때는 0으로 설정한다.

3. GPIO 설정 레지스터

다음은 Intel PXA27x Processor Family Developer's Manual에서 24장 General Purpose I/O Controller에 있는 테이블24-2를 가져온 것이다. SKKU 보드에서 사용하는 GPIO와 사용되는 기능을 노락색으로 색칠을 했다. GPIO 관련 설정을 하기 위해서는 이렇게 프로세서에 있는 GPIO들을 놓고 어떤 핀이 어떤 기능으로 사용될지를 먼저 결정하고 그에 맞춰서 레지스터들의 설정값들을 계산하면 된다.

Table 24-2. GPIO Alternate Functions (Sheet 1 of 4)

| GPIO Pin | Pin Name | Alternate Function 1 (In) | Alternate Function 2 (In) | Alternate Function 3 (In) | Alternate Function 1 (Out) | Alternate Function 2 (Out) | Alternate Function 3 (Out) |
|----------|--------------------------------------|---------------------------|---------------------------|---------------------------|----------------------------|----------------------------|----------------------------|
| 0 | GPIO<0> | | | | | | |
| 1 | GPIO<1>/ nRESET_GPIO ⁴ | | | | | | |
| 2 | SYS_EN ⁵ | | | | | | |
| 3 | GPIO<3>/ PWR_SCL | | | | | | |
| 4 | GPIO<4>/ PWR_SDA | | | | | | |
| 5 | PWR_CAP<0> ⁵ | | | | | | |
| 6 | PWR_CAP<1> ⁵ | | | | | | |
| 7 | PWR_CAP<2> ⁵ | | | | | | |
| 8 | PWR_CAP<3> ⁵ | | | | | | |
| 9 | GPIO<9> | | | FFCTS | HZ_CLK | | CHOUT<0> |
| 10 | GPIO<10> | FFDCD | | USB_P3_5 ⁷ | HZ_CLK | | CHOUT<1> |
| 11 | GPIO<11> | EXT_SYNC<0> | SSPRXD2 | USB_P3_1 | CHOUT<0> | PWM_OUT<2> | 48_MHz |
| 12 | GPIO<12> | EXT_SYNC<1> | CIF_DD<7> | | CHOUT<1> | PWM_OUT<3> | 48_MHz |
| 13 | GPIO<13> | CLK_EXT | KP_DKIN<7> | KP_MKIN<7> | SSPTXD2 | | |
| 14 | GPIO<14> | L_VSYNC | SSPSFRM2 | | | SSPSFRM2 | UCLK |
| 15 | GPIO<15> | | | | nPCE<1> | nCS<1> | |
| 16 | GPIO<16> | KP_MKIN<5> | | | | PWM_OUT<0> | FFTXD |
| 17 | GPIO<17> | KP_MKIN<6> | CIF_DD<6> | | | PWM_OUT<1> | |
| 18 | GPIO<18> | RDY | | | | | |
| 19 | GPIO<19> | SSPSCLK2 | | FFRXD | SSPSCLK2 | L_CS | nURST |
| 20 | GPIO<20> | DREQ<0> | MBREQ | | nSDCS<2> | | |
| 21 | GPIO<21> | | | | nSDCS<3> | DVAL<0> | MBGNT |
| 22 | GPIO<22> | SSPEXTCLK2 | SSPSCLK2EN | SSPSCLK2 | KP_MKOUT<7> | SSPSYSCLK2 | SSPSCLK2 |
| 23 | GPIO<23> | | SSPSCLK | | CIF_MCLK | SSPSCLK | |
| 24 | GPIO<24> | CIF_FV | SSPSFRM | | CIF_FV | SSPSFRM | |
| 25 | GPIO<25> | CIF_LV | | | CIF_LV | SSPTXD | |
| 26 | GPIO<26> | SSPRXD | CIF_PCLK | FFCTS | | | |
| 27 | GPIO<27> | SSPEXTCLK | SSPSCLKEN | CIF_DD<0> | SSPSYSCLK | | FFRTS |
| 28 | GPIO<28> | AC97_BITCLK | I2S_BITCLK | SSPSFRM | I2S_BITCLK | | SSPSFRM |
| 29 | GPIO<29> | AC97_SDATA_IN_0 | I2S_SDATA_IN | SSPSCLK | SSPRXD2 | | SSPSCLK |
| 30 | GPIO<30> | | | | I2S_SDATA_OUT | AC97_SDATA_OUT | USB_P3_2 |
| 31 | GPIO<31> | | | | I2S_SYNC | AC97_SYNC | USB_P3_6 |
| 32 | GPIO<32> | | | | MSSCLK | MMCLK | |

Table 24-2. GPIO Alternate Functions (Sheet 2 of 4)

| GPIO Pin | Pin Name | Alternate Function 1 (In) | Alternate Function 2 (In) | Alternate Function 3 (In) | Alternate Function 1 (Out) | Alternate Function 2 (Out) | Alternate Function 3 (Out) |
|----------|----------|---------------------------|---------------------------|---------------------------|----------------------------|----------------------------|----------------------------|
| 33 | GPIO<33> | FFRXD | FFDSR | | DVAL<1> | nCS<5> | MBGNT |
| 34 | GPIO<34> | FFRXD | KP_MKIN<3> | SSPSCLK3 | USB_P2_2 | | SSPSCLK3 |
| 35 | GPIO<35> | FFCTS | USB_P2_1 | SSPSFRM3 | | KP_MKOUT<6> | SSPTXD3 |
| 36 | GPIO<36> | FFDCD | SSPSCLK2 | KP_MKIN<7> | USB_P2_4 | SSPSCLK2 | |
| 37 | GPIO<37> | FFDSR | SSPSFRM2 | KP_MKIN<3> | USB_P2_8 | SSPSFRM2 | FFTXD |
| 38 | GPIO<38> | FFRI | KP_MKIN<4> | USB_P2_3 | SSPTXD3 | SSPTXD2 | PWM_OUT<1> |
| 39 | GPIO<39> | KP_MKIN<4> | | SSPSFRM3 | USB_P2_6 | FFTXD | SSPSFRM3 |
| 40 | GPIO<40> | SSPRXD2 | | USB_P2_5 | KP_MKOUT<6> | FFDTR | SSPSCLK3 |
| 41 | GPIO<41> | FFRXD | USB_P2_7 | SSPRXD3 | KP_MKOUT<7> | FFRTS | |
| 42 | GPIO<42> | BTRXD | ICP_RXD | | | | CIF_MCLK |
| 43 | GPIO<43> | | | CIF_FV | ICP_TXD | BTTXD | CIF_FV |
| 44 | GPIO<44> | BTCTS | | CIF_LV | | | CIF_LV |
| 45 | GPIO<45> | | | CIF_PCLK | AC97_SYSClk | BTRTS | SSPSYSClk3 |
| 46 | GPIO<46> | ICP_RXD | STD_RXD | | | PWM_OUT<2> | |
| 47 | GPIO<47> | CIF_DD<0> | | | STD_TXD | ICP_TXD | PWM_OUT<3> |
| 48 | GPIO<48> | CIF_DD<5> | | | BB_OB_DAT<1> | nPOE | |
| 49 | GPIO<49> | | | | | nPWE | |
| 50 | GPIO<50> | CIF_DD<3> | | SSPSCLK2 | BB_OB_DAT<2> | nPIOR | SSPSCLK2 |
| 51 | GPIO<51> | CIF_DD<2> | | | BB_OB_DAT<3> | nPIOW | |
| 52 | GPIO<52> | CIF_DD<4> | SSPSCLK3 | | BB_OB_CLK | SSPSCLK3 | |
| 53 | GPIO<53> | FFRXD | USB_P2_3 | | BB_OB_STB | CIF_MCLK | SSPSYSClk |
| 54 | GPIO<54> | | BB_OB_WAIT | CIF_PCLK | | nPCE<2> | |
| 55 | GPIO<55> | CIF_DD<1> | BB_IB_DAT<1> | | | nPREG | |
| 56 | GPIO<56> | nPWAIT | BB_IB_DAT<2> | | USB_P3_4 | | |
| 57 | GPIO<57> | nIOIS16 | BB_IB_DAT<3> | | | | SSPTXD |
| 58 | GPIO<58> | | LDD<0> | | | LDD<0> | |
| 59 | GPIO<59> | | LDD<1> | | | LDD<1> | |
| 60 | GPIO<60> | | LDD<2> | | | LDD<2> | |
| 61 | GPIO<61> | | LDD<3> | | | LDD<3> | |
| 62 | GPIO<62> | | LDD<4> | | | LDD<4> | |
| 63 | GPIO<63> | | LDD<5> | | | LDD<5> | |
| 64 | GPIO<64> | | LDD<6> | | | LDD<6> | |
| 65 | GPIO<65> | | LDD<7> | | | LDD<7> | |
| 66 | GPIO<66> | | LDD<8> | | | LDD<8> | |
| 67 | GPIO<67> | | LDD<9> | | | LDD<9> | |
| 68 | GPIO<68> | | LDD<10> | | | LDD<10> | |

Table 24-2. GPIO Alternate Functions (Sheet 3 of 4)

| GPIO Pin | Pin Name | Alternate Function 1 (In) | Alternate Function 2 (In) | Alternate Function 3 (In) | Alternate Function 1 (Out) | Alternate Function 2 (Out) | Alternate Function 3 (Out) |
|----------|-----------|---------------------------|---------------------------|---------------------------|----------------------------|----------------------------|----------------------------|
| 69 | GPIO<69> | | LDD<11> | | | LDD<11> | |
| 70 | GPIO<70> | | LDD<12> | | | LDD<12> | |
| 71 | GPIO<71> | | LDD<13> | | | LDD<13> | |
| 72 | GPIO<72> | | LDD<14> | | | LDD<14> | |
| 73 | GPIO<73> | | LDD<15> | | | LDD<15> | |
| 74 | GPIO<74> | | | | | L_FCLK_RD | |
| 75 | GPIO<75> | | | | | L_LCLK_A0 | |
| 76 | GPIO<76> | | | | | L_PCLK_WR | |
| 77 | GPIO<77> | | | | | L_BIAS | |
| 78 | GPIO<78> | | | | nPCE<2> | nCS<2> | |
| 79 | GPIO<79> | | | | PSKTSSEL | nCS<3> | PWM_OUT<2> |
| 80 | GPIO<80> | DREQ<1> | MBREQ | | | nCS<4> | PWM_OUT<3> |
| 81 | GPIO<81> | | CIF_DD<0> | | SSPTXD3 | BB_OB_DAT<0> | |
| 82 | GPIO<82> | SSPRXD3 | BB_IB_DAT<0> | CIF_DD<5> | | | FFDTR |
| 83 | GPIO<83> | SSPSFRM3 | BB_IB_CLK | CIF_DD<4> | SSPSFRM3 | FFTXD | FFRTS |
| 84 | GPIO<84> | SSPCLK3 | BB_IB_STB | CIF_FV | SSPCLK3 | | CIF_FV |
| 85 | GPIO<85> | FFRXD | DREQ<2> | CIF_LV | nPCE<1> | BB_IB_WAIT | CIF_LV |
| 86 | GPIO<86> | SSPRXD2 | LDD<16> | USB_P3_5 | nPCE<1> | LDD<16> | |
| 87 | GPIO<87> | nPCE<2> | LDD<17> | USB_P3_1 | SSPTXD2 | LDD<17> | SSPSFRM2 |
| 88 | GPIO<88> | USBHPWR<1> | SSPRXD2 | SSPSFRM2 | | | SSPSFRM2 |
| 89 | GPIO<89> | SSPRXD3 | | FFRI | AC97_SYSCLK | USBHPEN<1> | SSPTXD2 |
| 90 | GPIO<90> | KP_MKIN<5> | USB_P3_5 | CIF_DD<4> | | nURST | |
| 91 | GPIO<91> | KP_MKIN<6> | USB_P3_1 | CIF_DD<5> | | UCLK | |
| 92 | GPIO<92> | MMDAT<0> | | | MMDAT<0> | MSBS | |
| 93 | GPIO<93> | KP_DKIN<0> | CIF_DD<6> | | AC97_SDATA_OUT | | |
| 94 | GPIO<94> | KP_DKIN<1> | CIF_DD<5> | | AC97_SYNC | | |
| 95 | GPIO<95> | KP_DKIN<2> | CIF_DD<4> | KP_MKIN<6> | AC97_RESET_n | | |
| 96 | GPIO<96> | KP_DKIN<3> | MBREQ | FFRXD | | DVAL<1> | KP_MKOUT<6> |
| 97 | GPIO<97> | KP_DKIN<4> | DREQ<1> | KP_MKIN<3> | | MBGNT | |
| 98 | GPIO<98> | KP_DKIN<5> | CIF_DD<0> | KP_MKIN<4> | AC97_SYSCLK | | FFRTS |
| 99 | GPIO<99> | KP_DKIN<6> | AC97_SDATA_IN_1 | KP_MKIN<5> | | | FFTXD |
| 100 | GPIO<100> | KP_MKIN<0> | DREQ<2> | FFCTS | | | |
| 101 | GPIO<101> | KP_MKIN<1> | | | | | |
| 102 | GPIO<102> | KP_MKIN<2> | | FFRXD | nPCE<1> | | |
| 103 | GPIO<103> | CIF_DD<3> | | | | KP_MKOUT<0> | |

Table 24-2. GPIO Alternate Functions (Sheet 4 of 4)

| GPIO Pin | Pin Name | Alternate Function 1 (In) | Alternate Function 2 (In) | Alternate Function 3 (In) | Alternate Function 1 (Out) | Alternate Function 2 (Out) | Alternate Function 3 (Out) |
|------------------|-----------|---------------------------|---------------------------|---------------------------|----------------------------|----------------------------|----------------------------|
| 104 | GPIO<104> | CIF_DD<2> | | | PSKTSEL | KP_MKOUT<1> | |
| 105 | GPIO<105> | CIF_DD<1> | | | nPCE<2> | KP_MKOUT<2> | |
| 106 | GPIO<106> | CIF_DD<9> | | | | KP_MKOUT<3> | |
| 107 | GPIO<107> | CIF_DD<8> | | | | KP_MKOUT<4> | |
| 108 | GPIO<108> | CIF_DD<7> | | | CHOUT<0> | KP_MKOUT<5> | |
| 109 | GPIO<109> | MMDAT<1> | MSSDIO | | MMDAT<1> | MSSDIO | |
| 110 | GPIO<110> | MMDAT<2>/ MMCCS<0> | | | MMDAT<2>/ MMCCS<0> | | |
| 111 | GPIO<111> | MMDAT<3>/ MMCCS<1> | | | MMDAT<3>/ MMCCS<1> | | |
| 112 | GPIO<112> | MMCMD | nMSINS | | MMCMD | | |
| 113 | GPIO<113> | | | USB_P3_3 | I2S_SYSCLK | AC97_RESET_n | |
| 114 ¹ | GPIO<114> | CIF_DD<1> | | | UEN | UVS0 | |
| 115 ² | GPIO<115> | DREQ<0> | CIF_DD<3> | MBREQ | UEN | nUVS1 | PWM_OUT<1> |
| 116 ³ | GPIO<116> | CIF_DD<2> | AC97_SDATA_IN_0 | UDET | DVAL<0> | nUVS2 | MBGNT |
| 117 | GPIO<117> | SCL | | | SCL | | |
| 118 | GPIO<118> | SDA | | | SDA | | |
| 119 | GPIO<119> | USBHPWR<2> | | | | | |
| 120 | GPIO<120> | | | | | USBHPEN<2> | |

다음은 GPIO 관련 레지스터들의 설명이다.

- GPDR : 입출력 방향을 결정한다.
 - GPDR0<0>은 GPIO<1>을 설정한다. 마찬가지로 GPDR1<0>은 GPIO<32>를 설정한다. 즉 한 비트에 한 핀을 설정한다.
 - SKKU 보드에서 설정한 값은 다음과 같다.
 - GPDR<0> : 0xC0F393E4
 - GPDR<1> : 0xFCEFA8B3
 - GPDR<2> : 0xE2F1FFFF
 - GPDR<3> : 0xFE1FFFE5
- GPSR : GPIO 핀 셋팅 레지스터
 - GPIO 핀이 출력으로 설정되어 있고 GPSR에서 해당 핀이 1로 설정되면 하이 신호를 내보낸다.
 - SKKU 보드에서 설정한 값은 다음과 같다.
 - GPSR<0> : 0x00008004
 - GPSR<1> : 0x00020080
 - GPSR<2> : 0x16C14000
 - GPSR<3> : 0x0003E000
- GPCR : GPIO 핀 클리어 레지스터
 - GPIO 핀이 출력으로 설정되어 있고 GPSR에서 해당 핀을 1로 설정하면 로우 신호를 내보낸다.
 - SKKU 보드에서 설정한 값은 다음과 같다.
 - GPCR<0> : 0x0
 - GPCR<1> : 0x00000380, FFUART에서 로우 신호가 필요하다.
 - GPCR<2> : 0x0
 - GPCR<3> : 0x0

※GPCR과 GPSR에 값을 쓸 때는 가장 마지막으로 설정한 값을 따른다. 즉 GPCR<0>과 GPSR<0>을 둘 다 1로 설정해줘도 GPCR<0>을 나중에 써줬으면 로우 신호가 나가고 GPSR<0>을 나중에 써주면 하이 신호가 출력된다.

- GRER : 상승 엣지 확인 활성화
 - 해당 핀에서 상승 엣지가 검출되면 GEDR 레지스터의 해당 비트를 1로 설정한다.
 - SKKU 보드에서 설정한 값은 다음과 같다.
 - GRER<0> : 0x0
 - GRER<1> : 0x0
 - GRER<2> : 0x00080000, MMC_CD에서 필요하다.
 - GRER<3> : 0x0
- GFER : 하강 엣지 확인 활성화
 - 해당 핀에서 하강 엣지가 검출되면 GEDR 레지스터의 해당 비트를 1로 설정한다.
 - SKKU 보드에서 설정한 값은 다음과 같다.
 - GFER<0> : 0x00000001, On/Off 스위치에 사용됨.
 - GFER<1> : 0x0
 - GFER<2> : 0x04000000, PCMCIA의 인터럽트에 사용
 - GFER<3> : 0x00000008, USB의 인터럽트에 사용
- GAFR : GPIO 핀의 기능 설정
 - GPIO 핀이 일반 입출력이 아니라 프로세서 내부의 유닛에 의해 사용되도록 설정한다. 보통 세가지의 기능을 선택할 수 있고 일반 GPIO로 사용할 수 있으므로 총 4가지 경우가 있다. 따라서 한 핀당 두 개의 비트로 설정한다.
 - 예를 들어, GPIO<0>은 GAFR0_L<1:0>으로 설정할 수 있고 GPIO<16>은 GAFR0_U<1:0>으로 설정한다. 또 GPIO<32>는 GAFR1_L<1:0>으로 설정한다.
 - GAFR0_L : 0x830C0000
 - GAFR0_U : 0xA520051A
 - GAFR1_L : 0x999A955A
 - GAFR1_U : 0xAAA5A0AA
 - GAFR2_L : 0x6A8AAAAA
 - GAFR2_U : 0x0109A002
 - GAFR3_L : 0x5400100A
 - GAFR3_U : 0x00001409
- GPLR : 핀 상태 확인
 - 현재 핀의 신호 레벨을 알 수 있다. 읽기 전용 레지스터이다.
- GEDR : 엣지 검출 상태
 - GRER이나 GFER의 설정에 따라 엣지가 검출되면 1로 셋팅된다.

프로세서와 메모리 초기화

Intel PXA27x Processor Family Developer's Manual은 PXA27x 프로세서로 개발하기 위한 모든 사항을 담고 있는 문서이다. 그 중에서도 몇 가지 챕터를 골라서 요약하고 중요한 부분들을 한글로 번역해서 정리한다. 프로세서 초기화와 SDRAM, 플래시 메모리의 초기화 과정을 이해하도록 한다.

1. 프로세서 속도

page 3-20 Table 3-7

Table 3-7. Clock Frequencies

Note: Refer to the *Intel® PXA27x Processor Family Specification Update* for any changes to the supported frequency points in [Table 3-7](#).

| Core Run Freq (MHz) | CLKCFG[T] | Core Turbo Freq (MHz) | CLKCFG[T] | CLKCFG[HT] | CCCR[L] | CCCR[2N] | System Bus (MHz) | CLKCFG[B] | CLK_MEM (MHz) | CCCR[A] | SDCLK<2:1> SDRAM Clocks (MHz) | MDREFR[KxDB2]††† | Synchronous Flash (MHz) | MDREFR[K0DB4] | MDREFR[K0DB2] | LCD (MHz) |
|------------------------|-----------|--------------------------|-----------|------------|---------|----------|---------------------|-----------|------------------|---------|----------------------------------|------------------|----------------------------|---------------|---------------|--------------|
| 13† | X | — | X | X | X | X | 13 | X | 13 | X | 13 | X | 13 | X | X | 13 or 26†† |
| 91††† | 0 | — | — | 0 | 7 | 2 | 45 | 0 | 91 | 0 | 45 | 1 | 22.5 | 1 | 1 | 91 |
| 104 | 0 | 104 | 1 | 0 | 8 | 2 | 104 | 1 | 104 | 1 | 104 | 0 | 52 | 0 | 1 | 52 |
| 156 | 0 | 156 | 1 | 1 | 8 | 6 | 104 | 1 | 104 | 1 | 104 | 0 | 52 | 0 | 1 | 52 |
| 208 | 0 | 208 | 1 | 0 | 16 | 2 | 104 | 0 | 104 | 0 | 104 | 0 | 52 | 1 | X | 104 |
| 208 | 0 | 208 | 1 | 0 | 16 | 2 | 208 | 1 | 208 | 1 | 104 | 1 | 52 | 1 | X | 104 |
| 208 | 0 | 312 | 1 | 0 | 16 | 3 | 104 | 0 | 104 | 0 | 104 | 0 | 52 | 1 | X | 104 |
| 208 | 0 | 312 | 1 | 0 | 16 | 3 | 208 | 1 | 208 | 1 | 104 | 1 | 52 | 1 | X | 104 |
| 208 | 0 | 416 | 1 | 0 | 16 | 4 | 208 | 1 | 208 | 1 | 104 | 1 | 52 | 1 | X | 104 |
| 208 | 0 | 520 | 1 | 0 | 16 | 5 | 208 | 1 | 208 | 1 | 104 | 1 | 52 | 1 | X | 104 |
| 208 | 0 | 624††††† | 1 | 0 | 16 | 6 | 208 | 1 | 208 | 1 | 104 | 1 | 52 | 1 | X | 104 |

NOTES:
† Not a PLL clock frequency. Refer to [Section 3.5.7.7](#).
†† Use CCCR[LCD_26] to control this setting. See [Table 3-31](#).
††† L = 7 (Core = 91.0 MHz) is used for hardware boot-up frequency only and must not be used for normal operation.
†††† KxDB2 represents K1DB2 and K2DB2
††††† 624 MHz is available on the PXA270 processor only. See the *Intel® PXA270 Processor Electrical, Mechanical, and Thermal Specification* and *Intel® PXA27x Processor Family Electrical, Mechanical, and Thermal Specification* for supported frequency product points.

이 테이블에서 터보 모드로 진입하기 위해서는 CLKCFG[T] = 1, CLKCFG[HT] = 0, CCCR[L] = 16, CCCR[2N] = 5로 설정해주면 된다. 그 다음은 프로세서의 모드 이외에 관련 클럭들의 설정이다. 예를 들어, CLKCFG[B] = 1이면 시스템 버스가 208MHz가 되고 0으로 설정하면 104MHz가 된다. 마찬가지로 CCCR[A]가 1이면 메모리 클럭 설정에 관련된 CLK_MEM가 208MHz가 되는 식이다. SDRAM을 104MHz로 동작시키기 위해서는 CCCR[A]와 MDREFR[KxDB2]가 1이 되어야 한다. 현재 SKKU 보드에서는 CCCR[A] = 0, MDREFR[KxDB2] = 0으로 설정해서, CLK_MEM을 104MHz로 동작시키고 SDRAM을 52MHz로 동작시킨다.

일반적으로 PXA270은 520MHz로 동작하는 터보 모드를 사용한다. 이런 터보 모드를 사용하기 위해 필요한 레지스터 설정을 주의해서 봐야 한다. 다음 장에 설명하는 시피유와 메모리 컨트롤러 초기화 코드를 세밀하게 살펴보고 해당 레지스터의 설정에 주의해야 한다.

PXA270에 맞게 초기화 코드 수정하기

u-boot 1.1.2 를 포팅하기 위해서는 보통 다음 파일들을 수정해야 한다. 그 외에 코드들은 큰 흐름만 파악하고 있으면 별다른 문제없이 보드에 포팅할 수 있다.

- board/skku/memsetup.S : 메모리 컨트롤러 설정. SDRAM의 속도와 리프레시 주기, 플래시 메모리 동작에 대한 설정을 한다.
- board/skku/skku.c : 특정 보드에 맞는 하드웨어 설정을 한다. 커널에 넘기기 위한 부팅 파라미터의 주소를 설정하고 SDRAM의 시작 주소와 크기를 설정한다. ARM용 리눅스 커널에서는 주로 SDRAM의 시작 주소 + 0x100 번지에 파라미터를 넘긴다.
- board/skku/flash.c : 보드에서 사용하는 플래시 메모리의 동작에 관련된 함수들. u-boot에서는 flash_init, flash_erase, flash_print_info, write_buff을 공통적으로 작성해야 한다. 이 함수들은 보드마다 공통적으로 존재해야 하고 그 내부 코드는 보드에 사용된 플래시에 따라 다르게 작성해주면 된다. common/flash.c와 common/cmd_flash.c 파일에서 이 소스에 작성된 함수들을 사용한다.
- cpu/pxa270/start.S : 프로세서의 속도 설정과 SDRAM으로 부트 로더 자신을 복사하는 일을 한다.

실행되는 순서는 start.S에서 memsetup.S 파일을 호출한다. 그리고 메모리 컨트롤러 설정이 끝나면 부트 로더 자신을 SDRAM으로 복사하고 SDRAM으로 점프한다. 그러면 lib_arm/board.c 파일에 들어있는 start_armboot 함수가 실행되고 include/configs/skku.h에서 설정한 내용에 따라 나머지 하드웨어들에 대한 설정을 하고 사용자의 명령을 기다린다. 사용자가 커널을 로드하도록 명령하면 lib_arm/armlinux.c에 있는 do_bootm_linux 함수가 호출되고 커널 이미지를 로드하고 커널로 점프한다.

다음은 SKKU 보드를 위해 Lubbock 보드의 코드를 수정해서 만든 소스들이다.

cpu/pxa270/start.S

```
/*
 * armboot - Startup Code for XScale
 *
 * Copyright (C) 1998   Dan Malek <dmalek@jlc.net>
 * Copyright (C) 1999   Magnus Damm <kieraypc01.p.y.kie.era.ericsson.se>
 * Copyright (C) 2000   Wolfgang Denk <wd@denx.de>
 * Copyright (C) 2001   Alex Zuepke <azu@sysgo.de>
 * Copyright (C) 2002   Kyle Harris <kharris@nexus-tech.net>
 * Copyright (C) 2003   Robert Schwebel <r.schwebel@pengutronix.de>
 * Copyright (C) 2003   Kai-Uwe Bloem <kai-uwe.bloem@auerswald.de>
 *
 * See file CREDITS for list of people who contributed to this
 * project.
 *
 * This program is free software; you can redistribute it and/or
 * modify it under the terms of the GNU General Public License as
 * published by the Free Software Foundation; either version 2 of
 * the License, or (at your option) any later version.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.      See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program; if not, write to the Free Software
 * Foundation, Inc., 59 Temple Place, Suite 330, Boston,
 * MA 02111-1307 USA
 */

#include <config.h>
#include <version.h>

.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
_prefetch_abort:       .word prefetch_abort
_data_abort:           .word data_abort
_not_used:              .word not_used
_irq:                  .word irq
```

```

_fiq:                                .word fiq

                                .balignl 16,0xdeadbeef

/*
 * Startup Code (reset vector)
 *
 * do important init only if we don't start from RAM!
 * - relocate armboot to ram
 * - setup stack
 * - jump to second stage
 */

_TEXT_BASE:
        .word    TEXT_BASE

.globl _armboot_start
_armboot_start:
        .word _start

/*
 * These are defined in the board-specific linker script.
 */
.globl _bss_start
_bss_start:
        .word __bss_start

.globl _bss_end
_bss_end:
        .word _end

#ifdef CONFIG_USE_IRQ
/* IRQ stack memory (calculated at run-time) */
.globl IRQ_STACK_START
IRQ_STACK_START:
        .word    0x0badc0de

/* IRQ stack memory (calculated at run-time) */
.globl FIQ_STACK_START
FIQ_STACK_START:
        .word 0x0badc0de
#endif

/*****
/*
/* the actual reset code
/*
*****/

reset:

```



```

    mrs    r0,cpsr                /* set the cpu to SVC32 mode    */
    bic    r0,r0,#0x1f            /* (supervisor mode, M=10011) */
    orr    r0,r0,#0x13
    msr    cpsr,r0

/*
 * we do sys-critical inits only at reboot,
 * not when booting from ram!
 */
#ifdef CONFIG_INIT_CRITICAL
    bl     cpu_init_crit          /* we do sys-critical inits    */
#endif

relocate:                          /* relocate U-Boot to RAM      */
    adr    r0, _start             /* r0 <- current position of code */
    ldr    r1, _TEXT_BASE         /* test if we run from flash or RAM */
    cmp    r0, r1                /* don't reloc during debug    */
    beq    stack_setup

    ldr    r2, _armboot_start
    ldr    r3, _bss_start
    sub    r2, r3, r2             /* r2 <- size of armboot        */
    add    r2, r0, r2             /* r2 <- source end address     */

copy_loop:
    ldmia  r0!, {r3-r10}          /* copy from source address [r0] */
    stmia  r1!, {r3-r10}          /* copy to target address [r1]    */
    cmp    r0, r2                /* until source end addreee [r2] */
    ble    copy_loop

/* Set up the stack */
stack_setup:
    ldr    r0, _TEXT_BASE         /* upper 128 KiB: relocated uboot */
    sub    r0, r0, #CFG_MALLOC_LEN /* malloc area */
    sub    r0, r0, #CFG_GBL_DATA_SIZE /* bdfinfo */
#ifdef CONFIG_USE_IRQ
    sub    r0, r0, #(CONFIG_STACKSIZE_IRQ+CONFIG_STACKSIZE_FIQ)
#endif
    sub    sp, r0, #12            /* leave 3 words for abort-stack */

clear_bss:
    ldr    r0, _bss_start         /* find start of bss segment    */
    ldr    r1, _bss_end           /* stop here */
    mov    r2, #0x00000000        /* clear */

clbss_1:str    r2, [r0]           /* clear loop... */
    add    r0, r0, #4
    cmp    r0, r1
    bne    clbss_1

    ldr    pc, _start_armboot

```

_start_armboot: .word start_armboot

```

/*****
/*
/* CPU_init_critical registers */
/*
/* - setup important registers */
/* - setup memory timing */
/*
*****/

```

```

/* Interrupt-Controller base address */
IC_BASE:      .word    0x40d00000
#define ICMR    0x04

```

```

/* Reset-Controller */
RST_BASE:     .word    0x40f00030
#define RCSR    0x00

```

```

/* Operating System Timer */
OSTIMER_BASE: .word    0x40a00000
#define OSMR3   0x0C
#define OSCR     0x10
#define OWER     0x18
#define OIER     0x1C

```

```

/* Clock Manager Registers */
#ifdef CFG_CPUSPEED
CC_BASE:      .word    0x41300000
#define CCCR     0x00
cpuspeed:.word    CFG_CPUSPEED
#else
#error "You have to define CFG_CPUSPEED!!"
#endif

```

```

/* RS: ??? */
.macro CPWAIT
mrc    p15,0,r0,c2,c0,0
mov    r0,r0
sub    pc,pc,#4
.endm

```

cpu_init_crit:

```

/* mask all IRQs */
ldr    r0, IC_BASE
mov    r1, #0x00
str    r1, [r0, #ICMR]

```

```
#if defined(CFG_CPUSPEED)
```

```
    /* set clock speed */
    ldr    r0, CC_BASE
    ldr    r1, cpuspeed
    str    r1, [r0, #CCCR]

    //mov    r0, #2
    ldr r0, =CFG_CLKCFG_VAL          /* for Turbo mode */
    mcr     p14, 0, r0, c6, c0, 0 /* enter the frequency change sequence */

    /* NOW Turbo-mode is on! */
```

```
setspeed_done:
#endif
```

```
    /*
     * before relocating, we have to setup RAM timing
     * because memory timing is board-dependend, you will
     * find a memsetup.S in your board directory.
     */
    mov     ip,      lr
    bl      memsetup
    mov     lr,      ip

    /* Memory interfaces are working. Disable MMU and enable I-cache. */

    ldr     r0, =0x2001          /* enable access to all coproc. */
    mcr     p15, 0, r0, c15, c1, 0
    CPWAIT

    mcr     p15, 0, r0, c7, c10, 4 /* drain the write & fill buffers */
    CPWAIT

    mcr     p15, 0, r0, c7, c7, 0 /* flush Icache, Dcache and BTB */
    CPWAIT

    mcr     p15, 0, r0, c8, c7, 0 /* flush instuction and data TLBs */
    CPWAIT

    /* Enable the Icache */

/*
    mrc     p15, 0, r0, c1, c0, 0
    orr     r0, r0, #0x1800
    mcr     p15, 0, r0, c1, c0, 0
    CPWAIT
*/
    mov     pc, lr
```

```

/*****
/*
/* Interrupt handling
/*
*****/

/* IRQ stack frame */

#define S_FRAME_SIZE    72

#define S_OLD_R0        68
#define S_PSR           64
#define S_PC            60
#define S_LR            56
#define S_SP            52

#define S_IP            48
#define S_FP            44
#define S_R10           40
#define S_R9            36
#define S_R8            32
#define S_R7            28
#define S_R6            24
#define S_R5            20
#define S_R4            16
#define S_R3            12
#define S_R2            8
#define S_R1            4
#define S_R0            0

#define MODE_SVC 0x13

/* use bad_save_user_regs for abort/prefetch/undef/swi ... */

.macro    bad_save_user_regs
sub    sp, sp, #S_FRAME_SIZE
stmia  sp, {r0 - r12}          /* Calling r0-r12 */
add    r8, sp, #S_PC

ldr    r2, _armboot_start
sub    r2, r2, #(CONFIG_STACKSIZE+CFG_MALLOC_LEN)
sub    r2, r2, #(CFG_GBL_DATA_SIZE+8)    @ set base 2 words into abort stack
ldmia  r2, {r2 - r4}            /* get pc, cpsr, old_r0 */
add    r0, sp, #S_FRAME_SIZE    /* restore sp_SVC */

add    r5, sp, #S_SP
mov    r1, lr
stmia  r5, {r0 - r4}            /* save sp_SVC, lr_SVC, pc, cpsr, old_r */
mov    r0, sp
.endm

```

```

/* use irq_save_user_regs / irq_restore_user_regs for */
/* IRQ/FIQ handling */

.macro irq_save_user_regs
sub    sp, sp, #S_FRAME_SIZE
stmia  sp, {r0 - r12}          /* Calling r0-r12 */
add    r8, sp, #S_PC
stmdb  r8, {sp, lr}^          /* Calling SP, LR */
str     lr, [r8, #0]           /* Save calling PC */
mrs    r6, spsr
str     r6, [r8, #4]          /* Save CPSR */
str     r0, [r8, #8]          /* Save OLD_R0 */
mov     r0, sp
.endm

.macro irq_restore_user_regs
ldmia  sp, {r0 - lr}^          @ Calling r0 - lr
mov     r0, r0
ldr     lr, [sp, #S_PC]        @ Get PC
add     sp, sp, #S_FRAME_SIZE
subs   pc, lr, #4              @ return & move spsr_svc into cpsr
.endm

.macro get_bad_stack
ldr     r13, _armboot_start     @ setup our mode stack
sub     r13, r13, #(CONFIG_STACKSIZE+CFG_MALLOC_LEN)
sub     r13, r13, #(CFG_GBL_DATA_SIZE+8) @ reserved a couple spots in abort stack

str     lr, [r13]               @ save caller lr / spsr
mrs     lr, spsr
str     lr, [r13, #4]

mov     r13, #MODE_SVC          @ prepare SVC-Mode
msr     spsr_c, r13
mov     lr, pc
movs    pc, lr
.endm

.macro get_irq_stack              @ setup IRQ stack
ldr     sp, IRQ_STACK_START
.endm

.macro get_fiq_stack              @ setup FIQ stack
ldr     sp, FIQ_STACK_START
.endm

/*****
/*
/* exception handlers
/*
*****/

```

```

        .align    5
undefined_instruction:
    get_bad_stack
    bad_save_user_regs
    bl        do_undefined_instruction

```

```

        .align    5
software_interrupt:
    get_bad_stack
    bad_save_user_regs
    bl        do_software_interrupt

```

```

        .align    5
prefetch_abort:
    get_bad_stack
    bad_save_user_regs
    bl        do_prefetch_abort

```

```

        .align    5
data_abort:
    get_bad_stack
    bad_save_user_regs
    bl        do_data_abort

```

```

        .align    5
not_used:
    get_bad_stack
    bad_save_user_regs
    bl        do_not_used

```

```

#ifdef CONFIG_USE_IRQ

```

```

        .align    5
irq:
    get_irq_stack
    irq_save_user_regs
    bl        do_irq
    irq_restore_user_regs

```

```

        .align    5
fiq:
    get_fiq_stack
    irq_save_user_regs    /* someone ought to write a more */
    bl        do_fiq      /* efficient fiq_save_user_regs */
    irq_restore_user_regs

```

```

#else

```

```

        .align    5
irq:
    get_bad_stack

```

```

        bad_save_user_regs
        bl        do_irq

        .align    5
fiq:
        get_bad_stack
        bad_save_user_regs
        bl        do_fiq

#endif

/*****
/*
/* Reset function: the PXA250 doesn't have a reset function, so we have to */
/* perform a watchdog timeout for a soft reset.                               */
/*
*****/

        .align    5
.globl reset_cpu

        /* FIXME: this code is PXA250 specific. How is this handled on */
        /* other XScale processors?                                     */

reset_cpu:

        /* We set OWE:WME (watchdog enable) and wait until timeout happens */

        ldr        r0, OSTIMER_BASE
        ldr        r1, [r0, #OWER]
        orr        r1, r1, #0x0001                /* bit0: WME */
        str        r1, [r0, #OWER]

        /* OS timer does only wrap every 1165 seconds, so we have to set */
        /* the match register as well.                                     */

        ldr        r1, [r0, #OSCR]                /* read OS timer */
        add        r1, r1, #0x800                  /* let OSMR3 match after */
        add        r1, r1, #0x800                  /* 4096*(1/3.6864MHz)=1ms */
        str        r1, [r0, #OSMR3]

reset_endless:

        b         reset_endless

```

board/skku/memsetup.S

```
/*
 * Most of this taken from Redboot hal_platform_setup.h with cleanup
 *
 * NOTE: I haven't clean this up considerably, just enough to get it
 * running. See hal_platform_setup.h for the source. See
 * board/cradle/memsetup.S for another PXA250 setup that is
 * much cleaner.
 *
 * See file CREDITS for list of people who contributed to this
 * project.
 *
 * This program is free software; you can redistribute it and/or
 * modify it under the terms of the GNU General Public License as
 * published by the Free Software Foundation; either version 2 of
 * the License, or (at your option) any later version.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program; if not, write to the Free Software
 * Foundation, Inc., 59 Temple Place, Suite 330, Boston,
 * MA 02111-1307 USA
 */

#include <config.h>
#include <version.h>
#include <asm/arch/pxa-regs.h>

DRAM_SIZE: .long CFG_DRAM_SIZE

/* wait for coprocessor write complete */
.macro CPWAIT reg
mrc p15,0,\reg,c2,c0,0
mov \reg,\reg
sub pc,pc,#4
.endm

/*
 * Memory setup
 */

.globl memsetup
memsetup:

    mov    r10, lr
```



```
/* Set up GPIO pins first ----- */
```

```
ldr      r0,      =GPSR0
ldr      r1,      =CFG_GPSR0_VAL
str      r1,      [r0]

ldr      r0,      =GPSR1
ldr      r1,      =CFG_GPSR1_VAL
str      r1,      [r0]

ldr      r0,      =GPSR2
ldr      r1,      =CFG_GPSR2_VAL
str      r1,      [r0]

ldr      r0,      =GPCR0
ldr      r1,      =CFG_GPCR0_VAL
str      r1,      [r0]

ldr      r0,      =GPCR1
ldr      r1,      =CFG_GPCR1_VAL
str      r1,      [r0]

ldr      r0,      =GPCR2
ldr      r1,      =CFG_GPCR2_VAL
str      r1,      [r0]

ldr      r0,      =GPDR0
ldr      r1,      =CFG_GPDR0_VAL
str      r1,      [r0]

ldr      r0,      =GPDR1
ldr      r1,      =CFG_GPDR1_VAL
str      r1,      [r0]

ldr      r0,      =GPDR2
ldr      r1,      =CFG_GPDR2_VAL
str      r1,      [r0]

ldr      r0,      =GAFR0_L
ldr      r1,      =CFG_GAFR0_L_VAL
str      r1,      [r0]

ldr      r0,      =GAFR0_U
ldr      r1,      =CFG_GAFR0_U_VAL
str      r1,      [r0]

ldr      r0,      =GAFR1_L
ldr      r1,      =CFG_GAFR1_L_VAL
str      r1,      [r0]

ldr      r0,      =GAFR1_U
ldr      r1,      =CFG_GAFR1_U_VAL
```

```

str        r1, [r0]

ldr        r0,      =GAFR2_L
ldr        r1,      =CFG_GAFR2_L_VAL
str        r1, [r0]

ldr        r0,      =GAFR2_U
ldr        r1,      =CFG_GAFR2_U_VAL
str        r1, [r0]

ldr        r0,      =PSSR          /* enable GPIO pins */
ldr        r1,      =CFG_PSSR_VAL
str        r1, [r0]

/* ----- */
/* Enable memory interface */
/*
/* The sequence below is based on the recommended init steps */
/* detailed in the Intel PXA250 Operating Systems Developers Guide, */
/* Chapter 10. */
/* ----- */

/* ----- */
/* Step 1: Wait for at least 200 microseconds to allow internal */
/* clocks to settle. Only necessary after hard reset... */
/* FIXME: can be optimized later */
/* ----- */

ldr r3, =OSCR          /* reset the OS Timer Count to zero */
mov r2, #0
str r2, [r3]
ldr r4, =0x300          /* really 0x2E1 is about 200usec, */
                        /* so 0x300 should be plenty */

1:
ldr r2, [r3]
cmp r4, r2
bgt 1b

```

mem_init:

```

ldr        r1, =MEMC_BASE          /* get memory controller base addr. */

/* ----- */
/* Step 2a: Initialize Asynchronous static memory controller */
/* ----- */

/* MSC registers: timing, bus width, mem type */

/* MSC0: nCS(0,1) */
ldr        r2, =CFG_MSC0_VAL
str        r2, [r1, #MSC0_OFFSET]

```

```

ldr    r2,    [r1, #MSC0_OFFSET] /* read back to ensure */
                                     /* that data latches */
/* MSC1: nCS(2,3) */
ldr    r2,    =CFG_MSC1_VAL
str    r2,    [r1, #MSC1_OFFSET]
ldr    r2,    [r1, #MSC1_OFFSET]

/* MSC2: nCS(4,5) */
ldr    r2,    =CFG_MSC2_VAL
str    r2,    [r1, #MSC2_OFFSET]
ldr    r2,    [r1, #MSC2_OFFSET]

/* ----- */
/* Step 2b: Initialize Card Interface */
/* ----- */

/* MECR: Memory Expansion Card Register */
//ldr    r2,    =CFG_MECR_VAL
//str    r2,    [r1, #MECR_OFFSET]
//ldr    r2,    [r1, #MECR_OFFSET]

/* MCMEM0: Card Interface slot 0 timing */
//ldr    r2,    =CFG_MCMEM0_VAL
//str    r2,    [r1, #MCMEM0_OFFSET]
//ldr    r2,    [r1, #MCMEM0_OFFSET]

/* MCMEM1: Card Interface slot 1 timing */
//ldr    r2,    =CFG_MCMEM1_VAL
//str    r2,    [r1, #MCMEM1_OFFSET]
//ldr    r2,    [r1, #MCMEM1_OFFSET]

/* MCATT0: Card Interface Attribute Space Timing, slot 0 */
//ldr    r2,    =CFG_MCATT0_VAL
//str    r2,    [r1, #MCATT0_OFFSET]
//ldr    r2,    [r1, #MCATT0_OFFSET]

/* MCATT1: Card Interface Attribute Space Timing, slot 1 */
//ldr    r2,    =CFG_MCATT1_VAL
//str    r2,    [r1, #MCATT1_OFFSET]
//ldr    r2,    [r1, #MCATT1_OFFSET]

/* MCIO0: Card Interface I/O Space Timing, slot 0 */
//ldr    r2,    =CFG_MCIO0_VAL
//str    r2,    [r1, #MCIO0_OFFSET]
//ldr    r2,    [r1, #MCIO0_OFFSET]

/* MCIO1: Card Interface I/O Space Timing, slot 1 */
//ldr    r2,    =CFG_MCIO1_VAL
//str    r2,    [r1, #MCIO1_OFFSET]
//ldr    r2,    [r1, #MCIO1_OFFSET]

/* ----- */

```

```

/* Step 2c: Write FLYCNFG  FIXME: what's that??? */
/* ----- */

/* ----- */
/* Step 2d: Initialize Timing for Sync Memory (SDCLK0) */
/* ----- */

/* Before accessing MDREFR we need a valid DRI field, so we set */
/* this to power on defaults + DRI field. */

//ldr    r3,    =CFG_MDREFR_VAL
//ldr    r2,    =0xFFFF
//and     r3,    r3, r2
//ldr    r4,    =0x03ca4000
//orr     r4,    r4, r3
//str     r4,    [r1, #MDREFR_OFFSET] /* write back MDREFR */
//ldr     r4,    [r1, #MDREFR_OFFSET]

/* Note: preserve the mdrefr value in r4 */

/* ----- */
/* Step 3: Initialize Synchronous Static Memory (Flash/Peripherals) */
/* ----- */

/* Initialize SXCNFG register. Assert the enable bits */

/* Write SXMRS to cause an MRS command to all enabled banks of */
/* synchronous static memory. Note that SXLCR need not be written */
/* at this time. */

/* FIXME: we use async mode for now */

ldr     r2, =CFG_SXCNFG_VAL
str     r2, [r1, #SXCNFG_OFFSET]
ldr     r2, [r1, #SXCNFG_OFFSET]

/* ----- */
/* Step 4: Initialize SDRAM */
/* Codes for MDREFR are fixed by GIO according to bboot */
/* ----- */

ldr     r3, =CFG_MDREFR_VAL
ldr     r0, =(MDREFR_K0RUN | MDREFR_E0PIN | MDREFR_K0DB2)
and     r3, r3, r0
ldr     r2, [r1, #MDREFR_OFFSET]
ldr     r0, =(0xFFFF | MDREFR_APD | MDREFR_SLFRSH)
and     r2, r2, r0
orr     r0, r3, r2
str     r3, [r1, #MDREFR_OFFSET]
ldr     r3, [r1, #MDREFR_OFFSET]

```

```

nop
nop
nop

ldr      r3, [r1, #MDREFR_OFFSET]
ldr      r0, =CFG_MDREFR_VAL
and      r0, r0, #(MDREFR_K1RUN | MDREFR_K2RUN | MDREFR_K1DB2 |
MDREFR_K2DB2)
orr      r3, r3, r0
str      r3, [r1, #MDREFR_OFFSET]

ldr      r3, [r1, #MDREFR_OFFSET]
bic      r3, r3, #MDREFR_SLFRSH
str      r3, [r1, #MDREFR_OFFSET]
ldr      r3, [r1, #MDREFR_OFFSET]

ldr      r0, =CFG_MDREFR_VAL
and      r0, r0, #MDREFR_E1PIN
orr      r3, r3, r0
str      r3, [r1, #MDREFR_OFFSET]
ldr      r3, [r1, #MDREFR_OFFSET]

/*
/* ----- FIX by GIO ----- */

/* Step 4d: write MDCNFG with MDCNFG:DEx deasserted (set to 0), to
/*          configure but not enable each SDRAM partition pair. */

ldr      r4,      =CFG_MDCNFG_VAL
bic      r4,      r4,      #(MDCNFG_DE0|MDCNFG_DE1)

str      r4,      [r1, #MDCNFG_OFFSET] /* write back MDCNFG */
ldr      r4,      [r1, #MDCNFG_OFFSET]

/* Step 4e: Wait for the clock to the SDRAMs to stabilize,
/*          100..200  $\mu$ sec. */

ldr r3, =OSCR /* reset the OS Timer Count to zero */
mov r2, #0
str r2, [r3]
ldr r4, =0x300 /* really 0x2E1 is about 200usec,
/* so 0x300 should be plenty */

1:
ldr r2, [r3]
cmp r4, r2
bgt 1b

/* Step 4f: Trigger a number (usually 8) refresh cycles by
/*          attempting non-burst read or write accesses to disabled */

```

```

/*      SDRAM, as commonly specified in the power up sequence      */
/*      documented in SDRAM data sheets. The address(es) used      */
/*      for this purpose must not be cacheable.                    */

ldr     r3,     =CFG_DRAM_BASE
str     r2,     [r3]
str     r2,     [r3]
str     r2,     [r3]
str     r2,     [r3]
str     r2,     [r3]
str     r2,     [r3]
str     r2,     [r3]
str     r2,     [r3]

/* Step 4g: Write MDCNFG with enable bits asserted                */
/*      (MDCNFG:DEx set to 1).                                     */

//ldr     r3,   [r1, #MDCNFG_OFFSET]
//orr     r3,   r3,   #(MDCNFG_DE0|MDCNFG_DE1)
ldr r3, =CFG_MDCNFG_VAL
str     r3,   [r1, #MDCNFG_OFFSET]

/* Step 4h: Write MDMRS.                                          */

ldr     r2,   =CFG_MDMRS_VAL
str     r2,   [r1, #MDMRS_OFFSET]

ldr r2, =CFG_MDREFR_VAL
str r2, [r1, #MDREFR_OFFSET]

/* We are finished with Intel's memory controller initialisation */

/* ----- */
/* Disable (mask) all interrupts at interrupt controller          */
/* ----- */

```

initirqs:

```

mov     r1, #0          /* clear int. level register (IRQ, not FIQ) */
ldr     r2,   =ICLR
str     r1,   [r2]

ldr     r2,   =ICMR      /* mask all interrupts at the controller */
str     r1,   [r2]

/* ----- */
/* Clock initialisation                                          */

```

```

/* ----- */

initclks:

/* Disable the peripheral clocks, and set the core clock frequency */
/* (hard-coding at 398.12MHz for now). */

/* Turn Off ALL on-chip peripheral clocks for re-configuration */
/* Note: See label 'ENABLECLKS' for the re-enabling */
ldr    r1, =CKEN
/* FIXED BY GIO for SKKU */
/* enable memory controller, OS TIMER, FFUART, */
ldr     r2, =CFG_CKEN_VAL
//mov    r2, #0
str     r2, [r1]

/* ----- */
/* FIXED BY GIO for PXA270 */
/* Every setting for Turbo mode is already done in start.S */
/* ----- */

/* default value in case no valid rotary switch setting is found */
//ldr     r2, =(CCCR_L16|CCCR_LCD|CCCR_N25) /* 520Mhz, Turbo-mode */

/* ... and write the core clock config register */
//ldr     r1, =CCCR
//str     r2, [r1]

#ifdef RTC
/* enable the 32Khz oscillator for RTC and PowerManager */

ldr     r1, =OSCC
mov     r2, #OSCC_OON
str     r2, [r1]

/* NOTE: spin here until OSCC.OOK get set, meaning the PLL */
/* has settled. */

60:
ldr     r2, [r1]
ands    r2, r2, #1
beq     60b

#endif

/* ----- */
/*
/* ----- */

/* Save SDRAM size */
ldr     r1, =DRAM_SIZE
str     r8, [r1]

```

```

        /* Interrupt init: Mask all interrupts */
ldr    r0, =ICMR /* enable no sources */
        mov r1, #0
        str r1, [r0]

        /* FIXME */

#define NODEBUG
#ifdef NODEBUG
        /*Disable software and data breakpoints */
        mov    r0,#0
        mcr    p15,0,r0,c14,c8,0 /* ibcr0 */
        mcr    p15,0,r0,c14,c9,0 /* ibcr1 */
        mcr    p15,0,r0,c14,c4,0 /* dbcon */

        /*Enable all debug functionality */
        mov    r0,#0x80000000
        mcr    p14,0,r0,c10,c0,0 /* dcsr */

#endif

        /* ----- */
        /* End memsetup */
        /* ----- */

endmemsetup:

        mov    pc, lr

```


board/skku/skku.c

```
/*
 * (C) Copyright 2002
 * Kyle Harris, Nexus Technologies, Inc. kharris@nexus-tech.net
 *
 * (C) Copyright 2002
 * Sysgo Real-Time Solutions, GmbH <www.elinos.com>
 * Marius Groeger <mgroeger@sysgo.de>
 *
 * See file CREDITS for list of people who contributed to this
 * project.
 *
 * This program is free software; you can redistribute it and/or
 * modify it under the terms of the GNU General Public License as
 * published by the Free Software Foundation; either version 2 of
 * the License, or (at your option) any later version.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program; if not, write to the Free Software
 * Foundation, Inc., 59 Temple Place, Suite 330, Boston,
 * MA 02111-1307 USA
 */

#include <common.h>

/* ----- */

/*
 * Miscellaneous platform dependent initialisations
 */

int board_init (void)
{
    DECLARE_GLOBAL_DATA_PTR;

    /* memory and cpu-speed are setup before relocation */
    /* so we do _nothing_ here */

    /* arch number of Lubbock-Board */
    gd->bd->bi_arch_number = MACH_TYPE_LUBBOCK;

    /* adress of boot parameters */
    /* FIX BY GIO */
    gd->bd->bi_boot_params = 0xa4000100;
```

```

        return 0;
    }

int board_late_init(void)
{
    setenv("stdout", "serial");
    setenv("stderr", "serial");
    return 0;
}

int dram_init (void)
{
    DECLARE_GLOBAL_DATA_PTR;

    /* ADD BY GIO */
    /* SKKU board has only one bank */
    gd->bd->bi_dram[0].start = CFG_DRAM_BASE;
    gd->bd->bi_dram[0].size = CFG_DRAM_SIZE;

    return 0;
}

```

board/skku/flash.c

```
/*
 * (C) Copyright 2001
 * Kyle Harris, Nexus Technologies, Inc. kharris@nexus-tech.net
 *
 * (C) Copyright 2001
 * Wolfgang Denk, DENX Software Engineering, wd@denx.de.
 *
 * See file CREDITS for list of people who contributed to this
 * project.
 *
 * This program is free software; you can redistribute it and/or
 * modify it under the terms of the GNU General Public License as
 * published by the Free Software Foundation; either version 2 of
 * the License, or (at your option) any later version.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program; if not, write to the Free Software
 * Foundation, Inc., 59 Temple Place, Suite 330, Boston,
 * MA 02111-1307 USA
 */

#include <common.h>
#include <linux/byteorder/swab.h>

flash_info_t flash_info[CFG_MAX_FLASH_BANKS]; /* info for FLASH chips */

/* Board support for 1 or 2 flash devices */
#define FLASH_PORT_WIDTH32
#undef FLASH_PORT_WIDTH16

#ifdef FLASH_PORT_WIDTH16
#define FLASH_PORT_WIDTH ushort
#define FLASH_PORT_WIDTHV vu_short
#define SWAP(x) __swab16(x)
#else
#define FLASH_PORT_WIDTH ulong
#define FLASH_PORT_WIDTHV vu_long
#define SWAP(x) __swab32(x)
#endif

#define FPW FLASH_PORT_WIDTH
#define FPWV FLASH_PORT_WIDTHV

#define mb() __asm__ __volatile__ (" : : : \"memory\"")
```

```

/*-----
 * Functions
 */
static ulong flash_get_size (FPW *addr, flash_info_t *info);
static int write_data (flash_info_t *info, ulong dest, FPW data);
static void flash_get_offsets (ulong base, flash_info_t *info);
void inline spin_wheel (void);

/*-----
 */

unsigned long flash_init (void)
{
    int i;
    ulong size = 0;

    for (i = 0; i < CFG_MAX_FLASH_BANKS; i++) {
        switch (i) {
            case 0:
                flash_get_size ((FPW *) PHYS_FLASH_1, &flash_info[i]);
                flash_get_offsets (PHYS_FLASH_1, &flash_info[i]);
                printf("%s %d\n", __FILE__, __LINE__);
                break;

            case 1:
                flash_get_size ((FPW *) PHYS_FLASH_2, &flash_info[i]);
                flash_get_offsets (PHYS_FLASH_2, &flash_info[i]);
                break;

            default:
                panic ("configured too many flash banks!\n");
                break;

        }
        size += flash_info[i].size;
    }

    /* Protect monitor and environment sectors
     */
    flash_protect ( FLAG_PROTECT_SET,
                    CFG_FLASH_BASE,
                    CFG_FLASH_BASE + monitor_flash_len - 1,
                    &flash_info[0] );

    flash_protect ( FLAG_PROTECT_SET,
                    CFG_ENV_ADDR,
                    CFG_ENV_ADDR + CFG_ENV_SIZE - 1, &flash_info[0] );

    flash_print_info(&flash_info[0]);

    return size;
}

/*-----

```

```

    */
static void flash_get_offsets (ulong base, flash_info_t *info)
{
    int i;

    if (info->flash_id == FLASH_UNKNOWN) {
        printf("flash_get_offsets() failed.. %s %d\n", __FILE__, __LINE__);
        return;
    }

    if ((info->flash_id & FLASH_VENDMASK) == FLASH_MAN_INTEL) {
        for (i = 0; i < info->sector_count; i++) {
            info->start[i] = base + (i * PHYS_FLASH_SECT_SIZE);
            info->protect[i] = 0;
        }
    }
}

/*-----
    */
void flash_print_info (flash_info_t *info)
{
    int i;

    if (info->flash_id == FLASH_UNKNOWN) {
        printf ("missing or unknown FLASH type\n");
        return;
    }

    switch (info->flash_id & FLASH_VENDMASK) {
    case FLASH_MAN_INTEL:
        printf ("INTEL ");
        break;
    default:
        printf ("Unknown Vendor ");
        break;
    }

    switch (info->flash_id & FLASH_TYPEMASK) {
    case FLASH_28F128J3A:
        printf ("28F128J3A\n");
        break;

    /* FIX BY GIO */
    case FLASH_28F128K3:
        printf("28F128K3\n");
        break;

    default:
        printf ("Unknown Chip Type\n");
        break;
    }
}

```

```

printf ("   Size: %ld MB in %d Sectors\n",
        info->size >> 20, info->sector_count);

printf ("   Sector Start Addresses:");
for (i = 0; i < info->sector_count; ++i) {
    if ((i % 5) == 0)
        printf ("\n   ");
    printf (" %08lX%s",
            info->start[i],
            info->protect[i] ? " (RO)" : "   ");
}
printf ("\n");
return;
}

/*
 * The following code cannot be run from FLASH!
 */
static ulong flash_get_size (FPW *addr, flash_info_t *info)
{
    volatile FPW value;

    /* Write auto select command: read Manufacturer ID */
    addr[0x5555] = (FPW) 0x00AA00AA;
    addr[0x2AAA] = (FPW) 0x00550055;
    //addr[0x5555] = (FPW) 0x00900090;
    addr[0] = (FPW) 0x00900090;

    mb ();
    value = addr[0];

    switch (value) {

    case (FPW) INTEL_MANUFACT:
        info->flash_id = FLASH_MAN_INTEL;
        printf("info->flash_id = %x\n", info->flash_id);
        break;

    default:
        printf("flash_get_size() failed.. %s %d\n", __FILE__, __LINE__);
        info->flash_id = FLASH_UNKNOWN;
        info->sector_count = 0;
        info->size = 0;
        addr[0] = (FPW) 0x00FF00FF;          /* restore read mode */
        return (0);                          /* no or unknown flash */
    }

    mb ();
    value = addr[1];                        /* device ID */

    switch (value) {

```

```

case (FPW) INTEL_ID_28F128J3A:
    info->flash_id += FLASH_28F128J3A;
    info->sector_count = 128;
    info->size = 0x02000000;
    break;                                     /* => 16 MB */

/* FIX BY GIO */
case (FPW) INTEL_ID_28F128K3:
    printf("28F128K3 detected.. %s %d\n", __FILE__, __LINE__);
    info->flash_id += FLASH_28F128K3;
    info->sector_count = CFG_MAX_FLASH_SECT;
    info->size = CFG_FLASH_SIZE;
    printf("id = %x, sector_count = %x, size = %x\n",
           info->flash_id, info->sector_count, info->size);
    break;
default:
    info->flash_id = FLASH_UNKNOWN;
    break;
}

if (info->sector_count > CFG_MAX_FLASH_SECT) {
    printf("*** ERROR: sector count %d > max (%d) ***\n",
           info->sector_count, CFG_MAX_FLASH_SECT);
    info->sector_count = CFG_MAX_FLASH_SECT;
}

addr[0] = (FPW) 0xFFFFFFFF; /* 0x00FF00FF;*/          /* restore read mode */

printf("flash_get_size() completed.. %s %d\n", __FILE__, __LINE__);

return (info->size);
}

/*-----
*/

int flash_erase (flash_info_t *info, int s_first, int s_last)
{
    int flag, prot, sect;
    ulong type, start, last;
    int rcode = 0;

    if ((s_first < 0) || (s_first > s_last)) {
        if (info->flash_id == FLASH_UNKNOWN) {
            printf ("- missing\n");
        } else {
            printf ("- no sectors to erase\n");
        }
        return 1;
    }
}

```

```

type = (info->flash_id & FLASH_VENDMASK);
if ((type != FLASH_MAN_INTEL)) {
    printf ("Can't erase unknown flash type %08lx - aborted\n",
            info->flash_id);
    return 1;
}

prot = 0;
for (sect = s_first; sect <= s_last; ++sect) {
    if (info->protect[sect]) {
        prot++;
    }
}

if (prot) {
    printf ("- Warning: %d protected sectors will not be erased!\n",
            prot);
} else {
    printf ("\n");
}

start = get_timer (0);
last = start;

/* Disable interrupts which might cause a timeout here */
flag = disable_interrupts ();

/* Start erase on unprotected sectors */
for (sect = s_first; sect <= s_last; sect++) {
    if (info->protect[sect] == 0) { /* not protected */
        FPWV *addr = (FPWV *) (info->start[sect]);
        FPW status;

        printf ("Erasing sector %2d ... ", sect);

        /* arm simple, non interrupt dependent timer */
        reset_timer_masked ();

        *addr = (FPW) 0x00500050; /* clear status register */
        *addr = (FPW) 0x00200020; /* erase setup */
        *addr = (FPW) 0x00D000D0; /* erase confirm */

        while (((status = *addr) & (FPW) 0x00800080) != (FPW) 0x00800080) {
            if (get_timer_masked () > CFG_FLASH_ERASE_TOUT) {
                printf ("Timeout\n");
                *addr = (FPW) 0x00B000B0; /* suspend erase */
                *addr = (FPW) 0x00FF00FF; /* reset to read mode */
                rcode = 1;
                break;
            }
        }
    }
}

```



```

        *addr = 0x00500050;          /* clear status register cmd.    */
        *addr = 0xFFFFFFFF; /* 0x00FF00FF; */ /* reset to read mode */

        printf (" done\n");
    }
}
return rcode;
}

/*-----
 * Copy memory to flash, returns:
 * 0 - OK
 * 1 - write timeout
 * 2 - Flash not erased
 * 4 - Flash not identified
 */

int write_buff (flash_info_t *info, uchar *src, ulong addr, ulong cnt)
{
    ulong cp, wp;
    FPW data;
    int count, i, l, rc, port_width;

    if (info->flash_id == FLASH_UNKNOWN) {
        return 4;
    }

    /* get lower word aligned address */
#ifdef FLASH_PORT_WIDTH16
    wp = (addr & ~1);
    port_width = 2;
#else
    wp = (addr & ~3);
    port_width = 4;
#endif

    /*
     * handle unaligned start bytes
     */
    if ((l = addr - wp) != 0) {
        data = 0;
        for (i = 0, cp = wp; i < l; ++i, ++cp) {
            data = (data << 8) | (*(uchar *) cp);
        }
        for (; i < port_width && cnt > 0; ++i) {
            data = (data << 8) | *src++;
            --cnt;
            ++cp;
        }
        for (; cnt == 0 && i < port_width; ++i, ++cp) {
            data = (data << 8) | (*(uchar *) cp);

```

```

        }

        if ((rc = write_data (info, wp, SWAP (data))) != 0) {
            return (rc);
        }
        wp += port_width;
    }

    /*
     * handle word aligned part
     */
    count = 0;
    while (cnt >= port_width) {
        data = 0;
        for (i = 0; i < port_width; ++i) {
            data = (data << 8) | *src++;
        }
        if ((rc = write_data (info, wp, SWAP (data))) != 0) {
            return (rc);
        }
        wp += port_width;
        cnt -= port_width;
        if (count++ > 0x800) {
            spin_wheel ();
            count = 0;
        }
    }

    if (cnt == 0) {
        return (0);
    }

    /*
     * handle unaligned tail bytes
     */
    data = 0;
    for (i = 0, cp = wp; i < port_width && cnt > 0; ++i, ++cp) {
        data = (data << 8) | *src++;
        --cnt;
    }
    for (; i < port_width; ++i, ++cp) {
        data = (data << 8) | (*(uchar *) cp);
    }

    return (write_data (info, wp, SWAP (data)));
}

/*-----
 * Write a word or halfword to Flash, returns:
 * 0 - OK
 * 1 - write timeout
 * 2 - Flash not erased

```

```

*/
static int write_data (flash_info_t *info, ulong dest, FPW data)
{
    FPWV *addr = (FPWV *) dest;
    ulong status;
    int flag;

    /* Check if Flash is (sufficiently) erased */
    if ((*addr & data) != data) {
        printf ("not erased at %08lx (%lx)\n", (ulong) addr, *addr);
        return (2);
    }
    /* Disable interrupts which might cause a timeout here */
    flag = disable_interrupts ();

    *addr = (FPW) 0x00400040; /* write setup */
    *addr = data;

    /* arm simple, non interrupt dependent timer */
    reset_timer_masked ();

    /* wait while polling the status register */
    while (((status = *addr) & (FPW) 0x00800080) != (FPW) 0x00800080) {
        if (get_timer_masked () > CFG_FLASH_WRITE_TOUT) {
            *addr = (FPW) 0x00FF00FF; /* restore read mode */
            return (1);
        }
    }

    *addr = (FPW) 0xFFFFFFFF; /* 0x00FF00FF; */ /* restore read mode */

    return (0);
}

void inline spin_wheel (void)
{
    static int p = 0;
    static char w[] = "\\-/";

    printf ("\010%c", w[p]);
    (++p == 3) ? (p = 0) : 0;
}

```

기타 참고 사항 - 메모

- 부트로더가 플래시에 쓰이는 시작 주소, 끝주소
 - HYBUS :
 - SKKU
- 부트로더가 램으로 복사되는 시작 주소, 끝주소
 - HYBUS : 0xa3f00000 (64mb - 1mb)
 - SKKU
 -
- 부트 파라미터가 전달되는 주소
 - HYBUS : 0xa0000100
 - SKKU : 0xa0000100
- 커널이 복사되는 주소
 - HYBUS :
 - SKKU
- PXA27x에서 nCS<0:5>는 다음과 같은 시작 주소값을 가진다.
- nCS0 : 0x00000000
- nCS1 : 0x04000000
- nCS2 : 0x08000000
- nCS3 : 0x0c000000
- nCS4 : 0x10000000
- nCS5 : 0x14000000
- PCMCIA의 시작 주소
- CS8900의 시작 주소 : 0x04000300 (nCS0으로 설정했을 때)
-

```
#define CFI_CHIP_INTEL_28F128K3          0x8802
#define CFI_CHIPN_INTEL_28F128K3      "28F128K3"
```

28f128k3는 Intel StrataFlash® Synchronous Memory (K3/K18) 28F640K3, 28F640K18, 28F128K3, 28F128K18, 28F256K3, 28F256K18 (x16) Datasheet 를 참조하면 128Mbit크기이고 128섹터로 이루어져 있으므로 한 섹터는 16Mbyte/128=128kbyte라는 것을 알 수 있다. SKKU보드는 2개의 플래시 메모리를 사용하므로 한 섹터의 크기는 256Kbyte가 되고 총 메모리 크기는 32Mbyte가 된다.

linux-2.6.11/arch/arm/mach-pxa/mainstone.c 에 다음과 같이 추가해야 함. (sodiff 패치 참고)

```
+static struct resource cs8900_resources[] = {
+    [0] = {
+        .start = PA_CS8900_BASE + 0x300, -> cs8900칩 레지스터 시작 주소
+        .end   = PA_CS8900_BASE + 0xffff,
```

```

+         .flags      =      IORESOURCE_MEM,
+     },

    static struct map_desc mainstone_io_desc[] __initdata = {
-     { MST_FPGA_VIRT, MST_FPGA_PHYS, 0x00100000, MT_DEVICE }, /* CPLD */
+     {      VA_CS8900_BASE,  PA_CS8900_BASE,  0x00100000,      MT_DEVICE      },
+     /* CS8900 */
+     {      MST_FPGA_VIRT,  MST_FPGA_PHYS,  0x00100000,      MT_DEVICE      },
+     /* CPLD */
    };

```

start.S에 bl memsetup 이라는 코드가 있다. board/memsetup.S로 점프한다.
start.S는 PXA250과 다른 코드가 없다. 프로세서 속도와 부트로더 시작 주소만 주의할 것.
start.S -> board/memsetup.S ->

configs/sodiff.h 에서 memset.S에서 사용되는 다음 변수들을 수정함. (blob-mainstone과 bboot을 참조해서 값을 가져옴)

```

CFG_MDREFR_VAL
CFG_MSC0_VAL
CFG_MSC1_VAL
CFG_MDCNFG_VAL
CFG_MDMRS_VAL

```

➔ 이 값들이 옳은 값인지를 알 수 없음.

memsetup.S의 228라인에 SXCNFG와 SXMRS의 값을 쓰라고 나왔는데 코드는 없음.
memsetup.S에서 247라인에서 시작하는 init_sdram 코드는 고치기가 어렵다.

FFUART 동작하는 소스를 led test에 추가한다.

- . GPIO<34-41>이 FFUART에 사용되므로 이 GPIO들에 대한 설정 필요
- . CLK_REQ 레지스터 : 클럭의 소스 결정
- . CCRS, CKEN, CLKCFG 설정
- . Turbo mode로 들어가는 방법

- . 메모리 컨트롤러 각 레지스터에 대한 설명
- . GPIO 설정, 메모리 설정에 대한 설명
- . 설정하는 방법, 매뉴얼 읽는 방법

