Email:1192872431@qq.com

Transplant UBOOT on S5PV210

Development Environment

Hardware:

Processor:S5PV210

Board:X210

Softwawre:

UBOOT VERSION:1.3.4

Compiler:GCC(CROSS COMPILE)

OVERVIEW

I transplant uboot on x210 on the basis of that uboot offered by samsung.That uboot is suitable for SMDKV210 and I will provide this version uboot later.

Tasks before transplant

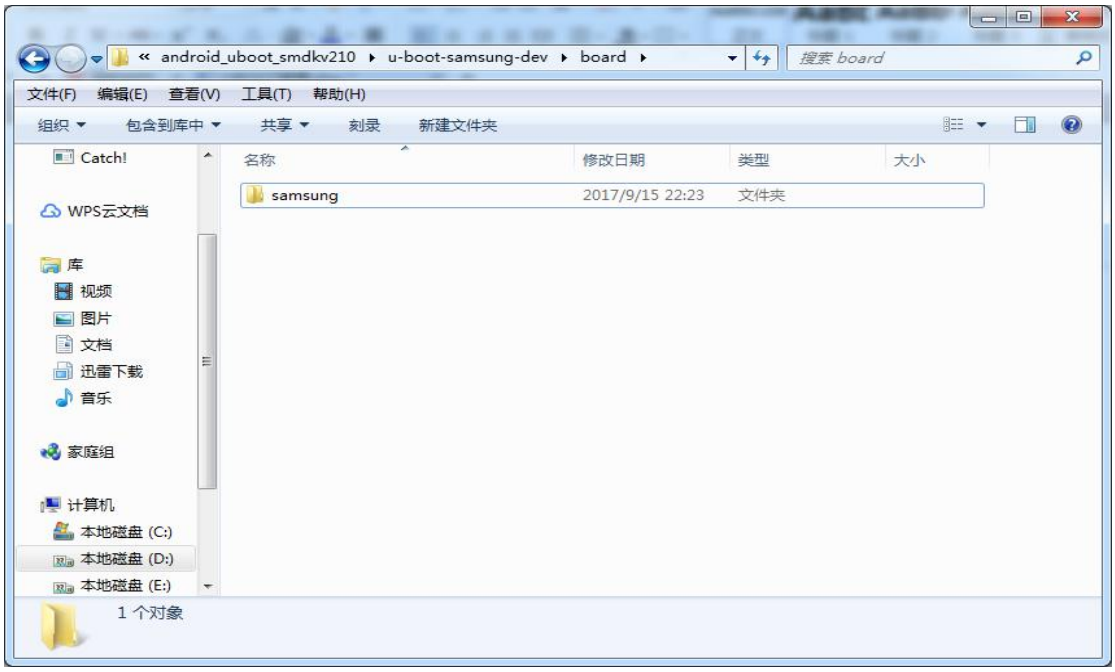
Delete irrelevant files

Note that this step is to make it clear while readding and transplanting, and it's not a must.

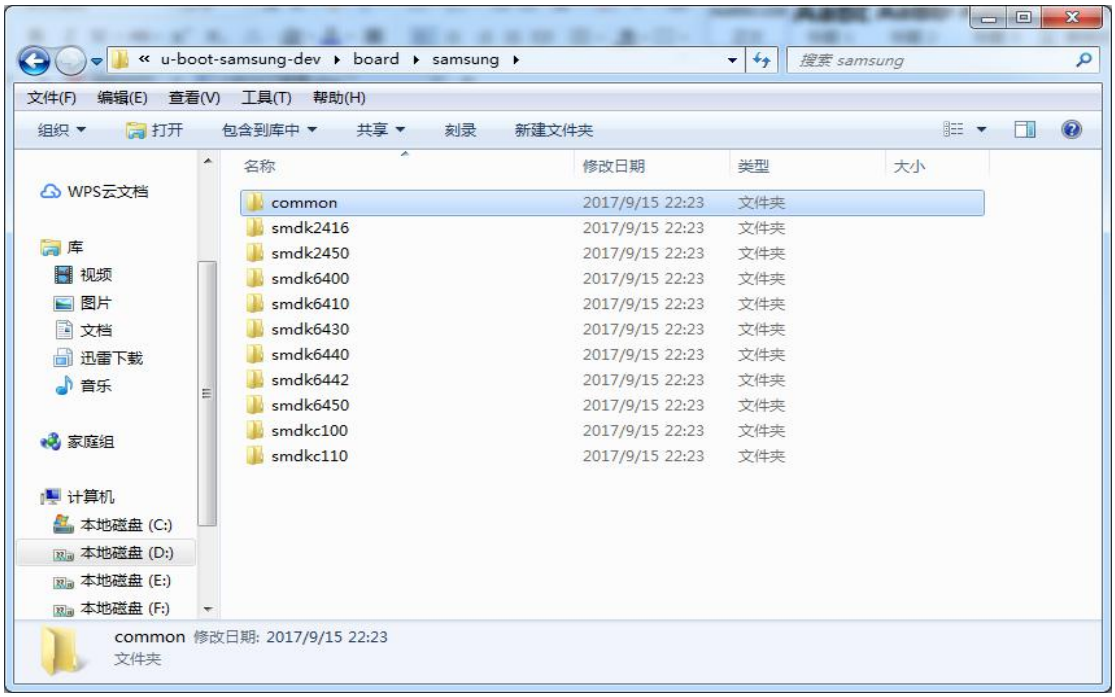
1. Director "/board"

There are different uboot code refered to different kinds of processors and boards.And these sub-director are primarily named after their manufacturers.

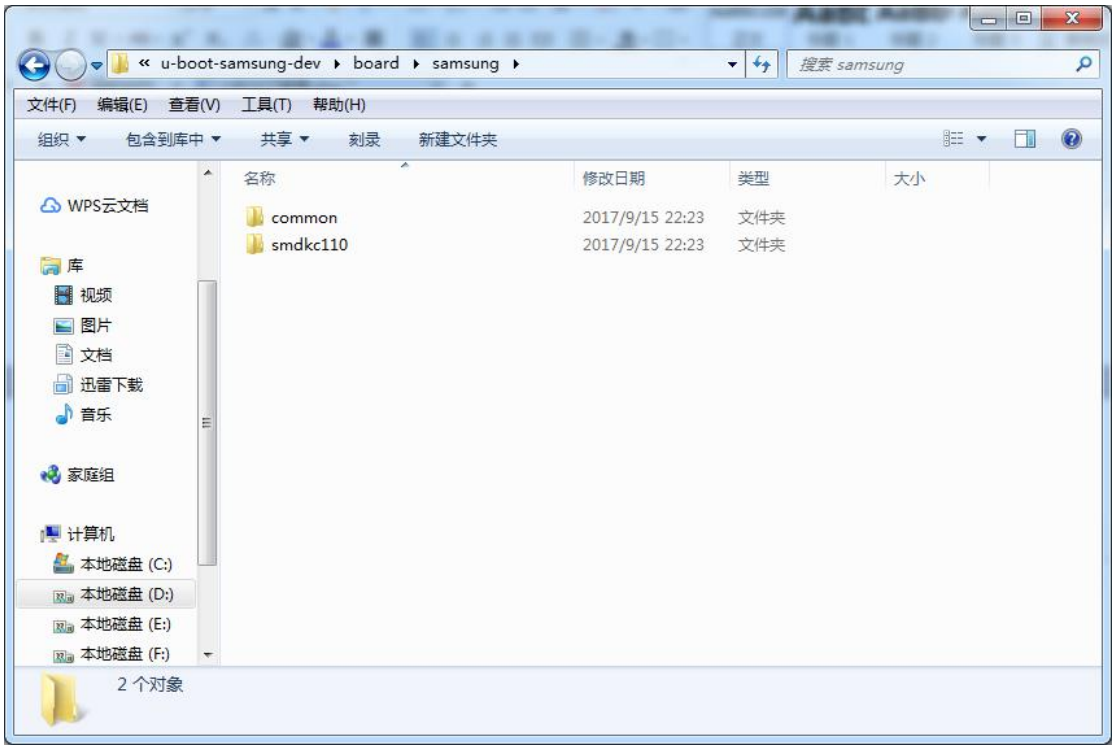
I only reserve folder "samsung".



1.1 Enter folder “samsung”, and there are several subfolders, and each folder represents one board from samsung.

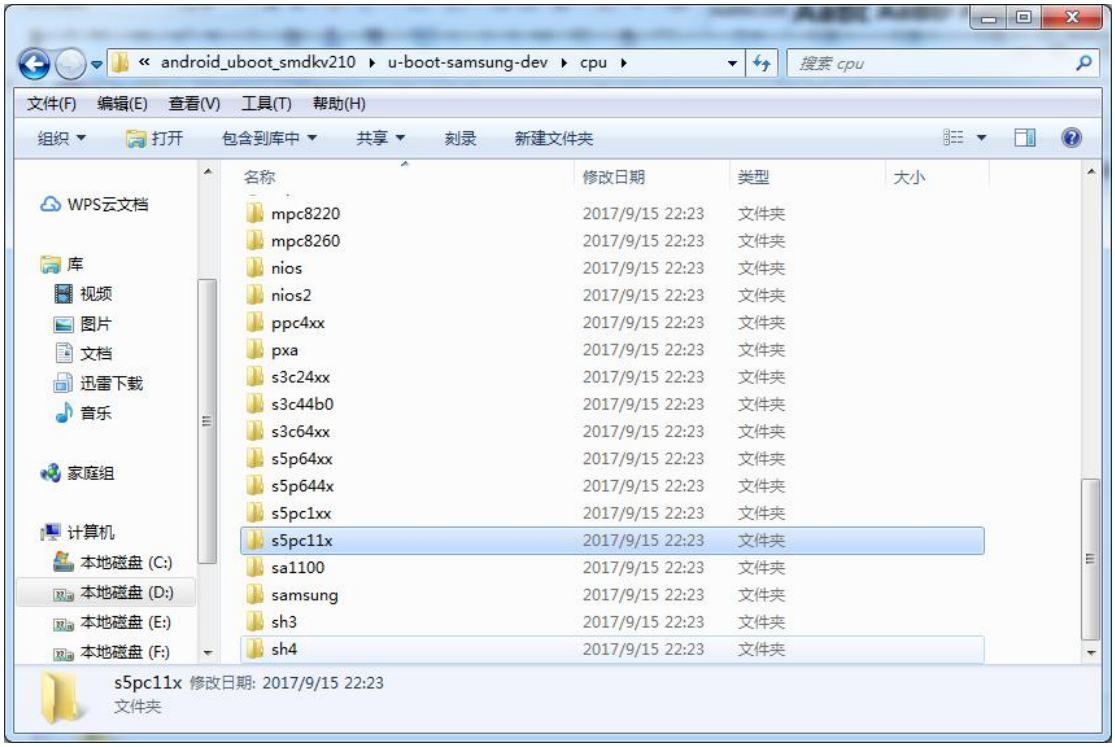


S5PV210 is similar to S5PC110, so I plan to transplant besed on smdkc110. As a result, I deleted some folders except “common” and “smdkc110”.



2. Director "/cpu"

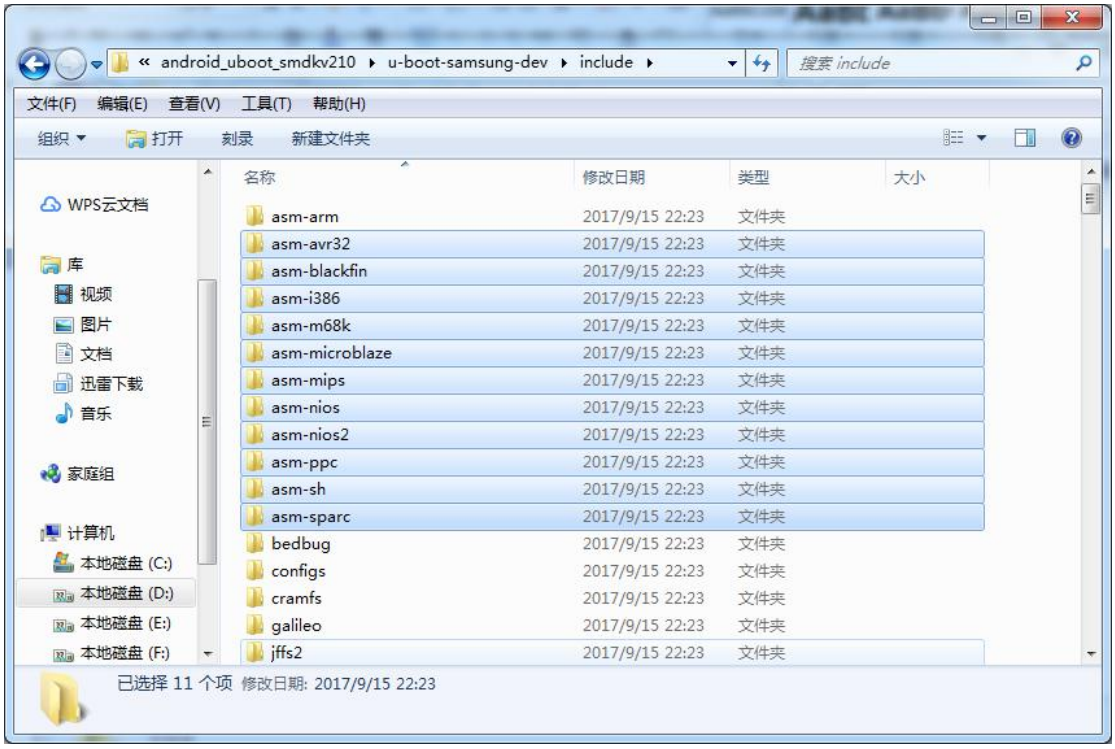
There are some kinds of code refered to different cpu architecture, and each subfolder represent one architecture.



I only reserved folder "s5pc11x" and deleted the others.

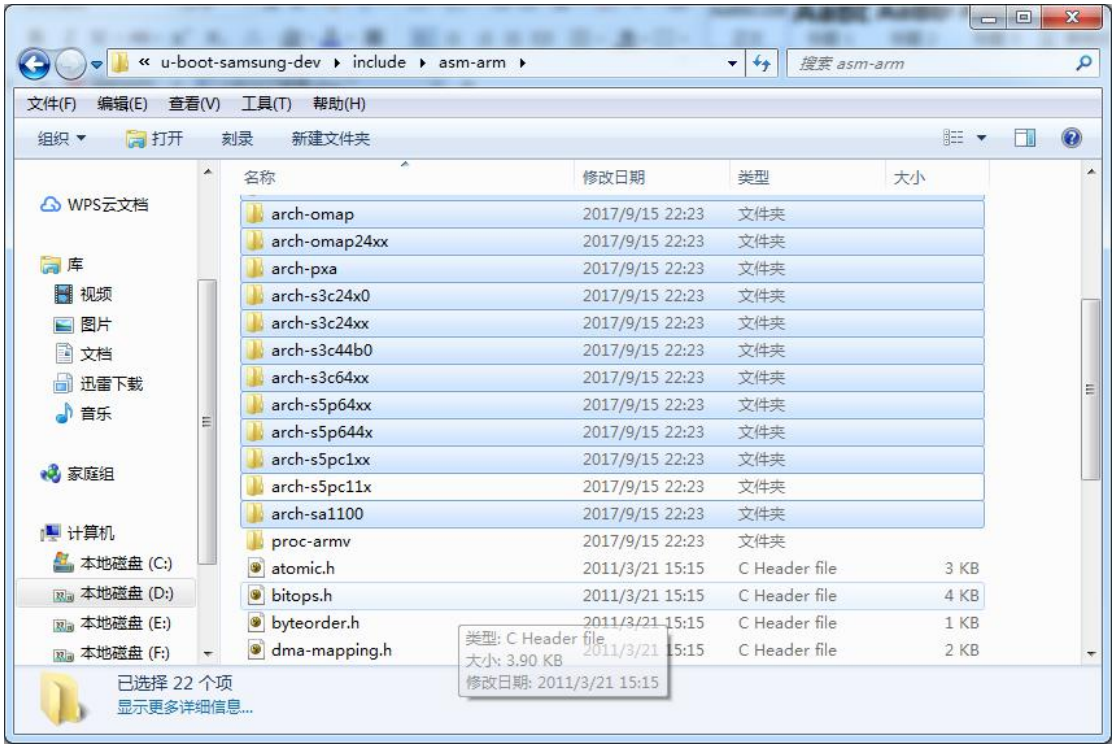
3. Director "/include"

This director contains the common header files refered to cpu architecture.

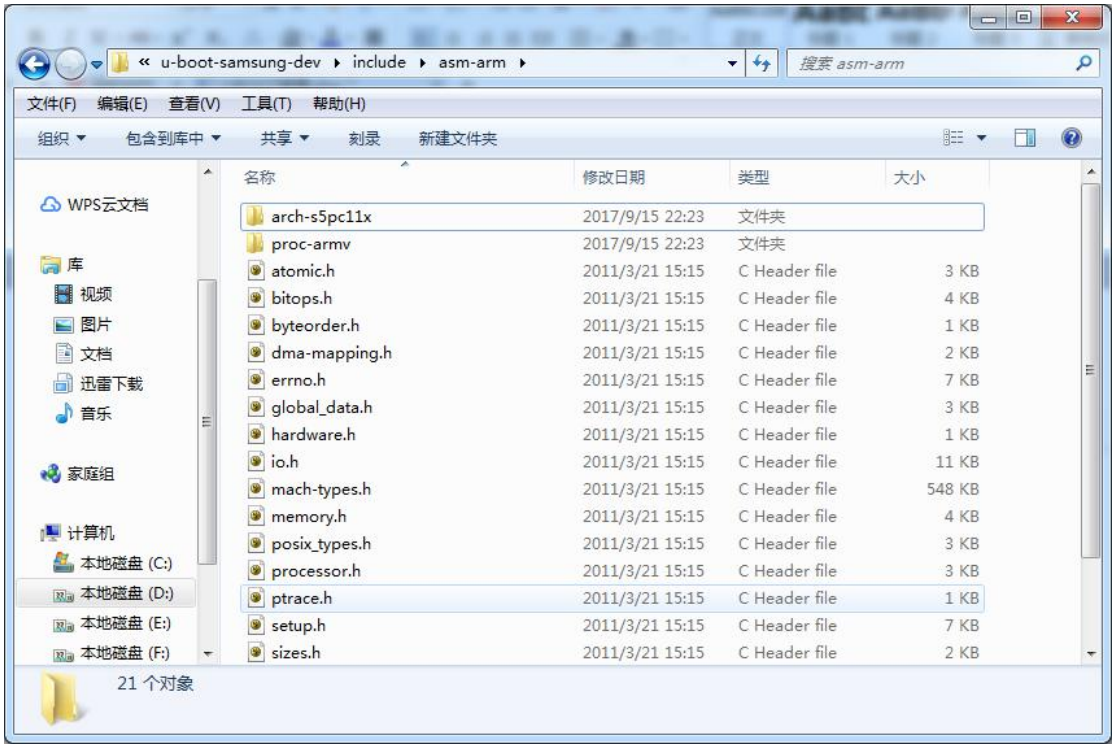


Because S5PV210 is a cpu based on arm, I deleted the irrelevant folders selected in above image.

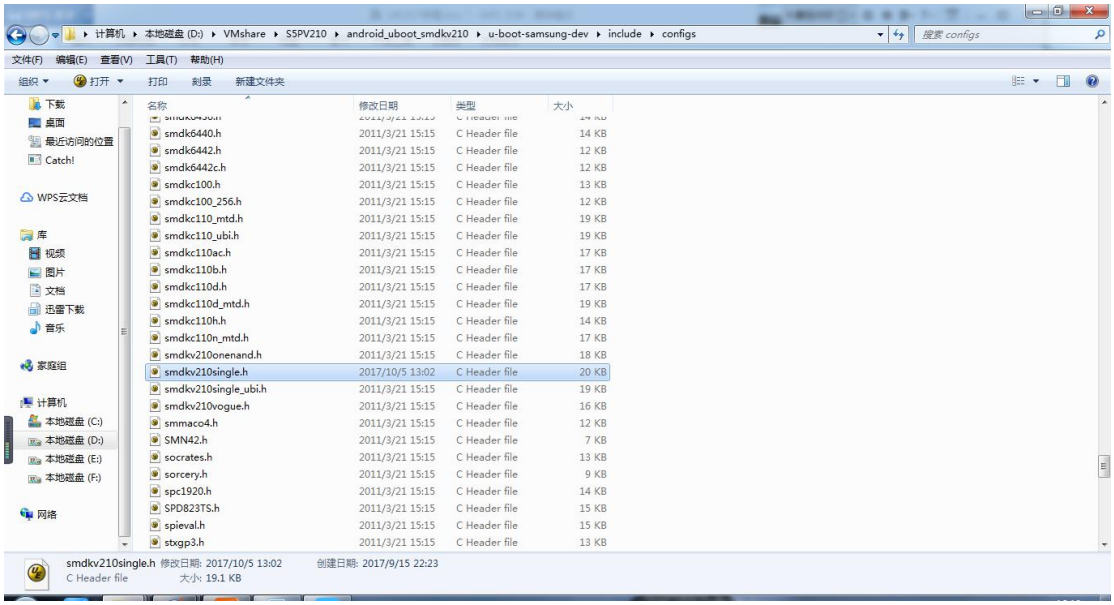
3.1 Enter director "asm-arm". There exists subfolders refered to specified processors based on arm.



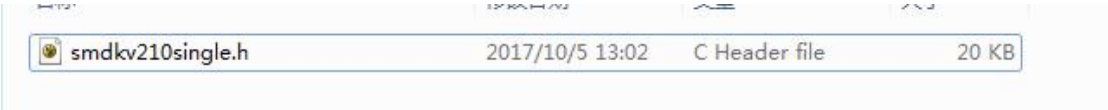
We only need to keep folder "arch-s5pc11x" and other common files.



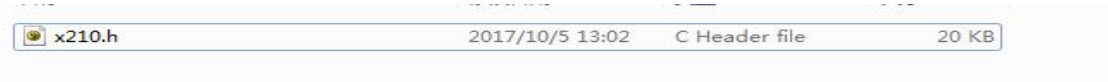
3.2 Enter director "\include\configs". There are header files to configure uboot. And one header file represent one type of configuration for uboot. And the key point to transplant uboot is to configure this header file.



We only keep one as our transplant template, and I choosed smdkv210single.h.

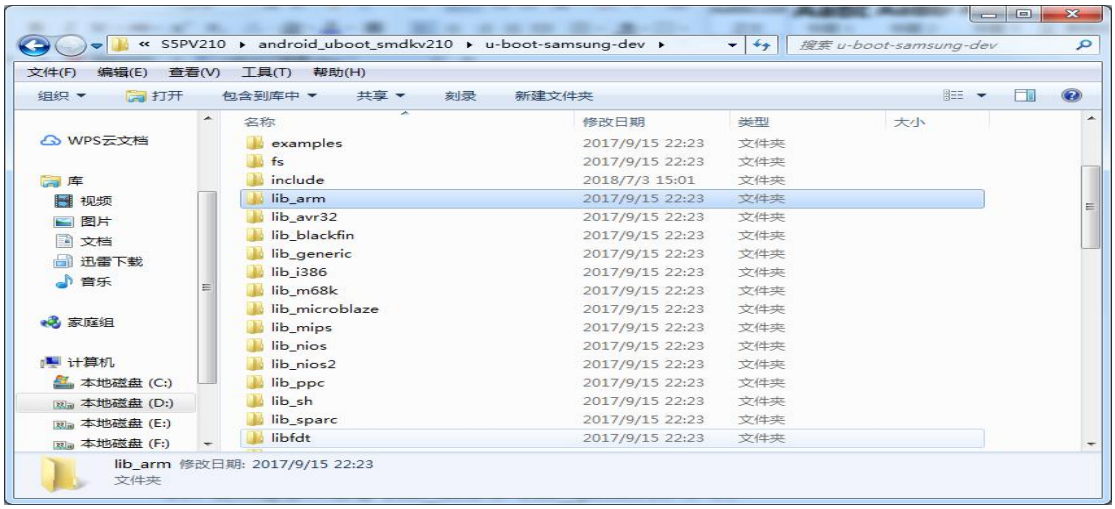


And then, rename it as x210.h.

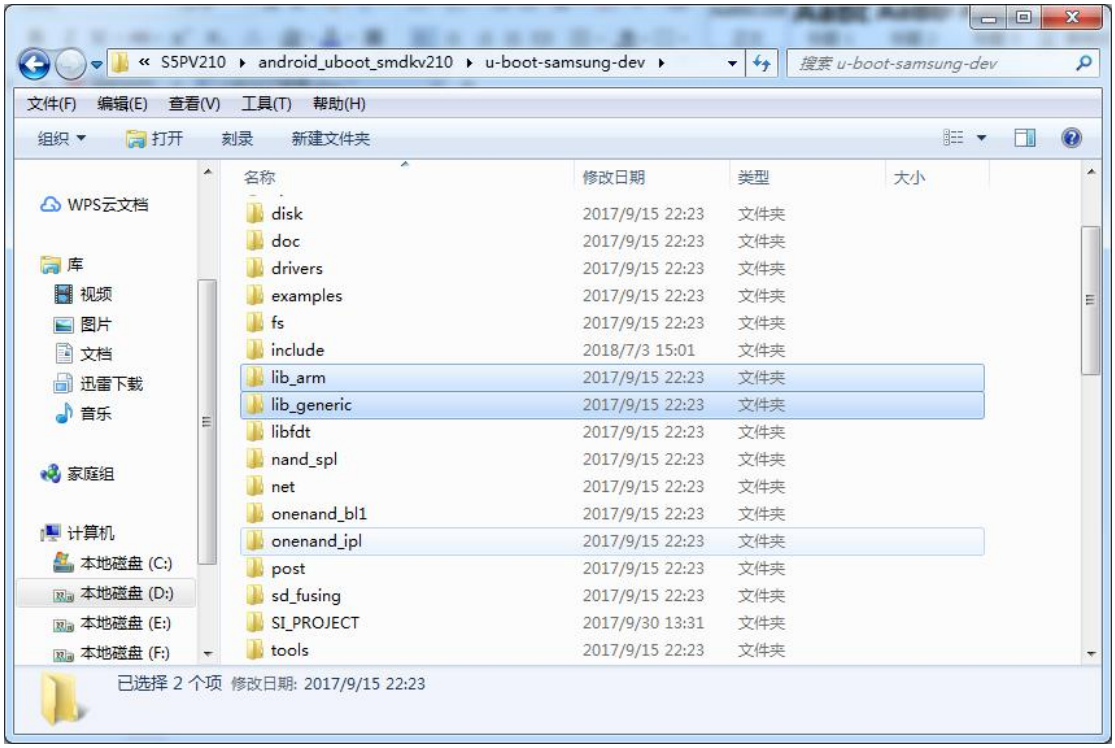


4. Folders "lib_xxx"

A series of lib_xxx folders, these are the library files associated with the cpu.

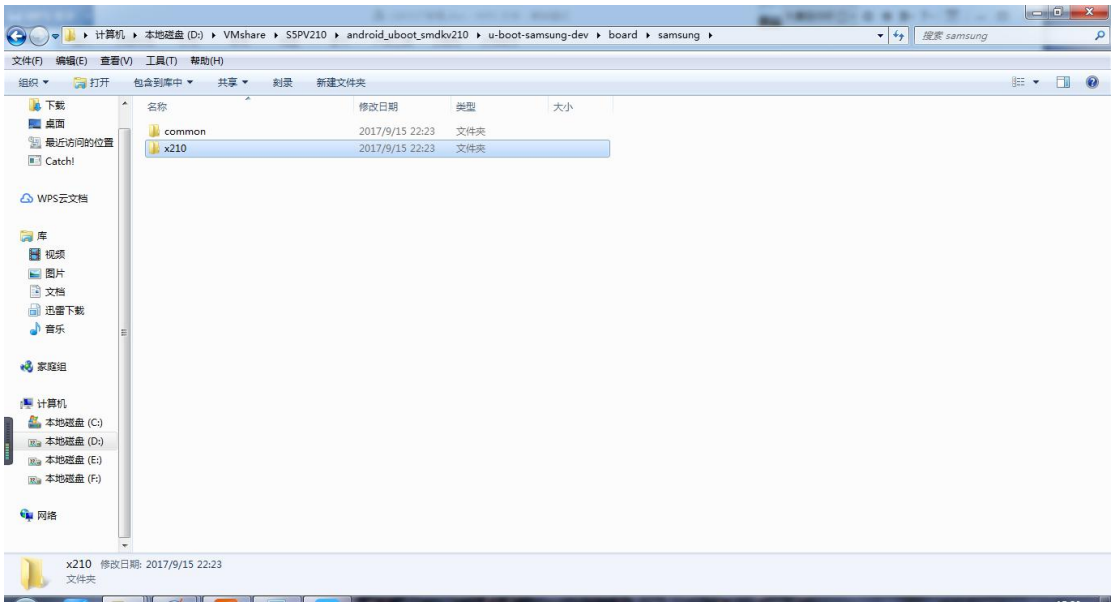


I only kept "lib_arm" and "lib_generic".

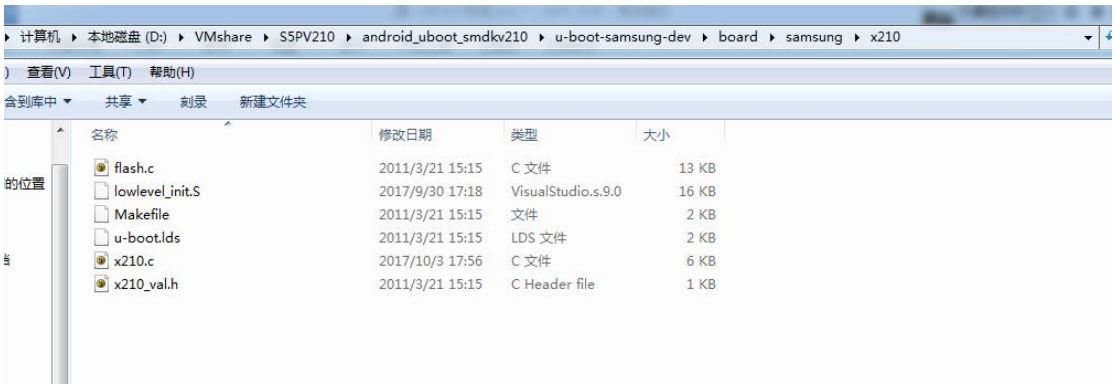


5.We need to rename some relevant pathname according to the board we use.

5.1Rename \board\samsung\smdkc110 to \board\samsung\x210



5.2 Enter \board\samsung\x210.Rename smdkc110.c to x210.c, and rename smdkc110_val.h to x210_val.h.



And then, revise makefile in current director:
change "OBJS := smdkc110.o flash.o"

```
OBJS      := smdkc110.o flash.o
```

to "OBJS := x210.0 flash.o"

```
OBJS      := x210.o flash.o
```

And revise "uboot_ld.s":
Old:

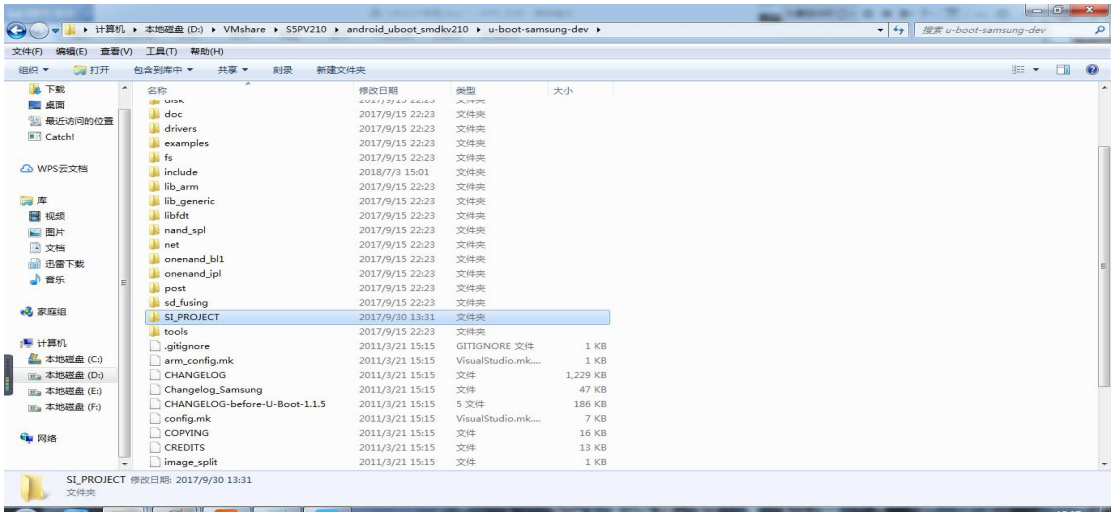
```
board/samsung/smdkc110/lowlevel_init.o     (.text)
```

New:

```
board/samsung/x210/lowlevel_init.o     (.text)
```

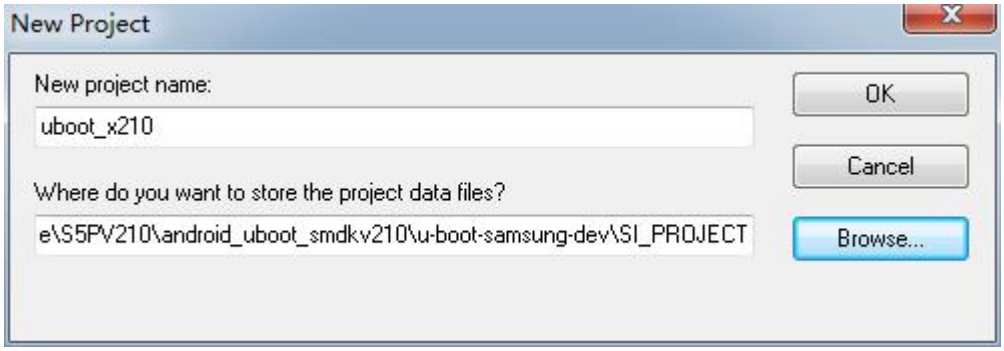
Build Source Insight project

I am used to make a folder named "SI_PROJECT" in the top level director, and to position source insight project files in SI_PROJECT.

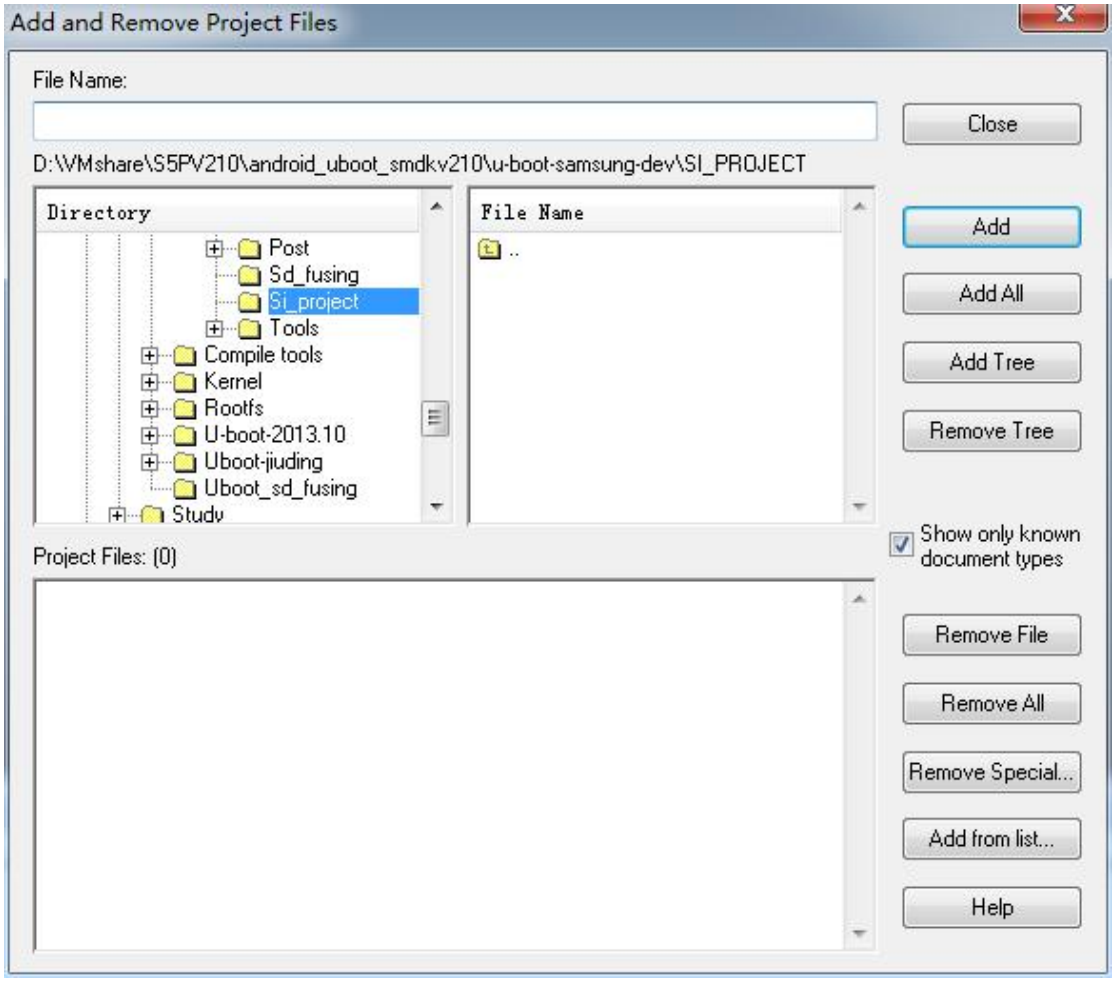


1. Open source insight , and click "Project->New project" to create a new Source Insight project.
2. My project name is "uboot_x210", and this name is up to you. And

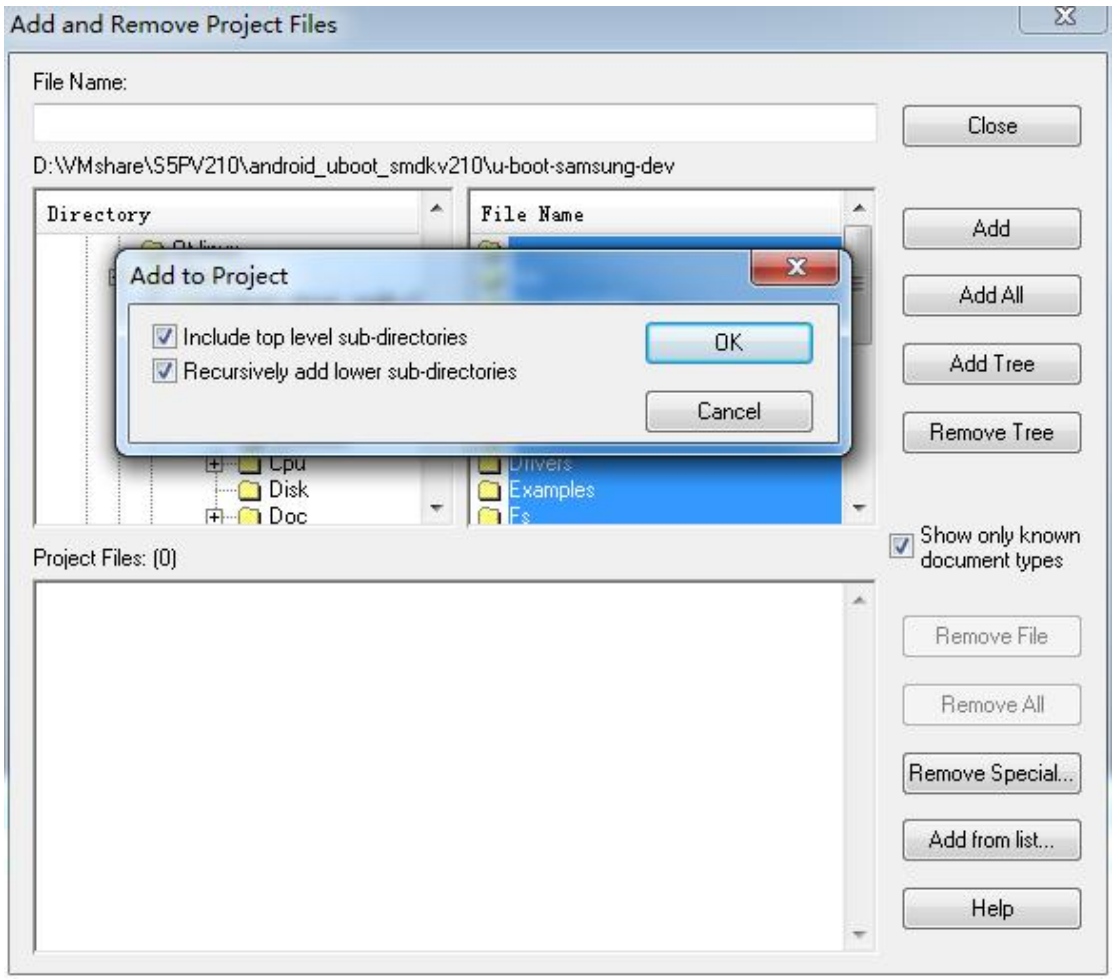
then, click “Browser” to find the folder “SI_PROJECT” you just created.Click “OK” in the end.



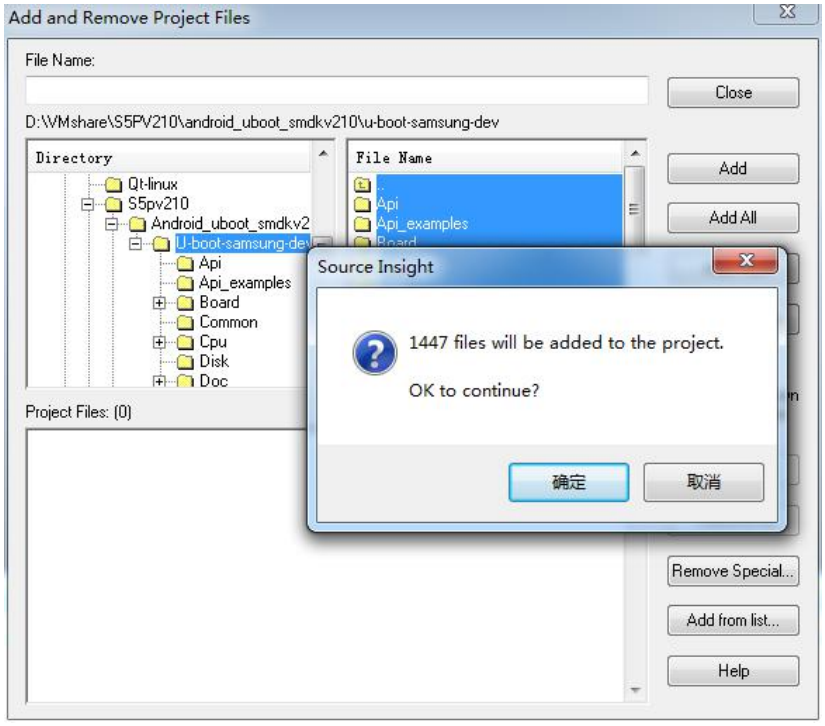
3. Click “OK” again, and the bellow image will appear.



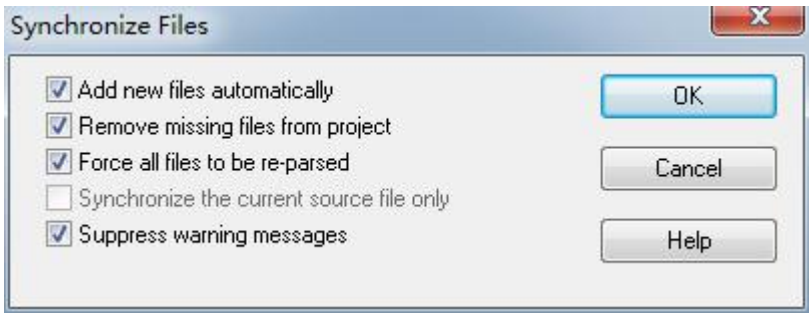
Click the top level director, and then click “Add all”. Select option “Recursively add lower sub-directories” in the new dialog box and click “OK”.



In the new dialog box, it will show the number of added files. The dialog box will disappear after you click "OK". Finally, please click "Close" to close the main dialog box.



4. Click “Project->synchronize files” to synchronize files. Select option “force all files to be re-parsed” and click “OK”.



Till now, the Source Insight project is done.

Tasks On Transplant

Entry of uboot ---start.S

Tips:

How do we know the entry of uboot?

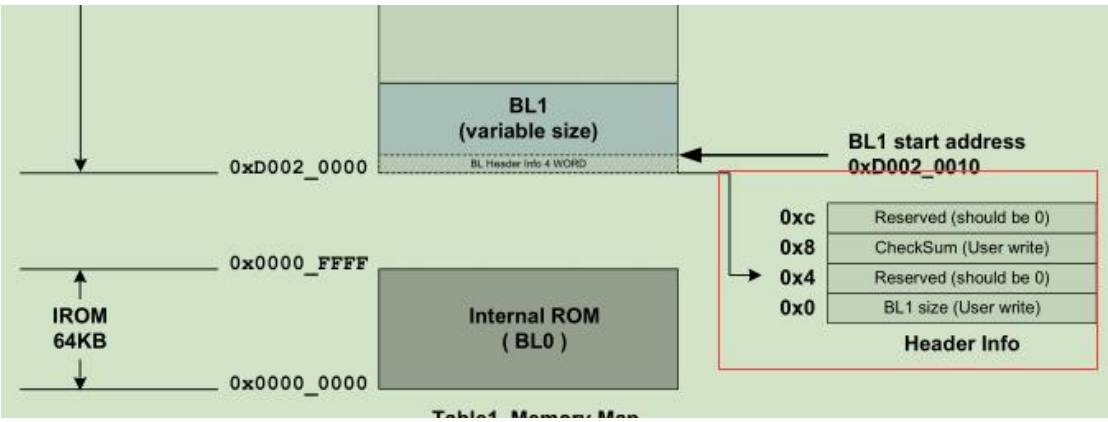
We could realize the entry via uboot_lds.S, ENTRY(_start)

and search "_start" to find the startup file.

1.Code around line49~54 in start.S

```
#if defined(CONFIG_EVT1) && !defined(CONFIG_FUSED)
    .word 0x2000
    .word 0x0
    .word 0x0
    .word 0x0
#endif
```

This part of code is to take 16 bytes space for header information of uboot, and it will compute header information and fill the empty 16 bytes space when execute script "sd_fusing.sh".



You can find the relevant descriptions at table1.Memory map(page13) and at "2.9 Header information data for Boot Code description"(page20) in file "S5PV210_iROM_ApplicationNote_Preliminary_20091126.pdf".

2. TEXT_BASE around line99 in start.S.

```
_TEXT_BASE:
    .word   TEXT_BASE
```

This variable is to specify linked address for uboot. If MMU is enabled, this address is supposed to virtual address. And we can find TEXT_BASE in Makefile which is in the top level director.

```
smdkv210single_config : unconfig
    @$(MKCONFIG) $(@:_config=) arm s5pc11x smdkc110 samsung s5pc110
    @echo "TEXT_BASE = 0xc3e00000" > $(obj)board/samsung/smdkc110/config.mk
```

Here, we should add a new configuration to support our board x210.

```
x210_config :   unconfig
    @$(MKCONFIG) $(@:_config=) arm s5pc11x x210 samsung s5pc110
    @echo "TEXT_BASE = 0xc3e00000" > $(obj)board/samsung/x210/config.mk
```

Here are some arguments configured, like arm, s5pc11x, x210, samsung, s5pc110. And these arguments are passed to mkconfig when mkconfig is called by Makefile. The effects of these arguments are easy to be understood, and you can read the mkconfig to get it.

3. "lowlevel_init"

3.1 Confirm "ELFIN_WATCHDOG_BASE"

```
/* Disable Watchdog */
ldr r0, =ELFIN_WATCHDOG_BASE      /* 0xE2700000 */
mov r1, #0
str r1, [r0]
```

We can realize the base address of WDT register is 0xE2700000 via datasheet "S5PV210_UM_REV1.1.pdf" p827.

3.4.1 REGISTER MAP

Register	Address	R/W	Description	Reset Value
WTCON	0xE270_0000	R/W	Watchdog Timer Control Register	0x00008021
WTDAT	0xE270_0004	R/W	Watchdog Timer Data Register	0x00008000
WTCNT	0xE270_0008	R/W	Watchdog Timer Count Register	0x00008000
WTCLRINT	0xE270_000C	W	Watchdog Timer Interrupt Clear Register	-

3.2.Confirm "ELFIN_GPIO_BASE"

```
ldr r0, =ELFIN_GPIO_BASE          /* 0xE0200000 */
```

We can realize the base address of GPIO register is 0xE0200000 via datasheet "S5PV210_UM_REV1.1.pdf" p32.

2.1.2 SPECIAL FUNCTION REGISTER MAP

Address		Description
0xE000_0000	0xE00F_FFFF	CHIPID
0xE010_0000	0xE01F_FFFF	SYSCON
0xE020_0000	0xE02F_FFFF	GPIO
0xE030_0000	0xE03F_FFFF	AXI_DMA
0xE040_0000	0xE04F_FFFF	AXI_PSYS
0xE050_0000	0xE05F_FFFF	AXI_PSFR
0xE060_0000	0xE06F_FFFF	TZPC2

3.3 Confirm "ELFIN_SROM_BASE"

```
ldr r0, =ELFIN_SROM_BASE      /* 0xE8000000 */
```

We can realize the base address of SROM_BANK register is 0xE8000000 via datasheet "S5PV210_UM_REV1.1.pdf" p647.

2.4.1 REGISTER MAP

Register	Address	R/W	Description	Reset Value
SROM_BW	0xE800_0000	R/W	Specifies the SROM Bus width & wait control	0x0000_0009
SROM_BC0	0xE800_0004	R/W	Specifies the SROM Bank0 control register	0x000F_0000
SROM_BC1	0xE800_0008	R/W	Specifies the SROM Bank1 control register	0x000F_0000
SROM_BC2	0xE800_000C	R/W	Specifies the SROM Bank2 control register	0x000F_0000
SROM_BC3	0xE800_0010	R/W	Specifies the SROM Bank3 control register	0x000F_0000
SROM_BC4	0xE800_0014	R/W	Specifies the SROM Bank4 control register	0x000F_0000
SROM_BC5	0xE800_0018	R/W	Specifies the SROM Bank5 control register	0x000F_0000

3.4. Deal with PS_HOLD

```
/* PS_HOLD pin(GPH0_0) set to high */
ldr r0, =(ELFIN_CLOCK_POWER_BASE + PS_HOLD_CONTROL_OFFSET)  /* 0xE010E81C */
ldr r1, [r0]
orr r1, r1, #0x300 |
orr r1, r1, #0x1
str r1, [r0]
```

Datasheet "S5PV210_UM_REV1.1.pdf", p470

4.10.5.8 MISC Register (PS_HOLD_CONTROL, R/W, Address = 0xE010_E81C)

PS_HOLD_CONTROL	Bit	Description	Initial State
Reserved	[31:12]	Reserved	0x00005
Reserved	[11:10]	Reserved	0
DIR	[9]	Direction (0: input, 1: output)	1
DATA	[8]	Driving value (0:low, 1:high)	0
Reserved	[7:1]	Reserved	0x00
PS_HOLD_OUT_EN	[0]	XEINT[0] pad is controlled by this register values and values of control registers for XEINT[0] of GPIO chapter is ignored when this field is '1'. (0: disable, 1: enable)	0

PS_HOLD (muxed with XEINT[0]) pin value is kept up in any power mode. This register is in alive region and reset

3.5 Is in Internal SRAM or DDR? If not in DDR, go to relocate.

```
ldr r0, =0xff000fff
bic r1, pc, r0      /* r0 <- current base addr of code */
ldr r2, _TEXT_BASE      /* r1 <- original base addr in ram */
bic r2, r2, r0      /* r0 <- current base addr of code */
cmp     r1, r2              /* compare r0, r1                  */
beq     1f          /* r0 == r1 then skip sdram init   */
```

The basic method is to compare the bi15~23 of current address with the bit15~23 of linked address. If they are no difference, it's in DDR.

3.6 Deal with PMIC

```
/* init PMIC chip */
bl PMIC_InitIp
```

Because there is no PMIC on our board x210, we need to mask this part

of code. Otherwise, cpu will be blocked.

3.7 system_clock_init

This part of the code is closely related to our hardware. And we can check it according to the clock diagram at p361 in datasheet "S5PV210_UM_REV1.1.pdf".

3.8 mem_ctrl_asm_init

This part of the code is closely related to our hardware. And we can check it according to the relevant content around p596~599 in datasheet "S5PV210_UM_REV1.1.pdf".

Start to transplant

Compiling, linking and executing

1. position project source code in the shared folder
2. open Makefile, modify cross-compilation toolchain.

```
CROSS_COMPILE = /usr/local/arm/gcc/arm-2009q3/bin/arm-none-linux-gnueabi-        //by Alion
```

3. Add a new configuration for x210 in Makefile

```
x210_config :   unconfig
    @$(MKCONFIG) $(@:_config=) arm s5pc11x x210 samsung s5pc110
    @echo "TEXT_BASE = 0xc3e00000" > $(obj)board/samsung/x210/config.mk
```

4. execute "make x210_config" in ubuntu.

Error: ln: failed to create symbolic link 'asm': Operation not supported.

```
root@ubuntu:/mnt/hgfs/VMshare/android_uboot_smdkv210/u-boot-samsung-dev# make x2
10_config
Configuring for x210 board...
ln: failed to create symbolic link 'asm': Operation not supported
Makefile:2587: recipe for target 'x210_config' failed
make: *** [x210_config] Error 1
```

Seek error source via adding print message, and locate the error at around line48 in mkconfig.

```
44        ln -s asm-$2 asm
45    else
46        cd ./include
47        rm -f asm
48        echo "123456"
49        ln -s asm-$2 asm
50        echo "==========="
51    fi
52
53    rm -f asm-$2/arch
54
```

Analysis: **Error said that the "ln" operation not be supported, and there indeed be an symbolic link operation. And symbolic link is not supported in shared folders.**

Solution: Copy project code to the unshared folder.

Result: Error disapper.

```
root@ubuntu:~/uboot_x210# make x210_config
Configuring for x210 board...
```

5.Execute “Make -j4 -s” to compile uboot.

uboot.bin will be created after compiling.

```
plit   mkmovi        SI_PROJECT
       nand_spl      System.map
       net           tools
       onenand_bl1   u-boot
eric   onenand_ipl   u-boot.bin
NERS   post          u-boot.dis
       README        u-boot.map
e      rules.mk      u-boot.srec
g      sd_fusing
```

6.Burn uboot.bin into SD card

cd sd_fusing, and execute “source sd_fusing.sh /dev/sdb”.

```
root@ubuntu:~/uboot_x210/sd_fusing# source sd_fusing2.sh /dev/sdb
/dev/sdb reader is identified.
make sd card partition
./sd_fdisk /dev/sdb
1+0 records in
1+0 records out
512 bytes copied, 0.0310471 s, 16.5 kB/s
mkfs.vfat -F 32 /dev/sdb1
mkfs.fat 3.0.28 (2015-05-16)
BL1 fusing
8+0 records in
8+0 records out
4096 bytes (4.1 kB, 4.0 KiB) copied, 0.0560567 s, 73.1 kB/s
u-boot fusing
16+0 records in
16+0 records out
8192 bytes (8.2 kB, 8.0 KiB) copied, 0.582171 s, 14.1 kB/s
544+0 records in
544+0 records out
278528 bytes (279 kB, 272 KiB) copied, 3.62401 s, 76.9 kB/s
U-boot image is fused successfully.
Eject SD card and insert it again.
```

7.operation result:

```
SD checksum Error
OK

U-Boot 1.3.4 (Jul  4 2018 - 21:25:50) for SMDKV210


CPU:  S5PV210@1000MHz(OK)
        APLL = 1000MHz, HclkMsys = 200MHz, PclkMsys = 100MHz
        MPLL = 667MHz, EPLL = 80MHz
                       HclkDsys = 166MHz, PclkDsys = 83MHz
                       HclkPsys = 133MHz, PclkPsys = 66MHz
                       SCLKA2M  = 200MHz
Serial = CLKUART
Board:   SMDKV210
DRAM:     1 GB
Flash:   8 MB
SD/MMC:  unrecognised EXT_CSD structure version 7
unrecognised EXT_CSD structure version 7
Card init fail!
0 MB
NAND:    0 MB
The input address don't need a virtual-to-physical translation : 23e9c008
*** Warning - using default environment

In:      serial
Out:     serial
Err:     serial
checking mode for fastboot ...
Hit any key to stop autoboot:  0

no devices available

no devices available
get_format
-------- 0 --------
Wrong Image Format for bootm command
ERROR: can't get kernel image!
SMDKV210 #
```

(1) Uart initialization is Ok, and baudrate is 115200.

(2) We need change "U-Boot 1.3.4 (Jul 4 2018 - 21:25:50) for SMDKV210" to "U-Boot 1.3.4 (Jul 4 2018 - 21:25:50) for x210".

Method: search "SMDKV210" in file /include/configs/x210.h.

You will get macro CONFIG_IDENT_STRING, and need to change it.

Old:

```
#define CONFIG_IDENT_STRING " for SMDKV210"
```

New:

```
//#define CONFIG_IDENT_STRING   " for SMDKV210"
#define CONFIG_IDENT_STRING " for x210"      /* by Alion */
```

(3) We need to change "Board: SMDKV210" to "Board: x210".

Method: Search "SMDKV210" in source code.

You will find it in x210.c.

```
int checkboard(void)
{
#ifdef CONFIG_MCP_SINGLE
#if defined(CONFIG_VOGUES)
    printf("\nBoard:   VOGUESV210\n");
#else
    printf("\nBoard:   SMDKV210\n");
#endif //CONFIG_VOGUES
#else
    printf("\nBoard:   SMDKC110\n");
#endif
    return (0);
}
```

change "printf("\nBoard: SMDKV210\n");" to "printf("\nBoard: x210\n");".

(4) "DRAM: 1 GB", the actual DRAM is 512MB on our board x210, so we need to revise it.

The problem is located in x210.h:

```
#define CONFIG_NR_DRAM_BANKS    2          /* we have 2 bank of DRAM */
#define SDRAM_BANK_SIZE         0x20000000    /* 512 MB */
#define PHYS_SDRAM_1            MEMORY_BASE_ADDRESS /* SDRAM Bank #1 */
#define PHYS_SDRAM_1_SIZE       SDRAM_BANK_SIZE
#define PHYS_SDRAM_2            (MEMORY_BASE_ADDRESS + SDRAM_BANK_SIZE) /* SDRAM Bank #2 */
#define PHYS_SDRAM_2_SIZE       SDRAM_BANK_SIZE

#define CFG_FLASH_BASE      0x80000000
```

There are two DRAMs, and each one is 256MB, so we should change the value of macro SDRAM_BANK_SIZE to 0x10000000.

We also need to change the value of macro MEMORY_BASE_ADDRESS to 0x30000000, because the DRAM on x210 is connected at 0x30000000.

(5) SD/MMC: unrecognised EXT_CSD structure version 7
unrecognised EXT_CSD structure version 7
Card init fail!

It turns out that SD/MMC initialization failed. I searched the error and located the error at function "mmc_read_ext_csd".

```
ext_csd_struct = ext_csd[EXT_CSD_REV];
if (ext_csd_struct > 5) {
    printf("unrecognised EXT_CSD structure "
        "version %d\n", ext_csd_struct);
    err = -1;
    goto out;
}
```

Because version7 is greater than 5, error occur. I change 5 to 7 so as to bypass this error.

Operation result:

```
SD checksum Error
OK

U-Boot 1.3.4 (Jul  6 2018 - 10:06:01) for x210


CPU:  S5PV210@1000MHz(OK)
        APLL = 1000MHz, HclkMsys = 200MHz, PclkMsys = 100MHz
        MPLL = 667MHz, EPLL = 80MHz
                       HclkDsys = 166MHz, PclkDsys = 83MHz
                       HclkPsys = 133MHz, PclkPsys = 66MHz
                       SCLKA2M  = 200MHz
Serial = CLKUART
Board:   x210
DRAM:    512 MB
Flash:   8 MB
SD/MMC:  3776MB
NAND:    0 MB
The input address don't need a virtual-to-physical translation : 23e9c008
In:      serial
Out:     serial
Err:     serial
checking mode for fastboot ...
Hit any key to stop autoboot:  0
ERROR: resetting DM9000 -> not responding
dm9000 not found at 0xa8000000 id: 0x2b2a2928
DM9000: Undefined IO-mode:0x3
MAC: 00:40:5c:26:0a:5b
could not establish link
TFTP from server 192.168.1.30; our IP address is 192.168.1.88
```

(6) The input address don't need a virtual-to-physical translation :23e9c008

I searched this sentence, and found the following code:

```
ulong virt_to_phy_smdkc110(ulong addr)
{
    if ((0xc0000000 <= addr) && (addr < 0xd0000000))
        return (addr - 0xc0000000 + 0x20000000);
    else
        printf("The input address don't need "\
            "a virtual-to-physical translation : %08lx\n", addr);

    return addr;
}
```

We need to map physical address 0x30000000~0x40000000 to virtual address 0xc0000000~0xd0000000.

do the modification:

A:

```
ulong virt_to_phy_smdkc110(ulong addr)
{
    if ((0xc0000000 <= addr) && (addr < 0xd0000000))
        //return (addr - 0xc0000000 + 0x20000000);
        return (addr - 0xc0000000 + 0x30000000);        /* by Alion */
    else
        printf("The input address don't need "\
            "a virtual-to-physical translation : %08lx\n", addr);

    return addr;
}
```

B:revise mapping table in lowlevel_init.S

```
00702:      .set __base,0x300
00703:      // 80MB for SDRAM with cacheable
00704:      .rept 0xD00 - 0xC00 /* by Alion */
00705:      //.rept 0xC50 - 0xC00
00706:      FL_SECTION_ENTRY __base,3,0,1,1
00707:      .set __base,__base+1
00708:      .endr
00709:
00710:      // Not Allowed
00711:      /*  by Alion
00712:      .rept 0xD00 - 0xC50
00713:      .word 0x00000000
00714:      .endr
00715:      */
00716: #endif
```

(8) This error indicates thar configuration for DM9000 is wrong。

```
ERROR: resetting DM9000 -> not responding
dm9000 not found at 0xa8000000 id: 0x2b2a2928
DM9000: Undefined IO-mode:0x3
MAC: 00:40:5c:26:0a:5b
could not establish link
```

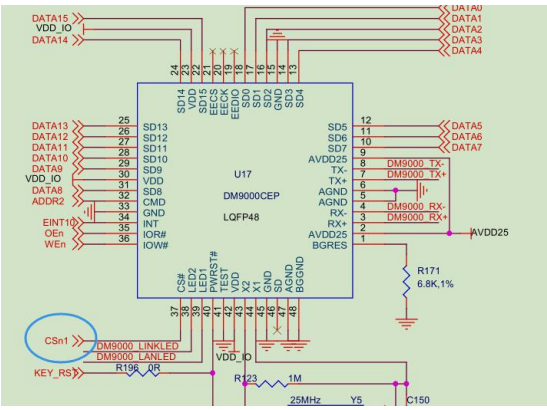
Search “dm9000 not found at”:

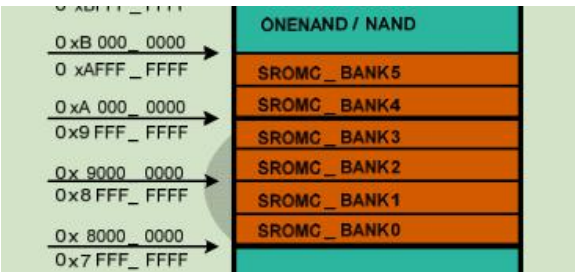
```
printf("dm9000 not found at 0x%08x id: 0x%08x\n",
        CONFIG_DM9000_BASE, id_val);
return -1;
```

Find that CONFIG_DM9000_BASE:

```
#ifdef CONFIG_DRIVER_DM9000
#define CONFIG_DM9000_BASE        (0xA8000000)
#define DM9000_IO                 (CONFIG_DM9000_BASE)
#if defined(DM9000_16BIT_DATA)
#define DM9000_DATA               (CONFIG_DM9000_BASE+2)
#else
#define DM9000_DATA               (CONFIG_DM9000_BASE+1)
```

The DM9000 on x210 is connected to SROMC_BANK1, and the base address is 0x88000000.



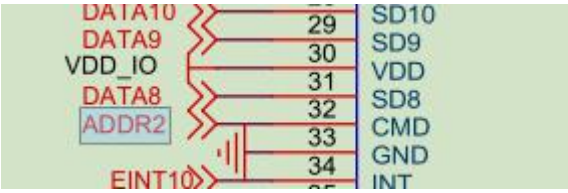


"DM9000_DATA":

Read datasheet of DM9000 to know:

			modified by EEPROM setting. See the EEPROM content description for detail.
32	CMD	I,PD	Command Type When high, the access of this command cycle is DATA port When low, the access of this command cycle is INDEX port

Read schematic to know:



"DM9000_DATA" is supposed to be "CONFIG_DM9000_BASE+4".

revise and execute:

```
checking mode for fastboot ...
Hit any key to stop autoboot:  0
dm9000 i/o: 0x88000000, id: 0x90000a46
DM9000: running in 16 bit mode
MAC: 00:40:5c:26:0a:5b
```

ping windows pc :

```
SMDKV210 # ping 192.168.1.10
dm9000 i/o: 0x88000000, id: 0x90000a46
DM9000: running in 16 bit mode
MAC: 00:40:5c:26:0a:5b
operating at 100M full duplex mode
rx fifo error
rx length too big
DM9000 error: status check fail: 0x3
```

It turns out that there still are some problems.

After some efforts, I found that I forget to revise initialization function "dm9000_pre_init".

```
static void dm9000_pre_init(void)
{
    unsigned int tmp;

#if defined(DM9000_16BIT_DATA)
    SROM_BW_REG &= ~(0xf << 20);
    SROM_BW_REG |= (0<<23) | (0<<22) | (0<<21) | (1<<20);

#else
    SROM_BW_REG &= ~(0xf << 20);
    SROM_BW_REG |= (0<<19) | (0<<18) | (0<<16);
#endif
    SROM_BC5_REG = ((0<<28)|(1<<24)|(5<<16)|(1<<12)|(4<<8)|(6<<4)|(0<<0));

    tmp = MP01CON_REG;
    tmp &=~(0xf<<20);
    tmp |=(2<<20);
    MP01CON_REG = tmp;
}
```

Revise it as follows:

```
static void dm9000_pre_init(void)
{
    unsigned int tmp;

#if defined(DM9000_16BIT_DATA)
/*
    SROM_BW_REG &= ~(0xf << 20);
    SROM_BW_REG |= (0<<23) | (0<<22) | (0<<21) | (1<<20);
*/
    SROM_BW_REG &= ~(0xf << 4);      //BANK1
    SROM_BW_REG |= (1<<7) | (1<<6) | (1<<5) | (1<<4);
```

```
#else
/*
    SROM_BW_REG &= ~(0xf << 20);
    SROM_BW_REG |= (0<<19) | (0<<18) | (0<<16);
*/
    SROM_BW_REG &= ~(0xf << 4);
    SROM_BW_REG |= (0<<7) | (0<<6) | (0<<4);

#endif
```

```
/*
    SROM_BC5_REG = ((0<<28)|(1<<24)|(5<<16)|(1<<12)|(4<<8)|(6<<4)|(0<<0));

    tmp = MP01CON_REG;
    tmp &=~(0xf<<20);
    tmp |=(2<<20);
    MP01CON_REG = tmp;
*/
    SROM_BC1_REG = ((0<<28)|(1<<24)|(5<<16)|(1<<12)|(4<<8)|(6<<4)|(0<<0));

    tmp = MP01CON_REG;
    tmp &=~(0xf<<4);
    tmp |=(2<<4);
    MP01CON_REG = tmp;

} ? end dm9000_pre_init ?
```

Testing: ping windows pc and ubuntu successfully, and download file from tftp server successfully.

```
SMDKV210 #
SMDKV210 #
SMDKV210 #
SMDKV210 #
SMDKV210 # ping 192.168.1.30
dm9000 i/o: 0x88000000, id: 0x90000a46
DM9000: running in 16 bit mode
MAC: 00:40:5c:26:0a:5b
operating at 100M full duplex mode
host 192.168.1.30 is alive
SMDKV210 #
dm9000 i/o: 0x88000000, id: 0x90000a46
DM9000: running in 16 bit mode
MAC: 00:40:5c:26:0a:5b
```

The prefix of cmdline is still SMDKV210, and we need change it to X210.

Method: Search SMDKV210 in x210.h.

```
#define CFG_PROMPT              "SMDKV210 # "    /* Monitor Command Prompt       */
```

Revise it as follows:

```
//#define CFG_PROMPT            "SMDKV210 # "    /* Monitor Command Prompt       */
#define CFG_PROMPT              "x210 # "    /* Monitor Command Prompt       */  /* by Alion */
```

complie and execute:

```
Hit any key to stop autoboot:  0
x210 #
x210 #
x210 #
x210 #
x210 #
```

Till now, transplant task is over.