SAMSUNG ELECTRONICS

Application Note
(Internal ROM Booting)

S5PV210

RISC Microprocessor

NOV 23, 2009

Preliminary REV 0.3

Preliminary product information describe products that are in development, for which full characterization data and associated errata are not yet available. Specifications and information herein are subject to change without notice.

Important Notice

The information in this publication has been carefully checked and is believed to be entirely accurate at the time of publication. Samsung assumes no responsibility, however, for possible errors or omissions, or for any consequences resulting from the use of the information contained herein.

Samsung reserves the right to make changes in its products or product specifications with the intent to improve function or design at any time and without notice and is not required to update this documentation to reflect such changes.

This publication does not convey to a purchaser of semiconductor devices described herein any license under the patent rights of Samsung or others.

Samsung makes no warranty, representation, or guarantee regarding the suitability of its products for any particular purpose, nor does Samsung assume any liability arising out of the application or use of any product or circuit and specifically disclaims any and all liability, including without limitation any consequential or incidental damages.

"Typical" parameters can and do vary in different applications. All operating parameters, including "Typicals" must be validated for each customer application by the customer's technical experts.

Samsung products are not designed, intended, or authorized for use as components in systems intended for surgical implant into the body, for other applications intended to support or sustain life, or for any other application in which the failure of the Samsung product could create a situation where personal injury or death may occur.

Should the Buyer purchase or use a Samsung product for any such unintended or unauthorized application, the Buyer shall indemnify and hold Samsung and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, expenses, and reasonable attorney fees arising out of, either directly or indirectly, any claim of personal injury or death that may be associated with such unintended or unauthorized use, even if such claim alleges that Samsung was negligent regarding the design or manufacture of said product

.

S5PV210 RISC Microprocessor
IROM(Internal ROM) Booting Application Note, Preliminary Revision 0.3

Samsung Electronics Co., Ltd.
San #24 Nongseo-Dong, Giheung-Gu
Yongin-City Gyeonggi-Do, Korea
446-711

Home Page: http://www.samsungsemi.com/

E-Mail: mobilesol.cs@samsung.com

Printed in the Republic of Korea

Revision History

Revision No	Description of Change	Refer to	Author(s)	Date
0.00	- Initial release for review	-	s.w baik	May 20, 2009
0.10	- Adding boot device option Changing boot code copy method		k.j kim	SEP 26, 2009
0.20	- Revised Figure 1, Figure 2		k.j kim	NOV 19, 2009
0.30	- Update boot flow Adding ERROR Handling		k.j kim	NOV 23, 2009

NOTE: Revised parts are written in blue.

Table of Contents

1 Overview

This chapter explains overall scheme of internal ROM(iROM) boot with memory devices such as Samsung MoviNAND/iNand, MMC/SD Card ,pure Nand, eMMC, eSSD, UART and USB boot with iROM is supported.

In S5PV210, iROM boot releases retention I/O(resets I/O) when it wakes up for recognizing Boot Device by OM pin. Refer section1.4.

1.1 H/W Feature

To use IROM boot loader, several hardware features are required.

- S5PV210X microprocessor based on CortexA8
- 64KB Internal ROM (iROM)
- 96KB Internal SRAM
- General SDRAM and Controller
- 4/8 Bit High Speed SD/MMC Controller

  4-bit SD / 4-bit MMC / 4 or 8-bit eMMC
- Nand Flash Controller
- OneNand Controller(AUDI)
- eSSD Controller
- UART/USB contorller

1.2 Feature

- OneNand Boot(Mux/Demux)
- Nand Boot (with H/W 8/16-Bit ECC)
- MMC Boot (MMC Specification 4.3 compatible including eMMC)
- eSSD Boot
- UART/USB Boot
- Secure boot mode support
  - Verify Integrity of Bootloader for all boot-up devices except for UART/USB boot.
  - To support secure boot mode, Security key value should be written in S5PV210.
    If no key is written in S5PV210, It is non-secure boot mode.
    Otherwise It's secure boot mode. To do that, Samsung should write security key in manufacture step.
- Second boot support
  - When 1st boot mode fails, SD/MMC boot will be tried through SD/MMC channel 2 with 4-bit data

1.3 Advantage of iROM booting

1. Reduce BOM Cost

=> In iROM booting with Movinand/iNAND/MMC/eMMC Card, eSSD. System can be booted without booting rom

So. There is no need other booting rom device(like nor flash)

2. Improve the Read Disturbance

=> In iROM booting with nand flash, S5PV210 can support 8/16-bit H/W ECC

All nand boot can be supported 8-bit H/W ECC.

But, 16-bit ECC is supported only one type of Nand which is 4KB 5cycle.

3. Reduce the production cost (Option)

=> You can program boot device memory using other boot device.

=> so. There is no need Gang programmer

1.4 Circuit Design Check Point

① To select iROM boot device OM pins are used. (**Refer 3 Boot configuration**)

② All of boot memory device of V210 have a SD/MMC second boot using MMC channel 2

③ OneNand boot - Xm0CSn4/NFCSn2/ONANDXL_CSn0 signal should be used for boot.
(**BL1 code should be include checksum data in the start of BL1 binary. Refer section 2.9)**

④ Nand boot - Xm0CSn2/NFCSn0 signal should be used for boot
(**BL1 code should be include checksum data in the start of BL1 binary. Refer section 2.9)**

⑤ SD/MMC/eMMC boot – MMC Channel 0 is used for first boot. And Channel 2 is used for Second boot
(**BL1 code should be include checksum data in start of BL1 binary. Refer section 2.9)**

⑥ UART boot – UART Channel port 2 is used for boot.

2 Operation

2.1 Operating Sequence

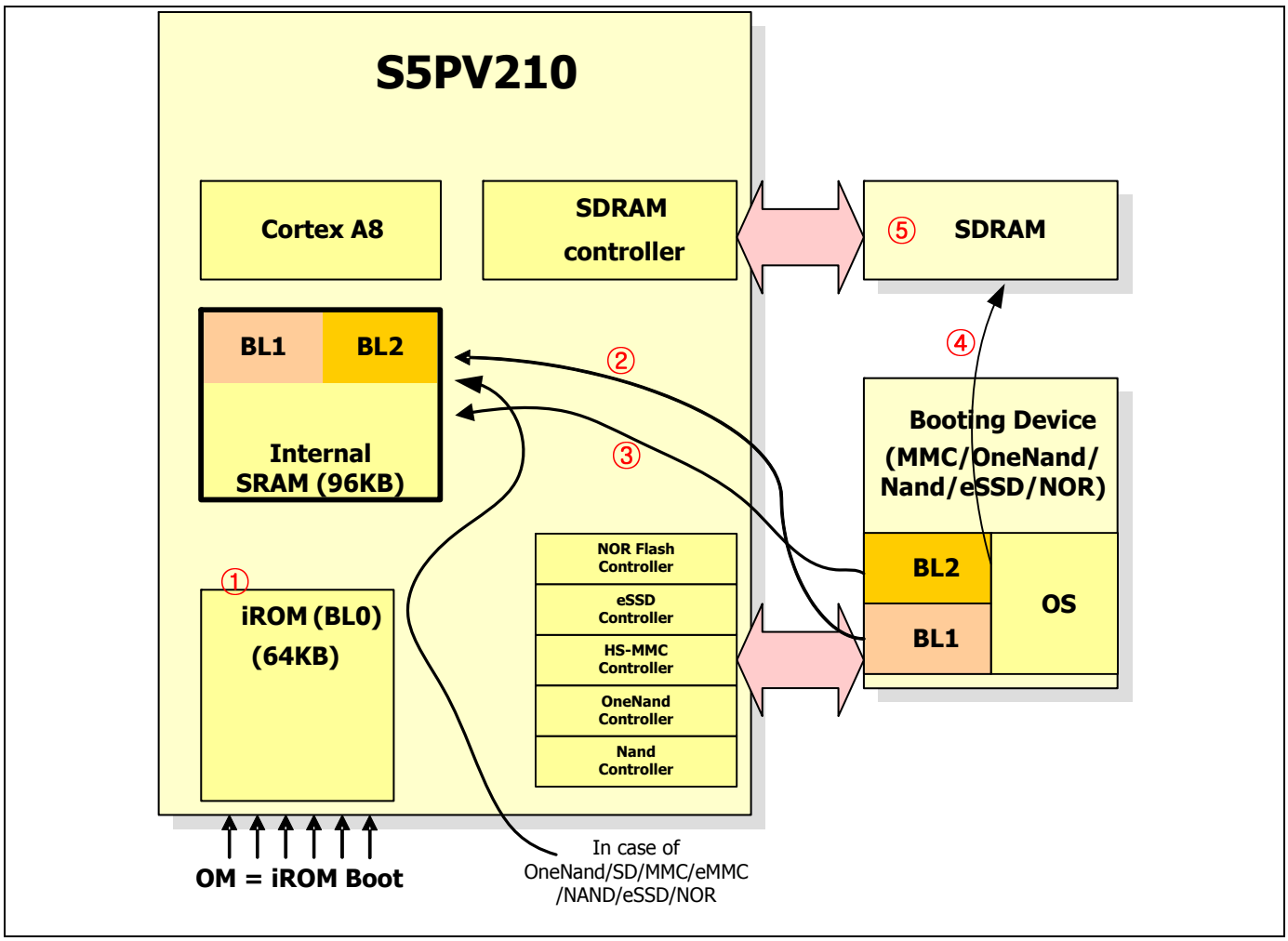


Figure 1. Overall boot-up diagram

BL1 / BL2 : It can be variable size copied from boot device to internal SRAM area.
BL1 max. size is 16KB. BL2 max. size is 80KB.

① iROM can do initial boot up : initialize system clock, device specific controller and booting device.

② iROM boot codes can load boot-loader to SRAM. The boot-loader is called BL1.
then iROM verify integrity of BL1 in case of secure boot mode.

③ BL1 will be executed: BL1 will load remained boot loader which is called BL2 on the SRAM
then BL1 verify integrity of BL2 in case of secure boot mode.

④ BL2 will be executed : BL2 initialize DRAM controller then load OS data to SDRAM.

⑤ Finally, jump to start address of OS. That will make good environment to use system.

2.2 iROM(BL0) boot-up sequence (Refer 2.3 V210 boot-up diagram)

1. Disable the Watch-Dog Timer
2. Initialize the instruction cache
3. Initialize the stack region (see “memory map” on chap 2.5)
4. Initialize the heap region. (see “memory map” on chap 2.5)
5. Initialize the Block Device Copy Function. (see “Device Copy Function” on chap 2.7)
6. Initialize the PLL and Set system clock. (see “clock configuration” on chap 2.11)
7. Copy the BL1 to the internal SRAM region (see “Device Copy Function” on chap 2.7)
8. Verify the checksum of BL1.
   If checksum fails, iROM will try the second boot up. (SD/MMC channel 2)
9. Check if it is secure-boot mode or not.
   If the security key value is written in S5PV210, It’s secure-boot mode.
   If it is secure-boot mode, verify the integrity of BL1.
10. Jump to the start address of BL1

2.3 V210 boot-up diagram

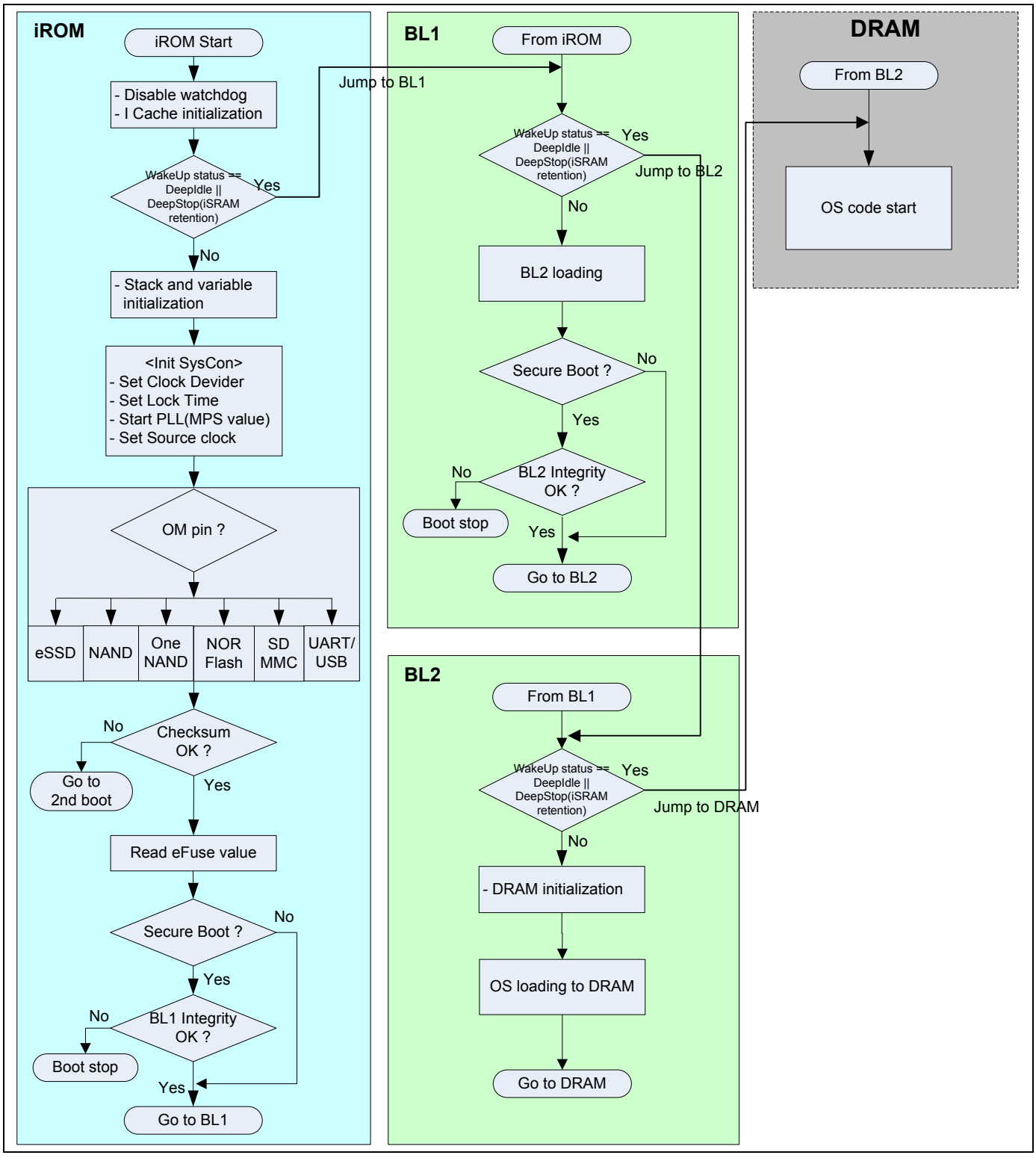
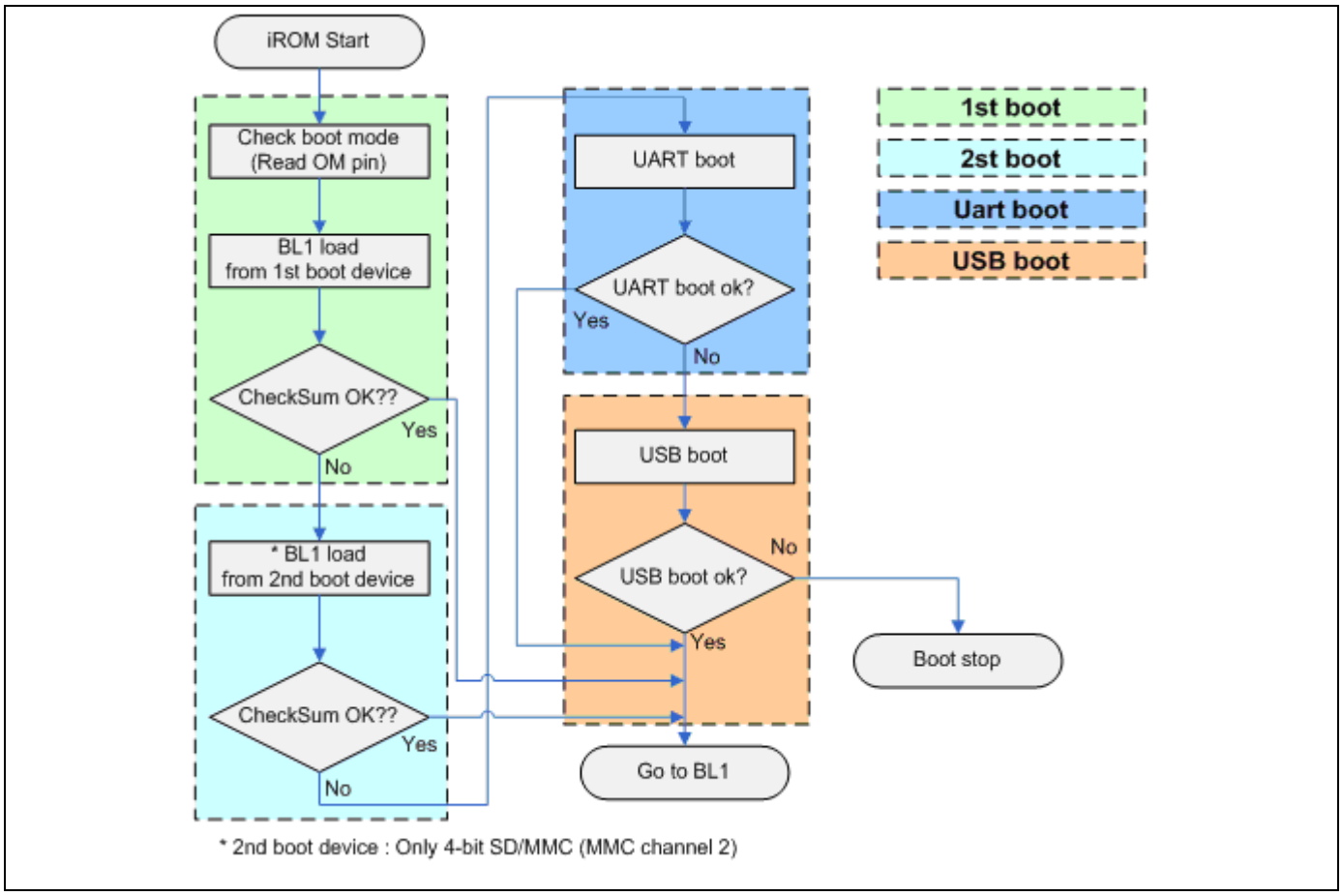


Figure 2. V210 boot-up diagram

2.4 iROM 2nd boot-up sequence when 1st boot fail



This is non-secure boot mode case.
In secure boot mode, Verification integrity of BL1 should be added behind checksum in 1st & 2nd boot step.

Figure 3. iROM BL1 copy flow

[Caution] Samsung recommend that the 2nd, UART and USB boot options should be used for only debugging.

2.4.1 UART boot mode

The S5PV210 iROM supports the UART download function. UART download is always checked by sending flag bit to DNW without regard to boot device selection. There is no selection signal for UART download. In order to avoid the UART timeout error, user should set the DNW configuration earlier than SMDK power on. Namely, when the BL1 code is selected and download process is started, the iROM should be started.

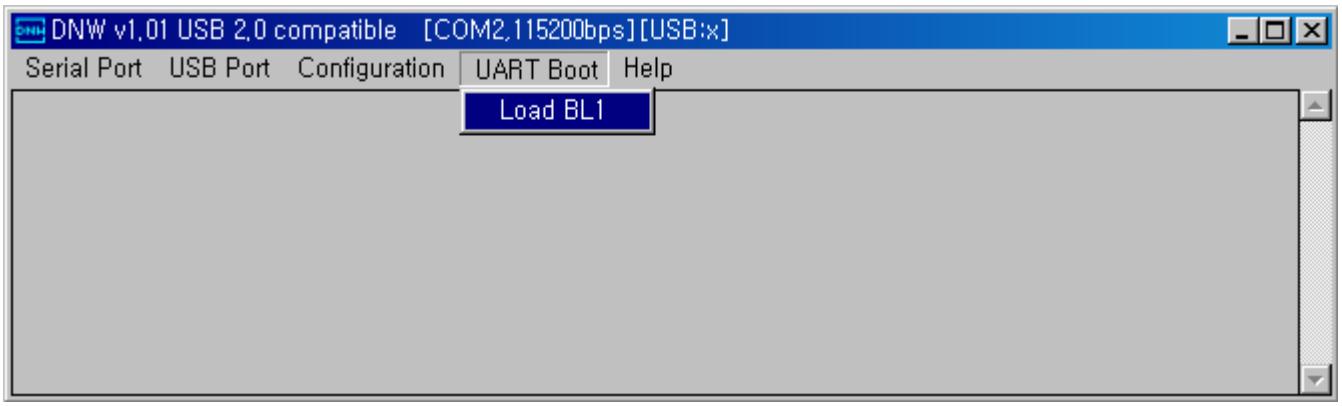


Figure 4. DNW usage for iROM uart boot

In order to set OM pins for uart boot, refer chapter 3 Boot configuration.

2.4.2 USB boot mode

The S5PV210 iROM supports the USB download function. If UART time-out occur, iROM try to USB boot mode. In order to avoid the USB negotiation time-out error, user should connect usb-cable between target board and PC in advance. If USB connection is OK, user can download BL1 image through USB like figure 5 – 7.

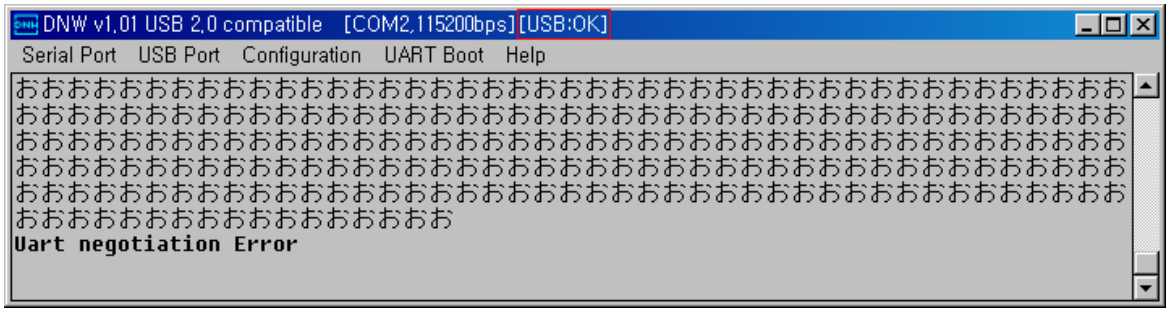


Figure 5. USB connection success

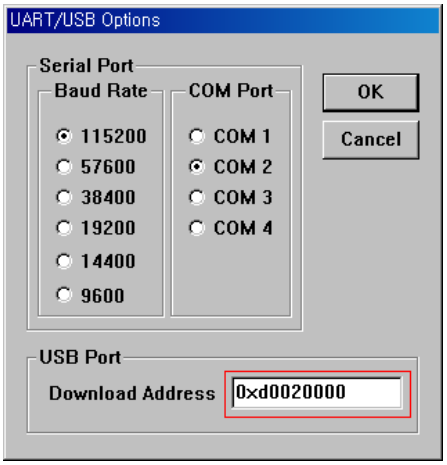


Figure 6. USB download address setting

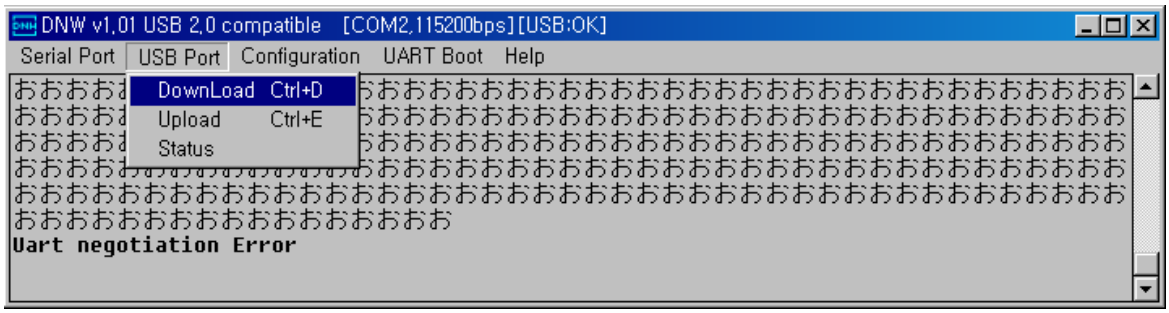


Figure 7. Selection menu USB download

As BL1 doesn't need header information through UART/USB boot mode, BL1's code base address is 0xd002_0000. In other cases except UART/USB boot mode, BL1 should have header information and It's code base address is 0xd0020010. (refer chapter 2.9)

2.5 Memory Map

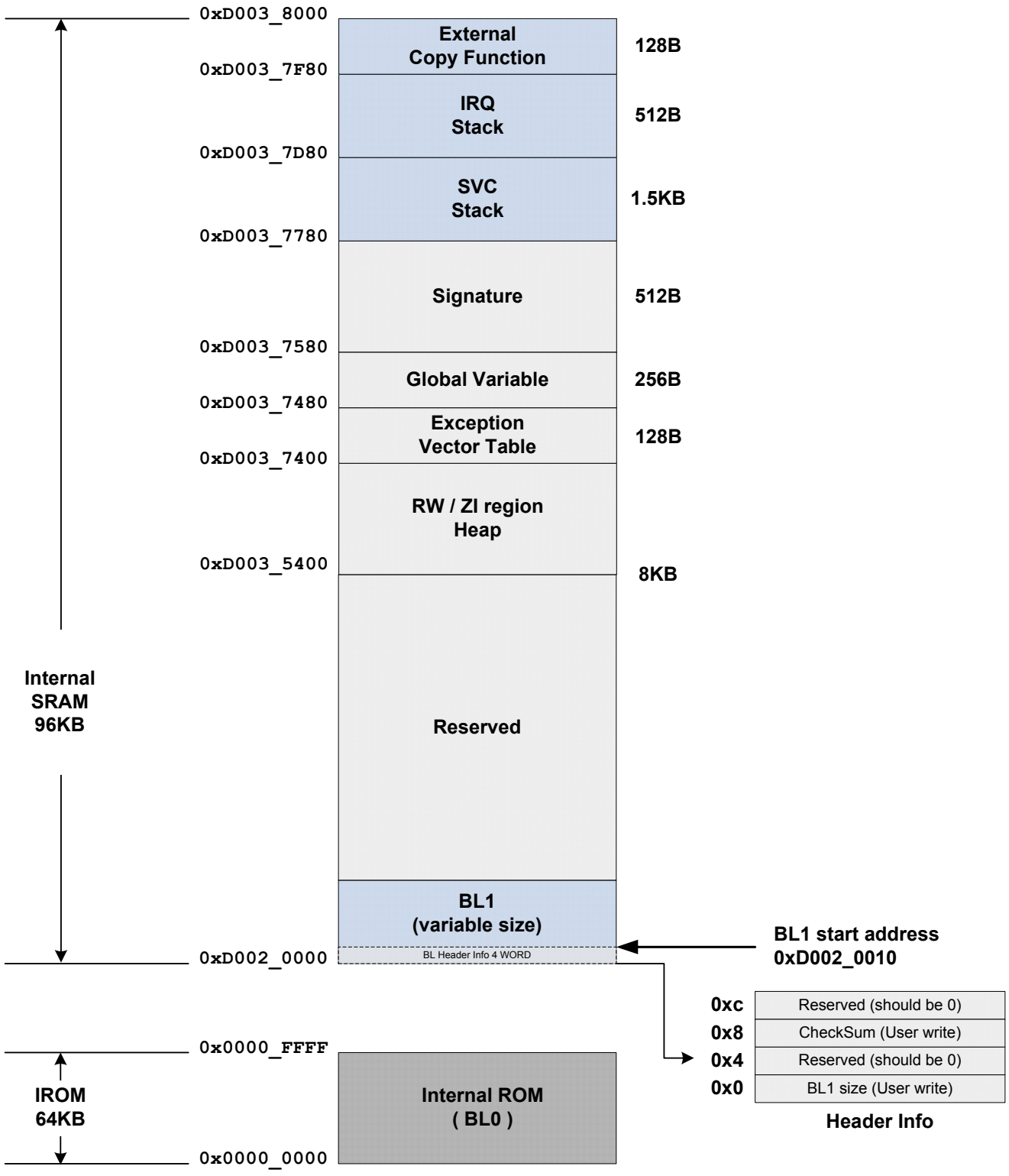


Table1. Memory Map

2.6 Global Variable

If the MMC device is used to boot up, the information of MMC card must be saved in the special area. Refer to table 2 and Figure 3.

Address	Name	Usage
0xD0037480	globalBlockSize	Total block count of the SD/MMC device.
0xD0037484	globalSDHCInfoBit	globalSDHCInfoBit[31:16] : RCA(Relative Card Address) Data globalSDHCInfoBit[2] : SD Card globalSDHCInfoBit[1] : MMC Card globalSDHCInfoBit[0] : High Capacity Enable
0xD0037488	V210_SDMMC_BASE	Current boot channel.

Table2. Special global variable for MMC & Nand boot mode.

2.7 Device Copy Function

The S5PV210 internally has a ROM code of block copy function for boot-u device. Therefore, developer may not needs to implements device copy functions. These internal functions can copy any data from memory devices to SDRAM. User can use these function after ending up the internal ROM boot process completely.

Address	Name	Usage
0xD0037F90	NF8_ReadPage_Adv	This internal function is advanced NF8_ReadPage function. (8-Bit ECC Check) Note. 2048, 4096 Page 8bits-bus Nand Only.
0xD0037F94	NF16_ReadPage_Adv	This internal function is advanced NF16_ReadPage function. (4-Bit ECC Check) Note. 2048 page size, 5 cycle address, 16bits-bus Nand Only.
0xD0037F98	CopySDMMCtoMem	This internal function can copy any data from SD/MMC device to SDRAM. User can use this function after the IROM boot process completely.
0xD0037F9C	CopyMMC4_3toMem	This internal function can copy any data from eMMC device to SDRAM. User can use this function after the IROM boot process completely.
0xD0037FA0	CopyOND_ReadMultiPages	This internal function can copy any data from OneNand device to SDRAM. User can use this function after the IROM boot process completely. (normal speed copy)
0xD0037FA4	CopyOND_ReadMultiPages_Adv	This internal function can copy any data from OneNand device to SDRAM. User can use this function after the IROM boot process completely. (fast speed copy)

0xD0037FA8	Copy_eSSDtoMem	This internal function can copy any data from eSSD device to SDRAM. User can use this function after the IROM boot process completely. (CPUPIO mode)
0xD0037FAC	Copy_eSSDtoMem_Adv	This internal function can copy any data from eSSD device to SDRAM. User can use this function after the IROM boot process completely. (UDMA mode)
0xD0037FB0	NF8_ReadPage_Adv128p	This internal function is advanced NF8_ReadPage function. (8-Bit ECC Check) Note. This is dedicated for only NAND which has 128 page per block with 2K page size.

Table3. Device Copy Function Pointer

- **Advanced Flash Copy Function Address (8-Bit ECC Check)**

```
/**
 * This Function copies a block of page to destination memory.( 8-Bit ECC only )
 * @param uint32 block : Source block address number to copy.
 * @param uint32 page : Source page address number to copy.
 * @param uint8 *buffer : Target Buffer pointer.
 * @return int32 - Success or failure.
 */
#define NF8_ReadPage_Adv (a,b,c) (((int(*)(uint32, uint32, uint8*))(*((uint32 *) 0xD0037F90)))(a,b,c))
```

Figure 8. Definition Nand Flash Block Copy Function for 8-Bit-ECC

- **Advanced Nand Flash Copy Function Address (4-Bit ECC Check)**

```
/**
 * This Function copies a block of page to destination memory( 4-Bit ECC only )
 * @param u32 block : Source block address number to copy.
 * @param u32 page : Source page address number to copy.
 * @param u8 *buffer : Target Buffer pointer.
 * @return int  - Success or failure.
 */
#define NF16_ReadPage_Adv(a,b,c) (((int(*)(uint32, uint32, uint8*))(*((uint32 *) 0xD0037F94)))(a,b,c))
```

Figure 9. Definition Nand Flash Block Copy Function for 8-Bit-ECC

- **OneNand Flash Copy Function Address**

```
/**
 * This Function copies a block of page to destination memory
 * @param uDataAddr – Destination Address
 * @param uBlockAddr - Block Address to read
 * @param uPageAddr - Page Address to read
 * @param uNumOfPages – Page Numbers  to read
 * @return bool(unsigend char) - Success or failure. */

#define CopyOND_ReadMultiPages(a,b,c,d,e) (((bool(*)(unsigned int, unsigned int, unsigned char, unsigned int, unsigned int))(*((unsigned int *) 0xD0037FA0)))(a,b,c,d,e))
```

- **OneNand Flash Copy Function Address**

```
/**
 * This Function copies a block of page to destination memory
 * @param uDataAddr – Destination Address
 * @param uBlockAddr - Block Address to read
 * @param uPageAddr - Page Address to read
 * @param uNumOfPages – Page Numbers  to read
 * @return bool(unsigend char) - Success or failure. */

#define CopyOND_ReadMultiPages_Adv (a,b,c,d,e) (((bool(*)(unsigned int, unsigned int, unsigned char, unsigned int, unsigned int))(*((unsigned int *) 0xD0037FA4)))(a,b,c,d,e))
```

Figure 10. Definition OneNAND Page Copy Function

- **SD/MMC Copy Function Address**

External source clock parameter is used to fit EPLL source clock at 20MHz.

```
/**
 * This Function copy MMC(MoviNAND/iNand) Card Data to memory.
 * Always use EPLL source clock.
 * This function works at 20Mhz.
 * @param u32 StartBlkAddress : Source card(MoviNAND/iNand MMC)) Address.(It must block address.)
 * @param u16 blockSize : Number of blocks to copy.
 * @param u32* memoryPtr : Buffer to copy from.
 * @param bool with_init : determined card initialization.
 * @return bool(u8) - Success or failure.
 */
#define CopySDMMCtoMem(z,a,b,c,e)(((bool(*)(int, unsigned int, unsigned short, unsigned int*, bool))(*((unsigned int *)0xD0037F98)))(z,a,b,c,e))
```

Figure 11. Definition SD/MMC Block Copy Function

- **eMMC Copy Function Address**

External source clock parameter is used to fit EPLL source clock at 20MHz.

```
/**
 * This Function copy MMC(MoviNAND/iNand) Card Data to memory.
 * Always use EPLL source clock.
 * This function works at 20Mhz.
 * @param u16 blockSize : Number of blocks to copy.
 * @param u32* memoryPtr : Buffer to copy from.
 * @param u32 busWidth : Data bus width.
 * @return bool(u8) - Success or failure.
 */
#define CopyMMC4_3toMem(a,b,c,d)(((bool(*)(bool, unsigned int, unsigned int*, int))(*((unsigned int *)0xD0037F9C)))(a,b,c,d))
```

Figure 12. Definition eMMC Block Copy Function

- **eSSD Copy Function Address (PIOCPU mode transfer)**

```
/**
 * This Function copy eSSD Data to memory.
 * PIOCPU mode transfer.
 * @param u32 uStBlock : Number of blocks to copy.
 * @param u32 uBlocks : Buffer to copy from.
 * @param u32 uBufAddr : Data bus width.
 * @return
 */
#define Copy_eSSDtoMem(a,b,c) (((void(*)(unsigned int, unsigned int, unsigned int))(*((unsigned int *)0xD0037FA8)))(a,b,c))
```

- **eSSD Copy Function Address (UDMA mode transfer)**

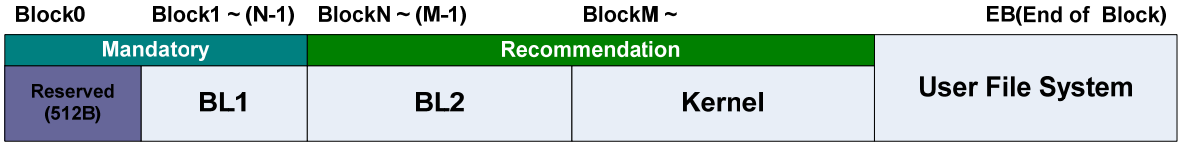
```
/**
 * This Function copy eSSD Data to memory.
 * UDMA mode transfer.
 * @param u32 uStBlock : Number of blocks to copy.
 * @param u32 uBlocks : Buffer to copy from.
 * @param u32 uBufAddr : Data bus width.
 * @return
 */
#define Copy_eSSDtoMem_Adv(a,b,c) (((void(*)(unsigned int, unsigned int, unsigned int))(*((unsigned int *)0xD0037FAC)))(a,b,c))
```

Figure 13. Definition eSSD Block Copy Function

2.8 Boot Block Assignment Guide

2.8.1 SD/MMC/eSSD Device Boot Block Assignment

<1Block=512B> (SD/MMC/eSSD)

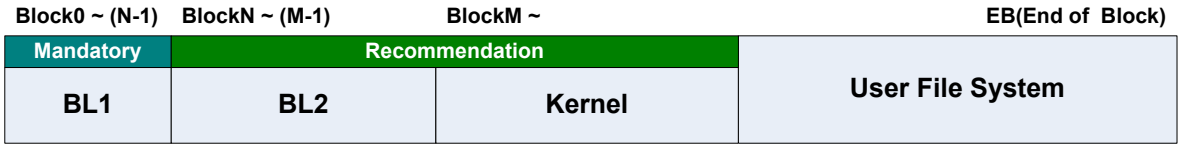


This guide is a sample but there is 1 mandatory rule.

- The first one block shouldn't be used. (Reserved)

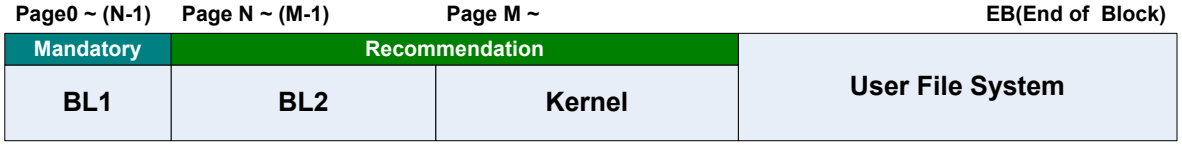
2.8.2 eMMC Device Boot Block Assignment

<1Block=512B> (eMMC)

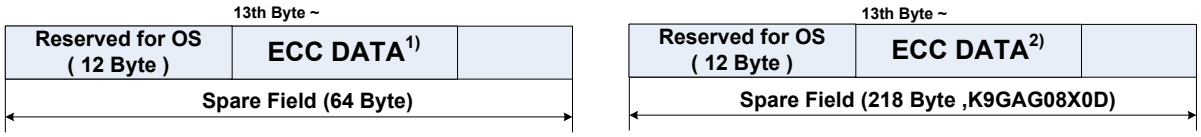


2.8.3 OneNAND/NAND Device Boot Block Assignment

(OneNAND/NAND)



[Caution] In case of NAND booting, NAND ECC Data should be located like this.



Note 1) ECC DATA size is 13Byte for 8 bit ECC.
Note 2) ECC DATA size is 26Byte for 16 bit ECC.
In case of 16bit ECC,
There are variable spare ECC size for each NAND Flash.
So need to check datasheet of NAND Flash.

2.9 Header information data for Boot Code description

The BL1 must have header data. The header data is used for being copied to internal SRAM by iROM code.

The header data has two information. One is size of BL1 and Another is checksum data of BL1.

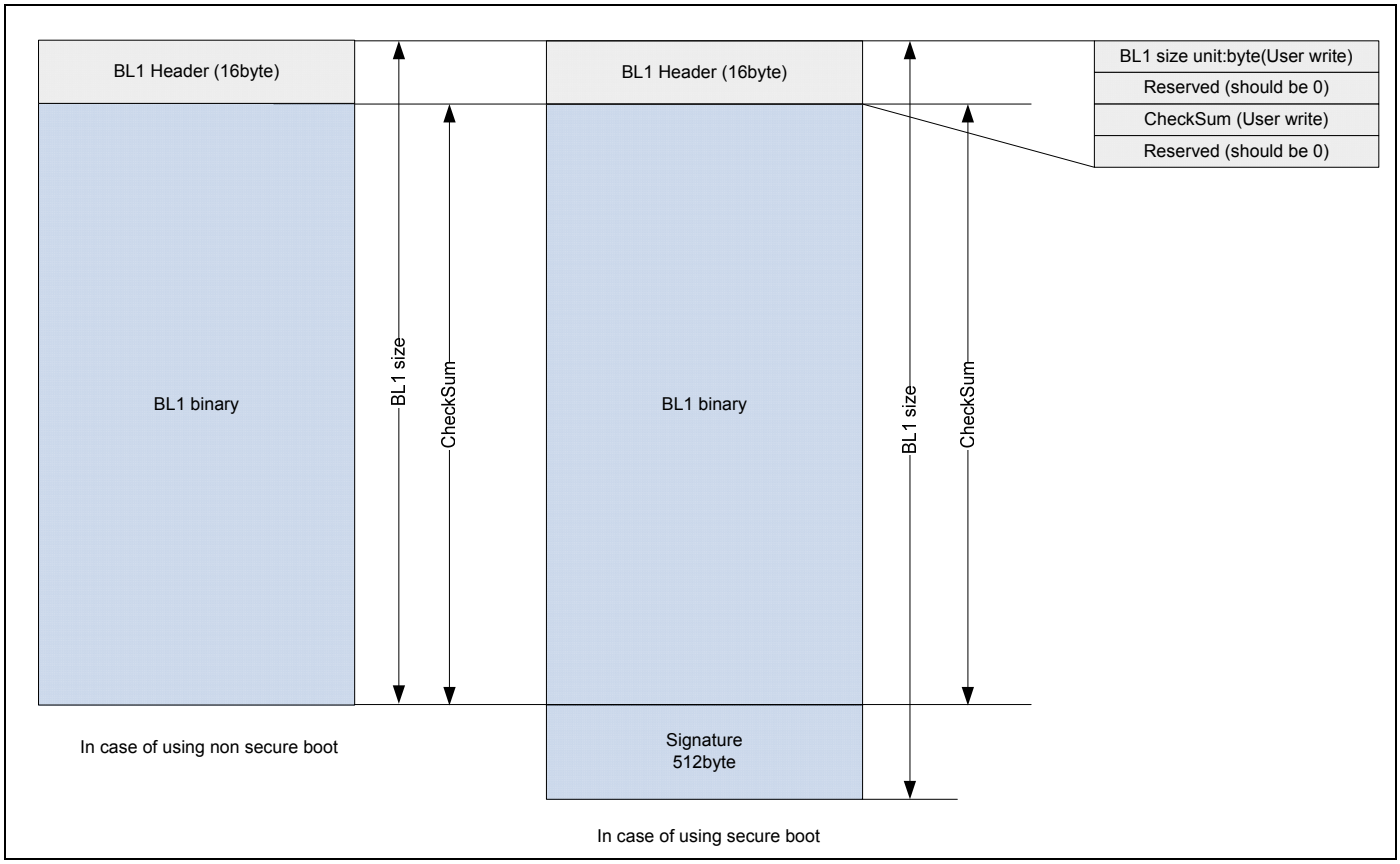


Figure 14. BL1 Structure

When loading BL1, iROM check size of BL1 in header data and copy BL1 to internal SRAM.

After coping BL1, iROM sum data of copied BL1 and compare it to checksum data in header data of BL1.

If it is success, BL1 start. otherwise iROM will try second boot(4-bit SD/MMC) from SDMMC channel 2 port.

2.10 Making checksum example code

```
for(count=0;count< dataLength;count+=1)
{
        buffer = (*(volatile u8*)(uBlAddr+count));
        checkSum = checkSum + buffer;
}
```

- **count** variable is unsigned int type.
- **dataLength** variable is unsigned int type. It contains size(byte) of BL1.
- **buffer** variable is unsigned short type. It is used reading 1 byte data from BL1.
- **checksum** variable is unsigned int type. It contains sum of BL1.

2.11 Clock Configuration

Only 24MHz external crystal/oscillator is available in V210. Refer to below.

- APLL : M:200, P:6, S:1 => 800MHz

ARMCLK	ACLK200	HCLK200	PCLK100	HCLK100
400	133	133	66	66

Note. APLL configuration

$F_{OUT} = (MDIV \times F_{IN}) / (PDIV \times 2^{SDIV-1})$

- MPLL : M:667, P:12, S:1 => 667MHz

HCLK166	PCLK83	SCLK_FIMC	ARMATCLK	HCLK133	PCLK66
133	66	133	133	133	66

Note. MPLL configuration

$F_{OUT} = (MDIV \times F_{IN}) / (PDIV \times 2^{SDIV})$

- EPLL : M:80, P:3, S:3, K:0 => 80MHz

Note. EPLL configuration

$F_{OUT} = ((MDIV+KDIV) \times F_{IN}) / (PDIV \times 2^{SDIV})$

3 Boot configuration

OM[5:0] pin should be tied with VDDSYS or GND, directly. It is aimed for minimize leakage current when entering the sleep mode. But if you have to get an option, you should add a pull-up and pull-down resistor with 100K ohms over.

OM[5]	OM[4]	OM[3]	OM[2]	OM[1]	OM[0]	OM[5]	OM[4]	OM[3] OM[2] OM[1]	OM[0]
1'b0	1'b0	1'b0	1'b0	1'b0	1'b0	Boot Mode	I-ROM	**eSSD**	X-TAL
					1'b1				X-TAL(USB)
				1'b1	1'b0			Nand 2KB, 5cycle (Nand 8bit ECC)	X-TAL
					1'b1				X-TAL(USB)
			1'b1	1'b0	1'b0			Nand 4KB, 5cycle (Nand 8bit ECC)	X-TAL
					1'b1				X-TAL(USB)
				1'b1	1'b0			Nand 4KB, 5cycle (Nand 16bit ECC)	X-TAL
					1'b1				X-TAL(USB)
		1'b1	1'b0	1'b0	1'b0			OnenandMux(Audi)	X-TAL
					1'b1				X-TAL(USB)
				1'b1	1'b0			OnenandDemux(Audi)	X-TAL
					1'b1				X-TAL(USB)
			1'b1	1'b0	1'b0			SD/MMC	X-TAL
					1'b1				X-TAL(USB)
				1'b1	1'b0			eMMC(4-bit)	X-TAL
					1'b1				X-TAL(USB)
	1'b1	1'b0	1'b0	1'b1	1'b0			**Nand 2KB, 5cycle (16-bit bus, 4-bit ECC)**	X-TAL
					1'b1				X-TAL(USB)
				1'b1	1'b0			Nand 2KB, 4cycle (Nand 8bit ECC)	X-TAL
					1'b1				X-TAL(USB)
			1'b1	1'b0	1'b0			iROM NOR boot	X-TAL
					1'b1				X-TAL(USB)
				1'b1	1'b0			eMMC(8-bit)	X-TAL
					1'b1				X-TAL(USB)
1'b1	1'b0	1'b0	1'b0	1'b0	1'b0		I-ROM **First boot UART ->USB**	**eSSD**	X-TAL
					1'b1				X-TAL(USB)
				1'b1	1'b0			Nand 2KB, 5cycle	X-TAL
					1'b1				X-TAL(USB)
			1'b1	1'b0	1'b0			Nand 4KB, 5cycle	X-TAL
					1'b1				X-TAL(USB)
				1'b1	1'b0			Nand 16bit ECC (Nand 4KB, 5cycle)	X-TAL
					1'b1				X-TAL(USB)
		1'b1	1'b0	1'b0	1'b0			OnenandMux(Audi)	X-TAL
					1'b1				X-TAL(USB)
				1'b1	1'b0			OnenandDemux(Audi)	X-TAL
					1'b1				X-TAL(USB)
			1'b1	1'b0	1'b0			SD/MMC	X-TAL
					1'b1				X-TAL(USB)
				1'b1	1'b0			eMMC(4-bit)	X-TAL
					1'b1				X-TAL(USB)

Table4. iROM OM pin description

4 Feature of the IROM Boot mode

1. OneNAND boot:

- Xm0CSn4/NFCSn2/ONANDXL_CSn0 signal should be used for boot.
- iROM code check a checksum data.
(BL1 code must include a checksum data. If BL1 code is loaded into internal SRAM. iROM code will compare a checksum data made by iROM with loaded one. Loading address is 0xD0020008.)
- OneNand clock is 33MHz using MPLL(667MHz) for boot

2. NAND boot:

- Using H/W 8bit ECC at boot page
- V210 supports 16bit ECC in case of 4KB, 5cycle Nand type.
- V210 supports 4bit ECC in case of 2KB, 5cycle 16-bit bus Nand type.
- iROM code check a checksum data.
- Xm0CSn2/NFCSn0 signal should be used for boot

3. SD/MMC and eMMC boot:

- First boot is using SDMMC CH0 as 4bit.
- Second boot is using SDMMC CH2 as 4bit.
- iROM code check a checksum data.
- SD/MMC clock is 20MHz using EPLL for boot
(BL1 code must include a checksum data. If BL1 code is loaded into internal SRAM. iROM code will compare a checksum data made by iROM with loaded one. Loading address is 0xD0020008.)

4. eMMC boot

- SDMMC CH0 can be used for eMMC boot(4/8 bit). Bus width is controlled by OM setting
- iROM code check a checksum data.
(BL1 code must include a checksum data. If BL1 code is loaded into internal SRAM. iROM code will compare a checksum data made by iROM with loaded one. Loading address is 0xD0020008.)
- SD/MMC clock is 20MHz using EPLL for boot

5. NOR boot

- Xm0CSn0 signal should be used for boot
- iROM code check a checksum data.
(BL1 code must include a checksum data. If BL1 code is loaded into internal SRAM. iROM code will compare a checksum data made by iROM with loaded one. Loading address is 0xD0020008.)

6. UART/USB boot :

- UART CH2 is used for UART boot and Debug message
- UART block frequency is 66.7MHz using MPLL for boot
- USB boot mode needs DNW tool.

Note) OM[4:0] signal don't need a pull-up/down register. But OM[5] signal needs a pull-down register. This register intends to change a boot mode between Normal storage and UART/USB boot.

5 ERROR HANDLING

When ERROR occurs during iROM boot, XPWMTOUT0 pin will be toggled to indicate what kind of error it is.

XPWMTOUT0 pin toggled with specific duty for each error case.

[Example : OneNAND checksum fail]

If checksum failure occurs during OneNAND boot mode, XPWMTOUT pin toggles with 40:60(High:Low) duty.

Figure 15. OneNAND checksum failure waveform

All error case have signals indicating error (refer table 5 iROM OM pin description)

Number	ERROR	SIGNAL OUTPUT DUTY (High : Low)
1	NOR boot checksum error	5 : 95
2	UART boot time out error	10 : 90
3	UART boot download data length over error	15 : 85
4	UART boot download data checksum error	20 : 80
5	NAND boot RnB No response error	25 : 75
6	NAND boot ECC error	30 : 70
7	NAND boot checksum error	35 : 65
8	OneNAND boot checksum error	40 : 60
9	No BL1 data in boot device	45 : 55
10	SDMMC boot card init error	50 : 50
11	SDMMC boot checksum error	60 : 40
12	Secure boot fail error	70 : 30
13	eSSD boot device init error	75 : 25
14	USB boot time out error	80 : 20
15	USB boot no input clock error	85 : 15
16	USB boot non-device mode error	90 : 10
17	eSSD boot checksum error	95 : 5

Table5. ERROR Signal Description

6 Hardware Guide

6.1 Power connection reference of eMMC booting

eMMC can support MMC 4.3 spec. Different feature between MMC 4.3 and MMC 4.2 is boot mode function.

To support eMMC boot mode, refer to figure-12.

(If you have any question about eMMC boot mode, refer MMC 4.3 spec document.)

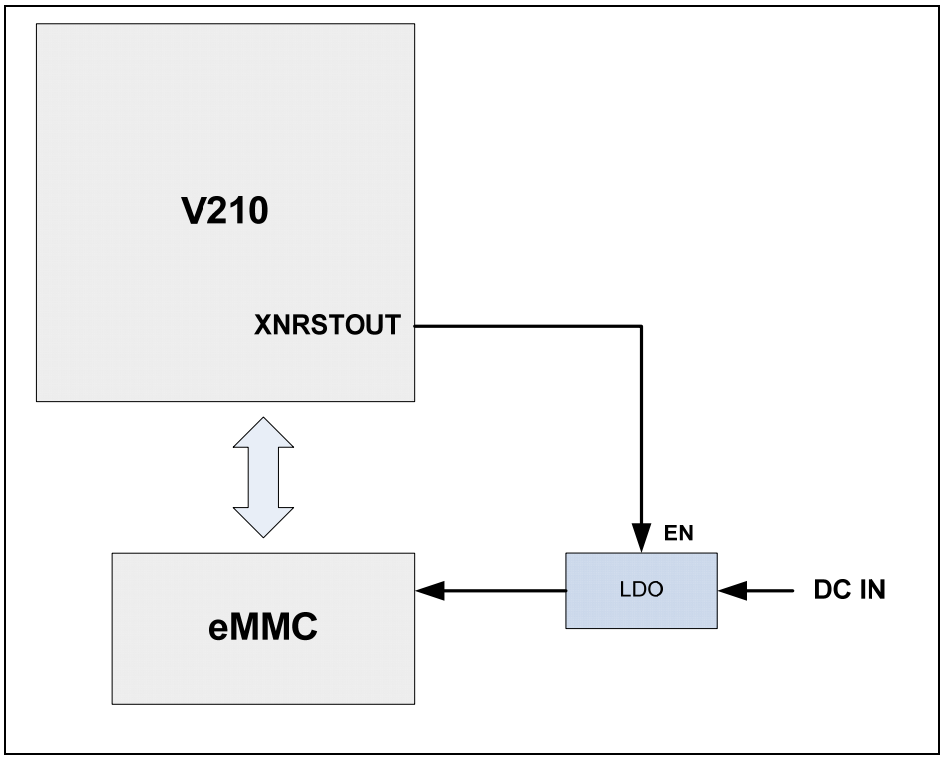


Figure 16. eMMC power connection reference