WinCE 5.0 MLC Solution (PocketMory) Porting Guide

WinCE 5.0 MLC NAND Solution (PocketMory) Porting Guide

Contact Email : mobilesol.cs@samsung.com

Revision History

Date	Version	Author	Amendment
2008.03.31	0.1	WinCE	Master Copy

Contents

Tables

Figures

1. Overview

1.1 What is MLC NAND Flash?

The MLC NAND Flash has some weakness point compared with SLC NAND.

A. The NOP number is only 1.
   i. The NOP means number of partial program times in one sector.
   ii. It means driver have to write down 1 page size at one times. And it can not be updated any more without erase.
B. The P/E cycle is only 5K or 10K. It depends on MLC NAND device specification.
   i. Therefore the NAND flash device driver has to support strong wear-leveling algorithm.
C. The write performance is slower than SLC NAND flash.
   i. Therefore, it needs write performance enhanced algorithm.
D. The 4bit ECC error can be occurred by 512bytes.
   i. Therefore, nand controller have to support hardware 4bit-ECC logic.
E. Initial bad block detection :
   i. SLC : Samsung makes sure that either the 1st or 2nd page of every initial invalid block has non-FFh data at the column address of 2,048.
   ii. MLC : Samsung makes sure that the last page of every initial invalid block has non-FFh data at the column address of 2,048.

1.2 What is PocketMory™ ?

The PocketMory™ code is based on Whimory solution. We develop it to work on WindowsCE 5.0. The PocketMory has Whimory core part.

1.3 What is Whimory™ ?

Whimory is Samsung Electronics' flash management software. Whimory has same functionalities like other well-know FTL (Flash Translation Layer).

1.4 PocketMory System Architecture

PocketMory exists between the file system and NAND flash memory.

Figure 1-1 shows the system architecture of PocketMory.

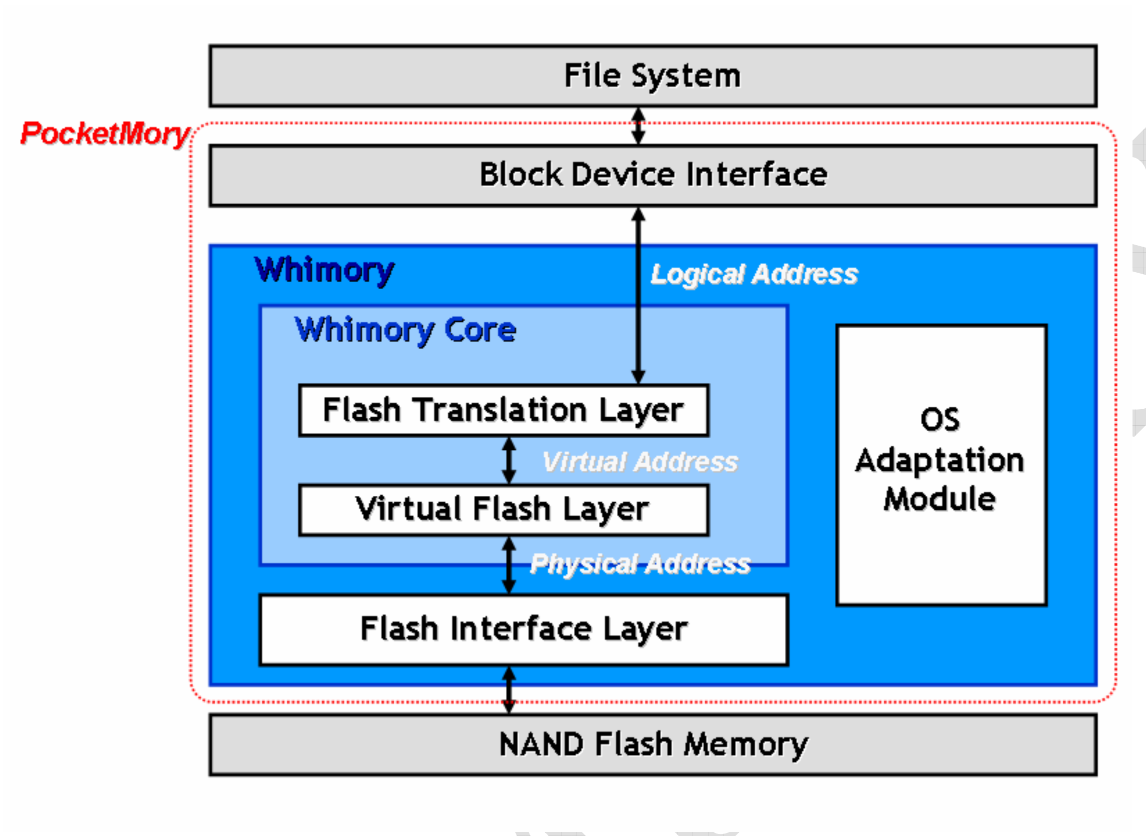


Figure 1-1. PocketMory System Architecture

 Whimory core: Whimory core is composed of two layers: FTL (Flash Translation Layer) at the top and VFL (Virtual Flash Layer) at the bottom. The layers have different features, but they perform the basic functionalities of Whimory as block device emulation and flash memory management. The main features of each layer are as follows.

• FTL: translates a logical address from the file system into the virtual flash address.

• VFL: translates the virtual address from the upper layer into the physical address. At this time, VFL does the address translation considering bad blocks and the number of NAND device in use. VFL accesses FIL, which actually performs read, write, or erase operation, with the physical address.

 OAM: OAM connects Whimory with the OS. OAM needs to be configured according to your OS environment to use NAND flash memory.

 FIL: There is a low level device driver between VFL and NAND flash memory. It reads, writes, or erases data on the physical sector address received from Whimory and is controlled by VFL.

1.5 Whimory Features

Following are the main benefits and features of Whimory implementation.

 It emulates a full block device and manages the data on NAND flash memory efficiently.

 It can be embedded in any kind of product using NAND flash memory, independent of OSs or platform types.

 It guarantees data integrity by managing a bad block and performing error detection or correction.

 It reliably works and reduces the data loss in case of sudden power failure with the advanced algorithms.

 Whimory uses smaller memory size than other FTL software. It can be used in embedded application that has small memory size.

1.6 Definitions and Acronyms

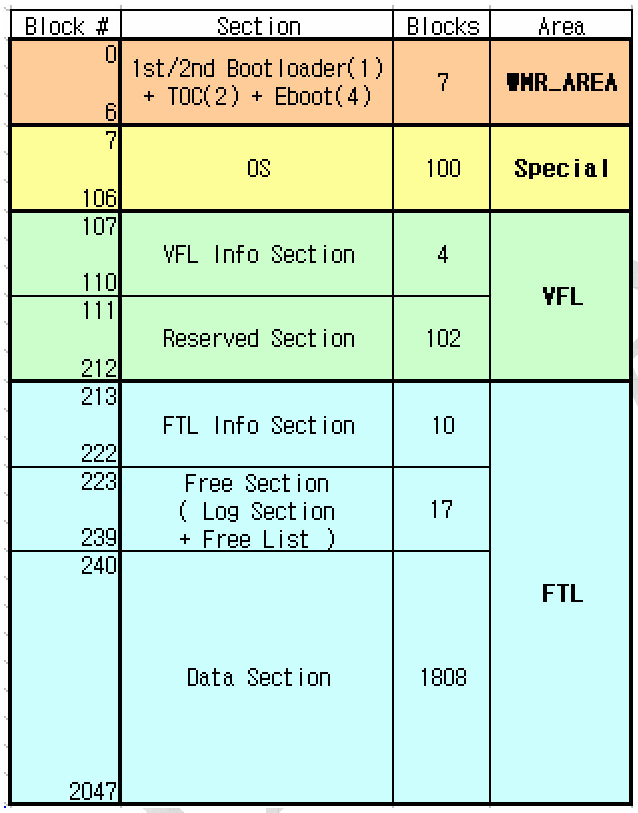
Block	NAND flash memory is partitioned into fixed-sized blocks. A block is 16K bytes or 128K bytes.
VFL	Virtual Flash Layer
FTL (Flash Translation Layer)	A software module which maps between logical addresses and physical addresses when accessing flash memory. FTL is One of the main layer in Whimory
Initial bad block	Invalid blocks upon arrival from the manufacturers
FIL	Flash Interface Layer
NAND flash controller	NAND flash controller is a controller for NAND flash memory
NAND flash device	NAND flash device is a device that contains NAND flash memory or NAND flash controller.
Page	NAND flash memory is partitioned into fixed-sized pages. A page is (512+16) bytes, (2048+64) bytes or (4096+128) bytes.
Run-time bad block	Additional invalid blocks may occur during the life of NAND flash usage
Sector	The file system performs read/write operations in a 512-byte unit called sector.
SLC	Single Level Cell
MLC	Multi Level Cell (Currently there is 4 level in one cell)
Wear-Leveling algorithm	Wear-leveling algorithm distributes the memory allocation in an evenly manner so as to increase the lifetime of NAND flash memory.
LSB	Least Significant Bit (First programmed data in one cell)
MSB	Most Significant Bit (Second programmed data in one cell)
BBM	Bad Block Mapping Manager

Table 1-1 Definitions and Acronyms

2. PocketMory block mapping.

2.1. Sample Block Mapping.

The Figure 2-1 shows the block mapping table when use K9G8G08 1GBytes NAND flash.



Block #	Section	Blocks	Area
0 – 6	1st/2nd Bootloader(1) + TOC(2) + Eboot(4)	7	WMR_AREA
7 – 106	OS	100	Special
107 – 110	VFL Info Section	4	VFL
111 – 212	Reserved Section	102	
213 – 222	FTL Info Section	10	FTL
223 – 239	Free Section (Log Section + Free List)	17	
240 – 2047	Data Section	1808	

Figure 2-1. NAND flash memory map

- WMR_AREA is combined which Block0 image and TOC area and Eboot area.

  Block 0 is used for 1st bootloader and 2nd bootloader.

  And Block 0's 126 page is used for initial bad block information and 127 page is used for signature. The signature of current version is "W220".

  TOC block is reserved with 3 blocks and Eboot is reserved with 5 blocks include Bad block reserving.

- The special area is used for OS area.

- The VFL Info Section is used for VFL information.
- The Reserved Section is for replace the blocks when runtime bad block is occurred.

- The FTL Info Section is for FTL information.
- The Free Section is used for write buffer. Log Section needs 14 blocks for the function of LSB recovery. If you do not need it, 7 blocks is needed.
- The Data Section is for storage region for OS.

These values are calculated in CalcGlobal function in WMRGlobal.c file.
And some values are defined in WMRTypes.h and WMRConfig.h file.
We have some tool for calculate the memory map automatically.

3. Porting procedure

3.1. SMDK2450 BSP Directory Structure with PocketMory Solution

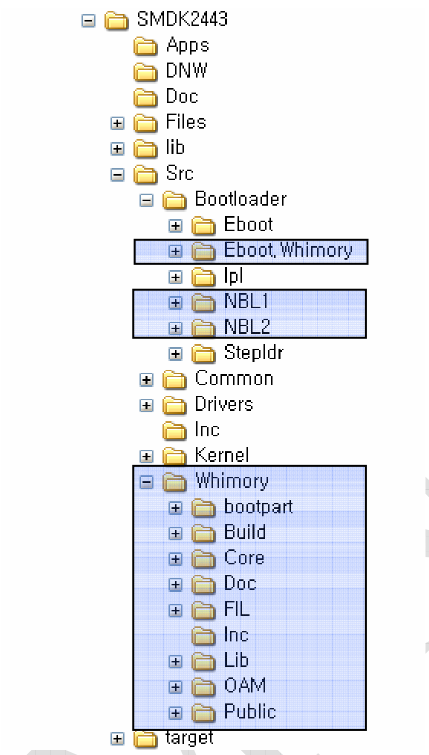


Figure 3-1. PocketMory Directory Structure

The blue boxed code is for PocketMory solution.

- ✓ Eboot.Whimory : The eboot loader directory for support MLC nand flash using PocketMory solution.
- ✓ NBL1 : First Bootloader code.
- ✓ NBL2 : Second Bootloader code.
- ✓ Whimory : PocketMory Directory.

3.2. Copy PocketMory related code to original BSP.

Copy Eboot.Whimory, NBL1, NBL2, Whomory directory to original BSP as Figure 3.1.

3.3. Change DIRS to build PocketMory Solution.

3.3.1. Src\Bootloader\dirs

```
DIRS=   \
            Eboot.Whimory      \
            NBL1      \
            NBL2     \
```

To use MLC type NAND like SLC type one for block number 0, you have to replace NBL1 with NBL1.SLC folder(this is default).

3.3.2. Src\Whimory\dirs

```
DIRS=    \
            Core       \
            FIL        \
            OAM        \
            Public     \
            BOOTPART          \
```

3.4. Change BSP for PocketMory

3.4.1. smdk2450.bat file

Call "src\Whimory\wmrenv.bat" file to set PocketMory related environment.

```
@REM - To support Multiple XIP
set IMGUPDATEXIP=1
set IMGMULTIXIP=1
set IMGBLOCKROM=1
set IMGNOKITL=1
set IMGNODEBUGGER=1
set MS_FLASH_DRIVER=

@REM - To support PocketMory
call %_TARGETPLATROOT%\src\Whimory\wmrenv.bat
```

3.4.2. config.bib file

Change the config.bib file to support multipleXIP function.

You can define the cbNKPagingPoolSize value in this file. This variable enables OEMs to limit memory used for paging read-only code/mapfile by setting the variable to the maximum memory allowed. This value has effect on the system performance.

If you increase this value, then the NAND flash access time can be reduced.

For more information, please refer to "Microsoft Windows CE" Documentation.

Windows CE Features > Core OS Services > Core OS Design Development > Kernel Reference > Kernel Global Variables > cbNKPagingPoolSize

```
MEMORY
IF IMGMULTIXIP !
IF WINCEDEBUG = retail
      NK          80200000    01900000    RAMIMAGE
      RAM          81B00000    02500000    RAM
ELSE
      NK          80200000    02800000    RAMIMAGE
      RAM          82A00000    01500000    RAM
ENDIF
      FLASH     92000000    00100000    RESERVED
ENDIF

IF IMGMULTIXIP
#define   CHAIN_ADDRESS             81700000
#define   CHAIN_LENGTH 00001000

            XIPKERNEL       80200000            00300000            RAMIMAGE
            NK              80500000            01200000            NANDIMAGE
            CHAIN           $(CHAIN_ADDRESS)        $(CHAIN_LENGTH)          RESERVED

            RAM             80500000            03B00000            RAM

            FLASH                   92000000    00100000    RESERVED
ENDIF



; Common RAM areas
            ARGS                    80020000    00000800    RESERVED
            SLEEP                   80028000    00002000  RESERVED
            EDBG                    80030000    00020000    RESERVED
            DISPLAY                 80100000    00100000    RESERVED
;           CF_DMA_BUF            83FE0000    00020000    RESERVED

CONFIG
      AUTOSIZE=ON
      COMPRESSION=ON
      KERNELFIXUPS=ON
      FSRAMPERCENT=0x40404040
```

```
IF IMGMULTIXIP
        RAM_AUTOSIZE=OFF
        ROM_AUTOSIZE=OFF
        DLLADDR_AUTOSIZE=ON

        XIPSCHAIN=$(CHAIN_ADDRESS)

        AUTOSIZE_ROMGAP=10000
        AUTOSIZE_DLLADDRGAP=0
        AUTOSIZE_DLLDATAADDRGAP=0
        AUTOSIZE_DLLCODEADDRGAP=0

        ROMFLAGS=0

ENDIF

IF IMGPROFILER
    PROFILE=ON
ELSE
    PROFILE=OFF
ENDIF


;
; ROMFLAGS is a bitmask of options for the kernel
;   ROMFLAGS      0x0001      Disallow Paging
;   ROMFLAGS      0x0002      Not all KMode
;   ROMFLAGS      0x0010      Trust Module only
;
IF IMGTRUSTROMONLY
    ROMFLAGS=10
ELSE
    ROMFLAGS=00
ENDIF


        ROMSTART = 80200000
```

```
            ROMWIDTH = 32

IF WINCEDEBUG = retail
            ROMSIZE = 01A00000
ELSE
            ROMSIZE = 02800000
ENDIF
```

3.4.3. platform.bib file

Remove smflash.dll from WinCE image.

Add BIBDrv.dll to support Multiple-XIP.

```
IF BSP_NONANDFS !
; -------------------------------------------------------------------------------
; File system drivers
;
    ; This is needed in the NK region because it is needed by BINFS
    ; to load other regions
;     smflash.dll             $(_FLATRELEASEDIR)\smflash.dll          NK          SH
    ; block driver that binfs interfaces with
      BIBDrv.dll              $(_FLATRELEASEDIR)\BIBDrv.dll           XIPKERNEL   SH
ENDIF ; BSP_NONANDFS
```

Add OnDisk.dll to WinCE Image.

```
FILES
;           Name            Path                                Memory Type
;           --------------  ----------------------------------  -----------
IF BSP_POCKETMORY
            ONDisk.dll      $(_FLATRELEASEDIR)\ONDisk.dll   NK          SH
ENDIF
```

3.4.4. platform.reg file.

Add below settings to support PocketMory storage folder.

This sample registry setting makes two partitions, the first one partition size is 32Mbytes and the second one partition size is the rest size of total nand flash. You can set this size using "WMRStartSector" and "WMRNumOfSector".

```
IF BSP_POCKETMORY
```

```
[HKEY_LOCAL_MACHINE\Drivers\BuiltIn\FlashDisk]
    "Prefix"="DSK"
    "Dll"="ONDisk.dll"
    "Order"=dword:1
    "IClass"=multi_sz:"{A4E7EDDA-E575-4252-9D6B-4195D48BB865}"
    "Profile"="FlashDisk"
    "BmlVolumeId"=dword:0; BML volume ID = 0
    "BmlPartitionId"=dword:8            ; BML parition ID = PARTITION_ID_FILESYSTEM
    "WMRStartSector"=dword:0
    "WMRNumOfSector"=dword:10000            ; 32MByte

[HKEY_LOCAL_MACHINE\System\StorageManager\Profiles\FlashDisk]
     "DefaultFileSystem"="FATFS"
     "PartitionDriver"="mspart.dll"
     "Name"="PocketMory MLC Disk"
     "Folder"="PocketMory"
     "AutoMount"=dword:1
     "AutoPart"=dword:1
     "AutoFormat"=dword:1
     "MountFlags"=dword:0

[HKEY_LOCAL_MACHINE\System\StorageManager\Profiles\FlashDisk\FATFS]
     "FriendlyName"="PocketMory FAT FileSystem"
     "Dll"="fatfsd.dll"
     "Flags"=dword:00000014    ;FATFS_ENABLE_BACKUP_FAT| FATFS_DISABLE_AUTOSCAN
     "FormatTfat"=dword:1
     "EnableCacheWarm"=dword:0


;-------------------------------------------------------------------------
; 2nd FAT Area
;-------------------------------------------------------------------------

[HKEY_LOCAL_MACHINE\Drivers\BuiltIn\FlashDisk1]
    "Prefix"="DSK"
    "Dll"="ONDisk.dll"
    "Order"=dword:1
    "IClass"=multi_sz:"{A4E7EDDA-E575-4252-9D6B-4195D48BB865}"
    "Profile"="FlashDisk1"
    "BmlVolumeId"=dword:0            ; BML volume ID = 0
    "BmlPartitionId"=dword:9         ; BML parition ID = PARTITION_ID_FILESYSTEM1
    "WMRStartSector"=dword:10000
    "WMRNumOfSector"=dword:ffffffff          ; last location

[HKEY_LOCAL_MACHINE\System\StorageManager\Profiles\FlashDisk1]
     "DefaultFileSystem"="FATFS"
     "PartitionDriver"="mspart.dll"
     "Name"="PocketMory MLC Disk1"
     "Folder"="PocketMory1"
     "AutoMount"=dword:1
     "AutoPart"=dword:1
     "AutoFormat"=dword:1
     "MountFlags"=dword:0

[HKEY_LOCAL_MACHINE\System\StorageManager\Profiles\FlashDisk1\FATFS]
     "FriendlyName"="PocketMory FAT FileSystem1"
     "Dll"="fatfsd.dll"
     "Flags"=dword:00000014 ; FATFS_ENABLE_BACKUP_FAT | FATFS_DISABLE_AUTOSCAN
     "FormatTfat"=dword:1
     "EnableCacheWarm"=dword:0

ENDIF BSP_POCKETMORY
```

Add below setting to support MultipleXIP.

```
; HIVE BOOT SECTION
IF BSP_NONANDFS !
[HKEY_LOCAL_MACHINE\System\StorageManager\AutoLoad\SMFlash]
          "DriverPath"="Drivers\\BlockDevice\\SMFlash"
          "LoadFlags"=dword:1
          "MountFlags"=dword:11
          "BootPhase"=dword:0

[HKEY_LOCAL_MACHINE\Drivers\BlockDevice\SMFlash]
          "Prefix"="DSK"
```

```
            "Dll"="BIBDrv.dll"
            "Order"=dword:0
            "Ioctl"=dword:4
            "Profile"="SMFlash"
            "FriendlyName"="Samsung Flash Driver"
            "MountFlags"=dword:11
            "BootPhase"=dword:0

; Bind BINFS to the block driver
[HKEY_LOCAL_MACHINE\System\StorageManager\Profiles\SMFlash]
            "DefaultFileSystem"="BINFS"
            "PartitionDriver"="mspart.dll"
            "AutoMount"=dword:1
            "AutoPart"=dword:1
            "MountFlags"=dword:11
            "Folder"="ResidentFlash"
            "Name"="Samsung Flash Disk"
            "BootPhase"=dword:0
ENDIF ; BSP_NONANDFS
; END HIVE BOOT SECTION
```

To use HIVE-based registry, you can refer below registry setting.

```
; @CESYSGEN IF FILESYS_FSREGHIVE
; HIVE BOOT SECTION
[HKEY_LOCAL_MACHINE\init\BootVars]
     "RegistryFlags"=dword:1
     "Flags"=dword:3
; END HIVE BOOT SECTION
; @CESYSGEN ENDIF FILESYS_FSREGHIVE
```

```
; HIVE BOOT SECTION
IF BSP_NONANDFS !
[HKEY_LOCAL_MACHINE\System\StorageManager\AutoLoad\SMFlash]
            "DriverPath"="Drivers\\BlockDevice\\SMFlash"
            "LoadFlags"=dword:1
            "MountFlags"=dword:11
            "BootPhase"=dword:0
```

```
[HKEY_LOCAL_MACHINE\Drivers\BlockDevice\SMFlash]
            "Prefix"="DSK"
            "Dll"="BIBDrv.dll"
            "Order"=dword:0
            "Ioctl"=dword:4
            "Profile"="SMFlash"
            "FriendlyName"="Samsung Flash Driver"
            "MountFlags"=dword:11
            "BootPhase"=dword:0

; Bind BINFS to the block driver
[HKEY_LOCAL_MACHINE\System\StorageManager\Profiles\SMFlash]
            "DefaultFileSystem"="BINFS"
            "PartitionDriver"="mspart.dll"
            "AutoMount"=dword:1
            "AutoPart"=dword:1
            "MountFlags"=dword:11
            "Folder"="ResidentFlash"
            "Name"="Samsung Flash Disk"
            "BootPhase"=dword:0
ENDIF ; BSP_NONANDFS
; END HIVE BOOT SECTION
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;---------------------------------------------------------------------------
; Added by MMS .. 30-SEP-2004
;---------------------------------------------------------------------------
IF BSP_POCKETMORY
; HIVE BOOT SECTION

[HKEY_LOCAL_MACHINE\Drivers\BuiltIn\FlashDisk]
    "Prefix"="DSK"
    "Dll"="ONDisk.dll"
    "Order"=dword:1
    "Flags"=dword:1000
    "IClass"=multi_sz:"{A4E7EDDA-E575-4252-9D6B-4195D48BB865}"
    "Profile"="FlashDisk"
    "BmlVolumeId"=dword:0; BML volume ID = 0
    "BmlPartitionId"=dword:8              ; BML parition ID = PARTITION_ID_FILESYSTEM
    "WMRStartSector"=dword:0
    "WMRNumOfSector"=dword:10000              ; 32MByte

[HKEY_LOCAL_MACHINE\System\StorageManager\Profiles\FlashDisk]
     "DefaultFileSystem"="FATFS"
     "PartitionDriver"="mspart.dll"
     "Name"="PocketMory MLC Disk"
     "Folder"="ROOT"
     "AutoMount"=dword:1
     "AutoPart"=dword:1
     "AutoFormat"=dword:1
     "MountAsROM"=dword:0
     "MountHidden"=dword:1
     "MountAsRoot"=dword:1
     "MountAsBootable"=dword:1

[HKEY_LOCAL_MACHINE\System\StorageManager\Profiles\FlashDisk\FATFS]
     "FriendlyName"="PocketMory FAT FileSystem"
     "Dll"="fatfsd.dll"
     "Flags"=dword:00000014 ; FATFS_ENABLE_BACKUP_FAT | FATFS_DISABLE_AUTOSCAN
     "FormatTfat"=dword:1
     "EnableCacheWarm"=dword:0

; END HIVE BOOT SECTION
;------------------------------------------------------------------------
; 2nd FAT Area
;------------------------------------------------------------------------

[HKEY_LOCAL_MACHINE\Drivers\BuiltIn\FlashDisk1]
    "Prefix"="DSK"
    "Dll"="ONDisk.dll"
    "Order"=dword:1
    "IClass"=multi_sz:"{A4E7EDDA-E575-4252-9D6B-4195D48BB865}"
    "Profile"="FlashDisk1"
    "BmlVolumeId"=dword:0              ; BML volume ID = 0
    "BmlPartitionId"=dword:9           ; BML parition ID = PARTITION_ID_FILESYSTEM1
    "WMRStartSector"=dword:10000
    "WMRNumOfSector"=dword:fffffff             ; last location

[HKEY_LOCAL_MACHINE\System\StorageManager\Profiles\FlashDisk1]
     "DefaultFileSystem"="FATFS"
     "PartitionDriver"="mspart.dll"
     "Name"="PocketMory MLC Disk1"
     "Folder"="PocketMory1"
     "AutoMount"=dword:1
     "AutoPart"=dword:1
     "AutoFormat"=dword:1
     "MountFlags"=dword:0

[HKEY_LOCAL_MACHINE\System\StorageManager\Profiles\FlashDisk1\FATFS]
     "FriendlyName"="PocketMory FAT FileSystem1"
     "Dll"="fatfsd.dll"
     "Flags"=dword:00000014 ; FATFS_ENABLE_BACKUP_FAT | FATFS_DISABLE_AUTOSCAN
     "FormatTfat"=dword:1
     "EnableCacheWarm"=dword:0

ENDIF BSP_POCKETMORY
```

3.4.5. ioctl.c file

Path: Src\Kernel\Oal\

Insert IO command for handling PocketMory

```
//------------------------------------------------------------------------------
//
//    define PSII control
//
#define __PSII_DEFINED__
CRITICAL_SECTION csPocketStoreBML;

#if defined(__PSII_DEFINED__)
UINT32 PSII_HALWrapper(VOID *pPacket, VOID *pInOutBuf, UINT32 *pResult);
```

```
#endif //#if defined(__PSII_DEFINED__)

#define IOCTL_POCKETSTORE_CMD             \
            CTL_CODE(FILE_DEVICE_HAL, 4070, METHOD_BUFFERED, FILE_ANY_ACCESS)
#define IOCTL_POCKETSTOREII_CMD           \
            CTL_CODE(FILE_DEVICE_HAL, 4080, METHOD_BUFFERED, FILE_ANY_ACCESS)

BOOL OALIoCtlPostInit(
            UINT32 code, VOID *pInpBuffer, UINT32 inpSize, VOID *pOutBuffer,
            UINT32 outSize, UINT32 *pOutSize)
{
            RETAILMSG(1,(TEXT("[OEMIO:INF]    + IOCTL_HAL_POSTINIT\r\n")));
            InitializeCriticalSection(&csPocketStoreBML);
            RETAILMSG(1,(TEXT("[OEMIO:INF]    - IOCTL_HAL_POSTINIT\r\n")));

            return TRUE;
}

BOOL OALIoCtlPocketStoreCMD(
            UINT32 code, VOID *pInpBuffer, UINT32 inpSize, VOID *pOutBuffer,
            UINT32 outSize, UINT32 *pOutSize)
{
            BOOL bResult;

            EnterCriticalSection(&csPocketStoreBML);
            bResult = PSII_HALWrapper(pInpBuffer, pOutBuffer, pOutSize);
            LeaveCriticalSection(&csPocketStoreBML);

            if (bResult == FALSE)
            {
                        RETAILMSG(1,(TEXT("[OEMIO:INF]        *    IOCTL_POCKETSTOREII_CMD
Failed\r\n")));
                        return FALSE;
            }
            return TRUE;
}
```

```
//------------------------------------------------------------------------------
//
//    Global: g_oalIoCtlTable[]
//
//    IOCTL handler table. This table includes the IOCTL code/handler pairs
//    defined in the IOCTL configuration file. This global array is exported
//    via oal_ioctl.h and is used by the OAL IOCTL component.
//
const OAL_IOCTL_HANDLER g_oalIoCtlTable[] = {
#if defined(__PSII_DEFINED__)
{ IOCTL_POCKETSTOREII_CMD,    0,    OALIoCtlPocketStoreCMD        },
{ IOCTL_HAL_POSTINIT,         0,    OALIoCtlPostInit              },
#endif //#if defined(__PSII_DEFINED__)
#include "ioctl_tab.h"
};
```

3.4.6. sources file

Path: Src\Kernel\Oal\

Link libraries of PocketMory.

```
INCLUDES=$(INCLUDES);..\..\inc;.\;$(_TARGETPLATROOT)\Src\Whimory\inc;\
              $(_TARGETPLATROOT)\Src\Whimory\Public\inc;\
              $(_TARGETPLATROOT)\Src\Whimory\OAM\WinCE;\
              $(_TARGETPLATROOT)\Src\Whimory\Core\VFL;

WHIMORYLIB=$(_TARGETPLATROOT)\src\Whimory\Lib

SOURCELIBS=$(SOURCELIBS) \
                  $(WHIMORYLIB)\$(_TGTCPU)\$(WINCEDEBUG)\WMRGlobal.lib        \
                  $(WHIMORYLIB)\$(_TGTCPU)\$(WINCEDEBUG)\FTL.lib              \
                  $(WHIMORYLIB)\$(_TGTCPU)\$(WINCEDEBUG)\VFL.lib              \
                  $(WHIMORYLIB)\$(_TGTCPU)\$(WINCEDEBUG)\PMHALWrapper.lib     \
                  $(WHIMORYLIB)\$(_TGTCPU)\$(WINCEDEBUG)\WinCEWMROAM.lib      \
                  $(WHIMORYLIB)\$(_TGTCPU)\$(WINCEDEBUG)\FIL.lib              \
                  $(WHIMORYLIB)\$(_TGTCPU)\$(WINCEDEBUG)\WMR_Utils.lib        \
```

3.4.7. oemaddrtab_cfg.inc file

Change table in this file in order to support multipleXIP.

```
        DCD         0x80000000, 0x30000000, 64          ; 64 MB DRAM BANK 6
```

3.4.8. Copy blcommon.c file

To fusing multiple XIP images, Copy blcommon.c file present in the [BSP]\Doc\ directory to X:\WINCE500\PLATFORM\COMMON\SRC\COMMON\BOOT\BLCOMMON and X:\WINCE500\PUBLIC\COMMON\OAK\DRIVERS\ETHDBG\BLCOMMON directory.

If you don’t copy this file, there will be error during fusing the OS image to NAND flash.

3.5. For about 0 block guarantee.

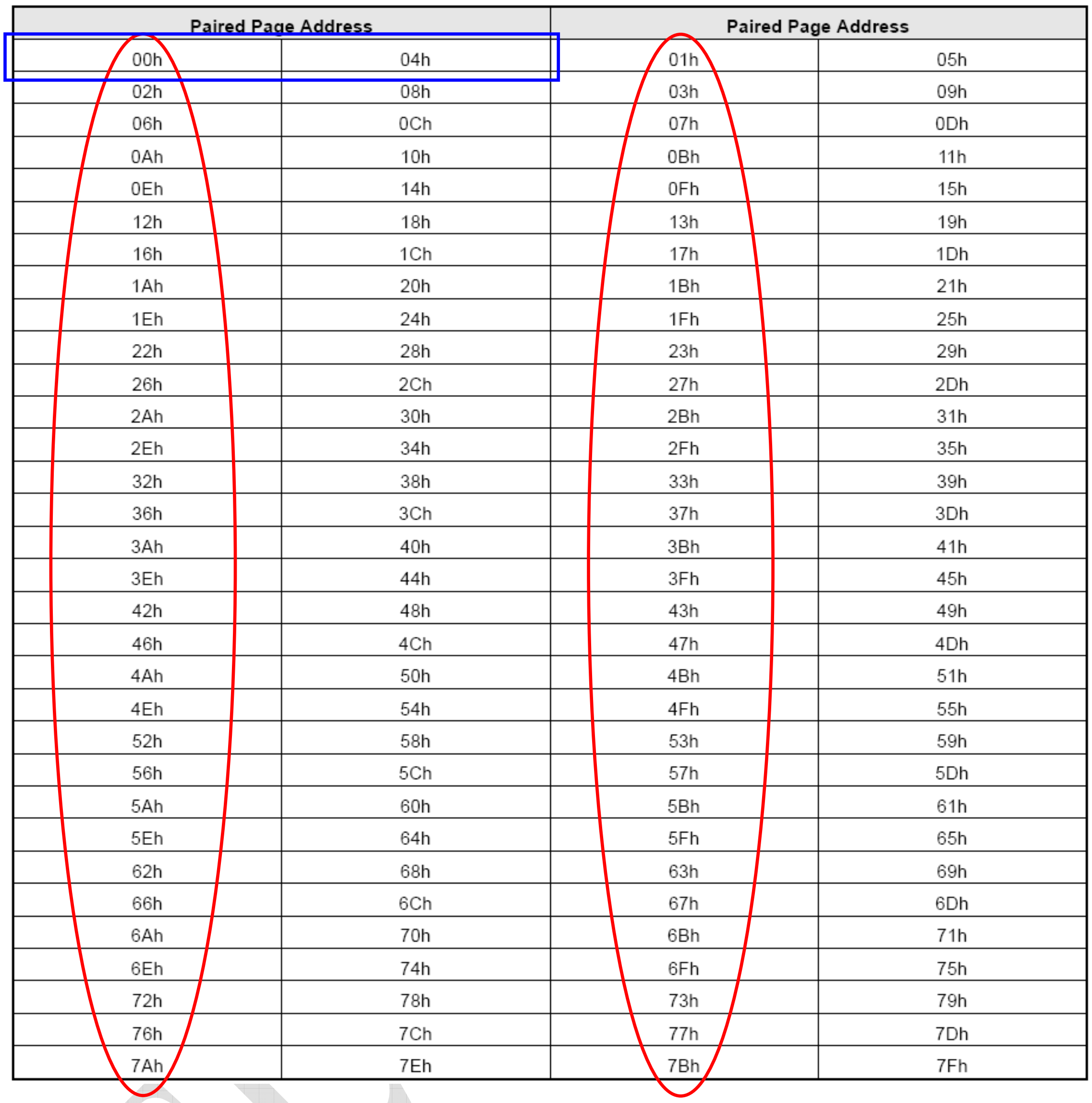
The boot code of 1st and 2nd page 4Kbytes is transferred into Steppingstone during reset. Therefore this two pages has to be guaranteed. But the MLC nand flash ECC error incidence is higher than SLC nand flash.

The MLC nand flash’s one cell is used by LSB(least significant bit) and MSB(most significant bit).

If we write only LSB, we can use the MLC nand like SLC nand.

This algorithm is used to guarantee 0th block.

The below table shows what is LSB and MSB paired page.



Paired Page Address		Paired Page Address	
00h	04h	01h	05h
02h	08h	03h	09h
06h	0Ch	07h	0Dh
0Ah	10h	0Bh	11h
0Eh	14h	0Fh	15h
12h	18h	13h	19h
16h	1Ch	17h	1Dh
1Ah	20h	1Bh	21h
1Eh	24h	1Fh	25h
22h	28h	23h	29h
26h	2Ch	27h	2Dh
2Ah	30h	2Bh	31h
2Eh	34h	2Fh	35h
32h	38h	33h	39h
36h	3Ch	37h	3Dh
3Ah	40h	3Bh	41h
3Eh	44h	3Fh	45h
42h	48h	43h	49h
46h	4Ch	47h	4Dh
4Ah	50h	4Bh	51h
4Eh	54h	4Fh	55h
52h	58h	53h	59h
56h	5Ch	57h	5Dh
5Ah	60h	5Bh	61h
5Eh	64h	5Fh	65h
62h	68h	63h	69h
66h	6Ch	67h	6Dh
6Ah	70h	6Bh	71h
6Eh	74h	6Fh	75h
72h	78h	73h	79h
76h	7Ch	77h	7Dh
7Ah	7Eh	7Bh	7Fh

Figure 3-2. Paired Page Address Information

For example, blue box in above table, 00h and 04h is pared page. It means the 00h page is LSB and the 04h page is MSB. If 00h page is programmed and 04h page is not programmed, this paired page's characteristic is same with SLC NAND. The only red circle is LSB in above picture.

To guarantee the StepLoader(the page 0 and 1), the paired page #4 and #5 have to be remain.

This two pages (#4 and #5) are not allowed to write any data even the all 0xFF data.

The other pages like from page 6 to last page of Block 0, the LSB and MSB page can be used.

Because the StepLdr can detects and corrects the ECC error.

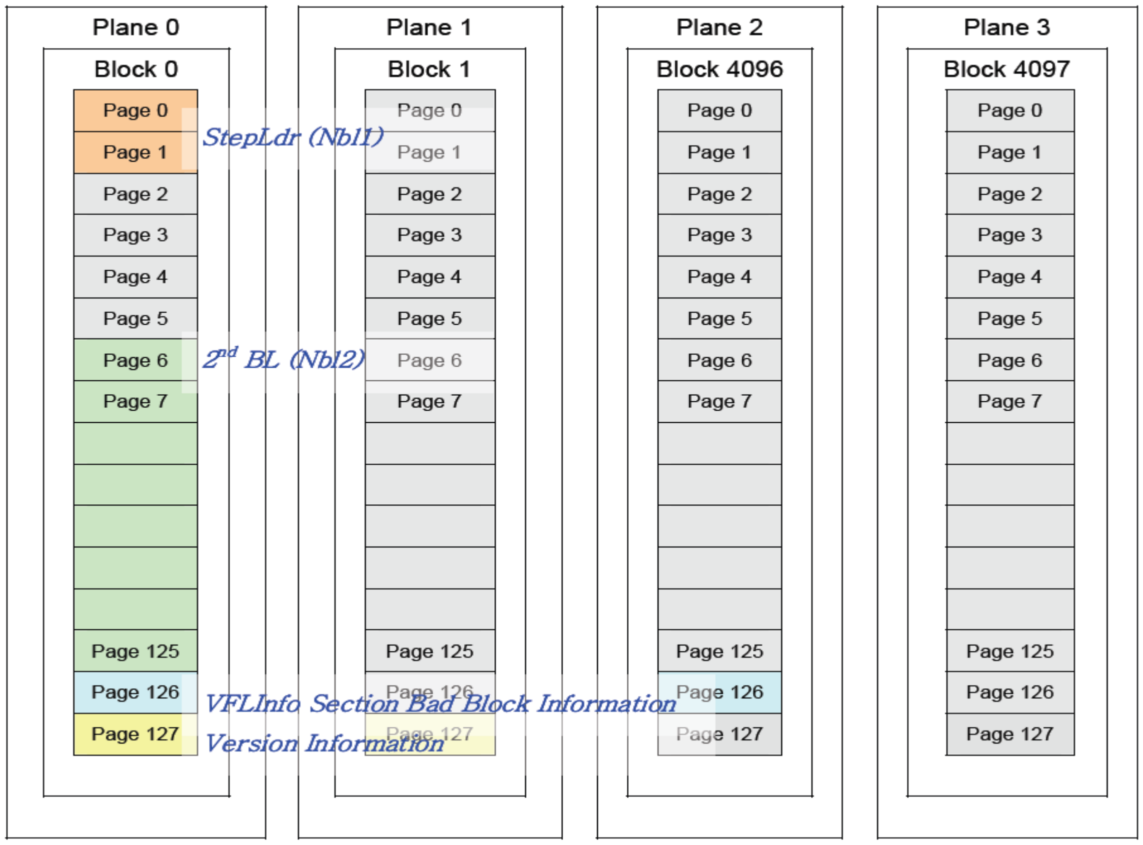


Figure 3-3. Sample Block 0 usage

In case of 4KByte page size NAND flash, for example K9GAG08/K9LBG08/K9HCG08, the Page 0 and 1 have to be written only 2048 bytes.

Because our NAND Flash controller read 2048bytes from page 0 and page 1 of Block 0 by H/W OM pin setting. For more detail information, please refer to "Users Manual – NAND Flash Controller" part.

And it doesn't need to write ECC parity value to page 0 and page 1.

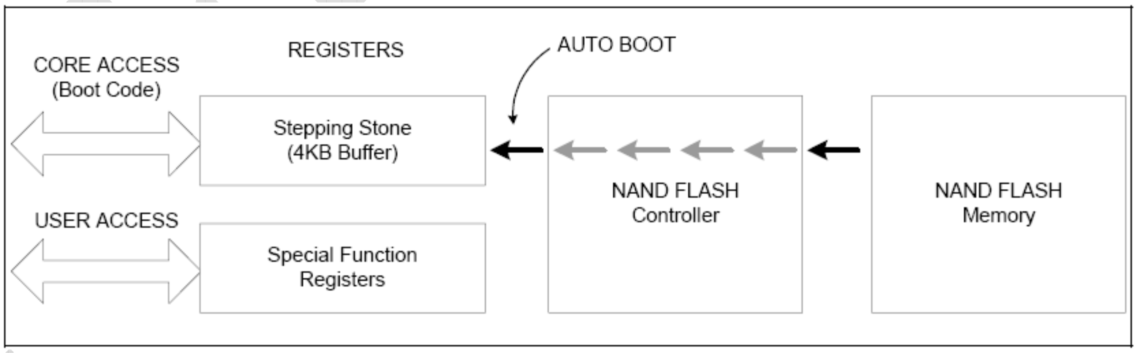


Figure 3-4. NAND Flash Controller Boot loader Block Diagram

4. What have to be modified to change NAND flash memory map?

4.1. WMR_USER_SUBLKS_RATIO

WMR_USER_SUBLKS_RATIO in WMRConfig.h file.

This value is reserved block ratio for whimory context & reserved section.

The MLC nand flash guaranteed bad block ration is 2.5%.

If you change this value, all block mapping is calculated by automatically.

You can emulate this values using “Whimory_NANDMap.xls” tool in Doc directory.

At now we set this value to 5% in sample BSP.

4.2. WMR_AREA_SIZE

WMR_AREA_SIZE in WMRTypes.h file.

This area is used for 1st Bootloader, 2nd Bootloader, TOC and Eboot.

From 0 to this defined value is not controlled by VFL area. You can only access this area using FIL functions. If you want to change the Eboot area can be controlled by VLF function, you have to move the Eboot area to special area.

4.3. SPECIAL_AREA_SIZE

SPECIAL_AREA_SIZE in WMRTypes.h file.

Special Area is used for OS area. You can change this value to fit your WinCE OS Image size.

And you have to change some definition in the loader.h file for eboot as below values.

The reserved block have to same value with SPECIAL_AREA_START number.

For example, if the SPECIAL_AREA_START number is 5, the RESERVED_BOOT_BLOCKS number have to be 5.

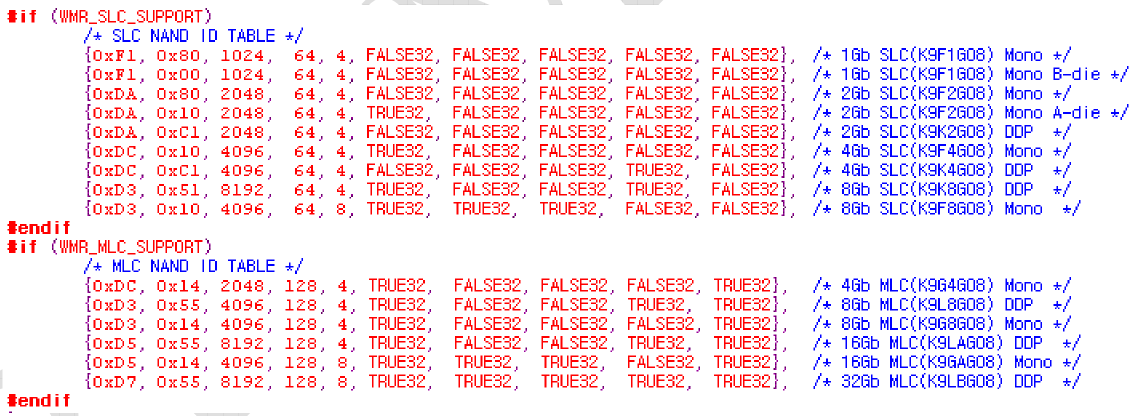
```
//
// OEM Reserved (Nand) Blocks for TOC and various bootloaders
//

// NAND Boot (loads into SteppingStone) @ Block 0
#define NBOOT_BLOCK                 (0)
#define NBOOT_BLOCK_SIZE            (1)
#define NBOOT_SECTOR                BLOCK_TO_SECTOR(NBOOT_BLOCK)

// TOC @ Block 1
#define TOC_BLOCK                   (1)
#define TOC_BLOCK_SIZE              (1)
```

```
#define TOC_BLOCK_RESERVED          (0)
#define TOC_SECTOR                  BLOCK_TO_SECTOR(TOC_BLOCK)

// Eboot @ Block 2
#define EBOOT_BLOCK                 (2)
#define EBOOT_BLOCK_SIZE            (3)
#define EBOOT_BLOCK_RESERVED        (0)
#define EBOOT_SECTOR                BLOCK_TO_SECTOR(EBOOT_BLOCK)

#define RESERVED_BOOT_BLOCKS        (NBOOT_BLOCK_SIZE + TOC_BLOCK_SIZE
+TOC_BLOCK_RESERVED + EBOOT_BLOCK_SIZE + EBOOT_BLOCK_RESERVED)

// Images start after OEM Reserved Blocks
#define IMAGE_START_BLOCK           RESERVED_BOOT_BLOCKS
#define IMAGE_START_SECTOR          BLOCK_TO_SECTOR(IMAGE_START_BLOCK)
```

4.4. BLOCKS_PER_BANK

BLOCKS_PER_BANK number is automatically set in FIL code.

For example, S3C2450X_NAND.c file defined below nand flash.



```
#if (WMR_SLC_SUPPORT)
        /* SLC NAND ID TABLE */
        {0xF1, 0x80, 1024,  64, 4, FALSE32, FALSE32, FALSE32, FALSE32, FALSE32},  /* 1Gb SLC(K9F1G08) Mono */
        {0xF1, 0x00, 1024,  64, 4, FALSE32, FALSE32, FALSE32, FALSE32, FALSE32},  /* 1Gb SLC(K9F1G08) Mono B-die */
        {0xDA, 0x80, 2048,  64, 4, FALSE32, FALSE32, FALSE32, FALSE32, FALSE32},  /* 2Gb SLC(K9F2G08) Mono */
        {0xDA, 0x10, 2048,  64, 4, TRUE32,  FALSE32, FALSE32, FALSE32, FALSE32},  /* 2Gb SLC(K9F2G08) Mono A-die */
        {0xDA, 0xC1, 2048,  64, 4, FALSE32, FALSE32, FALSE32, FALSE32, FALSE32},  /* 2Gb SLC(K9K2G08) DDP  */
        {0xDC, 0x10, 4096,  64, 4, TRUE32,  FALSE32, FALSE32, FALSE32, FALSE32},  /* 4Gb SLC(K9F4G08) Mono */
        {0xDC, 0xC1, 4096,  64, 4, FALSE32, FALSE32, FALSE32, TRUE32,  FALSE32},  /* 4Gb SLC(K9K4G08) DDP  */
        {0xD3, 0x51, 8192,  64, 4, TRUE32,  FALSE32, FALSE32, TRUE32,  FALSE32},  /* 8Gb SLC(K9K8G08) DDP  */
        {0xD3, 0x10, 4096,  64, 8, TRUE32,  TRUE32,  TRUE32,  FALSE32, FALSE32},  /* 8Gb SLC(K9F8G08) Mono  */
#endif
#if (WMR_MLC_SUPPORT)
        /* MLC NAND ID TABLE */
        {0xDC, 0x14, 2048, 128, 4, TRUE32,  FALSE32, FALSE32, FALSE32, TRUE32},   /* 4Gb MLC(K9G4G08) Mono */
        {0xD3, 0x55, 4096, 128, 4, TRUE32,  FALSE32, FALSE32, TRUE32,  TRUE32},   /* 8Gb MLC(K9L8G08) DDP  */
        {0xD3, 0x14, 4096, 128, 4, TRUE32,  FALSE32, FALSE32, FALSE32, TRUE32},   /* 8Gb MLC(K9G8G08) Mono */
        {0xD5, 0x55, 8192, 128, 4, TRUE32,  FALSE32, FALSE32, TRUE32,  TRUE32},   /* 16Gb MLC(K9LAG08) DDP  */
        {0xD5, 0x14, 4096, 128, 8, TRUE32,  TRUE32,  TRUE32,  FALSE32, TRUE32},   /* 16Gb MLC(K9GAG08) Mono */
        {0xD7, 0x55, 8192, 128, 8, TRUE32,  TRUE32,  TRUE32,  TRUE32,  TRUE32},   /* 32Gb MLC(K9LBG08) DDP  */
#endif
```

Figure 4-1. DEVInfo structure

The 1st value is device ID number.

The 2nd value is hidden ID number. For example, this value is different with DDP and Mono die.

The 3rd value is the number of total block.

The 4th value is the number of pages per one block.

The 5th value is the number of sectors per one page. One sector is 512 bytes.

The 6th value is for determine it can support 2 plane programming.

The 7th value is for determine it can support 2 plane read.

The 8th value is for determine it can support 2 plane read status.

The 9th value is for determine it can support internal interleaving function.

The 8th value is for determine it is MLC NAND flash or not.

4.5. Comparison Table

Below table is for compare with Mono die MLC nand (K9G8G08), DDP(K9L8G08), QDP(K9HAG08) for 2Kbyte/Page, Mono(K9GAG08) for 4Kbyte/Page.

	K9G8G08	K9G8G08 (2plane)	K9L8G08 (2plane)	K9HAG08 (2plane)	K9AG08 (2plane)
BANKS_TOTAL	1	1	2	4	1
SUBLKS_TOTAL	4096	2048	1024	1024	2048
BYTES_PER_SECTOR	512				
SECTORS_PER_SUPAGE	4	8	8	8	16
PAGES_PER_SUBLK	128	128	256	512	128
BYTES_PER_SUBLK	0x40000	0x80000	0x100000	0x200000	0x100000
SPECIAL_AREA_SIZE	200	100	50	25	100

Table 4-1 Comparison table for various MLC nand flash

4.6. Physical page write order.

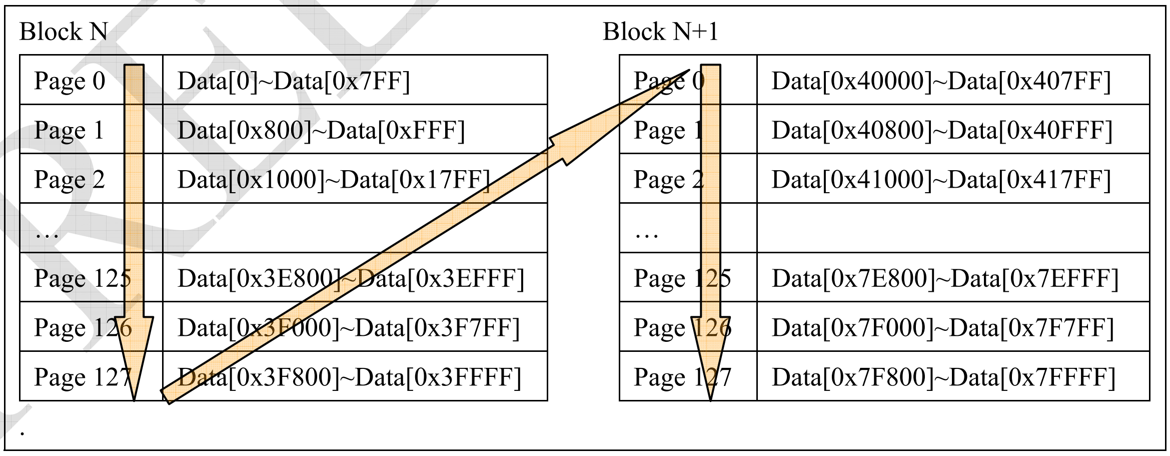
4.6.1. Normal programming

This picture shows the case of normal write. The data will be programmed like as below picture.

The page programming sequence is like as below.

[128 Block 0 Page]→[128 Block 1 Page]→[128 Block 2 Page]→…→[128 Block 127 Page]→

[129 Block 0 Page]→[129 Block 1 Page]→[129 Block 2 Page]→...



Block N

Page 0	Data[0]~Data[0x7FF]
Page 1	Data[0x800]~Data[0xFFF]
Page 2	Data[0x1000]~Data[0x17FF]
…	
Page 125	Data[0x3E800]~Data[0x3EFFF]
Page 126	Data[0x3F000]~Data[0x3F7FF]
Page 127	Data[0x3F800]~Data[0x3FFFF]

Block N+1

Page 0	Data[0x40000]~Data[0x407FF]
Page 1	Data[0x40800]~Data[0x40FFF]
Page 2	Data[0x41000]~Data[0x417FF]
…	
Page 125	Data[0x7E800]~Data[0x7EFFF]
Page 126	Data[0x7F000]~Data[0x7F7FF]
Page 127	Data[0x7F800]~Data[0x7FFFF]

.

Figure 4-2. 1-Plane Programming

4.6.2. 2 Plane programming

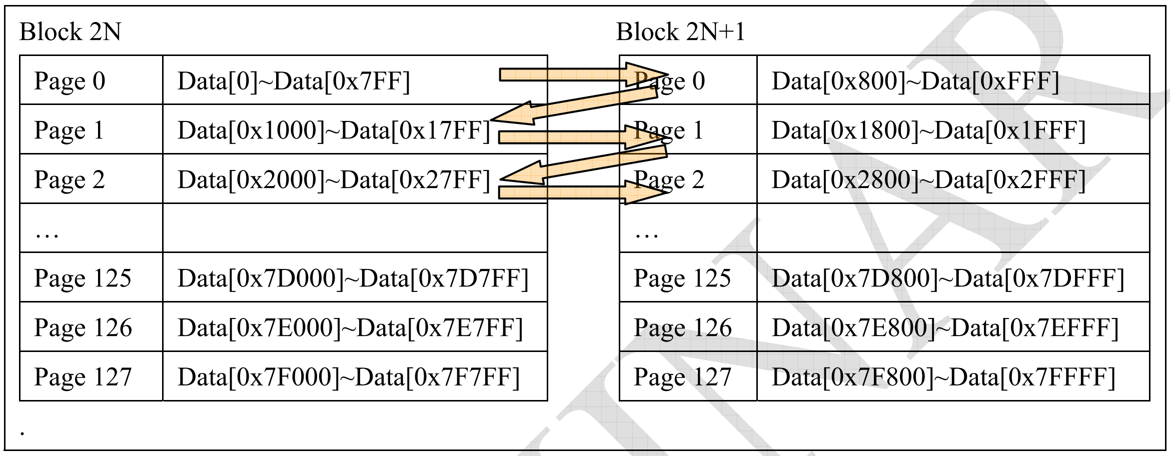
This picture shows the case of 2-Plane programming write. The data will be programmed like as below picture.

The page programming sequence is like as below.

[128 Block 0 Page]→[129 Block 0 Page] →

[128 Block 1 Page]→[129 Block 1 Page] →

[128 Block 2 Page]→[129 Block 2 Page] →…



Block 2N

Page 0	Data[0]~Data[0x7FF]
Page 1	Data[0x1000]~Data[0x17FF]
Page 2	Data[0x2000]~Data[0x27FF]
…	
Page 125	Data[0x7D000]~Data[0x7D7FF]
Page 126	Data[0x7E000]~Data[0x7E7FF]
Page 127	Data[0x7F000]~Data[0x7F7FF]

Block 2N+1

Page 0	Data[0x800]~Data[0xFFF]
Page 1	Data[0x1800]~Data[0x1FFF]
Page 2	Data[0x2800]~Data[0x2FFF]
…	
Page 125	Data[0x7D800]~Data[0x7DFFF]
Page 126	Data[0x7E800]~Data[0x7EFFF]
Page 127	Data[0x7F800]~Data[0x7FFFF]

.

Figure 4-3. 2-Plane Programming

4.6.3. 2 Plane and interleaving programming

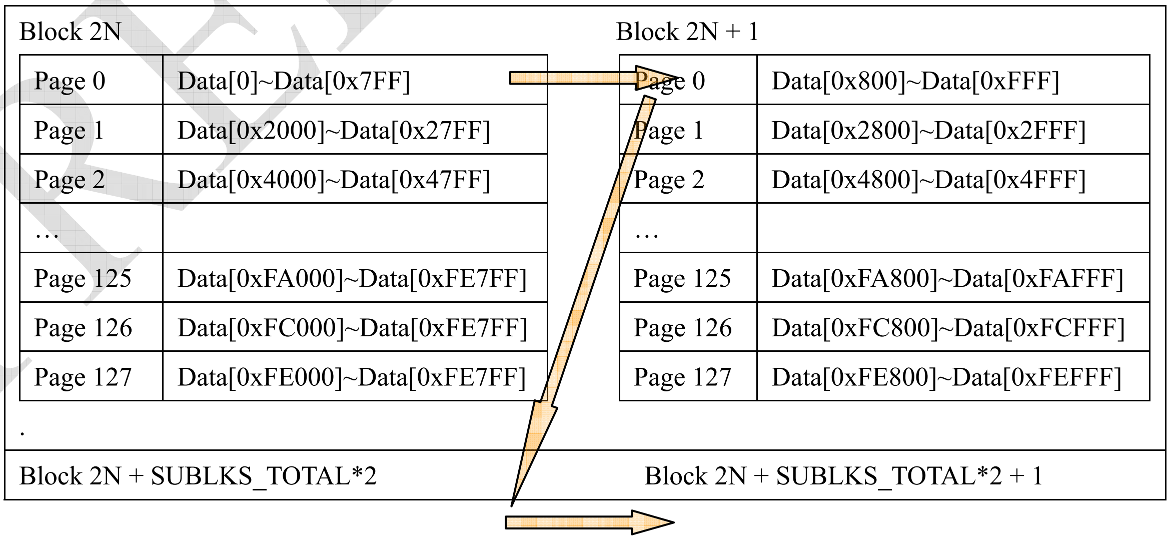
This picture shows the case of 2-Plane and interleaving programming write. The data will be programmed like as below picture.

The page programming sequence is like as below.

[128 Block 0 Page]→[129 Block 0 Page]→[128+4096 Block 0 Page]→[129+4096 Block 0 Page]→

[128 Block 1 Page]→[129 Block 1 Page]→[128+4096 Block 1 Page]→[129+4096 Block 1 Page]→

[128 Block 2 Page]→[129 Block 2 Page]→[128+4096 Block 2 Page]→[129+4096 Block 2 Page]→…



Block 2N

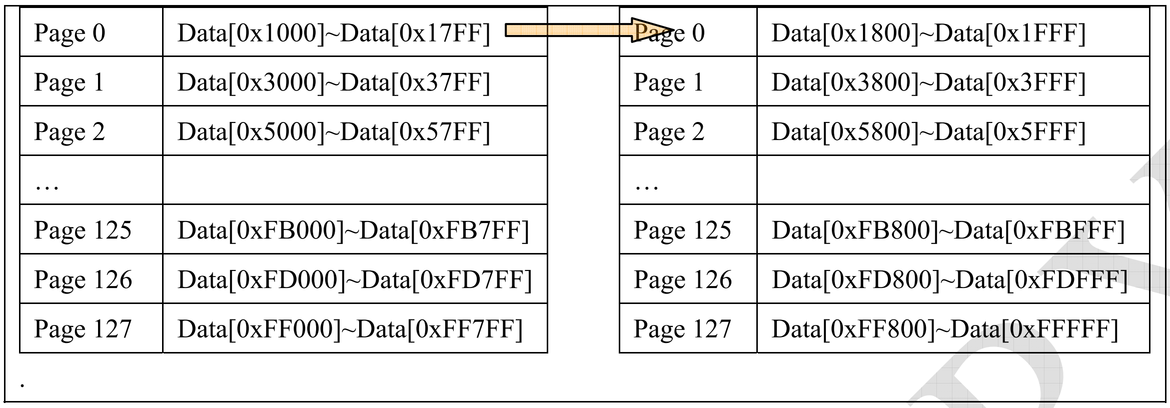
Page 0	Data[0]~Data[0x7FF]
Page 1	Data[0x2000]~Data[0x27FF]
Page 2	Data[0x4000]~Data[0x47FF]
…	
Page 125	Data[0xFA000]~Data[0xFE7FF]
Page 126	Data[0xFC000]~Data[0xFE7FF]
Page 127	Data[0xFE000]~Data[0xFE7FF]

Block 2N + 1

Page 0	Data[0x800]~Data[0xFFF]
Page 1	Data[0x2800]~Data[0x2FFF]
Page 2	Data[0x4800]~Data[0x4FFF]
…	
Page 125	Data[0xFA800]~Data[0xFAFFF]
Page 126	Data[0xFC800]~Data[0xFCFFF]
Page 127	Data[0xFE800]~Data[0xFEFFF]

.

Block 2N + SUBLKS_TOTAL*2 Block 2N + SUBLKS_TOTAL*2 + 1



Page 0	Data[0x1000]~Data[0x17FF]	Page 0	Data[0x1800]~Data[0x1FFF]
Page 1	Data[0x3000]~Data[0x37FF]	Page 1	Data[0x3800]~Data[0x3FFF]
Page 2	Data[0x5000]~Data[0x57FF]	Page 2	Data[0x5800]~Data[0x5FFF]
…		…	
Page 125	Data[0xFB000]~Data[0xFB7FF]	Page 125	Data[0xFB800]~Data[0xFBFFF]
Page 126	Data[0xFD000]~Data[0xFD7FF]	Page 126	Data[0xFD800]~Data[0xFDFFF]
Page 127	Data[0xFF000]~Data[0xFF7FF]	Page 127	Data[0xFF800]~Data[0xFFFFF]

.

Figure 4-4. 2-Plane, 2Way Programming

The SUBLKS_TOTAL number is block number for 1 plane.

For understanding NAND flash physical map, K9LAG08U0M NAND Flash map is as below.

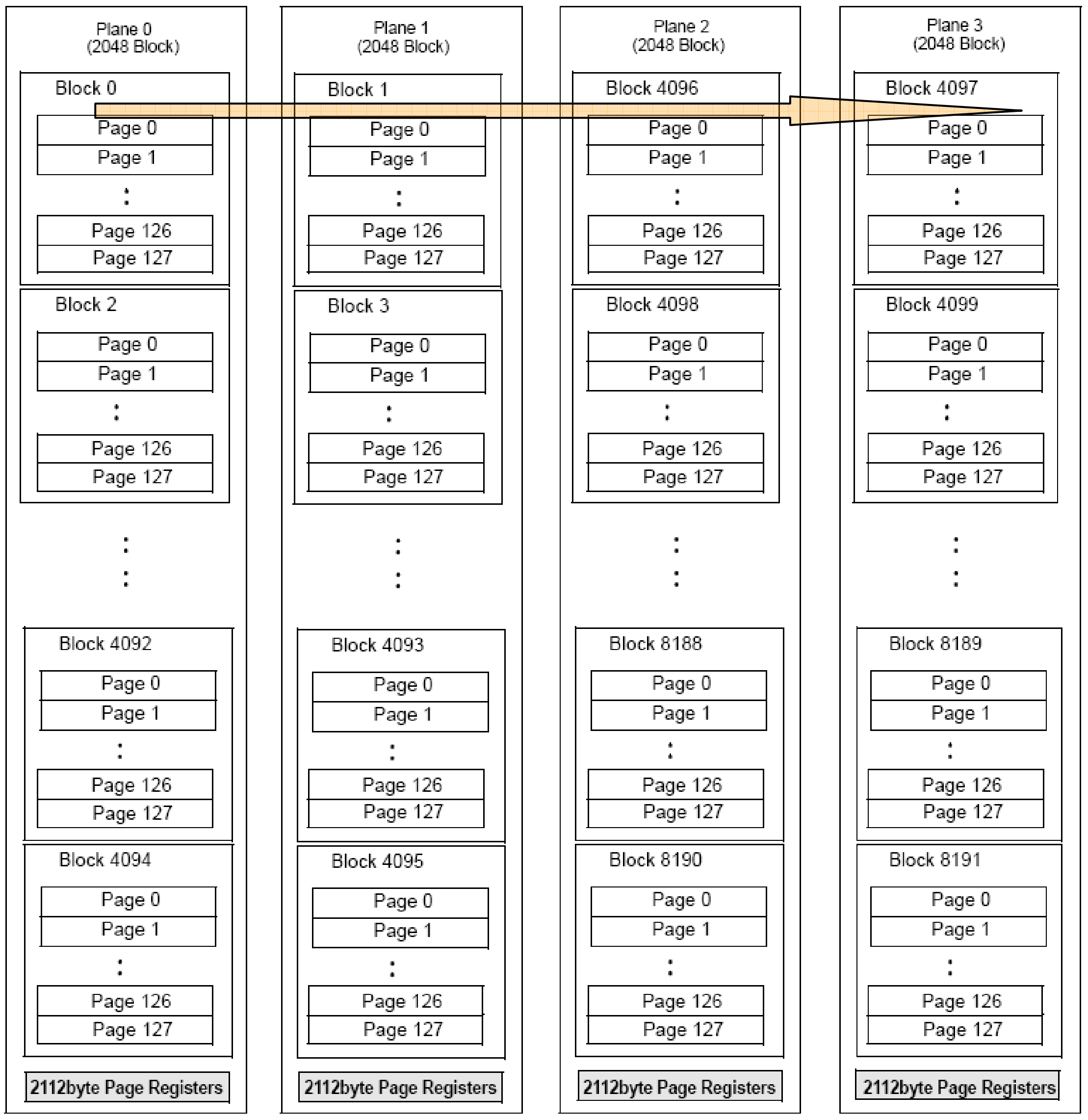
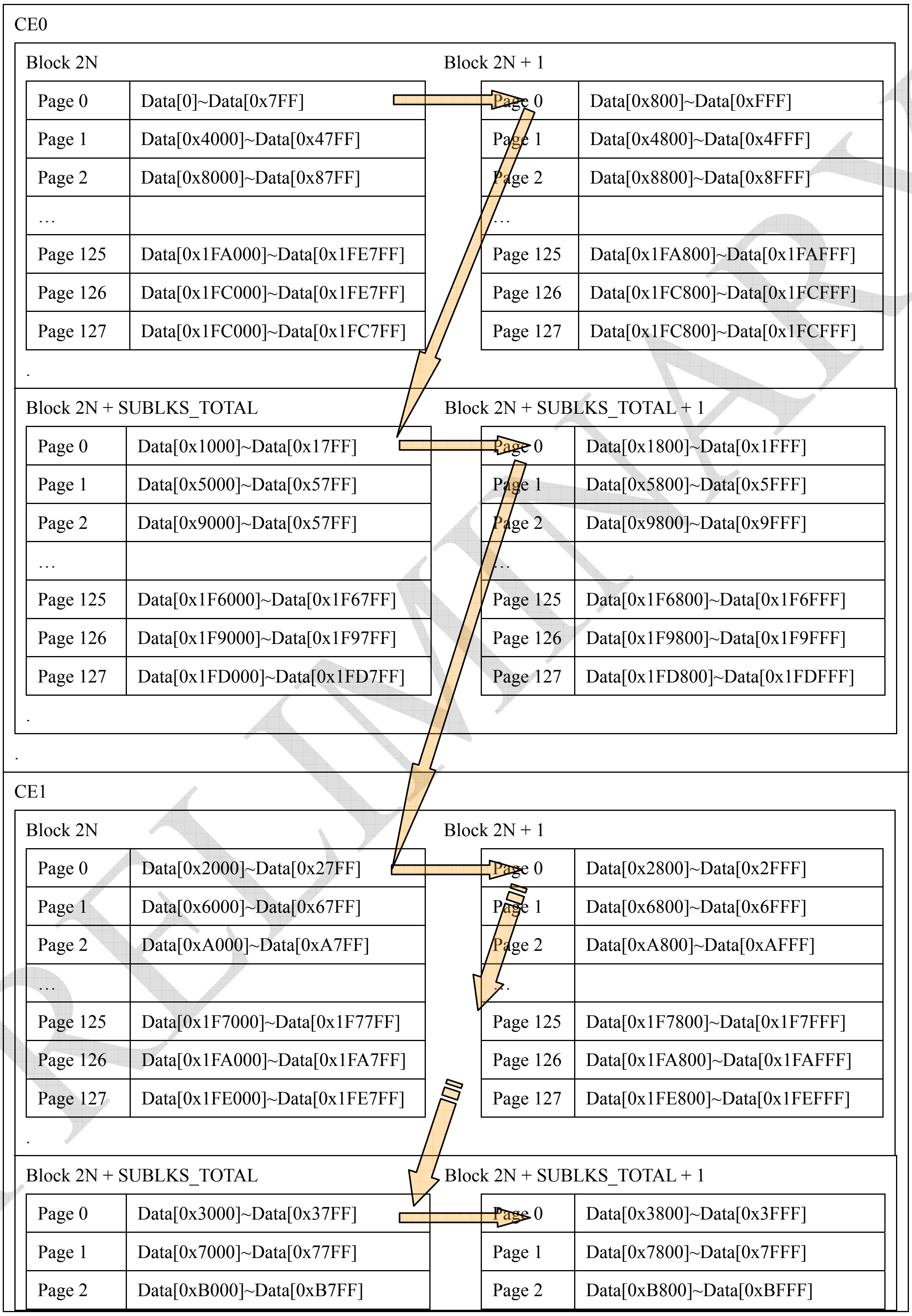


Figure 4-5. K9LAG08U0M Memory Map

K9LAG08U0M is arranged in four 4Gb memory planes. Each plane contains 2,048 blocks and 2112 byte page registers. This allows it to perform simultaneous page program and block erase by selecting one page or block from each plane. The block address map is configured so that two-plane program/erase operations can be executed by dividing the memory array into plane 0~1 or plane 2~3 separately.
For example, two-plane program/erase operation into plane 0 and plane 2 is prohibited. That is to say, two-plane program/erase operation into plane 0 and plane 1 or into plane 2 and plane 3 is allowed

4.6.4. 2 Plane, interleaving, 2CE programming

This picture shows the case of 2-Plane and interleaving programming write. The data will be programmed like as below picture.



CE0

Block 2N

Page 0	Data[0]~Data[0x7FF]
Page 1	Data[0x4000]~Data[0x47FF]
Page 2	Data[0x8000]~Data[0x87FF]
…	
Page 125	Data[0x1FA000]~Data[0x1FE7FF]
Page 126	Data[0x1FC000]~Data[0x1FE7FF]
Page 127	Data[0x1FC000]~Data[0x1FC7FF]

Block 2N + 1

Page 0	Data[0x800]~Data[0xFFF]
Page 1	Data[0x4800]~Data[0x4FFF]
Page 2	Data[0x8800]~Data[0x8FFF]
…	
Page 125	Data[0x1FA800]~Data[0x1FAFFF]
Page 126	Data[0x1FC800]~Data[0x1FCFFF]
Page 127	Data[0x1FC800]~Data[0x1FCFFF]

.

Block 2N + SUBLKS_TOTAL

Page 0	Data[0x1000]~Data[0x17FF]
Page 1	Data[0x5000]~Data[0x57FF]
Page 2	Data[0x9000]~Data[0x57FF]
…	
Page 125	Data[0x1F6000]~Data[0x1F67FF]
Page 126	Data[0x1F9000]~Data[0x1F97FF]
Page 127	Data[0x1FD000]~Data[0x1FD7FF]

Block 2N + SUBLKS_TOTAL + 1

Page 0	Data[0x1800]~Data[0x1FFF]
Page 1	Data[0x5800]~Data[0x5FFF]
Page 2	Data[0x9800]~Data[0x9FFF]
…	
Page 125	Data[0x1F6800]~Data[0x1F6FFF]
Page 126	Data[0x1F9800]~Data[0x1F9FFF]
Page 127	Data[0x1FD800]~Data[0x1FDFFF]

.

.

CE1

Block 2N

Page 0	Data[0x2000]~Data[0x27FF]
Page 1	Data[0x6000]~Data[0x67FF]
Page 2	Data[0xA000]~Data[0xA7FF]
…	
Page 125	Data[0x1F7000]~Data[0x1F77FF]
Page 126	Data[0x1FA000]~Data[0x1FA7FF]
Page 127	Data[0x1FE000]~Data[0x1FE7FF]

Block 2N + 1

Page 0	Data[0x2800]~Data[0x2FFF]
Page 1	Data[0x6800]~Data[0x6FFF]
Page 2	Data[0xA800]~Data[0xAFFF]
…	
Page 125	Data[0x1F7800]~Data[0x1F7FFF]
Page 126	Data[0x1FA800]~Data[0x1FAFFF]
Page 127	Data[0x1FE800]~Data[0x1FEFFF]

.

Block 2N + SUBLKS_TOTAL

Page 0	Data[0x3000]~Data[0x37FF]
Page 1	Data[0x7000]~Data[0x77FF]
Page 2	Data[0xB000]~Data[0xB7FF]

Block 2N + SUBLKS_TOTAL + 1

Page 0	Data[0x3800]~Data[0x3FFF]
Page 1	Data[0x7800]~Data[0x7FFF]
Page 2	Data[0xB800]~Data[0xBFFF]

…	
Page 125	Data[0x1F8000]~Data[0x1F87FF]
Page 126	Data[0x1FB000]~Data[0x1FB7FF]
Page 127	Data[0x1FF000]~Data[0x1FF7FF]

…	
Page 125	Data[0x1F8800]~Data[0x1F8FFF]
Page 126	Data[0x1FB800]~Data[0x1FBFFF]
Page 127	Data[0x1FF800]~Data[0x1FFFFF]

.

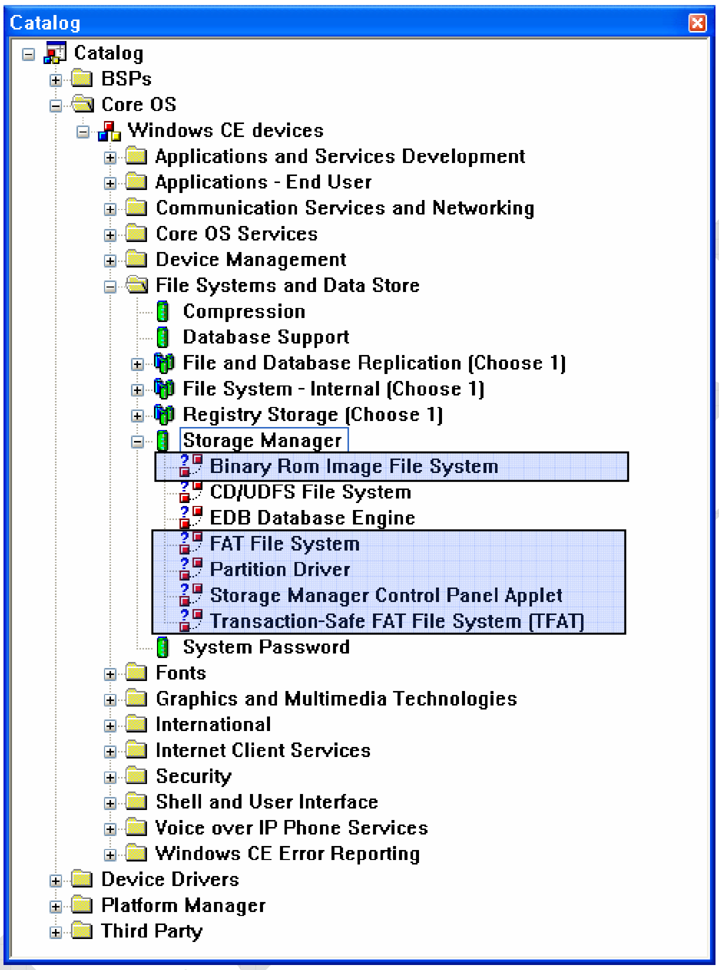
.

Figure 4-6. 2-Plane, 4Way Programming

5. Windows CE Fusing and Booting Procedure

5.1. Build BSP

Some catalog items has to be added to OS design to support multipleXIP with NAND flash.



```
Catalog
  Catalog
    BSPs
    Core OS
      Windows CE devices
        Applications and Services Development
        Applications - End User
        Communication Services and Networking
        Core OS Services
        Device Management
        File Systems and Data Store
          Compression
          Database Support
          File and Database Replication (Choose 1)
          File System - Internal (Choose 1)
          Registry Storage (Choose 1)
          Storage Manager
            Binary Rom Image File System
            CD/UDFS File System
            EDB Database Engine
            FAT File System
            Partition Driver
            Storage Manager Control Panel Applet
            Transaction-Safe FAT File System (TFAT)
          System Password
        Fonts
        Graphics and Multimedia Technologies
        International
        Internet Client Services
        Security
        Shell and User Interface
        Voice over IP Phone Services
        Windows CE Error Reporting
    Device Drivers
    Platform Manager
    Third Party
```

Figure 5-1. Catalog Items for MultipleXIP

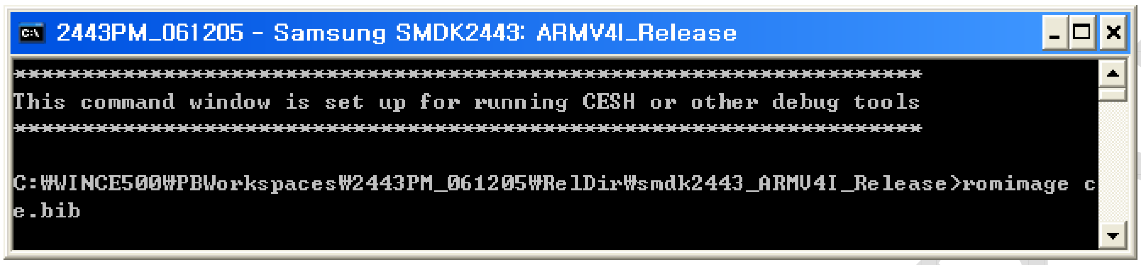
5.2. Change the ce.bib file in release directory to support MultipleXIP after build OS completed.

Change the region definition from NK to XIPKERNEL like as below.

The "[ReleaseDirectory]" string can be different depends on your build environment.

nk.exe	[ReleaseDirectory]\RelDir\smdk2450_ARMV4I_Release\nk.exe	XIPKERNEL	SH
coredll.dll	[ReleaseDirectory]\RelDir\smdk2450_ARMV4I_Release\coredll.dll	XIPKERNEL	SH
filesys.ese	[ReleaseDirectory]\RelDir\smdk2450_ARMV4I_Release\filesys.exe	XIPKERNEL	SH
binfs.dll	[ReleaseDirectory]\RelDir\smdk2450_ARMV4I_Release\binfs.dll	XIPKERNEL	SH
fsdmgr.dll	[ReleaseDirectory]\RelDir\smdk2450_ARMV4I_Release\fsdmgr.dll	XIPKERNEL	SH
mspart.dll	[ReleaseDirectory]\RelDir\smdk2450_ARMV4I_Release\mspart.dll	XIPKERNEL	SH
default.fdf	[ReleaseDirectory]\RelDir\smdk2450_ARMV4I_Release\default.fdf	XIPKERNEL	SH

5.3. Open the command window using platform builder menu [Build OS]->[Open Release Directory]. Enter the “romimage ce.bib” command.



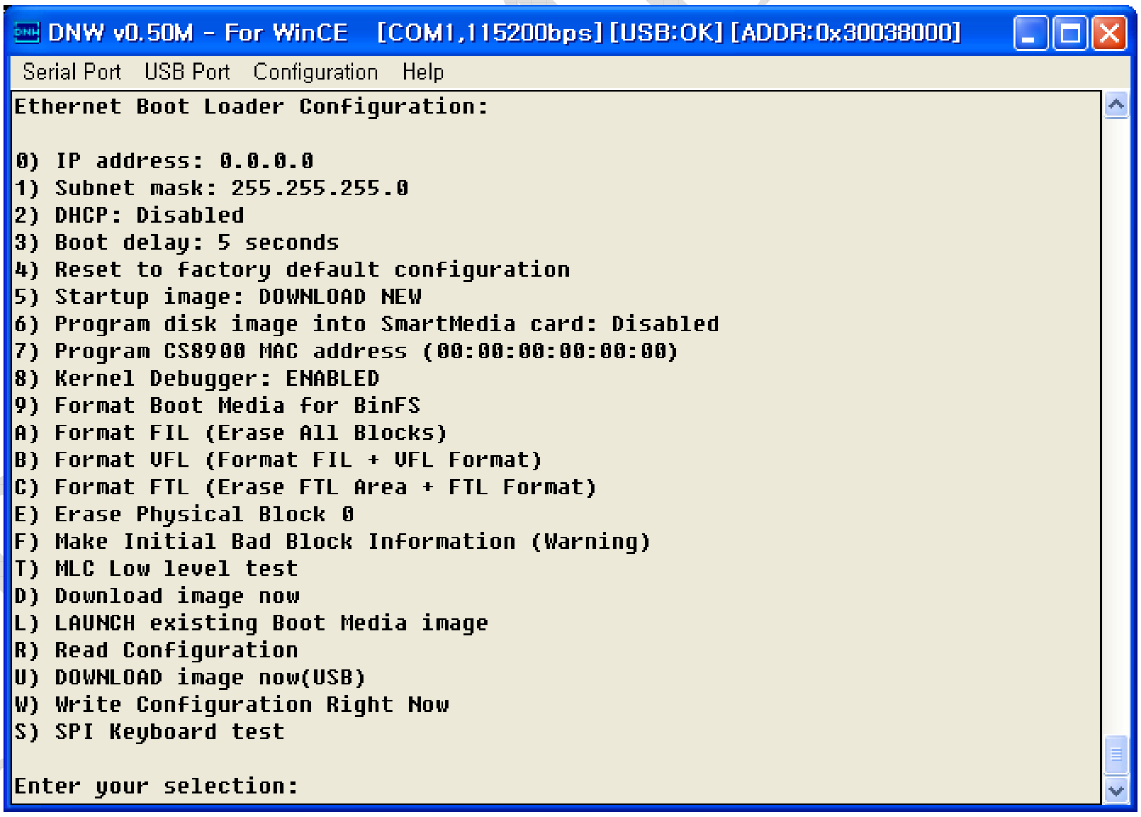
```
2443PM_061205 - Samsung SMDK2443: ARMV4I_Release
*************************************************************************
This command window is set up for running CESH or other debug tools
*************************************************************************

C:\WINCE500\PBWorkspaces\2443PM_061205\RelDir\smdk2443_ARMV4I_Release>romimage c
e.bib
```

Figure 5-2. Command window

Then below files will be generated.

- Chain.bin
- Nk.bin
- Xipkernel.bin
- Chain.lst

5.4. Download Eboot to 0x30038000 through USB on DNW menu.

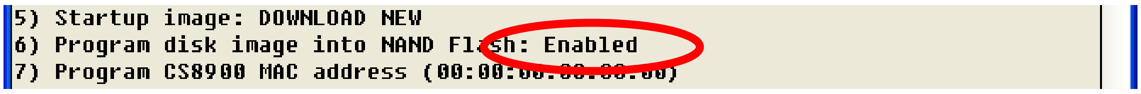


DNW v0.50M - For WinCE [COM1,115200bps] [USB:OK] [ADDR:0x30038000]
Serial Port USB Port Configuration Help

```
Ethernet Boot Loader Configuration:

0) IP address: 0.0.0.0
1) Subnet mask: 255.255.255.0
2) DHCP: Disabled
3) Boot delay: 5 seconds
4) Reset to factory default configuration
5) Startup image: DOWNLOAD NEW
6) Program disk image into SmartMedia card: Disabled
7) Program CS8900 MAC address (00:00:00:00:00:00)
8) Kernel Debugger: ENABLED
9) Format Boot Media for BinFS
A) Format FIL (Erase All Blocks)
B) Format VFL (Format FIL + VFL Format)
C) Format FTL (Erase FTL Area + FTL Format)
E) Erase Physical Block 0
F) Make Initial Bad Block Information (Warning)
T) MLC Low level test
D) Download image now
L) LAUNCH existing Boot Media image
R) Read Configuration
U) DOWNLOAD image now(USB)
W) Write Configuration Right Now
S) SPI Keyboard test

Enter your selection:
```

Figure 5-3. Download Eboot.nb0

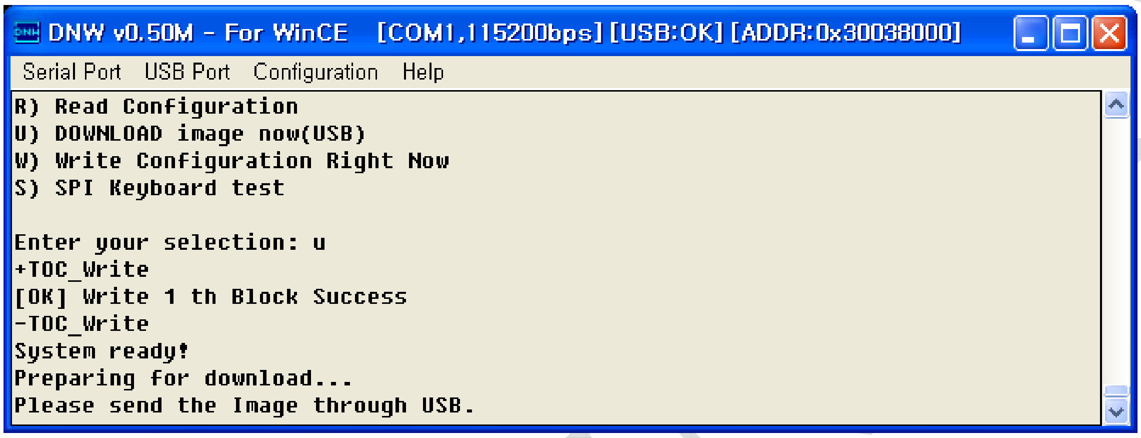
5.5. Press "6" to change "6) Program disk image into NAND Flash: Disabled" value to "Enable".



```
5) Startup image: DOWNLOAD NEW
6) Program disk image into NAND Flash: Enabled
7) Program CS8900 MAC address (00:00:00:00:00:00)
```

Figure 5-4. Change options

5.6. Press "U" to download and fusing the Block0Image.nb0.

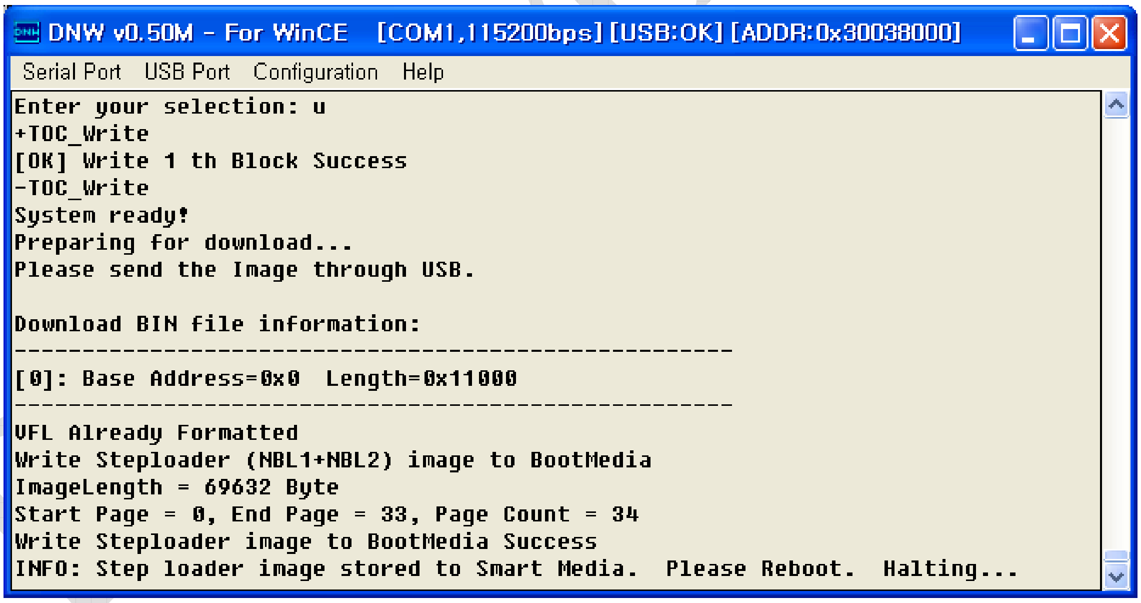


DNW v0.50M - For WinCE [COM1,115200bps] [USB:OK] [ADDR:0x30038000]
Serial Port USB Port Configuration Help

```
R) Read Configuration
U) DOWNLOAD image now(USB)
W) Write Configuration Right Now
S) SPI Keyboard test

Enter your selection: u
+TOC_Write
[OK] Write 1 th Block Success
-TOC_Write
System ready!
Preparing for download...
Please send the Image through USB.
```

Figure 5-5. Press "U" for transfer image through USB

5.7. Transfer "Block0img.nb0" image using Menu -> USB Port -> UBOOT menu.

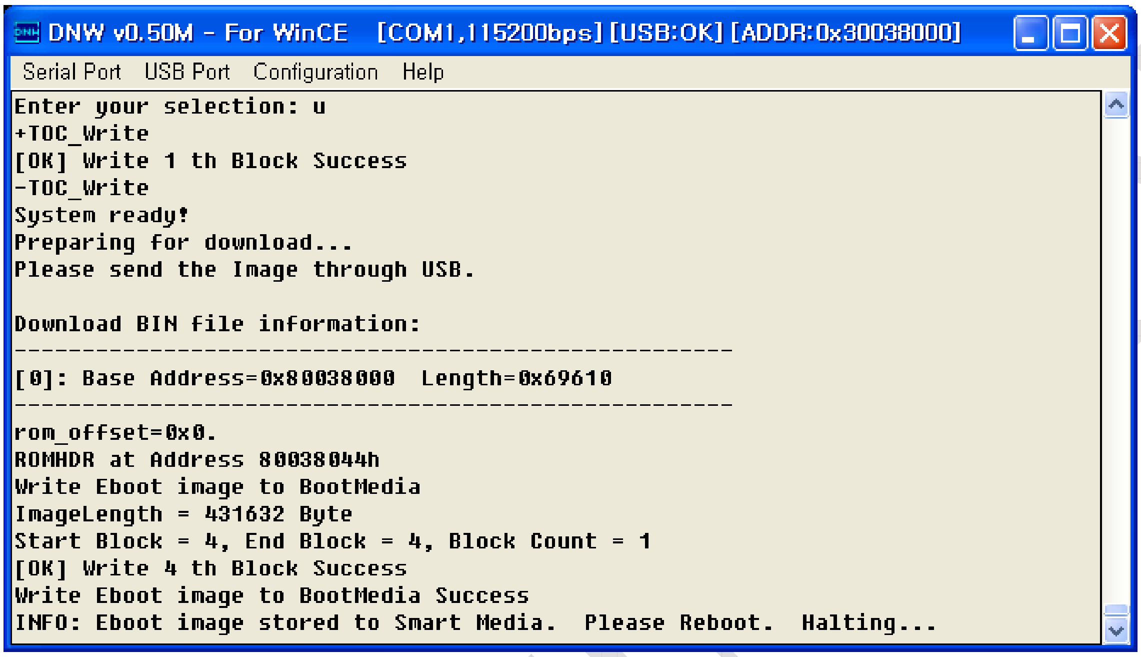


DNW v0.50M - For WinCE [COM1,115200bps] [USB:OK] [ADDR:0x30038000]
Serial Port USB Port Configuration Help

```
Enter your selection: u
+TOC_Write
[OK] Write 1 th Block Success
-TOC_Write
System ready!
Preparing for download...
Please send the Image through USB.

Download BIN file information:
-----------------------------------------------------
[0]: Base Address=0x0  Length=0x11000
-----------------------------------------------------
VFL Already Formatted
Write Steploader (NBL1+NBL2) image to BootMedia
ImageLength = 69632 Byte
Start Page = 0, End Page = 33, Page Count = 34
Write Steploader image to BootMedia Success
INFO: Step loader image stored to Smart Media.  Please Reboot.  Halting...
```

Figure 5-6. Fusing Block0img.nb0

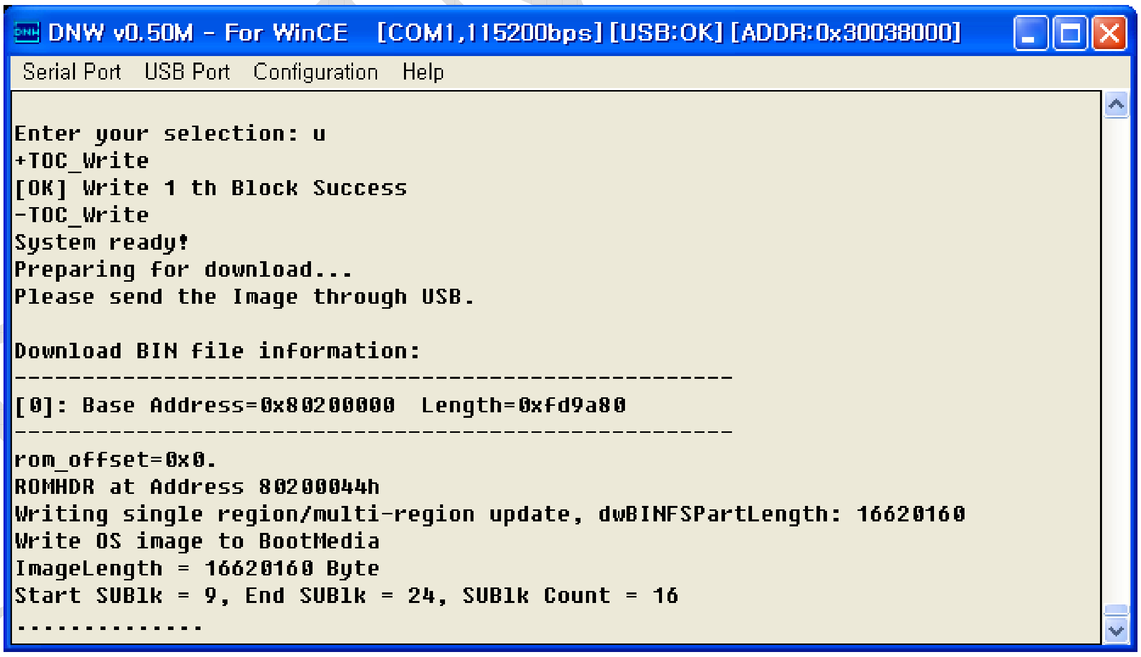
5.8. Turn Off and Turn On again. And download eboot.nb0 image again. And Press “U” for Download Eboot.bin. And transfer “Eboot.bin” image using UBOOT menu.


DNW v0.50M - For WinCE [COM1,115200bps] [USB:OK] [ADDR:0x30038000]
Serial Port USB Port Configuration Help

```
Enter your selection: u
+TOC_Write
[OK] Write 1 th Block Success
-TOC_Write
System ready!
Preparing for download...
Please send the Image through USB.

Download BIN file information:
-----------------------------------------------------
[0]: Base Address=0x80038000  Length=0x69610
-----------------------------------------------------
rom_offset=0x0.
ROMHDR at Address 80038044h
Write Eboot image to BootMedia
ImageLength = 431632 Byte
Start Block = 4, End Block = 4, Block Count = 1
[OK] Write 4 th Block Success
Write Eboot image to BootMedia Success
INFO: Eboot image stored to Smart Media.  Please Reboot.  Halting...
```

Figure 5-7. Fusing eboot.bin

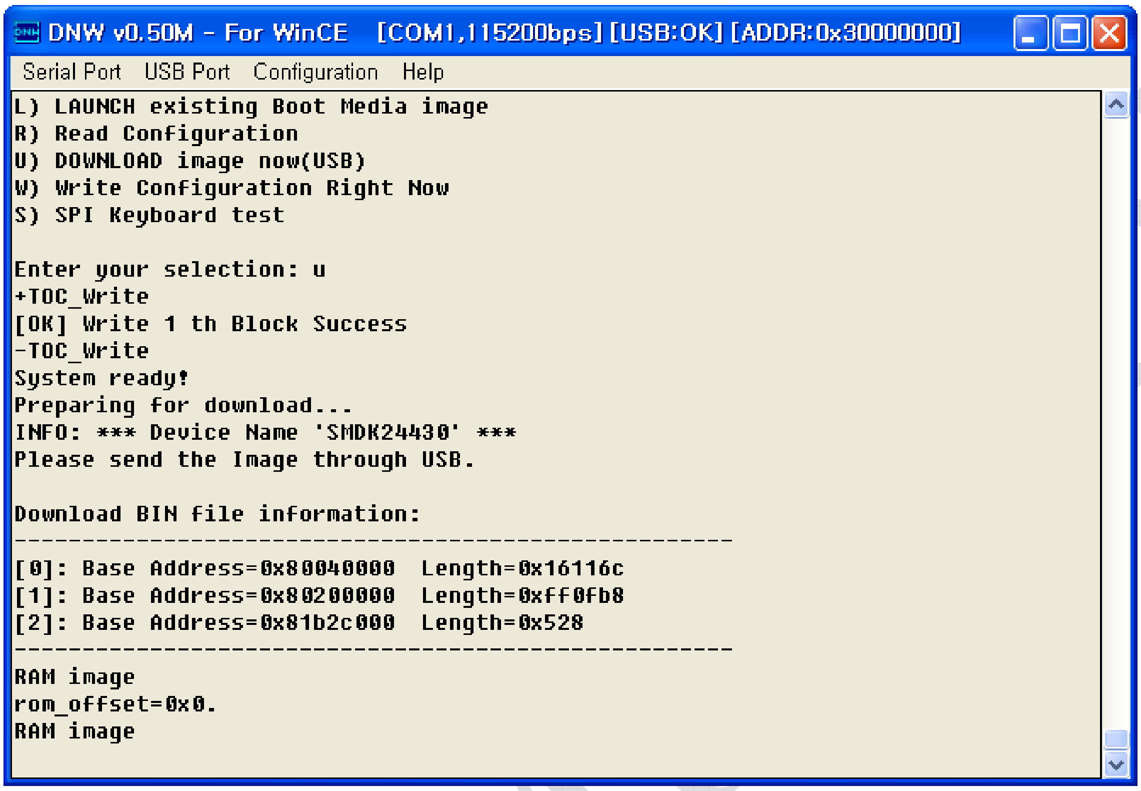
5.9. Turn Off and Turn On again. And download eboot.nb0 image again. And Press “U” for Download NK.bin. And transfer “NK.bin” image using UBOOT menu.


DNW v0.50M - For WinCE [COM1,115200bps] [USB:OK] [ADDR:0x30038000]
Serial Port USB Port Configuration Help

```

Enter your selection: u
+TOC_Write
[OK] Write 1 th Block Success
-TOC_Write
System ready!
Preparing for download...
Please send the Image through USB.

Download BIN file information:
-----------------------------------------------------
[0]: Base Address=0x80200000  Length=0xfd9a80
-----------------------------------------------------
rom_offset=0x0.
ROMHDR at Address 80200044h
Writing single region/multi-region update, dwBINFSPartLength: 16620160
Write OS image to BootMedia
ImageLength = 16620160 Byte
Start SUBlk = 9, End SUBlk = 24, SUBlk Count = 16
..............
```

Figure 5-8. Fusing nk.bin

5.10. In case of multipleXIP. Download chain.lst file using “UBOOT(WINCE500)” menu.


DNW v0.50M - For WinCE [COM1,115200bps] [USB:OK] [ADDR:0x30000000]
Serial Port
USB Port
Configuration
Help

```
L) LAUNCH existing Boot Media image
R) Read Configuration
U) DOWNLOAD image now(USB)
W) Write Configuration Right Now
S) SPI Keyboard test

Enter your selection: u
+TOC_Write
[OK] Write 1 th Block Success
-TOC_Write
System ready!
Preparing for download...
INFO: *** Device Name 'SMDK24430' ***
Please send the Image through USB.

Download BIN file information:
-----------------------------------------------------
[0]: Base Address=0x80040000  Length=0x16116c
[1]: Base Address=0x80200000  Length=0xff0fb8
[2]: Base Address=0x81b2c000  Length=0x528
-----------------------------------------------------
RAM image
rom_offset=0x0.
RAM image
```

Figure 5-9. Fusing multipleXIP image.

5.11. Then the WindowsCE image will boot up automatically.

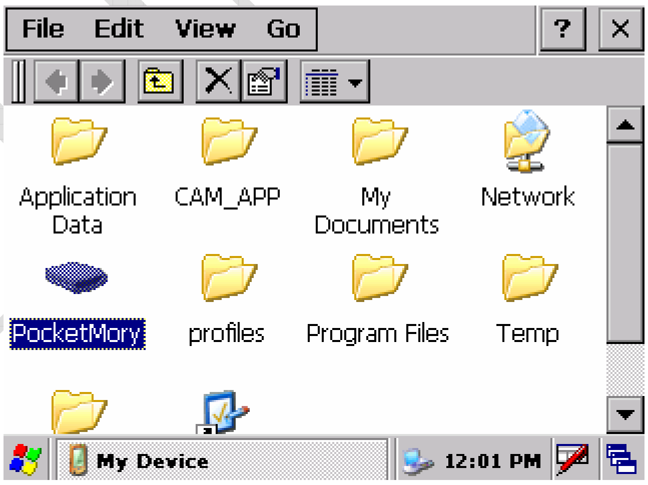


Figure 5-10. Windows CE screen

Below message is all messages for from fusing block0image, eboot, chain.lst and finally booting the Windows CE.

```
CLKOUT0:MPLL in, CLKOUT1:RTC clock.
```

```
+-----------------------------------------------+
| S3C2450X USB Downloader ver R0.1                    +
+-----------------------------------------------+
mSDR
MPLL=400.00MHz, ARMCLK=400.00MHz (1:1:2), HCLK=133.33MHz, PCLK=66.67MHz

USB: IN_ENDPOINT:1 OUT_ENDPOINT:3
FORMAT: <ADDR(DATA):4>+<SIZE(n+10):4>+<DATA:n>+<CS:2>
NOTE: 1. Power off/on or press the reset button for 1 sec
            in order to get a valid USB device address.
        2. For additional menu, Press any key.

reset status register = 00000001
Power-on Reset
USB host is not connected yet.
USB host is connected. Waiting a download.

Now, Downloading [ADDRESS:30038000h,TOTAL:524298]
RECEIVED FILE SIZE:       524298
(30806.6KB/S,0.0S)
RECEIVE FILE DONE !!

Checksum O.K.

Microsoft Windows CE Ethernet Bootloader Common Library Version 1.1 Built Dec  5 2006 10:42:36
Microsoft Windows CE Bootloader for the Samsung SMDK2450 Version 2.4 Built Dec  9 2006

InitDisplay
BP_Init
[FIL] ################
[FIL]  MID     = 0xec
[FIL]  DID     = 0xd3
[FIL]  HID[0] = 0x55
[FIL]  HID[1] = 0x25
```

```
[FIL]   HID[2] = 0x58
[FIL] ################
[FIL] ################
[FIL]   MID       = 0xec
[FIL]   DID       = 0xd3
[FIL]   HID[0] = 0x55
[FIL]   HID[1] = 0x25
[FIL]   HID[2] = 0x58
[FIL] ################
[FIL] ##############################
[FIL]   FIL Global Information
[FIL]   BANKS_TOTAL = 4
[FIL]   BLOCKS_PER_BANK = 1024
[FIL]   SUPPORT_INTERLEAVING = 1
[FIL]   SUBLKS_TOTAL = 1024
[FIL]   PAGES_PER_SUBLK = 512
[FIL]   PAGES_PER_BANK = 131072
[FIL]   SECTORS_PER_PAGE = 8
[FIL]   SECTORS_PER_SUBLK = 4096
[FIL]   USER_SECTORS_TOTAL = 3780608
[FIL]   ADDRESS_CYCLE = 5
[FIL] ##############################

[INFO] WMR_AREA_SIZE = 5
[INFO] SPECIAL_AREA_START = 5
[INFO] SPECIAL_AREA_SIZE = 25
[INFO] VFL_AREA_START = 30
[INFO] VFL_AREA_SIZE = 55
[INFO] VFL_INFO_SECTION_START = 30
[INFO] VFL_INFO_SECTION_SIZE = 4
[INFO] RESERVED_SECTION_START = 34
[INFO] RESERVED_SECTION_SIZE = 51
[INFO] FTL_INFO_SECTION_START = 85
[INFO] FTL_INFO_SECTION_SIZE = 6
[INFO] LOG_SECTION_SIZE = 7
[INFO] FREE_SECTION_START = 91
```

```
[INFO] FREE_SECTION_SIZE = 10
[INFO] FREE_LIST_SIZE = 3
[INFO] DATA_SECTION_START = 101
[INFO] DATA_SECTION_SIZE = 923
[INFO] FTL_AREA_START = 85
[INFO] FTL_AREA_SIZE = 939
[FTL:MSG] FIL_Init                              [OK]
[FTL:MSG] BUF_Init                              [OK]
[FTL:MSG] VFL_Init                              [OK]
[FTL:MSG] Not Formated !
[FTL:MSG] VFL_Format                            [OK]
[FTL:MSG] Write Signature                 [OK]
[FTL:MSG] VFL_Open                              [OK]
wNUM_BLOCKS : 1024(0x400)
TOC_Read
TOC_Read ERROR: INVALID_TOC Signature: 0xFFFFFFFF
TOC_Init: dwEntry:1, dwImageType: 0x2, dwImageStart: 0x0, dwImageLength: 0x0, dwLaunchAddr:
0x0
+BootConfigInit
-BootConfigInit
TOC {
dwSignature: 0x434F544E
BootCfg {
  ConfigFlags: 0x2020
  BootDelay: 0x5
  ImageIndex: 1
  IP: 0.0.0.0
  MAC Address: 00:00:00:00:00:00
  Port: 0.0.0.0
  SubnetMask: 255.255.255.0
}
ID[0] {
  dwVersion: 0x20004
  dwSignature: 0x45424F54
  String: 'eboot.nb0'
  dwImageType: 0x2
```

```
    dwTtlSectors: 0x400
    dwLoadAddress: 0x8C038000
    dwJumpAddress: 0x8C038000
    dwStoreOffset: 0x0
    sgList[0].dwSector: 0x2000
    sgList[0].dwLength: 0x400
}
ID[1] {
    dwVersion: 0x1
    dwSignature: 0x43465348
    String: ''
    dwImageType: 0x2
    dwTtlSectors: 0x0
    dwLoadAddress: 0x0
    dwJumpAddress: 0x0
    dwStoreOffset: 0x0
}
chainInfo.dwLoadAddress: 0X00000000
chainInfo.dwFlashAddress: 0X00000000
chainInfo.dwLength: 0X00000000
}
Press [ENTER] to download image stored on boot media, or [SPACE] to enter boot monitor.

Initiating image download in 4 seconds.

Ethernet Boot Loader Configuration:

0) IP address: 0.0.0.0
1) Subnet mask: 255.255.255.0
2) DHCP: Disabled
3) Boot delay: 5 seconds
4) Reset to factory default configuration
5) Startup image: DOWNLOAD NEW
6) Program disk image into NAND Flash: Disabled
7) Program CS8900 MAC address (00:00:00:00:00:00)
8) Kernel Debugger: ENABLED
```

```
9) Format Boot Media for BinFS
A) Format FIL (Erase All Blocks)
B) Format VFL (Format FIL + VFL Format)
C) Format FTL (Erase FTL Area + FTL Format)
E) Erase Physical Block 0
F) Make Initial Bad Block Information (Warning)
T) MLC Low level test
D) Download image now
L) LAUNCH existing Boot Media image
R) Read Configuration
U) DOWNLOAD image now(USB)
W) Write Configuration Right Now
S) SPI Keyboard test

Enter your selection: 6

Ethernet Boot Loader Configuration:

0) IP address: 0.0.0.0
1) Subnet mask: 255.255.255.0
2) DHCP: Disabled
3) Boot delay: 5 seconds
4) Reset to factory default configuration
5) Startup image: DOWNLOAD NEW
6) Program disk image into NAND Flash: Enabled
7) Program CS8900 MAC address (00:00:00:00:00:00)
8) Kernel Debugger: ENABLED
9) Format Boot Media for BinFS
A) Format FIL (Erase All Blocks)
B) Format VFL (Format FIL + VFL Format)
C) Format FTL (Erase FTL Area + FTL Format)
E) Erase Physical Block 0
F) Make Initial Bad Block Information (Warning)
T) MLC Low level test
D) Download image now
L) LAUNCH existing Boot Media image
```

```
R) Read Configuration
U) DOWNLOAD image now(USB)
W) Write Configuration Right Now
S) SPI Keyboard test

Enter your selection: u
+TOC_Write
[OK] Write 1 th Block Success
-TOC_Write
System ready!
Preparing for download...
INFO: *** Device Name 'SMDK24500' ***
Please send the Image through USB.

Download BIN file information:
-----------------------------------------------------
[0]: Base Address=0x0    Length=0x11000
-----------------------------------------------------
Stepldr image
VFL Already Formatted
Write Steploader (NBL1+NBL2) image to BootMedia
ImageLength = 69632 Byte
Start Page = 0, End Page = 33, Page Count = 34
Write Steploader image to BootMedia Success
INFO: Step loader image stored to Smart Media.    Please Reboot.    Halting...
CLKOUT0:MPLL in, CLKOUT1:RTC clock.


+------------------------------------------------+
| S3C2450X USB Downloader ver R0.1                         +
+------------------------------------------------+
mSDR
MPLL=400.00MHz, ARMCLK=400.00MHz (1:1:2), HCLK=133.33MHz, PCLK=66.67MHz

USB: IN_ENDPOINT:1 OUT_ENDPOINT:3
FORMAT: <ADDR(DATA):4>+<SIZE(n+10):4>+<DATA:n>+<CS:2>
```

```
NOTE: 1. Power off/on or press the reset button for 1 sec
              in order to get a valid USB device address.
         2. For additional menu, Press any key.

reset status register = 00000001
Power-on Reset
USB host is not connected yet.
USB host is connected. Waiting a download.

Now, Downloading [ADDRESS:30038000h,TOTAL:524298]
RECEIVED FILE SIZE:          524298
(30653.5KB/S,0.0S)
RECEIVE FILE DONE !!

Checksum O.K.
   C
Microsoft Windows CE Ethernet Bootloader Common Library Version 1.1 Built Dec   5 2006 10:42:36
Microsoft Windows CE Bootloader for the Samsung SMDK2450 Version 2.4 Built Dec   9 2006

InitDisplay
BP_Init
[FIL] ################
[FIL]   MID       = 0xec
[FIL]   DID       = 0xd3
[FIL]   HID[0] = 0x55
[FIL]   HID[1] = 0x25
[FIL]   HID[2] = 0x58
[FIL] ################
[FIL] ################
[FIL]   MID       = 0xec
[FIL]   DID       = 0xd3
[FIL]   HID[0] = 0x55
[FIL]   HID[1] = 0x25
[FIL]   HID[2] = 0x58
[FIL] ################
[FIL] ################################
```

```
[FIL]    FIL Global Information
[FIL]    BANKS_TOTAL = 4
[FIL]    BLOCKS_PER_BANK = 1024
[FIL]    SUPPORT_INTERLEAVING = 1
[FIL]    SUBLKS_TOTAL = 1024
[FIL]    PAGES_PER_SUBLK = 512
[FIL]    PAGES_PER_BANK = 131072
[FIL]    SECTORS_PER_PAGE = 8
[FIL]    SECTORS_PER_SUBLK = 4096
[FIL]    USER_SECTORS_TOTAL = 3780608
[FIL]    ADDRESS_CYCLE = 5
[FIL] ################################

[INFO] WMR_AREA_SIZE = 5
[INFO] SPECIAL_AREA_START = 5
[INFO] SPECIAL_AREA_SIZE = 25
[INFO] VFL_AREA_START = 30
[INFO] VFL_AREA_SIZE = 55
[INFO] VFL_INFO_SECTION_START = 30
[INFO] VFL_INFO_SECTION_SIZE = 4
[INFO] RESERVED_SECTION_START = 34
[INFO] RESERVED_SECTION_SIZE = 51
[INFO] FTL_INFO_SECTION_START = 85
[INFO] FTL_INFO_SECTION_SIZE = 6
[INFO] LOG_SECTION_SIZE = 7
[INFO] FREE_SECTION_START = 91
[INFO] FREE_SECTION_SIZE = 10
[INFO] FREE_LIST_SIZE = 3
[INFO] DATA_SECTION_START = 101
[INFO] DATA_SECTION_SIZE = 923
[INFO] FTL_AREA_START = 85
[INFO] FTL_AREA_SIZE = 939
[FTL:MSG] FIL_Init                          [OK]
[FTL:MSG] BUF_Init                          [OK]
[FTL:MSG] VFL_Init                          [OK]
[FTL:MSG] VFL_Open                          [OK]
```

```
wNUM_BLOCKS : 1024(0x400)
TOC_Read
-TOC_Read
Press [ENTER] to download image stored on boot media, or [SPACE] to enter boot monitor.

Initiating image download in 4 seconds.

Ethernet Boot Loader Configuration:

0) IP address: 0.0.0.0
1) Subnet mask: 255.255.255.0
2) DHCP: Disabled
3) Boot delay: 5 seconds
4) Reset to factory default configuration
5) Startup image: DOWNLOAD NEW
6) Program disk image into NAND Flash: Enabled
7) Program CS8900 MAC address (00:00:00:00:00:00)
8) Kernel Debugger: ENABLED
9) Format Boot Media for BinFS
A) Format FIL (Erase All Blocks)
B) Format VFL (Format FIL + VFL Format)
C) Format FTL (Erase FTL Area + FTL Format)
E) Erase Physical Block 0
F) Make Initial Bad Block Information (Warning)
T) MLC Low level test
D) Download image now
L) LAUNCH existing Boot Media image
R) Read Configuration
U) DOWNLOAD image now(USB)
W) Write Configuration Right Now
S) SPI Keyboard test

Enter your selection: u
+TOC_Write
[OK] Write 1 th Block Success
-TOC_Write
```

```
System ready!
Preparing for download...
INFO: *** Device Name 'SMDK24500' ***
Please send the Image through USB.

Download BIN file information:
-----------------------------------------------------
[0]: Base Address=0x8c038000    Length=0x6d6d8
-----------------------------------------------------
Eboot image
rom_offset=0x0.
ROMHDR at Address 8C038044h
Write Eboot image to BootMedia
ImageLength = 448216 Byte
Start Block = 2, End Block = 2, Block Count = 1
[OK] Write 2 th Block Success
Write Eboot image to BootMedia Success
INFO: Eboot image stored to Smart Media.    Please Reboot.    Halting...
CLKOUT0:MPLL in, CLKOUT1:RTC clock.



+------------------------------------------------+
| S3C2450X USB Downloader ver R0.1                         +
+------------------------------------------------+
mSDR
MPLL=400.00MHz, ARMCLK=400.00MHz (1:1:2), HCLK=133.33MHz, PCLK=66.67MHz

USB: IN_ENDPOINT:1 OUT_ENDPOINT:3
FORMAT: <ADDR(DATA):4>+<SIZE(n+10):4>+<DATA:n>+<CS:2>
NOTE: 1. Power off/on or press the reset button for 1 sec
            in order to get a valid USB device address.
      2. For additional menu, Press any key.

reset status register = 00000001
Power-on Reset
USB host is not connected yet.
USB host is connected. Waiting a download.

Now, Downloading [ADDRESS:30038000h,TOTAL:524298]
RECEIVED FILE SIZE:         524298
(30727.2KB/S,0.0S)
RECEIVE FILE DONE !!

Checksum O.K.

Microsoft Windows CE Ethernet Bootloader Common Library Version 1.1 Built Dec   5 2006 10:42:36
Microsoft Windows CE Bootloader for the Samsung SMDK2450 Version 2.4 Built Dec   9 2006

InitDisplay
BP_Init
[FIL] ################
[FIL]   MID      = 0xec
[FIL]   DID      = 0xd3
[FIL]   HID[0] = 0x55
[FIL]   HID[1] = 0x25
[FIL]   HID[2] = 0x58
[FIL] ################
[FIL] ################
[FIL]   MID      = 0xec
[FIL]   DID      = 0xd3
[FIL]   HID[0] = 0x55
[FIL]   HID[1] = 0x25
[FIL]   HID[2] = 0x58
[FIL] ################
[FIL] ##############################
[FIL]   FIL Global Information
[FIL]   BANKS_TOTAL = 4
[FIL]   BLOCKS_PER_BANK = 1024
[FIL]   SUPPORT_INTERLEAVING = 1
[FIL]   SUBLKS_TOTAL = 1024
[FIL]   PAGES_PER_SUBLK = 512
[FIL]   PAGES_PER_BANK = 131072
```

```
[FIL]   SECTORS_PER_PAGE = 8
[FIL]   SECTORS_PER_SUBLK = 4096
[FIL]   USER_SECTORS_TOTAL = 3780608
[FIL]   ADDRESS_CYCLE = 5
[FIL] ################################

[INFO] WMR_AREA_SIZE = 5
[INFO] SPECIAL_AREA_START = 5
[INFO] SPECIAL_AREA_SIZE = 25
[INFO] VFL_AREA_START = 30
[INFO] VFL_AREA_SIZE = 55
[INFO] VFL_INFO_SECTION_START = 30
[INFO] VFL_INFO_SECTION_SIZE = 4
[INFO] RESERVED_SECTION_START = 34
[INFO] RESERVED_SECTION_SIZE = 51
[INFO] FTL_INFO_SECTION_START = 85
[INFO] FTL_INFO_SECTION_SIZE = 6
[INFO] LOG_SECTION_SIZE = 7
[INFO] FREE_SECTION_START = 91
[INFO] FREE_SECTION_SIZE = 10
[INFO] FREE_LIST_SIZE = 3
[INFO] DATA_SECTION_START = 101
[INFO] DATA_SECTION_SIZE = 923
[INFO] FTL_AREA_START = 85
[INFO] FTL_AREA_SIZE = 939
[FTL:MSG] FIL_Init                      [OK]
[FTL:MSG] BUF_Init                      [OK]
[FTL:MSG] VFL_Init                      [OK]
[FTL:MSG] VFL_Open                      [OK]
wNUM_BLOCKS : 1024(x400)
TOC_Read
-TOC_Read
Press [ENTER] to download image stored on boot media, or [SPACE] to enter boot monitor.

Initiating image download in 4 seconds.
```

```
Ethernet Boot Loader Configuration:


0) IP address: 0.0.0.0
1) Subnet mask: 255.255.255.0
2) DHCP: Disabled
3) Boot delay: 5 seconds
4) Reset to factory default configuration
5) Startup image: DOWNLOAD NEW
6) Program disk image into NAND Flash: Enabled
7) Program CS8900 MAC address (00:00:00:00:00:00)
8) Kernel Debugger: ENABLED
9) Format Boot Media for BinFS
A) Format FIL (Erase All Blocks)
B) Format VFL (Format FIL + VFL Format)
C) Format FTL (Erase FTL Area + FTL Format)
E) Erase Physical Block 0
F) Make Initial Bad Block Information (Warning)
T) MLC Low level test
D) Download image now
L) LAUNCH existing Boot Media image
R) Read Configuration
U) DOWNLOAD image now(USB)
W) Write Configuration Right Now
S) SPI Keyboard test


Enter your selection: u
+TOC_Write
[OK] Write 1 th Block Success
-TOC_Write
System ready!
Preparing for download...
INFO: *** Device Name 'SMDK24500' ***
Please send the Image through USB.


Download BIN file information:
-----------------------------------------------------
```

```
[0]: Base Address=0x80040000    Length=0x16116c
[1]: Base Address=0x80200000    Length=0xff0fb8
[2]: Base Address=0x81b2c000    Length=0x528
----------------------------------------------------
RAM image
rom_offset=0x0.
RAM image
rom_offset=0x0.
RAM image
ROMHDR at Address 80040044h
+WriteOSImageToBootMedia: g_dwTocEntry =1, ImageStart: 0x80040000, ImageLength: 0x16116c,
LaunchAddr:0x800708c4
g_dwMBRSectorNum = 0x5000
Erase Block from 0x5, to 0x1d
Found the Chain region: StartAddress: 0x81B2C000; Length: 0x528
Writing multi-regions
BINFSPartMaxLength[0]: 0x4000, TtlBINFSPartLength: 0x4000
dwMaxRegionLength[2]: 0x4000
BINFSPartMaxLength[1]: 0x1c0000, TtlBINFSPartLength: 0x1c4000
dwMaxRegionLength[0]: 0x1c0000
BINFSPartMaxLength[2]: 0x1200000, TtlBINFSPartLength: 0x13c4000
dwMaxRegionLength[1]: 0x1200000
dwBlock = 5
OpenPartition: dwStartSector = 0x5000.
OpenPartition: dwNumSectors = 0x10000.
OpenPartition: dwPartType = 0x21.
OpenPartition: fActive = 0x1.
OpenPartition: dwCreationFlags = 0x2.
IsValidMBR: MBR sector = 0x5000
.OpenPartition: Invalid MBR.   Formatting flash.
Enter LowLevelFormat [0x5, 0x14].
BP_LowLevelFormat: // Erase all the flash blocks.
BP_LowLevelFormat: // Erase all the flash blocks.-End
WriteMBR: MBR block = 0x5.
WriteBlock: dwMBRBlockNum = 0x5.
Done.
```

```
CreatePartition: Enter CreatePartition for 0x21.
CreatePartition: Start = 0x1000, Num = 0x10000.
WriteMBR: MBR block = 0x5.
WriteBlock: dwMBRBlockNum = 0x5.
nCount = 0
BP_SetDataPointer at 0xc00000
BP_WriteData: Start = 0xc00000, Length = 0x16116c.
BP_WriteData: dwSectorAddr = 0x6000, dwOffsetSector = 0x0.
BP_WriteData: dwNumSects = 0xb08
BP_WriteData: dwNumExtraBytes = 0x16c
BP_WriteData: dwSectorAddr = 0x6000
BP_WriteData: dwNumSects = 0xb08
BP_WriteData: dwSectorAddr = 0x6b08
BP_WriteData: pbBuffer = 0xac961000
Updateded TOC!
nCount = 1
BP_SetDataPointer at 0xdc0000
BP_WriteData: Start = 0xdc0000, Length = 0xff0fb8.
BP_WriteData: dwSectorAddr = 0x6e00, dwOffsetSector = 0x0.
BP_WriteData: dwNumSects = 0x7f80
BP_WriteData: dwNumExtraBytes = 0xfb8
BP_WriteData: dwSectorAddr = 0x6e00
BP_WriteData: dwNumSects = 0x7f80
BP_WriteData: dwSectorAddr = 0xed80
BP_WriteData: pbBuffer = 0xad9b0000
nCount = 2
BP_SetDataPointer at 0x1fc0000
BP_WriteData: Start = 0x1fc0000, Length = 0x528.
BP_WriteData: dwSectorAddr = 0xfe00, dwOffsetSector = 0x0.
BP_WriteData: dwNumSects = 0x0
BP_WriteData: dwNumExtraBytes = 0x528
BP_WriteData: dwSectorAddr = 0xfe00
BP_WriteData: dwNumSects = 0x0
BP_WriteData: dwSectorAddr = 0xfe00
BP_WriteData: pbBuffer = 0xae2ec000
```

```
Written Chain Region to the Flash
LoadAddress = 0x81B2C000; FlashAddress = 0x0; Length = 0x20
 memcpy : g_pTOC->chainInfo.dwLoadAddress = 0x81B2C000; dwRegionStart = 0xAE2EC000;
dwRegionLength = 0x528
-WriteOSImageToBootMedia
+TOC_Write
[OK] Write 1 th Block Success
-TOC_Write
TOC {
dwSignature: 0x434F544E
BootCfg {
  ConfigFlags: 0x2820
  BootDelay: 0x5
  ImageIndex: 1
  IP: 0.0.0.0
  MAC Address: 00:00:00:00:00:00
  Port: 0.0.0.0
  SubnetMask: 255.255.255.0
}
ID[0] {
  dwVersion: 0x20004
  dwSignature: 0x45424F54
  String: 'eboot.nb0'
  dwImageType: 0x2
  dwTtlSectors: 0x400
  dwLoadAddress: 0x8C038000
  dwJumpAddress: 0x8C038000
  dwStoreOffset: 0x0
  sgList[0].dwSector: 0x2000
  sgList[0].dwLength: 0x400
}
ID[1] {
  dwVersion: 0x1
  dwSignature: 0x43465348
  String: ''
  dwImageType: 0x2
```

```
    dwTtlSectors: 0xB09
    dwLoadAddress: 0x80040000
    dwJumpAddress: 0x800708C4
    dwStoreOffset: 0x0
    sgList[0].dwSector: 0x0
    sgList[0].dwLength: 0xB09
}
chainInfo.dwLoadAddress: 0X81B2C000
chainInfo.dwFlashAddress: 0X00000000
chainInfo.dwLength: 0X00000020
}
RAM image
RAM image
Read OS image to BootMedia
ImageLength = 1446400 Byte
Start Page = 3072, End Page = 3425, Page Count = 354
..
Read OS image to BootMedia Success
waitforconnect
INFO: using TOC[1] dwJumpAddress: 0x800708c4
INFO: OEMLaunch: Jumping to Physical Address 0x33E708C4h (Virtual Address 0x800708C4h)...



Windows CE Kernel for ARM (Thumb Enabled) Built on Jun 24 2004 at 18:25:00
ProcessorType=0920    Revision=0
sp_abt=ffff5000 sp_irq=ffff2800 sp_undef=ffffc800 OEMAddressTable = 80070b88
DCache: 8 sets, 64 ways, 32 line size, 16384 size
ICache: 8 sets, 64 ways, 32 line size, 16384 size
FCLK:534000000, HCLK:133500000, PCLK:66750000
[MSMDD: IN] ++MSMDDInit()
[MSMDD:INF]    pdwXIPLoc=0x81b2c000, dwDontUseChain=0x1
[MSMDD:INF]    XIP chain location not fixed up
[MSMDD:OUT] --MSMDDInit()
Sp=ffffc7cc
[OEMIO:INF]    + IOCTL_HAL_POSTINIT
[OEMIO:INF]    - IOCTL_HAL_POSTINIT
```

```
+OALIoCtlHalInitRTC(...)
+OEMSetRealTime(2005/5/27 12:0:0.000)
-OEMSetRealTime(rc=1)
OEMIoControl: Unsupported Code 0x101008c - device 0x0101 func 35
[HALWP:INF]    nCtrlCode = PM_HAL_FIL_INIT
[FIL] ################
[FIL]    MID        = 0xec
[FIL]    DID        = 0xd3
[FIL]    HID[0] = 0x55
[FIL]    HID[1] = 0x25
[FIL]    HID[2] = 0x58
[FIL] ################
[FIL] ################
[FIL]    MID        = 0xec
[FIL]    DID        = 0xd3
[FIL]    HID[0] = 0x55
[FIL]    HID[1] = 0x25
[FIL]    HID[2] = 0x58
[FIL] ################
[FIL] ################################
[FIL]    FIL Global Information
[FIL]    BANKS_TOTAL = 4
[FIL]    BLOCKS_PER_BANK = 1024
[FIL]    SUPPORT_INTERLEAVING = 1
[FIL]    SUBLKS_TOTAL = 1024
[FIL]    PAGES_PER_SUBLK = 512
[FIL]    PAGES_PER_BANK = 131072
[FIL]    SECTORS_PER_PAGE = 8
[FIL]    SECTORS_PER_SUBLK = 4096
[FIL]    USER_SECTORS_TOTAL = 3780608
[FIL]    ADDRESS_CYCLE = 5
[FIL] ################################

[INFO] WMR_AREA_SIZE = 5
[INFO] SPECIAL_AREA_START = 5
[INFO] SPECIAL_AREA_SIZE = 25
```

```
[INFO] VFL_AREA_START = 30
[INFO] VFL_AREA_SIZE = 55
[INFO] VFL_INFO_SECTION_START = 30
[INFO] VFL_INFO_SECTION_SIZE = 4
[INFO] RESERVED_SECTION_START = 34
[INFO] RESERVED_SECTION_SIZE = 51
[INFO] FTL_INFO_SECTION_START = 85
[INFO] FTL_INFO_SECTION_SIZE = 6
[INFO] LOG_SECTION_SIZE = 7
[INFO] FREE_SECTION_START = 91
[INFO] FREE_SECTION_SIZE = 10
[INFO] FREE_LIST_SIZE = 3
[INFO] DATA_SECTION_START = 101
[INFO] DATA_SECTION_SIZE = 923
[INFO] FTL_AREA_START = 85
[INFO] FTL_AREA_SIZE = 939
[HALWP:INF]    nCtrlCode = PM_HAL_VFL_INIT
[HALWP:INF]    nCtrlCode = PM_HAL_VFL_OPEN
[BIBDRV:INF]    stNandInfo.dwPagesPerSuBlk          = 0x200
[BIBDRV:INF]    stNandInfo.dwSectorsPerPage         = 0x8
[BIBDRV:INF]    stNandInfo.dwSpecialAreaSize        = 0x19
[BIBDRV:INF]    stNandInfo.dwSpecialAreaStart      = 0x5
[BIBDRV:INF]    MBR MAGIC CODE (First): 0xe9,0xfd,0xff
[BIBDRV:INF]    MBR MAGIC CODE (Last) : 0x55,0xaa
[BIBDRV:INF]    Part_BootInd          = 0x0
[BIBDRV:INF]    Part_FirstHead        = 0x1
[BIBDRV:INF]    Part_FirstSector    = 0x1
[BIBDRV:INF]    Part_FirstTrack      = 0x0
[BIBDRV:INF]    Part_FileSystem      = 0x21
[BIBDRV:INF]    Part_LastHead          = 0x10
[BIBDRV:INF]    Part_LastSector      = 0x0
[BIBDRV:INF]    Part_LastTrack        = 0x0
[BIBDRV:INF]    Part_StartSector    = 4096
[BIBDRV:INF]    Part_TotalSectors = 65536
BIBDRV_PS:GetProfileName - Profile = SMFlash, length = 16
[BIBDRV:MSG]    DSK_IOControl unkonwn code(71c24)
```

```
OEMIoControl: Unsupported Code 0x1010104 - device 0x0101 func 65
OEMIoControl: Unsupported Code 0x10100d0 - device 0x0101 func 52
OEMIoControl: Unsupported Code 0x10100f8 - device 0x0101 func 62
DEBUG: CreateSerialObject 2
WARNING: CReg2440Uart::CReg2440Uart failed to obtain processor frequency - using default value
(66750000).
PWR: Process Attach
>PWR_Init(602ED64)
HW_Init : HW_InitRegisters
EXTINT0=0x0000000A
HW_Init : CreateEvent
HW_Init : IOCTL_HAL_REQUEST_SYSINTR
INFO: PwrButton: Mapped Irq 0x0 to SysIntr 0x14.
HW_Init : CreateThread
HW_Init : CeSetThreadPriority

PWR_IST: pPWR->State = 0x1
HW_Init : Done
<PWR_Init:0x37d00
>PWR_Open(0x37d00, 0x0, 0x3)
PCF: HW_Open
<PWR_Open:1
>PWR_IOControl(0x321000, 0x0, 0, 0x6037a48)
<PWR_IOControl:1
>PWR_Open(0x37d00, 0x0, 0x3)
PCF: HW_Open
<PWR_Open:2
PWR_Close(0x37d00)
PCF: HW_Close
v_pIOPregs->GPGDAT= 0x2
CLK:66750000, BaudRate:9600, UBRDIV:433, UDIVSLOTn:9
[OND:INF]   InitializeCriticalSection(&v_DiskCrit)
[OND:OUT] ++OpenDriverKey()
[OND:     ] OpenDriverKey RegOpenKeyEx(HLM\Drivers\Active\11) returned 0!!!
[OND:     ] OpenDriverKey - RegQueryValueEx(Key) returned 0
[OND:OUT] --OpenDriverKey()
```

```
[OND:MSG]   GetBmlId - BmlPartitionId = 8
[OND:MSG]   GetBmlId - BmlPartitionId = 8
[OND:MSG]   GetBmlId - BmlVolumeId = 0
[FTL:ERR] Can not find context!
[FTL:ERR]   there is error on _LoadFTLCxt!
[FTLP:ERR]   FTL_Open() failure. ERR Code=80010000
[OND:ERR] FTL_Open occurs Critical Error
[OND:OUT] ++OpenDriverKey()
[OND:     ] OpenDriverKey RegOpenKeyEx(HLM\Drivers\Active\11) returned 0!!!
[OND:     ] OpenDriverKey - RegQueryValueEx(Key) returned 0
[OND:OUT] --OpenDriverKey()
[OND:MSG]   GetBmlId - BmlPartitionId = 8
[OND:MSG]   GetBmlId - BmlPartitionId = 8
[OND:MSG]   GetBmlId - BmlVolumeId = 0
[OND:OUT] ++GetDeviceInfo()
[OND:OUT] ++OpenDriverKey()
[OND:     ] OpenDriverKey RegOpenKeyEx(HLM\Drivers\Active\11) returned 0!!!
[OND:     ] OpenDriverKey - RegQueryValueEx(Key) returned 0
[OND:OUT] --OpenDriverKey()
[OND:     ] GetProfileName - Profile = PocketMory, length = 22
[OND:     ] Order = 0, length = 4
[OND:OUT] --GetDeviceInfo()
[OND:MSG] ++GetStorageID
[OND:MSG] --GetStorageID
[OND:MSG]   +-------------------------+
[OND:MSG]   | IOCTL_DISK_FORMAT_MEDIA |
[OND:MSG]   +-------------------------+
<log P1="100" P2="FORMAT" />
[OND:INF]   FTL_Close() success
[WMR      ] ++WMR_Format_FTL()
[WMR:INF] WMR_Format_FTL() : All Virtual Block Erased (VFL) !!!
[WMR:INF] WMR_Format_FTL() : FTL_Init() Success
[WMR:INF] WMR_Format_FTL() : FTL_Format() Success
[WMR      ] --WMR_Format_FTL()
[OND:INF]   FTL_Format() success
[OND:INF]   FTL_Open() success
```

```
[OND:OUT] ++GetDeviceInfo()
[OND:OUT] ++OpenDriverKey()
[OND:    ] OpenDriverKey RegOpenKeyEx(HLM\Drivers\Active\11) returned 0!!!
[OND:    ] OpenDriverKey - RegQueryValueEx(Key) returned 0
[OND:OUT] --OpenDriverKey()
[OND:    ] GetProfileName - Profile = PocketMory, length = 22
[OND:    ] Order = 0, length = 4
[OND:OUT] --GetDeviceInfo()
SDHCD:SetClockRate() - Clock rate set to 260742 Hz
EXTINT0=0x000000FA
SDHCSetRate - Clock Control Reg = 201
SDHCSetRate - Actual clock rate = 24000000
CSDHCSlotBase::Start
USB Serial Function Class Enabled : Serial_Class
UFN_DETACH_3
-----------------------SDHCControllerIst - Card is Removed! 0x005B34D4
SC2450UsbFn!HandleUSBBusIrq: Suspend
SC2450UsbFn!HandleUSBBusIrq: Resume
SC2450UsbFn!HandleUSBBusIrq: Resume_Attach
OEMIoControl: Unsupported Code 0x10100fc - device 0x0101 func 63
CLK:66750000, BaudRate:9600, UBRDIV:433, UDIVSLOTn:9
CLK:66750000, BaudRate:9600, UBRDIV:433, UDIVSLOTn:9
+OEMSetAlarmTime(4/2/2006 2:0:0.000)
HIGH Speed
SC2450UsbFn!HandleUSBBusIrq: Reset
SC2450UsbFn!HandleUSBBusIrq: Reset
HIGH Speed
+OEMSetAlarmTime(4/2/2006 2:0:0.000)
```

6. PocketMory FIL layer

6.1. Main area and Spare area layout.

- Page Layout for 2Kbytes/Page NAND Flash Device:

0~511	512~1023	1024~1535	1536~2047	2048~2111
Sector 0 512B	Sector 1 512B	Sector 2 512B	Sector 3 512B	Spare Area 64B

Table 6-1. MLC Page Layout 2K page (2048+64 byte)

- Spare Area Layout for 2Kbyte/Page NAND Flash Device:

0(1)	1(1)	2~3(2)	4~15(12)	16~23(8)	24~39(8)	32~39(8)	40~47(8)	48~55(8)	56~63(8)
Bad Mark	Clean Mark	Reserved	Spare Context	Sector0 ECC	Sector1 ECC	Sector2 ECC	Sector3 ECC	Spare ECC	Spare ECC copy

Table 6-2. MLC Spare Area layout 2K page (64 Byte)

- Page Layout for 4Kbytes/Page NAND Flash Device:

0~511	512~1023	1024~1535	1536~2047	2048~2559	2560~3071	3072~3583	3584~4095	4096~4224
Sector 0 512B	Sector 1 512B	Sector 2 512B	Sector 3 512B	Sector 4 512B	Sector 5 512B	Sector 6 512B	Sector 7 512B	Spare Area 128B

Table 6-3. MLC Page Layout 4K page (4096+128 byte)

- Spare Area Layout for 4Kbyte/Page NAND Flash Device:

0 (1)	1 (1)	2~3 (2)	4~23 (20)	24~31 (8)	32~39 (8)	40~47 (8)	48~55 (8)	56~63 (8)	64~71 (8)	72~79 (8)	80~87 (8)	88~95 (8)	96~103 (8)
Bad Mark	Clean Mark	Reserved	Spare Context	Sector0 ECC	Sector1 ECC	Sector2 ECC	Sector3 ECC	Sector4 ECC	Sector5 ECC	Sector6 ECC	Sector7 ECC	Spare ECC	Spare ECC copy

Table 6-4. MLC Spare Area layout 4K page (128 Byte)

Bad Mark : If this value is not 0xFF then this block is Initial or Run-time Bad Block.

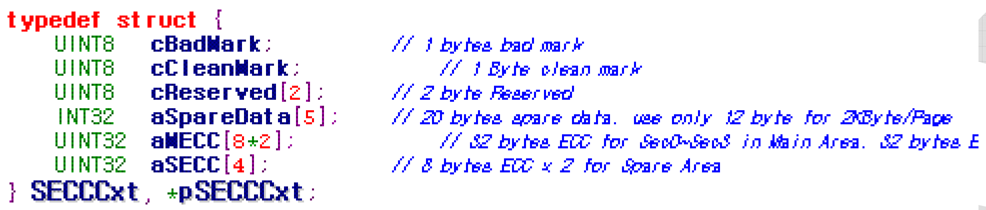
Clean Mark : If this page is programmed with any data (include all 0xFF), this value have to mark to 0x00.

Spare Context : VFL context value used by Upper layer.

Sector X ECC : 4-bit ECC parity code for Main area 512 bytes.

Spare ECC : 4-bit ECC parity code for Spare Context 12 bytes.

Spare ECC copy : Back up ECC value.



```
typedef struct {
    UINT8    cBadMark;              // 1 bytes bad mark
    UINT8    cCleanMark;                // 1 Byte clean mark
    UINT8    cReserved[2];          // 2 byte Reserved
     INT32   aSpareData[5];         // 20 bytes spare data. use only 12 byte for 2KByte/Page
    UINT32   aMECC[8+2];                // 32 bytes ECC for Sec0~Sec3 in Main Area. 32 bytes E
    UINT32   aSECC[4];              // 8 bytes ECC x 2 for Spare Area
} SECCCxt, *pSECCCxt;
```

Figure 6-1. FIL Spare Data Structure

6.2. Erase, Write, Read procedure.

- FIL Erase:

1) Erase command

2) Write Row-address

3) Erase Confirm command

4) Wait for R/B pin Ready

5) Read Status command

6) Read status

The step 4), 5), 6) is separated with NAND_Sync() function. This function is to support 2-plane programming, internal interleaving function to improve Read/Write/Erase performance. Multi-way operation is not implemented in this version.

- FIL Write:

1) Program command (sector 0) or Random Data Input command (sector 2,3,4)
2) Write address for Sector
3) ECC engine initialize for encoding
4) 512 byte Sector data write
5) Wait for ECC encoding finished
6) Keep ECC parity code in Memory
7) Loop 1)~6) for Sector 2,3,4
8) Random Data Input command
9) Write address for Spare area
10) Write 1 byte Bad mark (from Upper Layer)

11) Write 1 byte Clean mark (0x00)
12) Write 2 byte Reserved Data (0xFF)
13) ECC engine initialize for encoding
14) Write 12 byte Spare Context (from Upper layer)
15) Write 8 byte Sector 0 ECC parity code (7 byte is meaningful)
16) Write 8 byte Sector 1 ECC parity code (7 byte is meaningful)
17) Write 8 byte Sector 2 ECC parity code (7 byte is meaningful)
18) Write 8 byte Sector 3 ECC parity code (7 byte is meaningful)
19) Write 8 byte Sector 4 ECC parity code (7 byte is meaningful) in case 4Kbyte/Page
20) Write 8 byte Sector 5 ECC parity code (7 byte is meaningful) in case 4Kbyte/Page
21) Write 8 byte Sector 6 ECC parity code (7 byte is meaningful) in case 4Kbyte/Page
22) Write 8 byte Sector 7 ECC parity code (7 byte is meaningful) in case 4Kbyte/Page
23) **CE pin set to High (CE don't care state start)**
24) **Write 468 byte Dummy Data (0xFF)**
25) **CE pin set to Low (CE don't care state end)**
26) Wait for ECC encoding finished
27) Write 8 byte Spare ECC parity code (7 byte is meaningful)
28) Write 8 byte Spare ECC parity code again.
29) Program Confirm command
30) FIL Sync()

From 23) to 25) is work-around for generate 4-bit ECC of spare area.

- FIL Read:

1) Read command
2) Write address for Spare area
3) Read Confirm command
4) Read 1 byte Bad Mark
5) Read 1 byte Clean Mark
6) Read 2 byte Reserved Data
7) ECC engine initialize for decoding
8) Read 12 byte Spare Context
9) Read 8 byte Sector 0 ECC parity code (7 byte is meaningful)
10) Read 8 byte Sector 1 ECC parity code (7 byte is meaningful)
11) Read 8 byte Sector 2 ECC parity code (7 byte is meaningful)
12) Read 8 byte Sector 3 ECC parity code (7 byte is meaningful)

13) Read 8 byte Sector 4 ECC parity code (7 byte is meaningful) in case 4Kbyte/Page
14) Read 8 byte Sector 5 ECC parity code (7 byte is meaningful) in case 4Kbyte/Page
15) Read 8 byte Sector 6 ECC parity code (7 byte is meaningful) in case 4Kbyte/Page
16) Read 8 byte Sector 7 ECC parity code (7 byte is meaningful) in case 4Kbyte/Page
17) CE pin set to High (CE don't care state start)
18) Write 468 byte Dummy Data (0xFF)
19) CE pin set to Low (CE don't care state end)
20) Read 8 byte Spare ECC parity code from NAND flash device (7 byte is meaningful)
21) ECC decoding starts automatically
22) Wait for ECC decoding finished (detection and correction)
23) If there is uncorrectable Error, repeat from 7) to 15) and read 8 byte more for compare with next 8 byte spare ECC copy code.
24) Write address for Main area.
25) Read 512 byte.
26) CE pin set to High (CE don't care state start)
27) Write 7 byte ECC parity code which read from 9) to 12)
28) CE pin set to Low (CE don't care state end)
29) Read Next Sector with same method.

From 17) to 19) is work-around for detect 4-bit ECC of spare area.

7. Limitation of current device driver

1. We test the BSP with below nand flash part number.
   A. K9G8G08/K9G4G08 (Mono Die, support 2 plane programming)
   B. K9L8G08 (DDP, support 2 plane programming, internal interleaving)
   C. K9LAG08 (DDP, support 2 plane programming, internal interleaving)
   D. K9HAG08 (QDP, support 2 plane programming, internal interleaving, 2CE programming)
   E. K9GAG08 (Mono, support 2 plane programming, 4KByte/Page)
   F. K9LBG08 (DDP, support 2 plane programming, internal interleaving, 4KByte/Page)
   G. K9HCG08 (QDP, support 2 plane programming, internal interleaving, 2CE programming, 4K Byte/Page)