PocketStoreII v1.5.1

Quick Installation Guide

April 2007 , Version 2.4

SAMSUNG ELECTRONICS

Copyright notice

Trademarks

PocketStoreII is trademark of Flash Software Group, Samsung Electronics Co., Ltd in Korea and other countries

Restrictions on Use and Transfer

All software and documents of **PocketStoreII** are commercial software.
Therefore, you must install, use, redistribute, modify and purchase only in accordance with the terms of the license agreement you entered into with Flash Software Group, Samsung Electronics Co., Ltd.

All advertising materials mentioning features or use of this software must display the following acknowledgement:
"This product includes **PocketStoreII** written by Flash Software Group, Samsung Electronics Co., Ltd.

Preface

SEC-FSG PS2 IG-001

This document is a quick installation guide for PocketStoreII v1.5.1 for Windows Mobile 6.0 developed by Flash Software Group, Samsung Electronics.

■ Purpose

This document is intended for Project Manager, Project Leader, and Application Programmers who are installing PocketStoreII for the first time. All users must have experience with Microsoft Window Mobile software.

■ Scope

This document provides the requirements and instructions for installing, configuring and building PocketStoreII on target.

■ Definitions and Acronyms

Acronyms	Definition
BSP	Board Support Package
EBOOT	Ethernet BOOT / Ethernet Boot-loader
IPL	Initial Program Loader
IMGFS	Image File System
FATFS	FAT File System
Block	OneNAND is partitioned into fixed-sized blocks. A block is 128K bytes.
FTL	Flash Translation Layer A software module which maps between logical addresses and physical addresses when accessing OneNAND memory
HAL	Hardware Abstraction Layer
OTP	One Time Programmable
ONW	OneNAND Writer
XSR	eXtended Sector Remapper

■ Related Documents

- XSR v1.5.1_Part1.Sector Translation Layer Programmer's guide
- XSR v1.5.1_Part2.Block Management Layer Programmer's guide

■ History

Version	Date	Comment	Author
2.0	2006-06-23	Initial revision	Hyunsoo Cho
2.2	2007-04-13	Modified	Heejun Lim
2.3	2007-04-26	Modified	Jongsoon Park
2.4	2007-04-27	Review	Thomas

1. PocketStoreII Installation

PocketStoreII is a flash translation layer, made by Samsung Electronics, for OneNAND device that uses Windows Mobile as operating system. This section provides the prerequisites for installing the PocketStoreII v1.5.1 on Windows Mobile 6.0

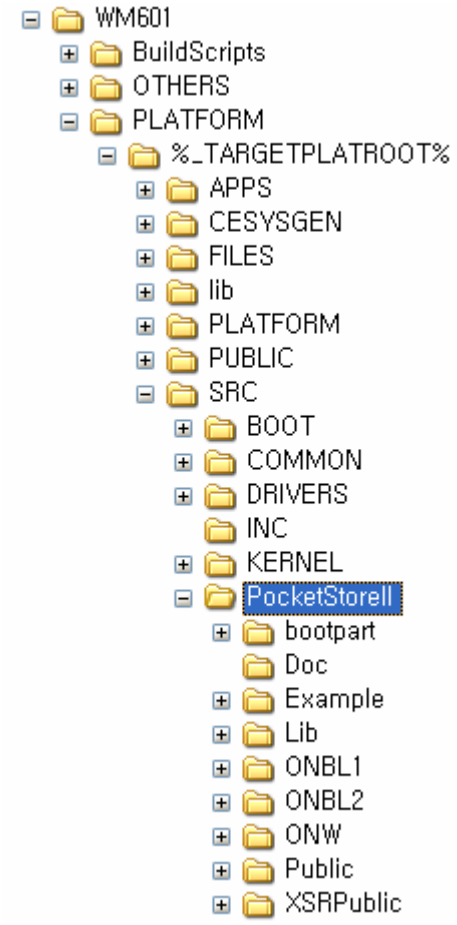
1.1. Prerequisites

- ☐ Target H/W with Samsung OneNAND
- ☐ Microsoft Windows Mobile AKU6.x
- ☐ Samsung PocketStoreII v1.5.1 for Windows Mobile Package

1.2. Install PocketStoreII Package

Unzip PocketStoreII package file.
(PocketStoreII_v1.5.1_pX_%_TARGETPLATROOT%.zip) into your BSP.

Recommended directory location is %_TARGETPLATROOT%\Src\PocketStoreII.



We also provide example of Eboot source in %_TARGETPLATROOT% \Platform\Src\PocketStoreII\Example\directory. But those sources are just a reference sample, so you may need to modify Eboot for your own target H/W.

1.3. PocketStoreII Configuration

1.3.1. Modify build configure files

Modify %_TARGETPLATROOT%\src\PocketStoreII\env.bat for your BSP
You should set these environment variables for building PocketStoreII

```
set BSP_POCKETSTORE=1
set XSR_PAM=<CPU_TYPE>
```

Add environment variables into %_TARGETPLATROOT%\%_TGTPLAT%.bat

```
…
@REM Include PocketStoreII configuration
call %_TARGETPLATROOT% \Platform\src\PocketStoreII\env.bat
…
```

Add PocketStoreII directory into
%_TARGETPLATROOT% \Platform\src\DIRS

```
DIRS =  ….              \
        PocketStoreII   \
        …
```

1.3.2. Modify OS configuration files

Add PocketStoreII DLL into %_TARGETPLATROOT%\files\platform.bib

```
MODULES

; Name                  Path                                Memory Type
; -------------         ---------------------------------   -----------
IF BSP_POCKETSTORE

ONDisk.dll              $(_FLATRELEASEDIR)\ONDisk.dll          NK    SH

ENDIF BSP_POCKETSTORE
…
```

Modify %_TARGETPLATROOT%\files\platform.reg

```
…
IF BSP_POCKETSTORE

#include "$(_TARGETPLATROOT)\Platform\src\PocketStoreII\PocketStoreII.reg"
```

```
ENDIF BSP_POCKETSTORE
…
```

You may refer to the
%_TARGETPLATROOT%\ Src\PocketStoreII\PocketStoreII.reg file.

Modify %_TARGETPLATROOT%\files\memory.cfg.xml

```
…
<HARDWARE>
        <RAM START="0x84000000" LENGTH="0x02000000"></RAM>
        <NAND SECTORSIZE="0x800" BLOCKSIZE="0x1F800"
LENGTH="0x01A16000" ID="FLASH" />
    </HARDWARE>
…
```

Memory.cfg.xml describes the platform's RAM and Flash memory layout. The layout will be used when disk image tool creates the OS image.
Following table shows information about configuration attributes that used in above sample.

Attribute name	**value**	**Description**
SECTORSIZE	0x800	Sector size in bytes
BLOCKSIZE	0x1F800	Block size of NAND flash in bytes. It is fixed in PocketStoreII.
LENGTH	0x1A16000	Size of OS image in bytes. It must be even multiple value of BLOCKSIZE
ID	FLASH	Unique, non-empty identifier

You may change the length if you want to change the image size or build the BSP in debug configuration.
USERSTORE element, showed bellow example, should be removed from memory.cfg.xml.

```
…
        <IMGFS ID="OS" STORAGE_ID="MBStrata"
FREE_SPACE_BUFFER="0x40000"/>
        <USERSTORE STORAGE_ID="MBStrata" ID="Storage"
PART_TYPE="0x04" />
    </PARTITIONS>
…
```

Add PocketStoreII code into
%_TARGETPLATROOT%\files\wpc\oem.cpm.csv or
%_TARGETPLATROOT%\files\smartfon\oem.cpm.csv

```
…
nk.exe,OEMXIPKERNEL
ONDisk.dll,OEMXIPKERNEL
…
```

1.3.3. Modify OAL(Os Adaptation Layer) & KERNEL

Add PocketStoreII handler into
%_TARGETPLATROOT%\Platform\src\inc\ioctl_tab.h

```
…
{ IOCTL_HAL_POSTINIT, 0,    OALIoctlPostInit                    },
{ IOCTL_POCKETSTOREII_CMD , 0 , OALIoCtlPocketStoreCMD },
…
```

Add PocketStoreII handler into
%_TARGETPLATROOT%\Platform\src\kernel\oal\ioctl.c

```
…
#include <XSR.h>
#include <HALWrapper.h>
…
CRITICAL_SECTION csPocketStoreBML;

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;
}
```

Add PocketStoreII libraries into
%_TARGETPLATROOT%\Platform\src\kernel\oal\sources and

```
…
SOURCELIBS= \
```

```
$(_TARGETPLATROOT)\src\PocketStoreII\Lib\$(_CPUINDPATH)\PSIIHALWrapper.
lib        \
$(_TARGETPLATROOT)\src\PocketStoreII\Lib\$(_CPUINDPATH)\PSIIGetUID.lib
           \
$(_TARGETPLATROOT)\src\PocketStoreII\Lib\$(_CPUINDPATH)\BML.lib
                   \
!IF ("$(IMGULDR)"=="1")
$(_TARGETPLATROOT)\src\PocketStoreII\Lib\$(_CPUINDPATH)\OSLessOAM_RW
.lib               \
!ELSE
$(_TARGETPLATROOT)\src\PocketStoreII\Lib\$(_CPUINDPATH)\OSLessOAM_KE
R.lib                          \
!ENDIF
$(_TARGETPLATROOT)\src\PocketStoreII\Lib\$(_CPUINDPATH)\$(XSR_PAM)PA
M.lib              \
$(_TARGETPLATROOT)\src\PocketStoreII\Lib\$(_CPUINDPATH)\ONLD.lib
```

1.3.4. Modify Config.h

%_TARGETPLATROOT%\Platform\src\PocketStoreII\config.h

```
#define PSII_VERSION            "[PocketStoreII] v1.5.1 P4 (16/FEB/2007)\r\n"

#undef PSII_DBGMSG_ENABLE
#include "config_debug_msg.h"

#define ONENAND_BASEADDR            0x08000000

#define LLD_OTP_UID             // OTP read enable
#define LLD_OTP_PBN             1000 // OTP block nPbn (should be in unlocked
region)
```

PSII_DBGMSG_ENABLE decides whether to take print out of PocketStoreII debug message or not. If it is undefined, the debug message will not be shown. And if it is defined, you can control the message category in config_debug_msg.h.

ONENAND_BASEADDR defines the physical base address of the OneNAND and it may be different to hardware configuration.

1.3.5. Modify BOOTLOADER

Add PocketStoreII libraries into
%_TARGETPLATROOT%\Platform\src\BOOTLOADER\EBOOT\sources

```
…
TARGETLIBS= \
…
      $(_COMMONOAKROOT)\lib\$(_CPUDEPPATH)\eboot.lib \
      $(_TARGETPLATROOT)\lib\$(_CPUDEPPATH)\bootpart.lib \
      $(_TARGETPLATROOT)\src\PocketStoreII\Lib\$(_CPUDEPPATH)\STL.lib
\
```

```
    $(_TARGETPLATROOT)\src\PocketStoreII\Lib\$(_CPUDEPPATH)\$(XSR_PAM)
PAM.lib         \
    $(_TARGETPLATROOT)\src\PocketStoreII\Lib\$(_CPUDEPPATH)\BML.lib
\
    $(_TARGETPLATROOT)\src\PocketStoreII\Lib\$(_CPUDEPPATH)\OSLessOAM.l
ib              \
    $(_TARGETPLATROOT)\src\PocketStoreII\Lib\$(_CPUDEPPATH)\ONLD.lib    \
  …
```

Also, add PocketStoreII libraries into
%_TARGETPLATROOT%\Platform\src\ BOOTLOADER\IPL\sources

```
…
TARGETLIBS= \
…
    $(_TARGETPLATROOT)\lib\$(_CPUDEPPATH)\bootpart.lib \
    $(_TARGETPLATROOT)\src\PocketStoreII\Lib\$(_CPUDEPPATH)\STL.lib
\
    $(_TARGETPLATROOT)\src\PocketStoreII\Lib\$(_CPUDEPPATH)\$(XSR_PAM)
PAM.lib         \
    $(_TARGETPLATROOT)\src\PocketStoreII\Lib\$(_CPUDEPPATH)\BML.lib
\
    $(_TARGETPLATROOT)\src\PocketStoreII\Lib\$(_CPUDEPPATH)\OSLessOAM.l
ib              \
    $(_TARGETPLATROOT)\src\PocketStoreII\Lib\$(_CPUDEPPATH)\ONLD.lib
…
```

1.4. Build & Run

After completing build & makeimg, you may have “ONBL1.nb0”, “ONBL2.nb0” from PocketStoreII. These two binaries should be written to 0-th block in OneNAND.

For writing images to OneNAND, you may simply use provided ONW or make your own customized utility. For both the cases, you can refer to the chapter 3.

2. Platform Customization

This chapter describes configuration changes based on your platform to deploy PocketStoreII.

2.1. Samsung S3C2410X

This platform only supports OneNAND asynchronous read mode with nCS0 booting.
You may use default configuration without changes.

Modify %_TARGETPLATROOT%\Src\PocketStoreII\env.bat

```
…
set BSP_POCKETSTORE=1
set XSR_PAM=S3C2410X
…
```

2.2. TI OMAP730 / OMAP850

This platform supports both asynchronous and synchronous read mode with internal and external booting. If you are using external boot, you may use default configuration.
But you should change following configurations for internal boot.

Modify %_TARGETPLATROOT%\Src\PocketStoreII\env.bat

```
…
set BSP_POCKETSTORE=1
set XSR_PAM=OMAP850
…
```

%_TARGETPLATROOT%\src\PocketStoreII\config.h

```
…
//PAM.cpp
#define ONENAND_BASEADDR            0x0C000000
#define PAM_USE_ASMCPY              // We will use assembly memcpy
…
```

%_TARGETPLATROOT%\Src\PocketStoreII\ONBL1\OMAP850\OneNAND.inc

```
…
ONLD_BASE_ADDRESS           EQU     0x0C000000
…
```

%_TARGETPLATROOT%\src\PocketStoreII\ONBL1\OMAP730\startup.s

```
…
ONBL2_BASE_ADDRESS          EQU     0x13150000
…
```

This address should be same with the entry point of ONBL2

2.3. INTEL PXA320

This platform only supports OneNAND asynchronous read mode with nCS0 booting. You may use default configuration without changes.

Modify %_TARGETPLATROOT%\Platform\ZYLONITE\Src\PocketStoreII\env.bat

```
…
set BSP_POCKETSTORE=1
set XSR_PAM=PXA320
…
```

2.4. OMAP2420

This platform has 2 different booting modes and OneNAND base address depends on the booting mode. So, two different base addresses are specified to config.h for each booting mode

Modify %_TARGETPLATROOT%\Src\PocketStoreII\config.h

```
…
#include "config_debug_msg.h"

#define ONENAND_BASEADDR            0x08000000    // OMAP2420
#define ONENAND_BASEADDR2           0x00000000    // OMAP2420 for external
booting
…
```

3. ONW

ONW (OneNAND Writer) is the OneNAND writing tool for writing binary images from host memory to OneNAND. For adopting PocketStoreII, you should format OneNAND by BML or STL layer before writing any binary images. For this, ONW may format OneNAND and write your own binary images. ONW is executed on RAM in target platform, so you can use JTAG tool or your own tool to download ONW. The memory location of written binary images for ONW is pre-defined in ONWConfig.h.

3.1. ONW Configuration

Some information have to be modified to appropriate value for target platform.
These addresses are used for the buffer of bootloader images.

%_TARGETPLATROOT%\src\PocketStoreII\ONW\ONWconfig.c

```
#define ONBL1_ADDRESS          0x80100000
#define ONBL2_ADDRESS          0x80150000
#define EBOOT_ADDRESS          0x80200000
#define IPL_ADDRESS            0x80300000
#define OS_ADDRESS             0x80400000

#define NUM_OF_IMAGE           4

#define ONW_BML_FORMAT
#define ONW_OS_FORMAT
#define ONW_OS_WRITE
#define ONW_FAT_FORMAT
```

c.f. Macro description

MACRO	Description
NUM_BML_IMG_WRITE	It means the number of binary images written by BML. In this case, 3 binary images mean ONBL1&ONBL2, EBOOT, and IPL. OS region isn't concerned with this configuration.
NUM_BML_PARTITION	It means the number of BML partition to be created in OneNAND
ONBL1_ADDRESS ONBL2_ADDRESS	It means the buffer location for ONBL1&ONBL2 image. The size of ONBL1 is 2KB (1KB is unused.) and the size of ONBL2 is 126KB. Always ONBL1 and ONBL2 should reside in the first block of OneNAND.
EBOOT_ADDRESS	It means the buffer location of EBOOT image.
IPL_ADDRESS	It means the buffer location of IPL image.
OS_ADDRESS	It means the buffer location of OS image. You should check the size of written dio file, and the size of RAM area. This macro can be used only if OS_REGION_WRITE is defined.
ONW_BML_FORMAT	It means BML format on all BML partitions. This operation doesn't erase OneNAND. So, you can always define this macro.
ONW_OS_FORMAT	It means STL_Format on OS partition. You should format

	OS partition before writing OS image. This operation may erase the partition.
ONW_OS_WRITE	It means to write OS image on OS partition.
ONW_FAT_FORMAT	It means STL_Format on FAT partition. You should format FAT partition before booting up. Format operation will do all FAT partitions.
NUM_OF_IMAGE	Total number of OneNAND to write image.

The following is a BML partition configuration. You may change this for your own purpose.
%_TARGETPLATROOT%\src\PocketStoreII\ONW\$(XSR_PAM)\ONWConfig.c

If you want to use OneNAND for booting & storage device, use below code.

```
VOID
_SetXSRPI(VOID)
{
    gstXSRPartI.nNumOfPartEntry        = NUM_BML_PARTITION;

    gstXSRPartI.stPEntry[0].nID        = PARTITION_ID_EBOOT;
    gstXSRPartI.stPEntry[0].nAttr      = BML_PI_ATTR_RO |
    BML_PI_ATTR_FROZEN;
    gstXSRPartI.stPEntry[0].n1stVbn    = EBOOT_START;
    gstXSRPartI.stPEntry[0].nNumOfBlks = EBOOT_NUM;

    gstXSRPartI.stPEntry[1].nID        = PARTITION_ID_IPL;
    gstXSRPartI.stPEntry[1].nAttr      = BML_PI_ATTR_RO |
    BML_PI_ATTR_FROZEN;
    gstXSRPartI.stPEntry[1].n1stVbn    = IPL_START;
    gstXSRPartI.stPEntry[1].nNumOfBlks = IPL_NUM;

    gstXSRPartI.stPEntry[2].nID        = PARTITION_ID_COPIEDOS;
    gstXSRPartI.stPEntry[2].nAttr      = BML_PI_ATTR_RO;
    gstXSRPartI.stPEntry[2].n1stVbn    = OS_START;
    gstXSRPartI.stPEntry[2].nNumOfBlks = OS_NUM;

    gstXSRPartI.stPEntry[3].nID        = PARTITION_ID_FILESYSTEM;
    gstXSRPartI.stPEntry[3].nAttr      = BML_PI_ATTR_RW;
    gstXSRPartI.stPEntry[3].n1stVbn    = FAT0_START;
    gstXSRPartI.stPEntry[3].nNumOfBlks = FAT0_NUM;

    gstXSRPartI.stPEntry[4].nID        = PARTITION_ID_FILESYSTEM+1;
    gstXSRPartI.stPEntry[4].nAttr      = BML_PI_ATTR_RW;
    gstXSRPartI.stPEntry[4].n1stVbn    = FAT1_START;
    gstXSRPartI.stPEntry[4].nNumOfBlks = FAT1_NUM;

    gstXSRPartI.stPEntry[5].nID        = PARTITION_ID_EBOOTCFG;
    gstXSRPartI.stPEntry[5].nAttr      = BML_PI_ATTR_RW;
    gstXSRPartI.stPEntry[5].n1stVbn    = EBOOTCFG_START;
    gstXSRPartI.stPEntry[5].nNumOfBlks = EBOOTCFG_NUM;
}
```

If you want to use OneNAND as only storage device, replace the above part to below part.

```
VOID
_SetXSRPI(VOID)
{
```

```
    gstXSRPartI.nNumOfPartEntry          = 1;

    gstXSRPartI.stPEntry[0].nID          = PARTITION_ID_FILESYSTEM;
    gstXSRPartI.stPEntry[0].nAttr        = BML_PI_ATTR_RW;
    gstXSRPartI.stPEntry[0].n1stVbn      = 0;
    gstXSRPartI.stPEntry[0].nNumOfBlks   = 1001;
}
```

Edit the memory configuration of ONW in
%_TARGETPLATROOT%\Platform\src\PocketStoreII\ ONW\$(XSR_PAM)\onw.bib
Following values are have to be modified to appropriate values.

```
MEMORY
;   Name        Start       Size        Type
;   -------     --------    --------    ----
    RAM         80060000    00050000    RAM
    STACK       80040000    00020000    RESERVED
    ONW         80000000    00040000    RAMIMAGE

    ROMSTART=80000000
    ROMWIDTH=32
    ROMSIZE=40000
```

Also you should customize startup.s code that is located
at %_TARGETPLATROOT%\Platform\src\PocketStoreII\ONW\startup.s

3.2. Executing ONW

You may use In-Circuit-Debugger or UART to download ONW, bootloader and OS image into your target RAM. And you should confirm that the downloading address in target matches with ONBL_ADDRESS, EBOOT_ADDRESS, IPL_ADDRESS and OS_ADDRESS.

Complete downloading these binaries, and execute ONW to write other images to OneNAND. Then, you may see several messages from ONW through UART.

If there is no errors in writing OneNAND, reboot the target and check log messages from OneNAND booting sequence.