



Austin Design Center  
11000 North Mopac Expressway  
Stonelake Bldg. 6 Suite 500  
Austin, Texas 78759

## ***USB2602 Software Release Notes*** ***v.542***

*Updated 6-22-2007*

The information contained herein is confidential and proprietary to SMSC, shall be used solely in accordance with the agreement pursuant to which it is provided, and shall not be reproduced or disclosed to others without the prior written consent of SMSC. Although the information is believed to be accurate, no responsibility is assumed for inaccuracies. SMSC reserves the right to make changes to this document and to specifications and product descriptions at any time without notice. Neither the provision of this information nor the sale of the described semiconductor devices conveys any licenses under any patent rights or other intellectual property rights of SMSC or others. The product may contain design defects or errors known as anomalies, including but not necessarily limited to any which may be identified in this document, which may cause the product to deviate from published specifications. SMSC products are not designed, intended, authorized or warranted for use in any life support or other application where product failure could cause or contribute to personal injury or severe property damage. Any and all such uses without prior written approval of an officer of SMSC will be fully at the risk of the customer. SMSC is a registered trademark of Standard Microsystems Corporation ("SMSC").

SMSC DISCLAIMS AND EXCLUDES ANY AND ALL WARRANTIES, INCLUDING WITHOUT LIMITATION ANY AND ALL IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE, AND AGAINST INFRINGEMENT AND THE LIKE, AND ANY AND ALL WARRANTIES ARISING FROM ANY COURSE OF DEALING OR USAGE OF TRADE. IN NO EVENT SHALL SMSC BE LIABLE FOR ANY DIRECT, INCIDENTAL, INDIRECT, SPECIAL, PUNITIVE, OR CONSEQUENTIAL DAMAGES; OR FOR LOST DATA, PROFITS, SAVINGS OR REVENUES OF ANY KIND; REGARDLESS OF THE FORM OF ACTION, WHETHER BASED ON CONTRACT; TORT; NEGLIGENCE OF SMSC OR OTHERS; STRICT LIABILITY; BREACH OF WARRANTY; OR OTHERWISE; WHETHER OR NOT ANY REMEDY OF BUYER IS HELD TO HAVE FAILED OF ITS ESSENTIAL PURPOSE, AND WHETHER OR NOT SMSC HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

## ***Software Compliance***

The software in this release conforms to the following industry flash card specifications. SMSC has tested to the best of its ability to ensure that this software conforms to these specifications. However, no other warranty is assured, express or implied, other than provided by SMSC's standard terms and conditions.

1. SmartMedia™ Electrical Specification Version 1.40
2. SmartMedia™ Physical Format Specifications Version 1.40
3. SmartMedia™ Logical Format Specifications Version 1.30
4. MultiMediaCard System Specification Version 4.2
5. SD Memory Card Specifications Version 2.0
6. Memory Stick Standard Format Specification Version 1.43-00
7. Memory Stick Pro Standard Format Specifications Version 1.02-00
8. Memory Stick Duo Standard Format Specifications Version 1.10-00
9. CompactFlash Specification Rev 2.0
10. xD Picture Card Specification Version 1.2
11. Universal Serial Bus Specification Rev 2.0
12. USB Mass Storage Class, Bulk Only Transport Version 1.0

**Table of Contents**

<b>Revision History .....</b>	<b>5</b>
<b>The Non-Volatile Store Data .....</b>	<b>8</b>
<b>Using Flash ROM to Store the NVStore Data .....</b>	<b>8</b>
<b>Creating the EEPROM.DAT File .....</b>	<b>8</b>
<b>Using the USB Drive Manager Application (for Windows XP only) .....</b>	<b>9</b>
The USBDM Toolbar .....	9
The Info Tab .....	10
The Branding Tab .....	10
Using .dat files with USBDM .....	10
The Configuration Tab .....	11
The Hub Tab .....	12
<b>Card Reader Attribute Bit Definitions .....</b>	<b>13</b>
<b>Hub Configuration Definitions: .....</b>	<b>15</b>
<b>Programming the NVStore Data .....</b>	<b>20</b>
<b>LUN Configuration and Icon Sharing .....</b>	<b>21</b>
LUN Configuration .....	21
Icon Sharing .....	21
<b>LUN Power Configuration .....</b>	<b>22</b>
LUN Power Masks .....	23
<b>Using Device Firmware Upgrade (DFU) .....</b>	<b>25</b>
Overview .....	25
Files Required for DFU for Windows .....	25
Creating the 128KB DFU Capable Flash Binary “both.bin” .....	26
Preparing a Device for DFU Operation .....	27
Choosing a Flash Eeprom for Your Device .....	27
Setting up the Hardware .....	27
Using the USBDM Application to Perform Device Firmware Upgrade (DFU) .....	28
Using the OEM.exe to Update Firmware .....	30
Creating a DFU Uploadable File .....	31
Using the DFU.exe Utility .....	32
<b>Performing a Firmware Upgrade with the DFU_App Application(Mac 10.X Only) .....</b>	<b>33</b>
Where to find DFU_App .....	33
Using an engineering version of DFU_App application .....	33
Creating a customer version of DFU .....	34
Using a customer version of DFU_App .....	35
<b>Using the USB2602 Custom Icons Package .....</b>	<b>36</b>
Contents of the USB2602 Custom Icons Package .....	36
Creating the Required SetIcon Ini Files .....	36
Manually Installing the Custom Icons Application Files .....	38
Troubleshooting the Custom Icons Application .....	40
<b>Windows Installer Packages .....</b>	<b>42</b>
<b>Using the Production Line Descriptor Update Utility (PLDU) .....</b>	<b>43</b>
Setting Up the PLDU Application .....	45
Using the PLDU to Update Device Descriptors .....	45
<b>Using the Production Line Test Utility (PLTU) .....</b>	<b>46</b>
Creating the PLTU ini File .....	46
A Sample PLTU ini File .....	47
Setting Up the PLTU Application .....	48
Using the PLTU to Test Multiple Devices .....	48
<b>Known Issues with the USB2602 Production Line Utilities .....</b>	<b>49</b>
<b>Using the QuickTest Production Line Read/Write Test Utility .....</b>	<b>50</b>
<b>Using the Windows XP Special Memory Stick Format Registry Key .....</b>	<b>51</b>
<b>Using the KillReg Utility .....</b>	<b>52</b>
<b>Using the Swapdrv Utility .....</b>	<b>53</b>
<b>Media Tested with the USB2602 .....</b>	<b>55</b>
<b>USB2602 Performance Benchmarks .....</b>	<b>57</b>

**GPIO Assignment Table..... 58**

**Known Firmware Related Issues ..... 59**

    General: ..... 59

    CF Devices: ..... 59

    MS Devices: ..... 59

    SM Devices: ..... 59

    SD/MMC Devices: ..... 59

    xD Devices: ..... 59

**Issues Not Related to Firmware..... 60**

**Revision History**

0.395:       **-ROM Mask 01.**

-Initial Release

0.444:       **-ROM Mask 02.**

**Firmware:**

- Added support for 4-bit HS-MMC.
- Added firmware changes to comply with the latest MSC BOT Compliance Test.
- Shortened timeout values to allow card reader to recover faster after surprise removal of media.
- Changed the value of hub default k\_hub\_default\_cb1 from 0x98 to 0X9B.
- Fixed an issue with high speed memory stick 2-bit ECC errors.
- Initialize MS and SM media ID strings – needed for MS Pro icon to work properly on Windows 2000 SP4 after device power up.

**Hardware:**

- Fixed issue with Attach on Card Insert/ Detach on Card Removal feature being non-functional.
- Fixed issue with SD/MMC not being recognized upon power up if card reader is configured with SD and MS sharing a LUN. (This was only an issue with the 128 pin package.)

**Applications:**

- Added PLDU to release packages.

0.464:

**Firmware:**

- SM-Player/Runtime xDPlayer.
- MS/MS Pro Compliance enhancements.
- xD Player mode compliance fix.

**Applications:**

- (USBDM 2.0.0.7, PLDU 2.0.1.0) Added attribute bit 29 for enabling SIR mode only or all modes.
- (USBDM 2.0.0.7, PLDU 2.0.1.0) Added attribute bit 30 for specifying whether blocks are to be erased or not while resolving mapping conflicts.
- (USBDM 2.0.0.7) Added "SMSC\_Recover\_Device()" fn in library which is called by USBDM to recover the device f/w in case of a previous f/w update failure.
- (USBDM 2.0.0.7, PLDU 2.0.1.0) Added attribute bit 31 for setting XD Player Mode.
- (USBDM 2.0.0.8, PLDU 2.0.1.1) DFU recovery enhancement for system power loss during DFU.
- (USBDM 2.0.0.8, PLDU 2.0.1.1) Allow 29 characters for MFG and PROD strings.
- (USBDM 2.0.0.8, PLDU 2.0.1.1) Added User Interface for updating VID/PID of DFU file during creation of OEM.exe.
- (USBDM 2.0.0.8, PLDU 2.0.1.1) Dragging and drop of a DFU file onto PLDU.exe is ignored.
- (SetIcon 1.4.0.0) Changed the way GetMediaID command is issued.
- (Software Installers 1.0.0.4) Generic.ini included for X86 platforms.
- (Software Installers 1.0.0.4) Modify/Repair options during uninstallation disabled.
- (Software Installers 1.0.0.4) Win2K INF files with support for disabling safe removal added.
- (Software Installers 1.0.0.4) Supports installation on 32-bit and 64-bit platforms.
- (Software Installers 1.0.0.4) Supports installation on all OSs from Windows 98 to Windows Vista.
- (Software Installers 1.0.0.4) Provides support for SMSC and GENERIC descriptor data.
- (Software Installers 1.0.0.4) Provides support for grouping MS LUN with MemoryStick device group. This feature is available only for 223 and 2228 devices having SMSC descriptor data.
- (Software Installers 1.0.0.4) Installer does not install any signed drivers.

0.0.0.501

**-External Evaluation Build.****Firmware:**

- SM-Player/Runtime support for xDPlayer
- MS/MS Pro Compliance enhancements.
- xD Player mode compliance fix
- Fix for extended reboot testing issue.
- Fix for intermittent HS-MMC issue.
- Added SD 2.0 HC-SD support.
- Fixed issue with some Memory Stick cards properly resuming operation after a USB bus Suspend when operating in self-powered mode.
- Fixed MMC inquiry response issue.
- Update of code for new MSPRO compliance test.
- Benchmark performance enhancements.
- Fixed improper clock speed setting for HS-MMC.
- fixed issue with SD clock starting before card power is settled and to always start at 200KHz.
- fixed intermittent card startup issue after USB Suspend.
- added additional SD vendor specific commands to provide SD Status Register information, such as Speed Class.
- Reversed SD Reverse Write-Protect bit for the OEM2 bond option default Configuration.
- Changed SD 2.0 Check Pattern to work with noncompliant Skymedi SD/MMC controllers.
- Fixed issue when sharing FET power for MS/SM/SD that SM/xD card is not recognized until re-inserted after host reboot..
- Fixed issue with device not re-attaching if SM card removed during SUSPEND and then re-inserted.
- Added MMC 4.2 support.
- Added performance enhancements
- Added MS-Pro Compliancy (tester version 1.06.070A).
- Fixed MMC 4.2 Identification issue.

**Applications:**

- (USBDM 2.0.0.6, PLDU 2.0.0.9) Added attribute bit 29 for enabling SIR mode only or all modes.
- (USBDM 2.0.0.6, PLDU 2.0.0.9) Added attribute bit 30 for specifying whether blocks are to be erased or not while resolving mapping conflicts.
- (USBDM 2.0.0.6) Added "SMSC\_Recover\_Device()" fn in library which is called by USBDM to recover the device f/w in case of a previous f/w update failure.
- (USBDM 2.0.0.7, PLDU 2.0.1.0) Added attribute bit 31 for setting XD Player Mode.
- (USBDM 2.0.0.8, PLDU 2.0.1.1) Added attribute bit 31 for setting xD Player mode.
- (USBDM 2.0.0.8, PLDU 2.0.1.1) DFU recovery enhancement for system power loss during DFU.
- (USBDM 2.0.0.8, PLDU 2.0.1.1) Allow 29 characters for MFG and PROD strings.
- (USBDM 2.0.0.8, PLDU 2.0.1.1) Added User Interface for updating VID/PID of DFU file during creation of OEM.exe.
- (USBDM 2.0.0.8, PLDU 2.0.1.1) Dragging and drop of a DFU file onto PLDU.exe is ignored.
- (SetIcon 1.4.0.0) Changed the way GetMediaID command is issued.
- (Software Installers 1.0.0.4) Generic.ini included for X86 platforms.
- (Software Installers 1.0.0.4) Modify/Repair options during uninstallation disabled.
- (Software Installers 1.0.0.4) Win2K INF files with support for disabling safe removal added.
- (Software Installers 1.0.0.4) INI files used by SwapDrvr and KillReg utilities modified (removed unwanted entries).
- (Software Installers 1.0.0.4) Supports installation on 32-bit and 64-bit platforms.
- (Software Installers 1.0.0.4) Supports installation on all OSs from Windows 98 to Windows Vista.
- (Software Installers 1.0.0.4) Installs all components (end-user applications, drivers, INF files, INI files and utility applications).
- (Software Installers 1.0.0.4) Provides support for SMSC and GENERIC descriptor data.
- (Software Installers 1.0.0.4) Provides support for grouping MS LUN with MemoryStick device group. This feature is available only for 223 and 2228 devices having SMSC descriptor data.
- (Software Installers 1.0.0.4) Installer does not install any signed drivers.
- (USBDM 2.0.0.9, PLDU 2.0.1.2) Added all four possible values for the bmAttribute under the configuration tab.

0.523:

**ROM Mask 04**

**Firmware:**

- Fixed minor Memory Stick Compliance test issue uncovered by latest test version.
- Added HS-MMC Compliancy mode with default that only allows MMC clock speed of 20MHz to ensure out of specification HS-MMC cards work. Optional high speed operation via attribute bit.
- MS/MS Pro Compliance enhancements.
- xD Player mode compliance fix.

**Applications:**

- (USBDM 2.0.1.0) Removed Admin check for OEM.exe under Vista. The OEM.exe app will check for admin rights under other OS only.
- (USBDM 2.0.1.0, PLDU 2.0.1.3) Changed the definition of bit20 of Attributes field to 0 = Compatibility Mode (Default value) and 1 = Allow High Speed Mode.
- (DosPLTU v2.2, CheckROM v1.6, EprmUpdt v2.2) Fixed the issue of infinite loop in certain OEM PCs for EHCI tests; Added code to try and enable A20 gate if it is not enabled.

0.539:

**ROM Mask 05**

**Firmware:**

- Forced maximum clock on HS-SD cards to 24MHz to help insure interoperability with 150X generation cards in more board layouts and configurations.

0.542:

**ROM Mask 06**

**Firmware:**

- Fixed a memory leak condition in the code which would cause, very rarely, erratic behavior after card insertion.
- Fixed issue with two specific MS Pro cards which incorrectly reported ready condition when they were not actually ready.

### **The Non-Volatile Store Data**

The NVStore is user modifiable data that is stored in either serial EEPROM or external program flash ROM and used by the device during operation. Some of the values that can be modified in the NVStore data include the serial number, VID/PID, Manufacturers ID String, Product ID String, LUN ID Strings, the modifiable device descriptors such as bmAttributes and MaxPower, number of LUNs, LUN order, and other modifiable bytes which customize the operation of the USB2602.

The NVStore data is programmed into the device using a text file “EEPROM.DAT,” which contains the bytes of data that are written to the EEPROM.

SMSC provides a utility to program the NVStore data called “USBDM.exe.” The procedure for using the USBDM Utility to write the NVStore data is described in the following paragraphs.

### **Using Flash ROM to Store the NVStore Data**

If you are using external program flash you can, as a cost reduction measure, eliminate the need for a serial eeprom in your device by using the SST39VF010 Flash ROM and the “NO EEPROM” version of the USB2602 firmware. The NO EEPROM firmware uses a portion of the memory storage area in the SST39VF010 Flash to hold all of the NVStore data. Currently, the SST39VF010 is the only chip supported by the NO EEPROM firmware. If you have a requirement to use another flash, please contact SMSC Sales to inquire about adding support for your chip.

Note: The USB2602 contains internal masked ROM program code. If you are running the 2602 from internal ROM code, you must use an external eeprom to store the NVStore data (VID/PID/Manufacturer and Product ID Strings, Attribute Bytes, etc.).

### **Creating the EEPROM.DAT File**

An eeprom.dat file can be created using the USBDM application by altering all fields in the Configuration and Branding tab as desired for the new file and saving the file using the “Save” button. This can be done with or without a device attached to the host computer. The following section describes these tabs and the USBDM application in detail.



**Using the USB Drive Manager Application (for Windows XP only)**

The USB Drive Manager (USBDM) application can be used to create the eeprom.dat file and program the USB2602 device via USB, plus some additional functions such as creating end-user firmware updates contained within a single, easily distributable exe and having the ability to instantly read the NVStore data from the device without the need for a driver swap.

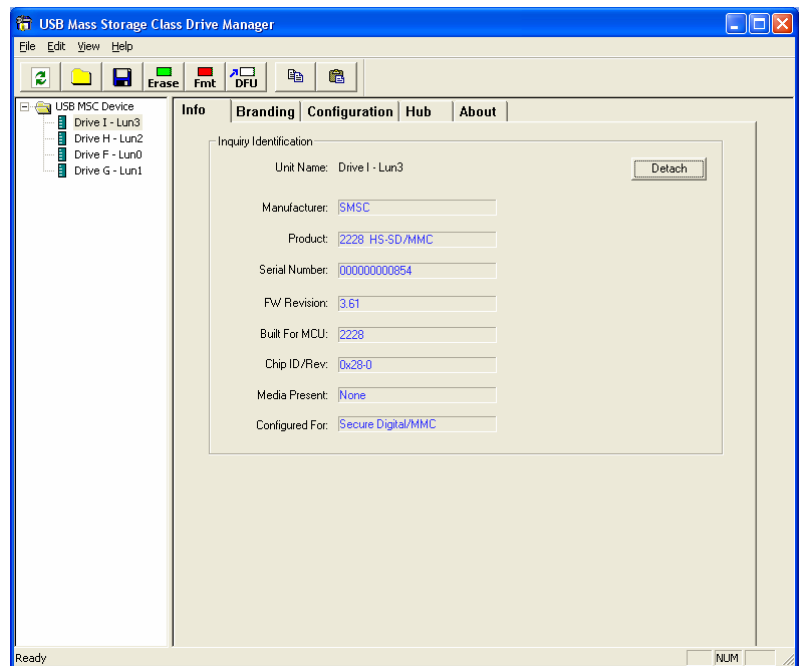
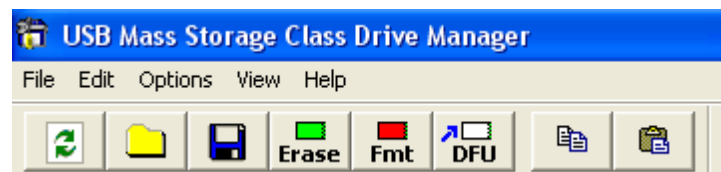
**Note: Only USBDM version 2.0.0.4 or newer will work properly for updating 2602.**

**Note: USBDM will not work for updating a SMSC standalone hub.**

**Note: The USBDM Application is supported in Windows XP only.**

**Getting Started:**

To start the USB Drive Manager application, simply double click on the “USBDM.exe” executable. Once the application opens you will see the screen shown to the right if there is a device attached to the host computer. If there is no device present, a virtual device will be listed instead of the USB MSC Device information shown in this example. This virtual device allows a .dat file to be edited without the need for a device to be attached to the host computer.

**The USBDM Toolbar**

The toolbar buttons shown above are displayed at the top left hand side of the application. Starting from left to right, they perform the following functions:

Button 1: Refresh Drive List

Button 2: Load .dat file

Button 3: Save .dat file

Button 4: Erase Media (Not Used With 2602)

Button 5: Format Drive (Not Used With 2602)

Button 6: Upload Firmware

Button 7: Copy

Button 8: Paste

\*If you do not see these buttons displayed, go to “View” in menu bar and make sure there is a check next to the “Toolbar” option.

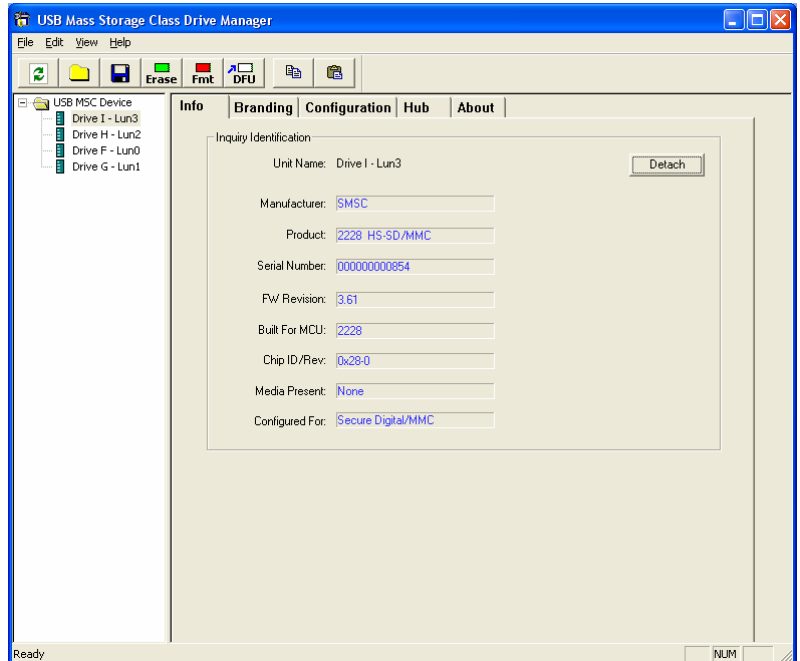
\*Clicking on the “Help” option above the toolbar and selecting “About Drive Manager” will display the version of the USBDM application.

### The Info Tab

The info tab is displayed whenever a USB mass storage class device is attached to the host while USBDM is running. This tab displays the key fields in the NVStore data for the device. Note: Unless the device contains the SMSC USBDM firmware extensions, most of the data fields will display INVALID.

Attach a device containing the USBDM firmware extensions to the PC via a USB cable. The USB Drive Manager application will read the NVStore data for this device if valid data exists. It will display information for each drive that is available on the device. The example to the right has information for Drive F, Drive G, Drive H, and Drive I. You can toggle between the information for each of these drives by single clicking on the Drive entry under the “USB MSC Device” folder on the left side of the application.

Note: The detach button seen on this tab will momentarily detach the target device from the system.



### The Branding Tab

The Branding tab is used to write vendor specific data to the NVStore. Programmable fields include: Vendor ID, Product ID, Language ID, Manufacturing String, Product String, and Serial Number String. Any of this information can be changed on the device. Once you have entered the information for your device, click on the “Update Now” button to program the NVStore.

**Vendor ID:** Unique for every vendor. Assigned by the USB Implementers Forum.

**Product ID:** Unique to product. Assigned by vendor.

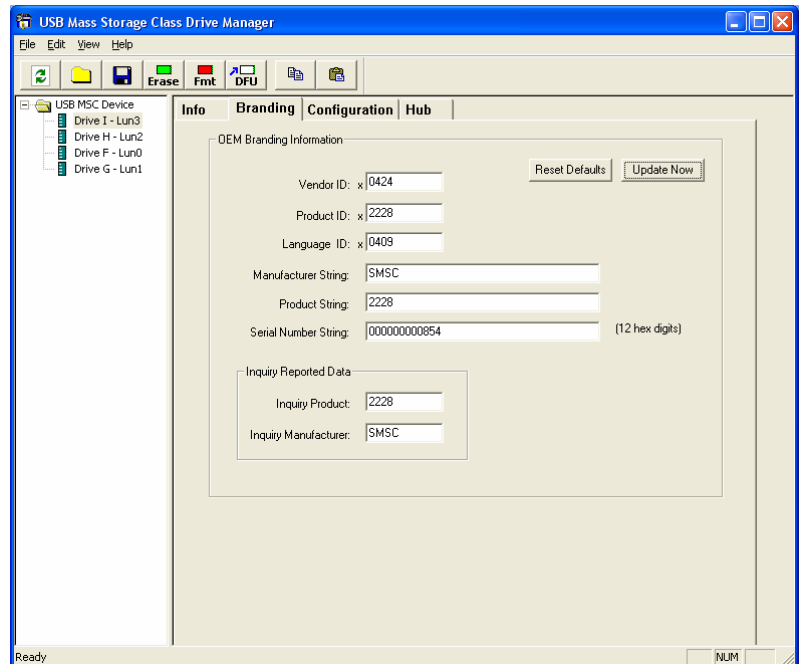
**Language ID:** 0409 is the Language Code for English. Other Language Codes may be found in the USB specification.

**Product String:** 28 characters max. Used to identify the product. This string will be used during the USB enumeration process in Windows.

**Manufacturing String:** 28 characters max. Used to identify the manufacturer.

**Serial Number:** 12 hex digits max. Must be unique to each device.

**Inquiry Manufacturer (8 Bytes) and Product (5 Bytes) ID Strings:** If bit 4 of the 1<sup>st</sup> attribute byte is set, the device will use these strings in response to a USB inquiry command, instead of the USB Descriptor Manufacturer and Product ID Strings.



### Using .dat files with USBDM

The Load .dat file button can be used to populate these fields from a valid .dat file. After clicking the Load .dat file button, you will be prompted to specify a .dat file. Once the .dat file has loaded, the text fields will be updated to reflect the data in the .dat file. Any changes made to the text fields can also be saved into a .dat format using the Save .dat file button at the top of the application.

## The Configuration Tab

The Configuration tab contains all of the other NVStore programmable fields not found in the Branding Tab.

The Configuration Tab is where you set:

- 1) The NVStore signature which is always “ATA1” for the USB2602
- 2) The attribute bits
- 3) The LUN assignments
- 4) The LUN IDs
- 5) NAND Profile (Not Used for USB2602)
- 6) Miscellaneous settings such as the USB descriptors bMaxPower and bmAttribute

These user programmable fields are described in detail in the following paragraphs.

**Signature:** The signature should remain set to ATA1 for USB2602.

**Attribute Bits:** The attribute bits are used to customize the functionality of the USB2602

firmware. A complete list of all programmable attribute bits and their function is listed in the section of this document entitled “Card Reader Attribute Bit Definitions.” In the image shown above “Reverse SD Card Write Protect Sense” is the only option selected. Placing a check to the left of an option sets an attribute bit. If the box is unchecked, the attribute bit will be cleared. Any of these options may be checked or unchecked depending on the various needs for the product being programmed. There are also dropdown options in the “Infra-Red” and “CIR” sections. These dropdown options do not apply to the 2602.

### **LUN Configuration:**

**LUN ID Strings (7 bytes each)**—There are four LUN ID strings corresponding to LUN# 0,1,2 and 3.

**Number of Icons to Display, CF Lun #, MS Lun #, NAND Lun #, SD/MMC Lun #, SM Lun #**—These bytes are used to specify the number of LUNs the device exposes to the

host. These bytes are also used for icon sharing—assigning more than one LUN to a single icon. This is used in applications where the device utilizes a combo socket, and the OEM wishes to have only a single icon displayed for one or more interfaces. For more information, see the section of this document entitled “LUN Configuration and Icon Sharing.” If this field is set to “FF”, the program assumes that you are using the default value of “04” and will display icons for CF, MS, SM, and SD. If this field is any other value besides “FF”, you must specify the LUN# assignments in the boxes below starting with LUN 00 and going to (# of Icons to Display -1)

**NAND Profile (2 Bytes) (Not used for the USB2602)**—This is where the NAND performance profile is specified for controllers that use it.

Note that more than one interface (CF, MS, SM, or SD) can share a LUN. Remember **LUN numbering always starts at 00**.

The configuration to the right directs the firmware to show three LUNs in the order of CF, SD/MMC, and SM. Note that Memory Stick is not enabled in this configuration.

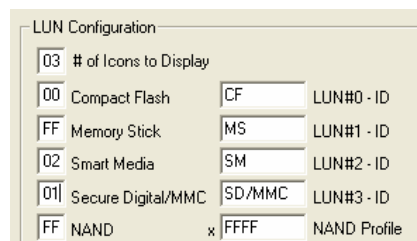
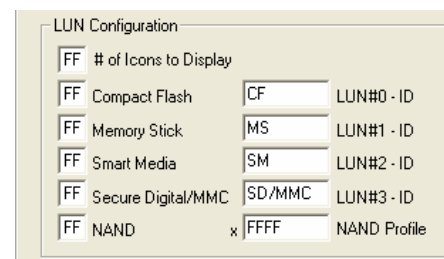
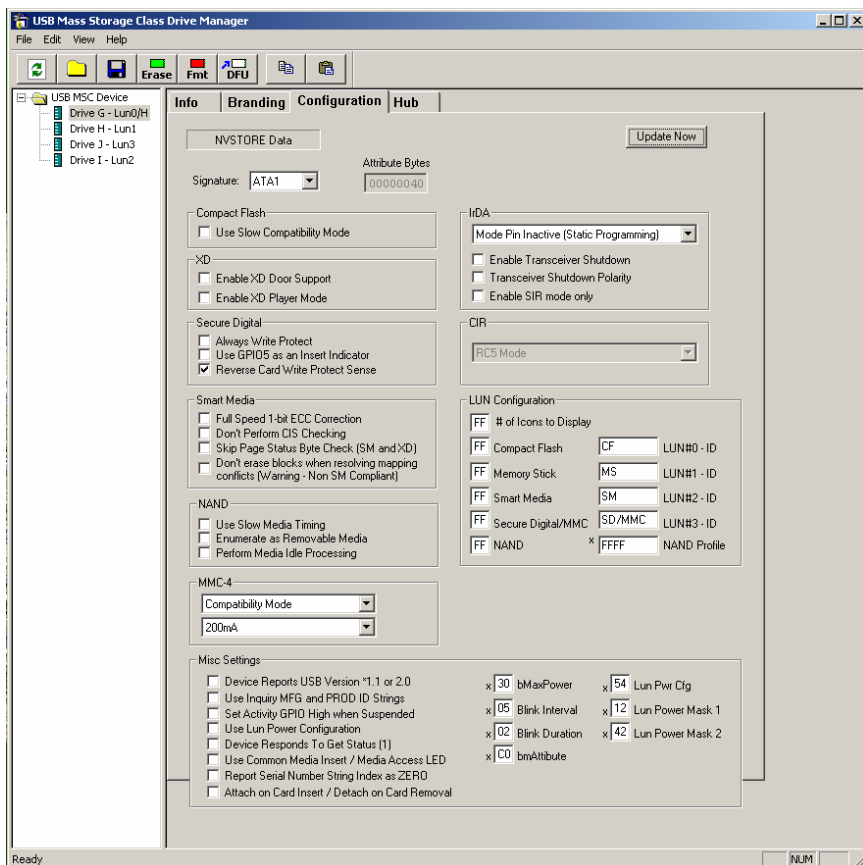
# of Icons to Display: 03

Compact Flash (1<sup>st</sup> LUN): 00

Memory Stick (will not display): FF

Smart Media (2<sup>nd</sup> LUN): 02

Secure Digital/MMC (3<sup>rd</sup> LUN): 01



**Misc. Settings:** The Misc. Settings section is used to program the other miscellaneous NVStore editable values. They are:

- 1) **bMaxPower** (1 byte): Per USB specification. Do not set this value greater than 100mA
- 2) **Blink Interval** (1 byte): Programmable in 10ms intervals. Hi bit indicates idle state: 0–Off, 1–On. The remaining bits are used to determine the blink interval up to a max of 128 x 10 ms.
- 3) **Blink Duration** (1 byte): This byte is used to designate the number of seconds that the GPIO 0 LED will continue to blink after a drive access. Setting this byte to “05” will cause the GPIO 0 LED to blink for 5 seconds after a drive access.
- 4) **bmAttribute** (1 byte): Per USB Specification.  
80 – Device is Bus Powered  
C0 – Device is Self Powered
- 5) **Lun Pwr Cfg** (1 byte): Should be a valid hexadecimal number. Default = 54. Refer to the “Lun Power Configuration” section for additional information on how to calculate this byte.
- 6) **Lun Power Mask 1** (1 byte): contains the power mask setting for CF and MS controllers. The mask used depends on how the LUN is configured in the LUN Power Configuration byte. Refer to the “Lun Power Configuration” section for additional information on how to calculate this byte.
- 7) **Lun Power Mask 2** (1 byte): contains the power mask setting for SM and SD controllers. The mask used depends on how the LUN is configured in the LUN Power Configuration byte.

x	01	bMaxPower	x	54	Lun Pwr Cfg
x	05	Blink Interval	x	12	Lun Power Mask 1
x	02	Blink Duration	x	42	Lun Power Mask 2
x	C0	bmAttribute			

## The Hub Tab

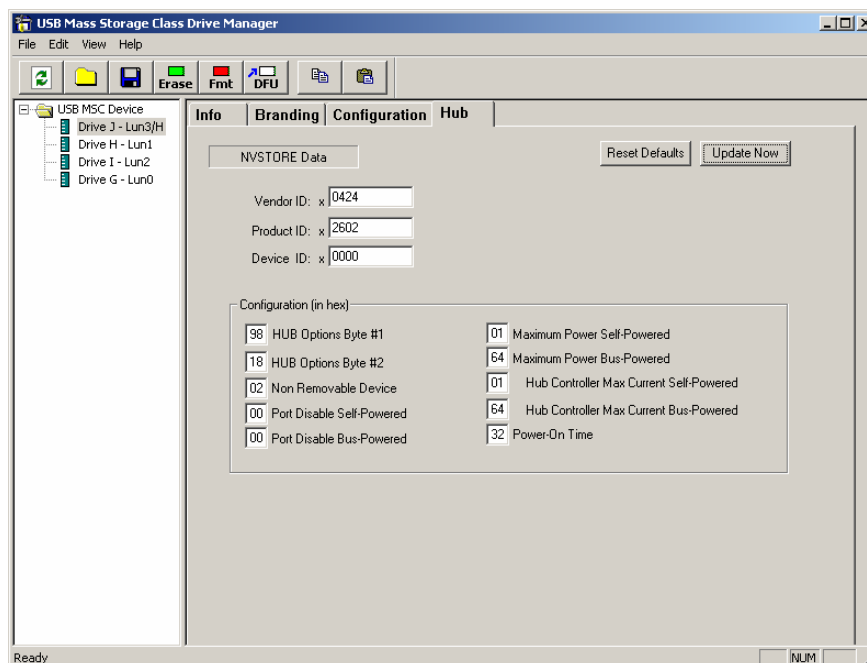
The Hub tab contains all of the NVStore programmable fields that apply to the hub portion of the USB2602 hub/card reader combo.

**Vendor ID:** Unique for every vendor. Assigned by the USB Implementers Forum.

**Product ID:** Unique to product. Assigned by vendor.

**Device ID:** Unique to device. Assigned by vendor.

**Configuration Bytes:** The configuration bytes are used to customize the functionality of the hub portion of the USB2602 firmware. A complete list of all programmable configuration bytes and their functions are listed in the section of this document entitled “Hub Configuration Definitions.”



**Note:** It is strongly recommended not to change the hub configuration bytes from the default values. Certain configurations could cause the USB2602 to become non-functional. Please contact a SMSC Applications Engineer to discuss any needed changes to this tab before updating non-default hub configuration information into a USB2602.

**Card Reader Attribute Bit Definitions**

**Attributes (4 bytes):** The attribute value for your device is determined by the options selected in the USBDM utility provided by SMSC. Changing the checkboxes and updating the device can update this information. These bits are defined below and organized by the Byte/Bit order. In the USBDM GUI, these bits are organized by which media type/feature they affect. The majority of these bits are displayed as checkboxes in the USBDM GUI. A few of them are displayed in dropdown options. The “Infra-Red (IrDA)” and “CIR” dropdown options are not used with the USB2602 and changing these settings will have no effect on the device. The bit definitions are as follows:

**Note:** The bit names are shown in bold below and correlate to how the attribute checkbox is labeled in USBDM. Not all checkboxes apply to the USB2602. Those bits that do not apply will specify, “Reserved – always set to 0” in the definitions below.

**Byte 1, bit 0:** Reserved – always set to 0. **Use Slow NAND FLASH Media Timing**

**Byte 1, bit 1:** Reserved – always set to 0. **Enumerate NAND Device as Removable**

**Byte 1, bit 2:** Reserved – always set to 0. **Use GPIO5 as an SD Card Insert Indicator**

**Byte 1, bit 3: Report Serial Number String Index as ZERO**

1 – Always report iSerial as zero in the device descriptor.

0 (default) – Report non-zero iSerial in device descriptor if serial number is valid.

**Byte 1, bit 4: Use the Inquiry Manufacturer and Product ID Strings**

1 – Use the Inquiry Manufacturer and Product ID Strings.

0 (default) – Use the USB Descriptor Manufacturer and Product ID Strings.

**Byte 1, bit 5: Set Activity GPIO High when Suspended**

1 – The activity LED GPIO is set to High when suspended.

0 (default) – The activity LED GPIO is set to Low when suspended.

**Byte 1, bit 6: Reverse SD Card Write Protect Sense**

1 (default) – SD cards will be write protected when SW\_nWP is high, and writable when SW\_nWP is low.

0 – SD cards will be write protected when SW\_nWP is low, and writable when SW\_nWP is high.

**Byte 1, bit 7: Make SD Cards Write Protected Always (Read Only)**

1 – SD cards will always be write protected, regardless of the state of the card's write protect switch.

0 (default) – SD cards will only be write protected when the write protect switch on the SD card is engaged.

**Byte 2, bit 0: Don't Perform Smart Media CIS Checking**

1 – Ignore CIS check for Smart Media to allow the USB2602 to work with non-compliant cards.

0 (default) – Enforce Strict CIS checking for Smart Media cards.

**Byte 2, bit 1:** Reserved – always set to 0. **Perform NAND Media Idle processing**

**Byte 2, bit 2: Use Slow Compact Flash Compatibility Mode**

1 – Compact Flash will operate in slow PIO-0 mode only regardless of CF card's actual capability.

0 (default) – Compact Flash will operate at the fastest mode the card reports it can support.

**Byte 2, bit 3: Device Responds To Get Status (1)**

1 – Device will report itself as SELF POWERED in response to a GET STATUS from the host.

0 (default) – Device will report itself as BUS POWERED in response to a GET STATUS from the host.

**Byte 2, bit 4: Device Reports USB Version \*1.1 or 2.0 (Warning: Setting this bit will result in the device being non-compliant with the USB 2.0 specification.)**

1 – Device will report itself as USB version 1.1 in the bcdUSB device descriptor.

0 (default) – Device will report itself as USB version 2.0 in the bcdUSB device descriptor.

**Byte 2, bit 5: Use a Common Media Insert / Media Activity LED**

1 – The activity LED will function as a common media inserted/media access LED.

0 (default) – The activity LED will remain in its idle state until media is accessed.

**Byte 2, bit 6: Reserved**

**Byte 2, bit 7: Skip Page Status Byte Check on SM and xD**

1 – Ignore data status byte check (increases performance, but risks sending corrupted data to host).

0 (default) – Perform normal checking.

**Card Reader Attribute Bit Definitions (cont.)****Byte 3, bit 0: Attach on Card Insert / Detach on Card Removal**

- 1 – The device will attach to the host when media is inserted and detach from the host when media is removed.
- 0 (default) – The device will always remain attached while powered, regardless of the presence or absence of media.

**Byte 3, bit 1: Enable xD Door Support**

- 1 – Adds support for using an xD door by moving the activity LED to GPIO 12, using GPIO 1 as an xD door input, and using GPIO 4 as a media detect pin (see hardware schematic).
- 0 (default) – All GPIOs retain their normal function.

**Byte 3, bit 2: Use Lun Power Configuration**

- 1 – Custom LUN Power Configuration stored in the NVSTORE is used. Refer to section “LUN Power Configuration” section for additional information about this feature.
- 0 (default) – Default LUN Power Configuration is used.

**Byte 3, bit 3: Reserved – always set to 0.****Byte 3, bit 4: HS-MMC enable bit. (Set or cleared by dropdown option in MMC-4 section)**

- 1 – Allows HS-MMC cards to operate at the maximum clock frequency (24MHz or 48MHz) specified by the card. Will automatically adjust the clock rate back to 20MHz maximum if any card errors are detected for a given card insertion. Although this mode has been shown to work with known non-compliant HS-MMC cards, SMSC cannot guarantee that any non-compliant card can operate, either with this bit set or cleared.
- 0 (default) – Forces all MMC cards to operate at a maximum of 20MHz. Some HS-MMC cards have been found to have non-compliant data hold timing and will fail with SMSC’s product if operated above 20MHz.

**Byte 4, bit 6 Smart Media Player Mode**

- 1 – Do not erase blocks while resolving mapping conflicts.
- 0 (default) – Erase blocks while resolving mapping conflicts.

**Byte 4, bit 7 xD Player Mode**

- 1 – Enable xD player mode. Do not erase blocks when resolving mapping conflicts. Media marked as write-protected.
- 0 (default) – xD player mode disabled.

**Note:** Bits that are used for the dropdown options in the “IrDA” and “CIR” sections of the Configuration tab are not used with the USB2602 and changing these settings will have no effect on the device.

**All other bits in the Attribute fields are reserved and should be set to 0.**

**Note:** The above defined attribute bits pertain to the card reader portion of the USB2602 Hub/Card Reader combo only. The configuration bytes in the following section pertain to the hub portion only.

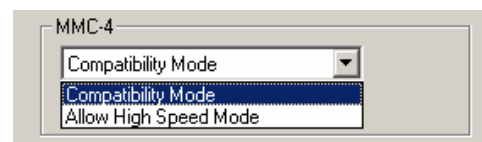
**Setting the MMC-4 Clock Speed and Card Power Management Bits:**

The MMC-4 Card Power Management bits are Byte 3, bit 5 and Byte 3, bit 6.

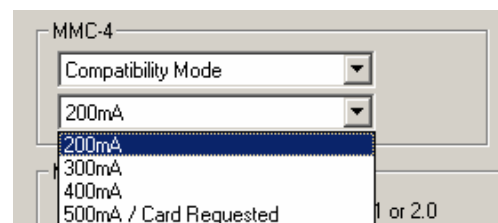
Most attribute bits are set by placing a check to the left of an option. This is true for all attribute bits except Byte 3, bit 4; Byte 3, bit 5; and Byte 3, bit 6. A dropdown box sets or clears these bits. Dropdown A, shown to the right, is used to set or clear Byte 3, bit 4.

Dropdown B, shown to the right, is used to set Byte 3, bit 5 and Byte 3, bit 6 by selecting the appropriate option in the dropdown box. The table below shows how each the dropdown choice correlates to the MMC-4 Card Power Management bits.

Dropdown option (Current allowed)	Byte 3, bit 6	Byte 3, bit 5
200mA (default)	0	0
300mA	0	1
400mA	1	0
500mA or whatever the card requests	1	1



Dropdown A



Dropdown B



**Hub Configuration Definitions:**

**Note:** It is strongly recommended not to change the hub configuration bytes from the default values. Certain configurations could cause the USB2602 to become non-functional. Please contact a SMSC Applications Engineer to discuss any needed changes to this tab before updating non-default hub configuration information into a USB2602.

Configuration (in hex)	
9B HUB Options Byte #1	01 Maximum Power Self-Powered
18 HUB Options Byte #2	64 Maximum Power Bus-Powered
02 Non Removable Device	01 Hub Controller Max Current Self-Powered
00 Port Disable Self-Powered	64 Hub Controller Max Current Bus-Powered
00 Port Disable Bus-Powered	32 Power-On Time

**Configuration (10 bytes):** The configuration for the hub portion of the 2602 device is determined by the bytes (in hex) that are entered in the hub tab of USBDM. The default values are explained in detail below, followed by tables that describe all of the possible options for these configurations. Please see a SMSC Applications Engineer if you wish to program your device with non-default values to ensure that the configuration is valid.

**HUB Options Byte #1: 9B (default)**

Configures hub to be self-powered, with no LED indicators, High/Full speed supported, multi TTs supported, EOP generation disabled, and individual port-by-port switching and over current sense.

See below table **7.2.9.8 Register 07h: CONFIG\_BYTE\_1 (Reset = 0x00)** for additional details

**HUB Options Byte #2: 18 (default)**

Configures hub to not have dynamic auto-switching, have a 2ms over current timer delay, report it is part of a compound device, and have normal electrical drive strength on the USB D+ and D- lines.

See below table **7.2.9.9 Register 08h: Configuration Data Byte 2 (Reset = 0x00)** for additional details.

**Non-Removable Device: 02 (default)**

Configures port 1 as non-removable and ports 2-4 as removable.

See below table **7.2.9.10 Register 09h: Non-Removable Device (Reset = 0x00)** for additional details.

**Port Disable Self-Powered: 00 (default)**

Configures all ports to be enabled.

See below table **7.2.9.11 Register 0Ah: Port Disable For Self-Powered Operation (Reset = 0x00)** for additional details.

**CAUTION:** Bit 1 is reserved and must always be set to 1 to ensure that **port 1 is always enabled**. Disabling this port will result in the USB2602 becoming non-functional until the eeprom is removed and replaced with a blank eeprom.

**Port Disable Bus-Powered: 00 (default)**

Configures all ports to be enabled.

See below table **7.2.9.12 Register 0Bh: Port Disable For Bus Powered Operation (Reset = 0x00)** for additional details.

**CAUTION:** Bit 1 is reserved and must always be set to 1 to ensure that **port 1 is always enabled**. Disabling this port will result in the USB2602 becoming non-functional until the eeprom is removed and replaced with a blank eeprom.

**Maximum Power Self-Powered: 01 (default)**

Configures the maximum power that the hub consumes from an upstream port when operating as a self-powered hub to be 2mA.

See below table **7.2.9.13 Register 0Ch: Max Power For Self-Powered Operation (Reset = 0x00)** for additional details.

**Maximum Power Bus-Powered: 64 (default)**

Configures the maximum power that the hub consumes from an upstream port when operating as a bus-powered hub to be 200mA.

See below table **7.2.9.14 Register 0Dh: Max Power For Bus Powered Operation (Reset = 0x00)** for additional details.

**Hub Controller Max Current Self-Powered: 01 (default)**

Configures the maximum power that the hub consumes from an upstream port when operating as a bus-powered hub to be 2mA.

See below table **7.2.9.15 Register 0Eh: Hub Controller Max Current For Self Powered Operation (Reset = 0x00)** for additional details.

**Hub Controller Max Current Bus-Powered: 64 (default)**

Configures the maximum power that the hub consumes from an upstream port when operating as a bus-powered hub to be 200mA.

See below table **7.2.9.16 Register 0Fh: Hub Controller Max Current For Bus Powered Operation (Reset = 0x00)** for additional details.

**Power-On Time: 32 (default)**

Specifies that it takes 100ms from the time the host initiated power-on sequence begins on a port until power is good on that port.

See below table **7.2.9.17 Register 10h: Power-On Time (Reset = 0x00)** for additional details.

**7.2.9.8 Register 07h: CONFIG\_BYTE\_1 (Reset = 0x00)**

<b>BIT NUMBER</b>	<b>BIT NAME</b>	<b>DESCRIPTION</b>
7	SELF_BUS_PWR	Self or Bus Power: Selects between Self- and Bus-Powered operation.  0 = Bus-Powered operation. 1 = Self-Powered operation.
6	PORT_IND	Port Indicator Support: Indicates implementation of LED indicators  0 = No LED indicators. 1 = LED indicators.
5	HS_DISABLE	High Speed Disable: Disables the capability to attach as either a High/Full-speed device, and forces attachment as Full-speed only i.e. (no High-Speed support).  0 = High-/Full-Speed. 1 = Full-Speed-Only (High-Speed disabled!)
4	MTT_ENABLE	Multi-TT enable: Enables one transaction translator per port operation.  0 = single TT for all ports. 1 = one TT per port (multiple TTs supported)
3	EOP_DISABLE	EOP Disable: Disables EOP generation of EOF1 when in Full-Speed mode.  0 = EOP generation is normal. 1 = EOP generation is disabled.
2:1	CURRENT_SNS	Over Current Sense: Indicates whether current sensing is on a port-by-port basis, ganged, or no over current sensing.  00 = Ganged sensing (all ports together). 01 = Individual port-by-port. 1x = Over current sensing not supported. (must only be used with Bus-Powered configurations!)
0	PORT_PWR	Port Power Switching: Indicates whether port power switching is on a port-by-port basis or ganged.  0 = Ganged switching (all ports together) 1 = Individual port-by-port switching.



**7.2.9.9 Register 08h: Configuration Data Byte 2 (Reset = 0x00)**

BIT NUMBER	BIT NAME	DESCRIPTION
7	Reserved	Always set to 0
6	Reserved	Always set to 0
5:4	OC_TIMER	Over Current Timer: Over Current Timer delay.  00 = 0.1ms 01 = 2ms 10 = 4ms 11 = 6ms
3	Reserved	Always set to 1
2:1	Reserved	Always set to 00
0	BOOST_IOUT	Boost the Drive strength on the USB D+ and D- lines.  '0' = Normal Electrical Drive Strength '1' = Elevated Electrical Drive Strength

**7.2.9.10 Register 09h: Non-Removable Device (Reset = 0x00)**

BIT NUMBER	BIT NAME	DESCRIPTION
7:0	NR_DEVICE	Non-Removable Device: Indicates which port(s) include non-removable devices. '0' = port is removable, '1' = port is non-removable.  Bit 7= 1; Reserved Bit 6= 1; Reserved Bit 5= 1; Reserved Bit 4= 1; Port 4 non-removable. Bit 3= 1; Port 3 non-removable. Bit 2= 1; Port 2 non-removable. Bit 1 is Reserved, always = '1'. Bit 0 is Reserved, always = '0'.

**7.2.9.11 Register 0Ah: Port Disable For Self Powered Operation (Reset = 0x00)**

BIT NUMBER	BIT NAME	DESCRIPTION
7:0	PORT_DIS_SP	Port Disable Self-Powered: Disables 1 or more contiguous ports. '0' = port is available, '1' = port is disabled.  Bit 7= Reserved Bit 6= Reserved Bit 5= Reserved Bit 4= 1; Port 4 is disabled. Bit 3= 1; Port 3 is disabled. Bit 2= 1; Port 2 is disabled. Bit 0 is Reserved, always = '0' Bit 0 is Reserved, always = '0'

**7.2.9.12 Register 0Bh: Port Disable For Bus Powered Operation (Reset = 0x00)**

<b>BIT NUMBER</b>	<b>BIT NAME</b>	<b>DESCRIPTION</b>
7:0	PORT_DIS_BP	<p>7:0 Port Disable Bus-Powered: Disables 1 or more contiguous ports. '0' = port is available, '1' = port is disabled.</p> <p>Bit 7= Reserved            Bit 6= Reserved            Bit 5= Reserved            Bit 4= 1; Port 4 is disabled. (note: this bit is reserved for the 3-Port)            Bit 3= 1; Port 3 is disabled.            Bit 2= 1; Port 2 is disabled.            Bit 0 is Reserved, always = '0'            Bit 0 is Reserved, always = '0'</p>

**7.2.9.13 Register 0Ch: Max Power For Self Powered Operation (Reset = 0x00)**

<b>BIT NUMBER</b>	<b>BIT NAME</b>	<b>DESCRIPTION</b>
7:0	MAX_PWR_SP	<p>Max Power Self Powered: Value in 2mA increments that the Hub consumes from an upstream port (VBUS) when operating as a self-powered hub. This value includes the hub silicon along with the combined power consumption (from VBUS) of all associated circuitry on the board. This value also includes the power consumption of a permanently attached peripheral if the hub is configured as a compound device, and the embedded peripheral reports 0mA in its descriptors.</p> <p>Note: The USB2.0 Specification does not permit this value to exceed 100mA</p> <p>A value of 50 (decimal) indicates 100mA.</p>

**7.2.9.14 Register 0Dh: Max Power For Bus Powered Operation (Reset = 0x00)**

<b>BIT NUMBER</b>	<b>BIT NAME</b>	<b>DESCRIPTION</b>
7:0	MAX_PWR_BP	<p>Max Power Self Powered: Value in 2mA increments that the Hub consumes from an upstream port (VBUS) when operating as a self-powered hub. This value includes the hub silicon along with the combined power consumption (from VBUS) of all associated circuitry on the board. This value also includes the power consumption of a permanently attached peripheral if the hub is configured as a compound device, and the embedded peripheral reports 0mA in its descriptors.</p> <p>A value of 50 (decimal) indicates 100mA.</p>

**7.2.9.15 Register 0Eh: Hub Controller Max Current For Self Powered Operation (Reset = 0x00)**

<b>BIT NUMBER</b>	<b>BIT NAME</b>	<b>DESCRIPTION</b>
7:0	HC_MAX_C_SP	<p>Hub Controller Max Current Self-Powered: Value in 2mA increments that the Hub consumes from an upstream port (VBUS) when operating as a self-powered hub. This value includes the hub silicon along with the combined power consumption (from VBUS) of all associated circuitry on the board. This value does NOT include the power consumption of a permanently attached peripheral if the hub is configured as a compound device.</p> <p>Note: The USB2.0 Specification does not permit this value to exceed 100mA</p> <p>A value of 50 (decimal) indicates 100mA, which is the default value.</p>

**7.2.9.16 Register 0Fh: Hub Controller Max Current For Bus Powered Operation (Reset = 0x00)**

<b>BIT NUMBER</b>	<b>BIT NAME</b>	<b>DESCRIPTION</b>
7:0	HC_MAX_C_BP	<p>Hub Controller Max Current Self-Powered: Value in 2mA increments that the Hub consumes from an upstream port (VBUS) when operating as a self-powered hub. This value will include the hub silicon along with the combined power consumption (from VBUS) of all associated circuitry on the board. This value will NOT include the power consumption of a permanently attached peripheral if the hub is configured as a compound device.</p> <p>A value of 50 (decimal) would indicate 100mA, which is the default value.</p>

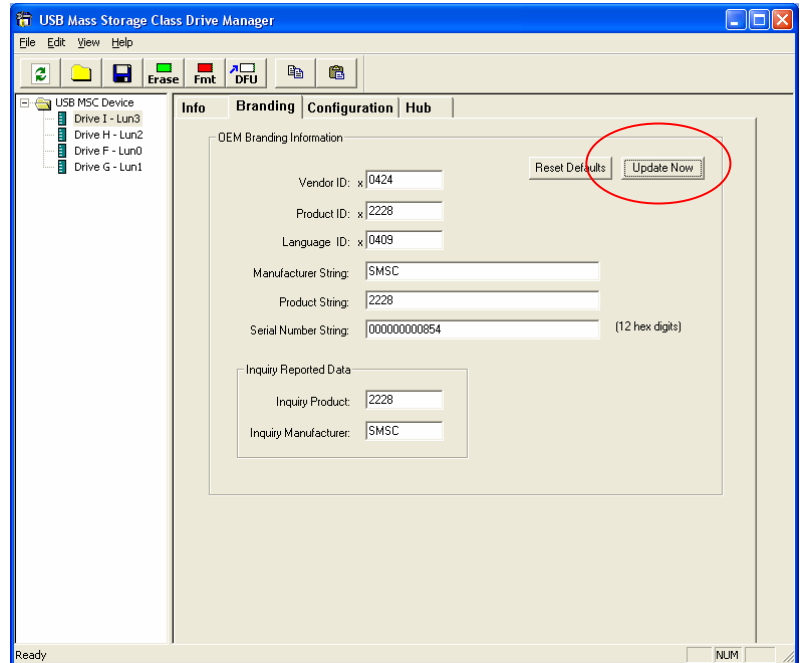
**7.2.9.17 Register 10h: Power-On Time (Reset = 0x00)**

<b>BIT NUMBER</b>	<b>BIT NAME</b>	<b>DESCRIPTION</b>
7:0	POWER_ON_TIME	<p>Power On Time: The length of time that it takes (in 2 ms intervals) from the time the host initiated power-on sequence begins on a port until power is good on that port.</p>

### Programming the NVStore Data

Once the eeprom.dat file has been created and loaded into USBDM, you are ready to program the NVStore data into your device.

Press the “Update Now” button on either the Branding or Configuration Tab of the USBDM application. Both buttons will update all of the information displayed on any tab in USBDM. The operation will report that the Update completed successfully once the data has been programmed.



## LUN Configuration and Icon Sharing

### LUN Configuration

LUN (Logical Unit Number) is the term given to each available media type in the USB2602. The USB2602 has a total of 4 LUNs available for use: Compact Flash, Memory Stick, Smart Media, and Secure Digital/Multimedia Card. OEMs can specify the number and order of LUNs exposed to the user by setting the LUN Configuration section of the Configuration tab in USBDM and updating the NVSTORE with these new settings.

**Example:** The example on the right shows the correct settings for a 2602 device that exposes icons for MS, SM and CF in that order. Note the following bytes:

**Number of Icons to Display: “03”** (The user will see 3 icons)  
**MS LUN #: “00”** (Memory Stick will be the 1<sup>st</sup> icon displayed)  
**SM LUN #: “01”** (Smart Media will be the 2<sup>nd</sup> icon displayed)  
**CF LUN #: “02”** (Compact Flash will be the 3<sup>rd</sup> icon displayed)  
**SD/MMC LUN #: “FF”** (An icon for SD/MMC will not be displayed)

**Note:** LUN numbering always starts at “00”.

LUN Configuration			
03	# of Icons to Display		
02	Compact Flash	CF	LUN#0 - ID
00	Memory Stick	MS	LUN#1 - ID
01	Smart Media	SM	LUN#2 - ID
FF	Secure Digital/MMC	SD/MMC	LUN#3 - ID
FF	NAND	x FFFF	NAND Profile

### Icon Sharing

In addition to LUN configuration, the USB2602 can be further customized to allow more than one LUN to share an icon. This functionality would most likely be used for devices that contain multi-card adapters (adapters that can read more than one type of card). So if you wanted to use a “5-in-1” or a “6-in-1” adapter, the USB2602 could be configured to only display a single icon to the user, rather than an icon for each individual media type. Alternatively, if you wanted to use a “4-in-1” adapter for Memory Stick, Smart Media, Secure Digital, and Multimedia Card, but have a separate adapter for Compact Flash, you could configure the USB2602 to display 2 icons to the user (one for the 4-in-1 adapter and one for the Compact Flash) as shown in the example on the right.

**Example:** The example on the right shows the correct settings for a 2602 device that exposes 2 icons: 1 for (CF) and 1 for (MS, SM, and SD/MMC) in that order. Note the following bytes:

**Number of Icons to Display: “02”** (The user will see 2 icons)  
**CF LUN #: “00”** (Compact Flash will be the 1<sup>st</sup> icon displayed)  
**MS LUN #: “01”**  
**SM LUN #: “01”**  
**SD/MMC LUN #: “01”** } (These media will all share a single icon)

LUN Configuration			
02	# of Icons to Display		
00	Compact Flash	CF	LUN#0 - ID
01	Memory Stick	MS	LUN#1 - ID
01	Smart Media	SM	LUN#2 - ID
01	Secure Digital/MMC	SD/MMC	LUN#3 - ID
FF	NAND	x FFFF	NAND Profile

## LUN Power Configuration

The LUN Power Configuration allows the user to customize which GPIOs control power to which LUNs. Without this feature, users designing card readers that utilize multi-card sockets (sockets which can accept different flash card types) must include one FET for each card that the socket supports. Therefore, if a socket can accept any card type, the board design must include 4 FETs even though only 1 FET is active at a time. In order to reduce cost, only one FET is needed per socket. Users can set the LUN Power Configuration to have a single GPIO control power to the FET to deliver power to the multi-card socket, instead of requiring 4 GPIOs to power 4 FETs independently.

An additional feature for the 2602 is that it has 3 internal FETs which can be utilized instead of external FETs. The LUN Power Configuration feature allows any card (except CF cards) to be powered either by an external FET or internal FET (CF cards can ONLY be powered by an external FET). Also, any card (except CF cards) can be powered by any combination of internal FETs. These features are configured via the NVSTORE settings. These configurations are described below.

**In order to use this feature the user must set the “Use LUN Power Configuration” bit (Attribute byte 3 bit 2) and assign a valid hexadecimal number to the “LUN Pwr Cfg” byte (byte 172), “LUN Power Mask 1,” and “LUN Power Mask 2” in the NVSTORE.**

The format of the NVStore LUN Pwr Cfg byte is as follows:

Bit							
7	6	5	4	3	2	1	0
SD Power GPIO		SM Power GPIO		MS Power GPIO		CF Power GPIO	

The Power GPIO field for each of the sockets shall be defined as follows:

Bit 1	Bit 0	Power GPIO
0	0	Use external FET, connected with GPIO 8,9,10 and/or 11
0	1	Use Internal FET, connected with GPIO 8, 10, or 11
1	0	Reserved
1	1	Reserved

By default the LUN Power Configuration byte will be as follows:

	LUN Power Configuration								Definition
	7	6	5	4	3	2	1	0	
CF							0	0	Use External FET
MS					0	1			Use Internal FET
SM			0	1					Use Internal FET
SD/MMC	0	1							Use Internal FET
	54h								

The above chart shows SD being powered by internal FET, SM powered by internal FET, MS powered by internal FET, and CF powered by external FET.

Note: When the appropriate attribute byte LUN Power GPIO is changed, the behavior of the Power GPIOs will change to that specified above regardless of LUN configuration.

## LUN Power Masks

The LUN Power Masks are 4-bit fields that represent which GPIOs or FETs are configured for use with each LUN. The mask definition is different, depending on how the LUN is configured.

Power Mask Table

Config	Mask	FET(s)	PIN(s)
00	0001	External	GPIO 8
00	0010	External	GPIO 9
00	0100	External	GPIO 10
00	1000	External	GPIO 11
01	0001	Internal FET 0	GPIO 8
01	0010	Internal FET 1	GPIO 10
01	0011	Internal FET 0 and Internal FET 1	GPIO 8 and GPIO 10
01	0100	Internal FET 2	GPIO 11
01	0101	Internal FET 0 and Internal FET 2	GPIO 8 and GPIO 11
01	0110	Internal FET 1 and Internal FET 2	GPIO 10 and GPIO 11
01	0111	Internal FET 1 and Internal FET 2 and Internal FET 3	GPIO 8 and GPIO 10 and GPIO 11
01	1xxx	Invalid	Invalid

### LUN Power Mask 1

LUN Power Mask 1 contains the power mask setting for CF and MS controllers. The mask used depends on how the LUN is configured in the LUN Power Configuration byte.

Bit							
7	6	5	4	3	2	1	0
MS Power Mask Default: 0001 – Internal FET 0				CF Power Mask Default: 0010 – External FET GPIO 9			

Default value for LUN Power Mask 1 is 0x12

### LUN Power Mask 2

LUN Power Mask 2 contains the power mask setting for SM and SD controllers. The mask used depends on how the LUN is configured in the LUN Power Configuration byte.

Bit							
7	6	5	4	3	2	1	0
SD Power Mask Default: 0100 – Internal FET 2				SM Power Mask Default: 0010 – Internal FET 1			

Default value for LUN Power Mask 2 is 0x42

**Example:** The Icon Sharing example in the previous sections describes a device with 2 icons: 1 for (CF) and 1 for (MS, SM and SD/MMC) in that order. Since MS, SM and SD/MMC are all sharing a socket in that example only 2 FETs would be needed. The Lun Power Configuration feature can be used to assign two GPIOs to power these LUNs instead of the four GPIOs used by default. Suppose for example that the user would like the CF slot to be powered externally by GPIO 9 and the combo slot to be powered internally by FET0. First the user would set the attribute bit “Use LUN Pwr Config.” Then the user would set the LUN Power Configuration byte to 0x54, the LUN Power Mask 1 to 0x12, and the LUN Power Mask 2 to 0x11. (See tables below for how this value is found.)

0x54 Example:

	LUN Power Configuration								Definition
	7	6	5	4	3	2	1	0	
CF							0	0	Use External FET
MS					0	1			Use Internal FET
SM			0	1					Use Internal FET
SD/MMC	0	1							Use Internal FET
	54h								

For MS, SM, and SD combo slot: From the Power Mask Table above we know that if the LUN Power Configuration for the media slot type is set to 0x01 and the desired power is from Internal FET 0, then the Power Mask for that media slot type is 0x0001.

For CF slot: From the Power Mask Table above we know that if the LUN Power Configuration for the media slot type is set to 0x00 and the desired power is from External GPIO9, then the Power Mask for that media slot type is 0x0010.

This information can be used to determine the LUN Power Mask bytes as follows:

Bit							
7	6	5	4	3	2	1	0
MS Power Mask				CF Power Mask			
0001 – Internal FET 0				0010 – External FET GPIO 9			
1				2			

LUN Power Mask 1 is 0x12

Bit							
7	6	5	4	3	2	1	0
SD Power Mask				SM Power Mask			
0001 – Internal FET 0				0001 – Internal FET 0			
1				1			

LUN Power Mask 2 is 0x11



## **Using Device Firmware Upgrade (DFU)**

### **Overview**

Device Firmware Upgrade (DFU) is the process by which device firmware is updated through a standard USB cable, eliminating the need to remove, reprogram, and replace flash memory. This operation is accomplished by placing special code into an external flash memory chip at the time it is initially programmed. This code can then later be called upon to essentially change the USB device into a flash programmable device. New firmware can then be uploaded to the device and reprogrammed into the flash. Once the operation is complete, the device configures itself back to a normal USB device and begins utilizing the new firmware. **Please note that you can not perform a device firmware upgrade if you are running from the internal USB2602 ROM code. You must use an external flash if you want to have device firmware upgrade capability.**

SMSC's Device Firmware Upgrade (DFU) package gives manufacturers the ability to easily utilize DFU to dynamically update the firmware and descriptor information in their devices. This allows for in circuit programming of new device firmware both on the assembly line, and by the end user in the field. This affords both the manufacturer and the end user a great opportunity to utilize the feature enhancements and bug fixes of new code immediately once it becomes available.

In order to help customers evaluate the DFU technology, SMSC provides a DFU package that consists of the DFU driver, device firmware, sample DFU applications and source code, and a DFU driver API which customers can use to quickly develop custom DFU applications. SMSC also provides a DFU package for Mac 10.X systems. This document serves to describe the use of these tools, and the implementation of Device Firmware Upgrade in a typical device application.

### **Files Required for DFU for Windows**

**USBDM.exe**—A sample DFU application that demonstrates the use of the API and the procedure for updating the firmware and NVStore data.

**eeprom.dat**—A text file containing the changeable descriptor information used to update the NVStore. This file can be created and edited in the DAT Editor (under the Tools menu) in the DFUTest application.

**hex2bin.exe**—A batch capable utility that converts INTEL HEX, MOTOROLA 'S', or TEKTRONIX HEX files to Binary Format.

**dfu.exe**—A utility used to add, remove, or check for the presence of a DFU file suffix. Any firmware image that is to be uploaded to a device via DFU should contain a valid DFU file suffix.

**dfu.hex**—The DFU execution code that is inserted into the lower 64kb of a 128kb flash when it is initially programmed. This hex file is converted to a 64kb binary file with the "hex2bin.exe" utility, and then appended to the 64kb "fmc.bin" file to create the 128kb flash image (included with the USB2602 firmware).

**fmc.hex**—The USB2602 device firmware that is inserted into the upper 64kb of a 128kb flash when it is initially programmed. This hex file is converted to a 64kb binary file with the "hex2bin.exe" utility, and then appended to the 64kb "dfu.bin" file to create the 128kb flash image (included with the USB2602 firmware).

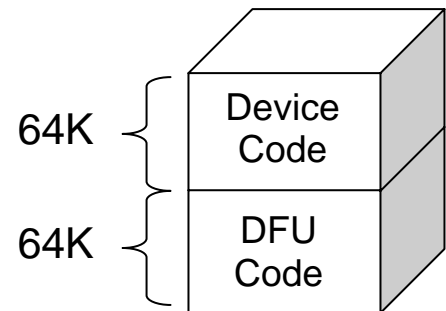
**fmc.dfu**—A firmware image that can be uploaded to the device. This file is created by the user. This document explains in detail how to make downloadable DFU images through the use of the "DFU.exe" utility, which appends a DFU file suffix to the firmware file to be uploaded to the device. (This file is created by the user.)

**Application Source Code**—All of the source code for the USBDM sample application.

**Creating the 128KB DFU Capable Flash Binary “both.bin”**

In order to prepare a device for DFU operation, the flash must be programmed with both the DFU code, and the normal USB2602 device code. The device code is converted to a 64KB binary file, and appended to the DFU code, which has also been converted to a 64KB binary file. Together they form the 128KB binary file which is uploaded to the flash eeprom. When this file is uploaded to the flash, the DFU code occupies the lower 64KB block, and the device code occupies the upper 64KB block.

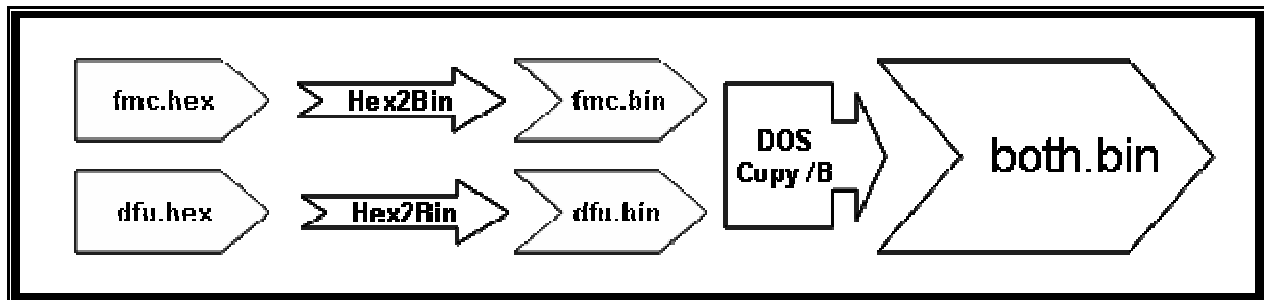
In normal operation, a DFU capable USB2602 device executes only the device code in the upper 64KB block of memory. This code allows it to function as a normal USB 2.0 flash media controller. However, when the device is switched to DFU mode, the DFU code in the lower 64KB block begins executing and the device ceases to be a flash media device. Essentially, it changes to become an eeprom programming device. In this mode it is capable of reprogramming the USB2602 device code in the upper 64KB block of flash memory. Once the operation is complete, the device switches code execution back to the upper bank and begins operating with the newly updated code. At this point it ceases to be an eeprom programming device, and returns to being a flash media device.

**128KB Flash EEPROM**

To create the 128KB DFU capable flash binary file that will initially be programmed into the flash eeprom, you will need two files:

- 1) `fmc.hex` (The device code)
- 2) `dfu.hex` (The DFU code)

The “dfu.hex” file is provided by SMSC and provides programming support for a limited number of eeproms. The “fmc.hex” file is the standard USB2602 device firmware. These two files, “dfu.hex” and “fmc.hex,” are both converted to 64KB binary files with the “hex2bin.exe” utility, and then appended to each other with a DOS copy command. Together they become the 128KB binary file “both.bin.” The procedure for creating “both.bin” is outlined below.



Note that this entire procedure can be accomplished easily using a simple DOS batch file:

```

hex2bin -L65536 dfu.hex dfu.bin
hex2bin -L65534 fmc.hex fmc.bin
copy /Y /B dfu.bin /B + fmc.bin /B both.bin /B
  
```

### **Preparing a Device for DFU Operation**

In order to prepare a device for DFU operation, the flash must initially be programmed with the “both.bin” code. The “both.bin” file contains both the device code as well as the DFU code. The DFU code must preexist on the flash in order for it to be capable of receiving a DFU upload. The DFU code remains dormant in the lower 64KB of memory until it is called upon to perform a device firmware upgrade operation.

Once the flash has been programmed with the “both.bin” file, it may be inserted into the 2602’s flash socket in preparation for DFU operation.

### **Choosing a Flash Eeprom for Your Device**

SMSC provides customers the “dfu.hex” file that supports only the **SST39VF010**, **AM29LV010B**, **AM29LV040B**, **STM29W010B**, **MBM29LV400TC**, and the **MBM29LV200TC** flash eeproms. While all of these flash support DFU firmware uploads, only the SST39VF010 supports NO EEPROM operation.

If you wish to use another flash in your device, it would most likely require some modification to the existing DFU code by SMSC to support the electrical characteristics of the new chip. If this is the case, please contact SMSC sales to have the project scheduled.

If you do decide to use another flash eeprom, there are a few requirements to look for to make sure it will work with DFU. First of all it should be 128KB and byte writable. Also, it should have equivalent programming characteristics as the three supported chips, i.e. block size, erase size, read/write/erase speed, command set, and command address. Provided the chip meets all of the above requirements, there is a good chance that it will support DFU.

### **Setting up the Hardware**

Either a USB 1.1 or 2.0 controller may be used for the DFU operation; however, some USB 2.0 host controller drivers such as OMIs have been found to have defects which prevent DFU from performing normally. If you are going to use a USB 2.0 host controller, it is recommended that you use Microsoft’s host controller drivers in order to achieve the best results. Once the board is attached and powered up, it should enumerate as a normal USB flash media controller. When you see the drive icon(s) appear, the device is ready. Currently only USB 2.0 may be used for the DFU operation when using Macintosh operating systems. The following section describes the next step in the process, which is setting up the software application to perform the DFU.

**Using the USBDM Application to Perform Device Firmware Upgrade (DFU)**

The following files are needed to perform a device firmware upgrade with the USBDM application:

1. The USBDM application executable (USBDM.exe).
2. The device code (both.bin). **\*Must be preprogrammed in the device flash in order to accept DFU**
3. A HEX to BIN converter (hex2bin.exe).
4. Utility to add the .dfu suffix (dfu.exe).
5. The updated firmware image. Steps to create this file are explained below (fmc.dfu).

\* Note that if you also want to perform an update of the serial eeprom, you will need a 6th file, “eeprom.dat” which contains the descriptor information for the serial eeprom.

A firmware update can only be done using this application if a valid both.bin file is already programmed onto the device. See the section of this document entitled “Creating the 128KB DFU Capable Flash Binary ‘both.bin’” for steps on how to create the both.bin file.

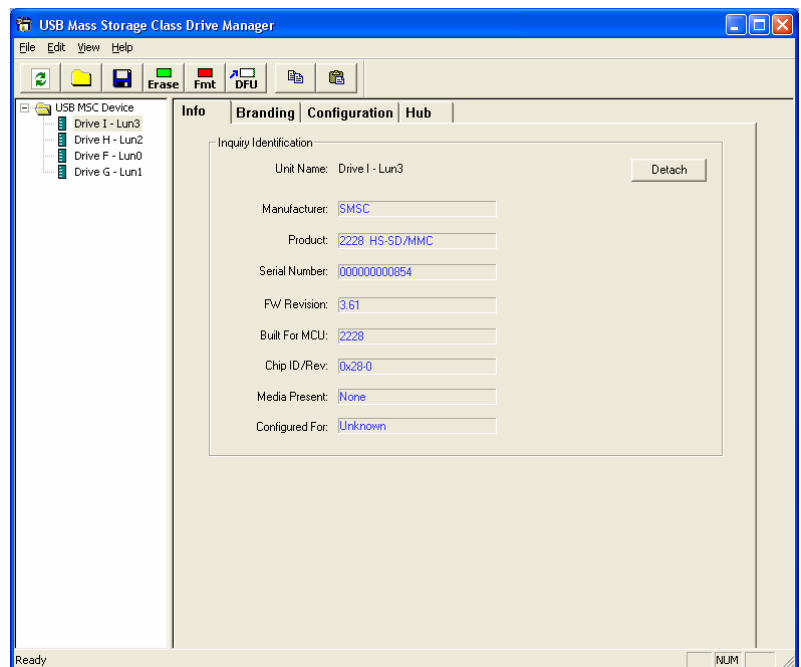
**Creating the .dfu File:**

The .dfu file is a DFU uploadable firmware image. It is the USB2602 firmware converted to binary format using the hex2bin.exe utility, with a DFU suffix appended to it. For information on creating the .dfu file, please see the section of this document entitled “Creating a DFU Uploadable File.” Please note that the USBDM application uses the device ID field (DID) to check firmware version information. The DID field should be filled with the major and minor firmware version (for this example, v3.00, the DID would be 0x0300).

This procedure can be completed using a simple DOS batch file:

```
hex2bin -l65534 fmc.hex fmc.bin
dfu fmc.bin -did 0x0395 -pid 0x2228 -vid 0x0424
ren fmc.bin fmc.dfu
```

USBDM is used for the firmware update. To begin the firmware update, start USBDM by double clicking on the icon.

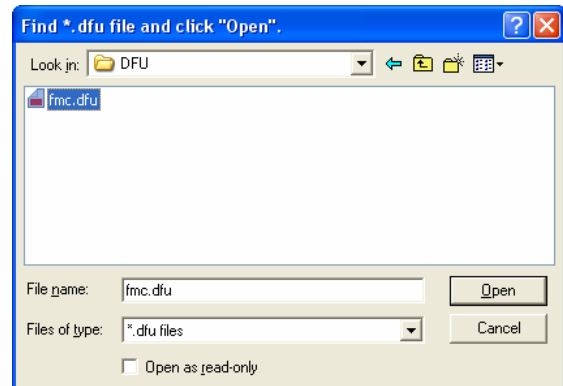


## Updating the Firmware:

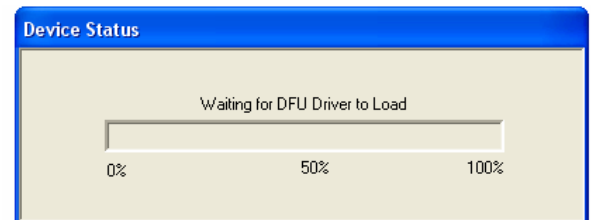
To perform a firmware update, click on the “Upload Firmware” button at the top of the application.



You will then be prompted to select the .dfu file that you wish to upload to your device. Navigate to the .dfu file (if it is not already listed in the current folder) and click open.



You will see a pop up box on your screen that displays the status of the firmware upload. This status will cycle through “Waiting for DFU Driver to Load,” “Switching to DFU Mode,” “Uploading New Firmware,” “Validating New Firmware,” and “Firmware Upload Successful.” Once the loading is complete you will be prompted to unplug the device and reattach it to continue (or to restart the host if the device is internally mounted). Once the device is reattached, the device will enumerate and the information for the updated firmware will be loaded into the USB Drive Manager application.



Note: The first time USBDM is used for DFU on a Windows XP host, the found new hardware wizard will be seen when the dfu driver is used during the firmware update process. This will only happen the first time a DFU is performed on a host. When this comes up, choose to have windows automatically install the driver. Choose to continue loading the SMSC DFU driver even though it is unsigned. While this is occurring, you may receive a message from USBDM asking you if you wish to continue waiting for the device to respond. Select yes to continue waiting.

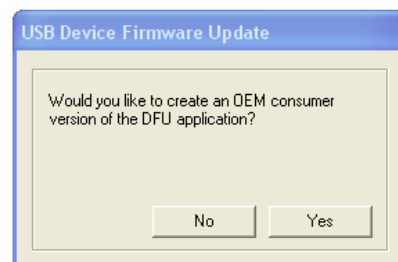
### Using USB Drive Manager to Create a Consumer Firmware Update Executable

USBDM can be used to create a very simple, easy to use, and easy to distribute firmware update that OEMs can give to their customers to allow firmware upgrades. To create the executable, you only need two files:

1. The Drive Manager application (USBDM.exe).
2. The updated firmware image (fmc.dfu).

**Note: Ensure that the DID set in the DFU file matches the Major and Minor firmware revision.**

Simply drag and drop the .dfu file on the USBDM.exe icon in Windows. You will see a popup box asking if you would like to create an OEM consumer version of the DFU application. Click yes and the application will build the consumer firmware update executable. The executable will be given the default name of “OEM.exe.” You can rename this file to whatever you like. This is the file that is distributed to the customer to allow firmware upgrades.



**Note: The target device must be preprogrammed with a valid “both.bin” file to allow firmware upgrades.**

### Using the OEM.exe to Update Firmware

The OEM executable icon is shown to the right.



1) Double click on this executable to begin updating the firmware in your target device.

2) You will be prompted to attach a supported USB device.

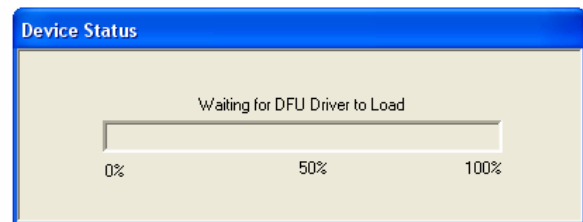
This prompt also displays which firmware version the executable will use to update your device. For this example, Firmware Version 3.00 is used.

3) Connect your device (if not connected already) and click “Continue.”

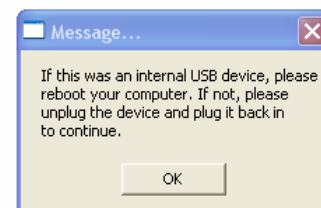


Note: This application allows consumers to make firmware updates to their device provided that 1) a valid both.bin file is already programmed on the target device and 2) the firmware that they are attempting to upgrade to is equal to or newer than the firmware version already on the device. This application will not allow an update to a version of firmware that is older than what is currently on the device. You will be asked if you would like to update your device firmware; click “yes” to verify the update and the application will begin to update your device.

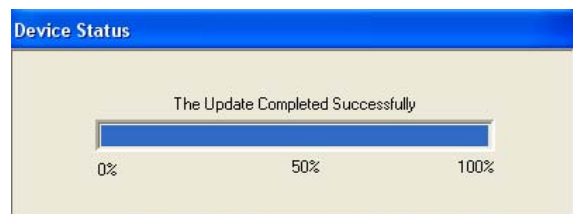
The application will show the status of the update. It will cycle through “Waiting for DFU Driver to Load,” “Switching to DFU Mode,” “Uploading New Firmware,” “Validating New Firmware,” and “Firmware Upload Successful.”



4) The USB Drive Manager application will prompt you to either reboot your computer (if an internal USB device was updated) or unplug the device and plug it back in (if an external device was updated).



After this is completed, you will see the device status pop up return with the message “The Update Completed Successfully.” The firmware is now updated on your device.

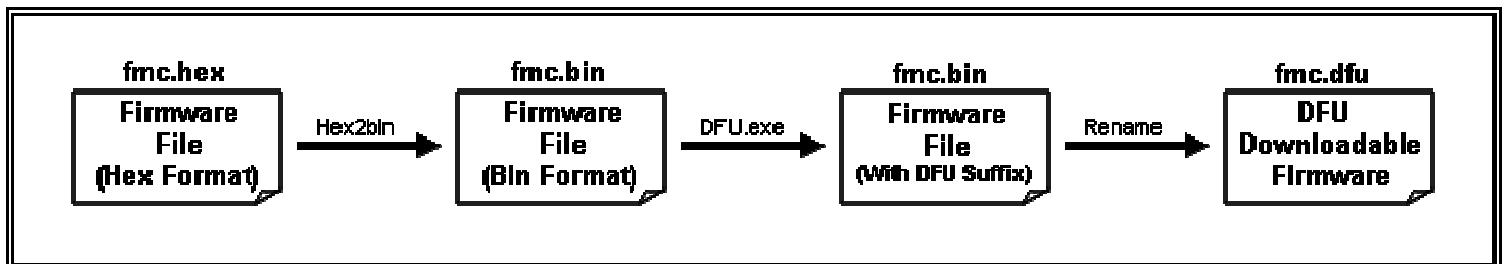


### Creating a DFU Uploadable File

In order for a file to be uploadable via a DFU operation, it must contain a valid DFU file suffix. The DFU file suffix contains a CRC of the entire file, a DFU signature, and the VID, PID, and DID for the device to be upgraded. The following table was extracted from the USB Device Firmware Upgrade Specification (Rev 1.0), and shows the composition of the DFU file suffix.

Offset	Field	Size	Value	Description
-0	<i>dwCRC</i>	4	Number	The CRC of the entire file, excluding <i>dwCRC</i> . (Calculation specified in the following section).
-4	<i>bLength</i>	1	16	The length of this DFU suffix including <i>dwCRC</i> .
-5	<i>ucDfuSignature</i>	3	uc	The unique DFU signature field.
-8	<i>bcdDFU</i>	2	BCD	DFU specification number.
-10	<i>idVendor</i>	2	ID	The vendor ID associated with this file. Either FFFFh or must match device's vendor ID.
-12	<i>idProduct</i>	2	ID	The product ID associated with this file. Either FFFFh or must match device's product ID.
-14	<i>bcdDevice</i>	2	BCD	The release number of the device associated with this file. Either FFFFh or a BCD firmware release or version number.

In the SMSC DFU application, DFU downloadable files are given the extension “.dfu”. This is strictly arbitrary; the files can be of any extension as long as the application is designed to handle them. In order to create your own DFU downloadable file, you begin with the firmware file that is going to be used to upgrade the device. If the new firmware file is not already in binary format, it should be converted to binary using the Hex2Bin utility provided. Once in binary format, the “dfu.exe” utility is used to append a valid DFU file suffix to the firmware file (See the next section titled “Using the DFU.exe Utility”). Once the DFU file suffix has been added, you may rename the file with a .dfu extension to indicate that it is DFU downloadable. The entire procedure for creating the DFU downloadable file is summarized below.



### Using the DFU.exe Utility

The “DFU.exe” utility can be used to add a DFU suffix to a file, or to check for the presence of a valid DFU suffix on an existing file. If required, the “DFU.exe” utility can also be used to remove a DFU suffix from a file. The “DFU.exe” utility is run from a command box in Windows.

The usage of DFU.exe is:

**DFU.exe <filename> [options]**

To check for the presence of a DFU file suffix:

**DFU.exe <filename>**

To remove a DFU suffix from a file:

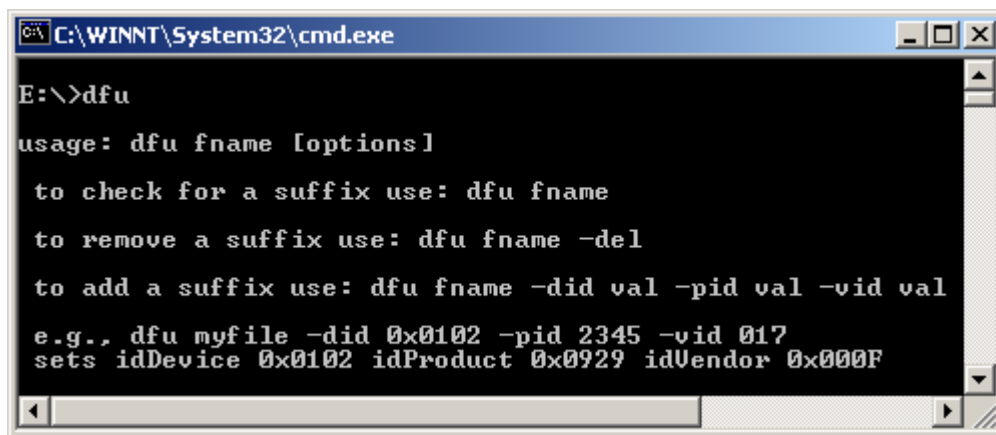
**DFU.exe <filename> -del**

To add a DFU suffix to a file:

**DFU.exe <filename> -did <val> -pid <val> -vid <val>**

Example of adding a DFU suffix to “fmc.bin”:

**DFU.exe fmc.bin -did 0xFFFF -pid 0x2602 -vid 0x0424**



```
C:\WINNT\System32\cmd.exe

E:\>dfu

usage: dfu fname [options]

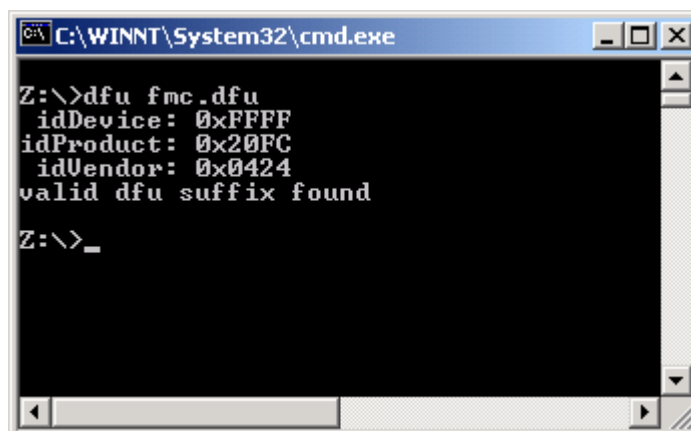
to check for a suffix use: dfu fname

to remove a suffix use: dfu fname -del

to add a suffix use: dfu fname -did val -pid val -vid val

e.g., dfu myfile -did 0x0102 -pid 2345 -vid 017
sets idDevice 0x0102 idProduct 0x0929 idVendor 0x000F
```

Once the DFU suffix has been added to the file, the last step is to give it a file extension that matches the type expected by your application. The dfuTest sample application is programmed to accept DFU uploadable files that have the “.dfu” extension. Finally, check and make sure that the file has a valid suffix:



```
C:\WINNT\System32\cmd.exe

Z:\>dfu fmc.dfu
idDevice: 0xFFFF
idProduct: 0x20FC
idVendor: 0x0424
valid dfu suffix found

Z:\>_
```



## **Performing a Firmware Upgrade with the DFU App Application(Mac 10.X Only)**

\*Note: Before attempting to use this DFU application, ensure that your device is set up properly for DFU by reviewing the section “Using Device Firmware Upgrade (DFU)”

The following files are required in order to perform a device firmware upgrade using Mac 10.X:

1. The DFU application (DFU\_App)
2. The updated firmware image (fmc.dfu)
3. smsckext.kext
4. DFU\_Drvr.framework
5. smsctoolslib.framework

### **Where to find DFU\_App**

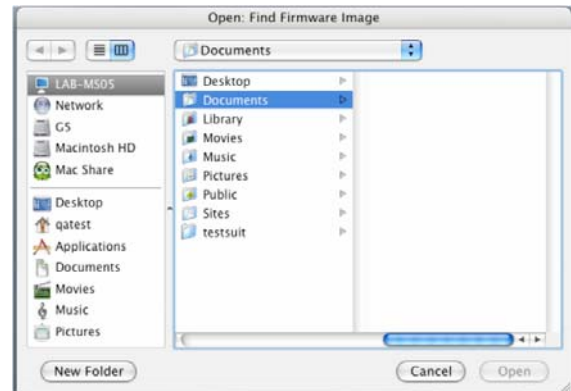
The 2602-installer package will automatically load the DFU application in the hard drive that was selected during the installation process. Open the Applications folder in this hard drive. Once in the Applications folder, open the folder created during installation called “USB Mass Storage Software.” In this folder you will find a ReadME.txt and a DFU\_App icon. (Note: if your installer package does not include the DFU feature then you will only find a ReadME.txt.) The DFU\_App.app may be copied to the desktop if desired for ease of use.

### **Using an engineering version of DFU\_App application**



To start the DFU\_App application, simply double click the DFU\_App icon.

The application will open a dialog box, which allows you to browse to the desired firmware image. The file you select must have a .dfu suffix. Refer to the previous section “Creating a DFU Uploadable File” for instructions on how to create this file. Once you have navigated to the .dfu file that contains the version of firmware you wish to upgrade to, click open. You must have a valid USB device attached to a 2.0 host controller in order for the firmware upload to complete properly. DFU for Mac 10.X is currently only supported for use with a 2.0 host controller.



After opening the file, the firmware upload will begin. The first screen you see will verify that you wish to upgrade the firmware of the USB device detected. (You may upgrade to a version of firmware that is newer than the current firmware on the device, the same as the version currently on the device, or older than the version currently on the device.) Click yes if this is the upgrade that you want. The application will then detach and reattach the device. You may get a pop up message warning you of a removal of the device. This message can be ignored.



After the device reattaches, the device will switch to DFU mode and begin downloading the new firmware. The progress of this upgrade will be shown on the message box. After the download is complete, the new firmware will be verified and the message box will display either a successful firmware update or a failure message.



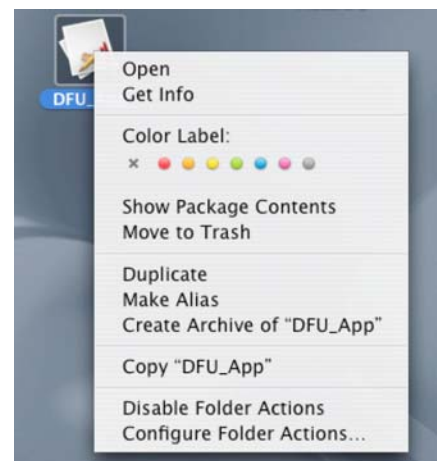
A typical firmware update takes about 1 minute to complete. Once the success message is displayed you must unplug and replug the device in order to complete the DFU process.



## Creating a customer version of DFU

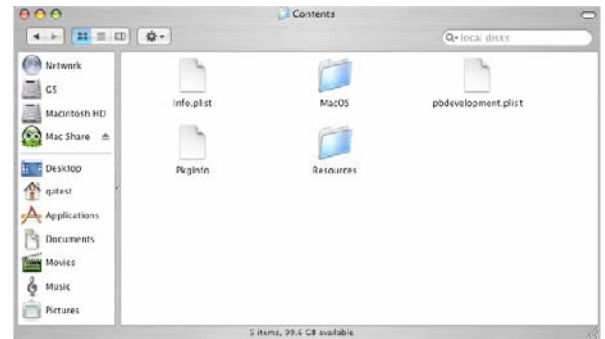
The engineering version of DFU\_App can be used to upgrade firmware or to create a customer version of the DFU\_App application. In order to prepare the DFU\_App application for customer use, a file named "fmc.dfu" that contains the firmware required by the customer must be placed in the resource folder of the application. The file must be named "fmc.dfu" in order for the application to properly recognize it as a customer version.

To navigate to the resource folder, right click on the DFU\_App icon. Select "Show Package Contents" from the drop down menu. There will be only one folder icon displayed in the DFU\_App contents. It is titled "Contents." Double click on the "Contents" folder.



The contents folder contains the items shown to the right. Drop a file named `fmc.dfu` (that has the firmware you would like the customer version of DFU\_App to contain) into the “Resources” folder. The next time the DFU\_App is started it will now recognize the `fmc.dfu` file in the resources folder and act as a customer version instead of an engineering version. Refer to the previous section “Creating a DFU Uploadable File” for instructions on how to create `fmc.dfu`.

At any time the `fmc.dfu` file can be moved from the resources folder, and the DFU\_App will act as an engineering version again, or it can be replaced with a file that is loaded with a different version of firmware.



## Using a customer version of DFU\_App

The process for uploading firmware using the customer version of DFU\_App is extremely similar to the way the firmware is uploaded using the engineering version of this application. The icon for the customer version of DFU\_App is identical to the engineering DFU\_App icon. The only difference between the engineering version and the customer version is that when the customer icon is double clicked instead of being prompted to navigate to the `dfu` file to upgrade to, the first screen the user will see is the prompt verifying that they wish to upgrade. The customer option does not give the option to choose different versions of firmware to upgrade to; whichever version was loaded into the `fmc.dfu` file contained in the resource folder is the only upgrade that can be done on the device.

The only option the customer version gives the user is whether or not they want to update to the version of firmware stored in the application. The steps to upload the firmware are identical to the steps in the previous section “Using the engineering version of DFU\_App” beginning with the screen shown to the right.



### Known Issues with DFU\_App Application (Mac 10.X Only)

1. Must use a USB 2.0 port to perform firmware update.
2. If user is unable to select an `fmc.dfu` file when using the engineering version, a customer version must be created and used as a work around.
3. If an error occurs during creating the customer version, check to see if the user has write access to the DFU\_App; if not, create a copy that is writable and try to create the customer version from this copy.
4. Not supported for Mac OS versions newer than 10.3.4.

### **Using the USB2602 Custom Icons Package**

The USB2602 custom icons package allows OEMs to assign custom icons to the drives associated with the USB2602 flash media controller. This allows the end user to easily distinguish between the different media types in Windows Explorer. The application works with Windows 2000 and Windows XP. A new feature available in SetIcon is the ability to dynamically change icons based on media state. In other words, you can specify that one icon appear if there is media in the reader slot, and another icon appear when there is no media in the reader slot. Also, the dynamic icon functionality enables the detection of MMC, MS Pro, and xD, allowing the user to display custom icons for those media types as well.

### **Contents of the USB2602 Custom Icons Package**

The USB2602 Custom Icons Package consists of the following:

**SetIcon.exe**—The custom icon application.

**Smsc.ini**—A sample Windows ini file.

**Sample Icons**—The sample icons distributed with this package are for evaluation use only.

**Eeprom.dat**—A text file containing the changeable descriptor information used to update the serial eeprom with the DFUTest utility.

### **Creating the Required SetIcon Ini Files**

In order for the SetIcon application to work properly, an ini file with a specific file name and format must be installed on the host computer. The ini file tells the SetIcon application which icons are associated with which drives, and provides a full path to each icon. The following four paragraphs describe the procedure for creating, naming, formatting, and installing the ini file on the host PC.

#### **1) Setting the Ini File Name:**

**Windows 2000 and XP** - The name of the ini file should be the same as the device's Manufacturer string, but be no longer than 8 characters. If the Manufacturer string is greater than 8 characters, then only the first 8 characters of the string should be used. If the Manufacturer string is less than 8 characters, then the ini file should use the entire Manufacturer's string.

Example: If MFG string is "Standard Microsystems Corp", the ini filename should be "Standard.ini"

Example: If MFG string is "SMSC", the ini filename should be "SMSC.ini"

(**Note:** The Manufacturer's string may be set or viewed using the USBDM.)

## Creating the Required SetIcon Ini Files (Cont.)

### 2) Setting the Ini Section Name:

**Windows 2000 and XP** - The name of the section should be same as the first 5 characters of the Device's Product ID string enclosed in square brackets, including any spaces if present.

Example: If the Product ID string is "223 USB Controller", the section name should be "[223 U]"

Example: If the Product ID string is "223US", the section name should be "[223US]"

Example: If the Product ID string is "223", the section name should be "[223]"

Example: If the Product ID string is "", the section name should be "[]"

(**Note:** The Manufacturer's string may be set or viewed using the USBDM utility.)

### 3) Creating the Ini Section Content:

Under the Ini Section name should be a two line entry for each media type. The format for the two line entry is "Prod=Path\IconName.ico", where "Prod" is the string following the dash (-) in the Disk Drives section of the Device Manager for that drive (as seen in the screenshot to the right).

Path\IconName.ico is the full path and icon name for the icon to be used for that drive. "ProdLABEL=Label Name" – (A declaration used to display a descriptive label in Windows Explorer for disk volumes with no names) where "ProdLABEL" is the same as "Prod" as explained above appended with the word "LABEL" and "Label Name" is the label that is to be displayed for the corresponding drive.

Note: The string length of "Label Name" should be less than 32 characters and should only contain alpha-numerical characters and special characters 'space' (' ') and 'under score' ('\_').

Example: CF=\Program Files\Icons\CF.ico

Example: CFLABEL=Compact Flash Drive

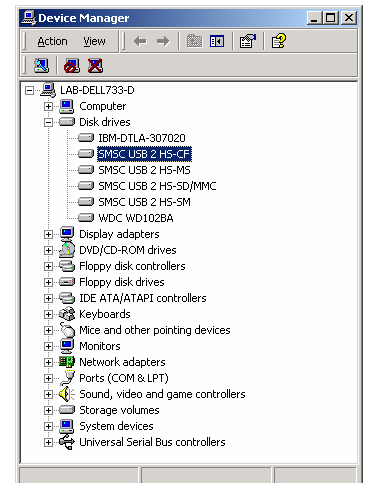
Example: SD/MMC=\Program Files\Icons\SDMMC.ico

Example: SD/MMCLABEL=SDMMC Drive (**Note there is no slash “/”**)

#### *Important Notes:*

- 1) The full path to the icon should be less than 64 characters.
- 2) The file containing the icon should only be an .ico, .dll or .exe file.
- 3) There should not be any extra spaces before and after the '=' sign

To use the dynamic icon functionality, you also need to add lines for each LUN number and interface type (i.e. CF, SM, XD, etc.) for both the media present “L#\_” and media not present “L#\_NM” states. Please see the sample ini file that follows for clarification.



### 4) Placing the Ini File in the Correct Location on the Target PC:

In order for the custom icon application to work correctly, the ini file must be placed in one of the Windows System directories, depending on which operating system is being used. Those directories are:

**Windows 2000** - "Windows\System32"

**Windows XP** - "Windows\System32"

**Manually Installing the Custom Icons Application Files**

In order to perform a manual installation of the custom icons application files, the following steps should be performed:

1. Copy the SetIcon.exe file to a location on the target computer's hard drive (i.e. "C:\Program Files\Icons\SetIcon.exe").
2. Copy the icon files to a location on the target computer's hard drive (i.e. "C:\Program Files\Icons\").
3. Add a String entry to the Windows registry key "HKEY\_LOCAL\_MACHINE\Software\Microsoft\Windows\CurrentVersion\Run" that will automatically start the SetIcon application each time the host computer is booted.

**String:** SetIcon   **Value:** C:\Program Files\Icons\SetIcon.exe

4. Copy the ini file to the appropriate Windows System directory on the host PC. (See the previous section "Creating the Ini Files" for details.)
5. Manually start the SetIcon.exe application by double clicking it, or simply reboot the host PC. The entry placed in the registry during Step 3 will automatically start the application after the PC is rebooted.

## A Sample Ini File

```
[2602 ]
CF=C:\Program Files\Icons\CF.ico
CFLABEL=Compact Flash Drive
MS=C:\Program Files\Icons\MS.ico
MSLABEL=Memory Stick Drive
SM=C:\Program Files\Icons\SM.ico
SMLABEL=Smart Media Drive
SD/MMC=C:\Program Files\Icons\SDMMC.ico
SD/MMCLABEL=SDMMC Drive

L0_CF=\Program Files\SMSC\Cf.ico
L0_CFLABEL=Compact Flash Drive
L0_NM=\Program Files\SMSC\cf-gray.ico
L0_NMLABEL=Compact Flash Drive

L1_MS=\Program Files\SMSC\Ms.ico
L1_MSLABEL=Memory Stick Drive
L1_MSPR=\Program Files\SMSC\MsPro.ico
L1_MSPRLABEL=Memory Stick Pro Drive
L1_NM=\Program Files\SMSC\ms-gray.ico
L1_NMLABEL=Memory Stick Drive

L2_SM=\Program Files\SMSC\Sm.ico
L2_SMLABEL=Smart Media Drive
L2_XD=\Program Files\SMSC\Xd.ico
L2_XDLABEL=xD Media Drive
L2_NM=\Program Files\SMSC\sm-gray.ico
L2_NMLABEL=Smart Media Drive

L3_SD=\Program Files\SMSC\Sd.ico
L3_SDLABEL=SD Media Drive
L3_MMC=\Program Files\SMSC\Mmc.ico
L3_MMCLABEL=MMC Media Drive
L3_NM=\Program Files\SMSC\sdmmc-gray.ico
L3_NMLABEL=SDMMC Media Drive
```

## Creating a Windows Installer for the Custom Icons Application Files

Using an automated installer is the preferred method for installing and setting up the Custom Icons application to run on an end user's PC. As part of the USB2602 Custom Icons Application Package, a sample Windows installer is included which demonstrates a practical example of using a Windows installer to install, setup, and run the Custom Icons application. To use the installer, simply run it and then reboot the host PC once the installation is complete. When the reboot is complete, the custom icons for the 2602 should appear in Windows Explorer.

**Important Note:** The ini files that are installed by the SMSC provided installer are hard coded to match SMSC's VID/PID, Manufacturer String, and Product ID String. The EEPROM.DAT file that is included with the software distribution contains the required data, and should be used to program evaluation boards to be used with the installer. Otherwise the ini files will not match the data in your board, and the icons will not appear. In general, to create a Windows Installer you should configure it to do the following:

1. Copy the SetIcon.exe file to a location on the target computer's hard drive (i.e. "C:\Program Files\Icons\SetIcon.exe").
2. Copy the icon files to a location on the target computer's hard drive (i.e. "C:\Program Files\Icons\").
3. Add a String entry to the Windows registry key "HKEY\_LOCAL\_MACHINE\Software\Microsoft\Windows\CurrentVersion\Run" that will automatically start the SetIcon application each time the host computer is booted.

**String:** SetIcon **Value:** C:\Program Files\Icons\SetIcon.exe

4. Configure the installer to do a conditional installation depending on the operating system, to copy the ini files to the appropriate Windows System directory. (See the section "Creating the Ini Files" for details.)
5. Configure the installer to run the "SetIcon.exe" application once the install is complete. Alternatively, you could force the user to reboot the PC.

## Troubleshooting the Custom Icons Application

Issue:

Cause:

After installing the Custom Icons application and rebooting, the custom icons do not appear.	<ol style="list-style-type: none"> <li>1) If you used the custom installer it is likely that the contents of your serial eeprom do not match the ini files that are installed with the installer. Read the section "Programming the Serial EEPROM" and program the eeprom to match SMSC's VID/PID, Manufacturers String, and Product ID String for the 2602. An EEPROM.DAT file with this data is included in the SetIcon software release for your convenience.</li> <li>2) If you created your own ini files and installed the application files manually, the cause is most likely an incorrectly named or formatted ini file. Refer to the section "Creating the Ini Files" and double check to make sure that the ini files are correctly named, formatted, and placed in the proper location.</li> <li>3) Check to see that the "SetIcon.exe" application is running by checking the Processes tab in the Task Manager.</li> </ol>
After installing the Custom Icons application the drives still show the original icon.	Unplug the USB cable and then reattach it. Icons are only displayed when the device is attached with the SetIcon application running. If this does not correct the problem, try the troubleshooting steps above.
In Windows XP (SP1) the custom icons do not appear after a reboot of the host. However if the USB cable is detached and reattached, or media is either inserted or ejected, the icon(s) appear.	This is a bug in Windows XP. Microsoft has developed a fix (KB823293).
In Windows XP, the drive media label is not updated when a card is inserted.	This is a known issue in Windows XP. As a workaround, you can either hit F5 to refresh the label, or remove and reinsert the media.
Card Reader Software Safe Removal Installer does not install properly when attempting to run the installer after removing a previous version of the installer.	When the Card Reader Software Installer is removed, the uninstall stops the device and removes all registry entries associated with the device. The device must be unplugged and reattached before it will enumerate.



Memory Stick Custom icon does not appear after running the Card Reader Safe Removal Installer.	Limitation of the OS.
--	-----------------------

## Windows Installer Packages

There are two sample installers included in the USB2602 DFU and Driver Package. Both of these installers demonstrate a practical example of using a Windows installer to install, setup, and run the Custom Icons application as mentioned in the previous section. In addition, one of these installers "Cardreader Software Installer Safe Removal vXXX.exe" has the added feature of not allowing a SMSC device to be removed by the safe removal icon available in Windows 2000 and Windows XP. The Safe Removal program will still function properly for other non-SMSC VID/PIDs; it will just not list any SMSC VID/PIDs as options to remove. The other installer "Cardreader Software Installer vXXX.exe" will allow a SMSC VID/PID to be removed via the safe removal program. The safe removal functionality is the only difference between these two installers.

**Important Note:** The SMSCMSC.ini file that is installed by the SMSC provided installer is hard coded to match SMSC's VID/PID, Manufacturer String, and Product ID String. In order to enable this feature for end products with different VID/PIDs this file should be renamed and all SMSC VID/PIDs should be removed and replaced with the end product VID/PID. A sample SMSCMSC.ini is included below. This file is installed by the SMSC provided installer at C:\Windows\inf.

```
[Version]
Signature="$WINDOWS NT$"
Class=USB
ClassGuid={36FC9E60-C465-11CF-8056-444553540000}
Provider=%SMSC%
DriverVer=11/14/1999,5.00.2183.1

[Manufacturer]
%SMSC%=SMSC

[SMSC]
%USB\VID_0424&PID_223A.DeviceDesc%= SMSC_BULK, USB\VID_0424&PID_2602

[SMSC_BULK]
Include = usbstor.inf
Needs = USBSTOR_BULK
Addreg = SMSC_BULK.AddReg,SMSC_BULK_NR.AddReg

[SMSC_BULK.HW]
Include = usbstor.inf
Needs = USBSTOR_BULK.HW

[SMSC_BULK.NT]
Include = usbstor.inf
Needs = USBSTOR_BULK.NT
Addreg = SMSC_BULK.AddReg,SMSC_BULK_NR.AddReg

[SMSC_BULK.NT.HW]
;

[SMSC_BULK.NT.Services]
Include = usbstor.inf
Needs = USBSTOR_BULK.NT.Services

;Registry Sections
[SMSC_BULK.AddReg]
HKR,,DriverFlags,0x00010001,0x00000001

[SMSC_BULK_NR.AddReg]
HKR,,NonRemovable,0x00010001,0x00000001

[SourceDisksNames]
1= %InstallDisk%, ""

[SourceDisksFiles]
SmscMsc.INF = 1

[Strings]
SMSC = "Standard Microsystems Corporation"
USB\VID_0424&PID_223A.DeviceDesc = "USB Mass Storage Device"
InstallDisk = ""
```

### **Using the Production Line Descriptor Update Utility (PLDU)**

**Purpose:** The PLDU is used to update device firmware and/or device descriptors such as the VID/PID, Manufacturer, and Product ID strings in a production line environment using Windows 2000 SP3 and SP4. Under Windows XP, this can be used to update device descriptors or firmware if all the devices have same descriptor data. Otherwise, each device will enumerate as a MSC device and the utility needs to keep swapping drivers which is a time consuming operation and not really effective under a production line environment. This application is intended to be used by OEMs in their production line environment and is not intended for other users. The utility features a simple interface that displays success or failure of the programming operation in graphical form using either a green box with a checkmark (PASS), or a red box with an "X" (FAIL). The PLDU is capable of programming one device at a time and takes approximately 12 seconds to complete.

#### **Features:**

1. Firmware update.
2. Descriptor (NVRAM) update.
3. Read descriptor (NVRAM) data from device.
4. GUI editor to edit and create DAT files.
5. Graphical and Text status display.
6. Automatic serial number increment after every descriptor update.
7. Break up of serial number to YY-MM-DD-S-SN format where
  - YY - Year (2 digits)
  - MM - Month (2 digits)
  - DD - Day (2 digits)
  - S - Station number (1 digit)
  - SN - Serial number (5 digits)

#### **Application Behavior:**

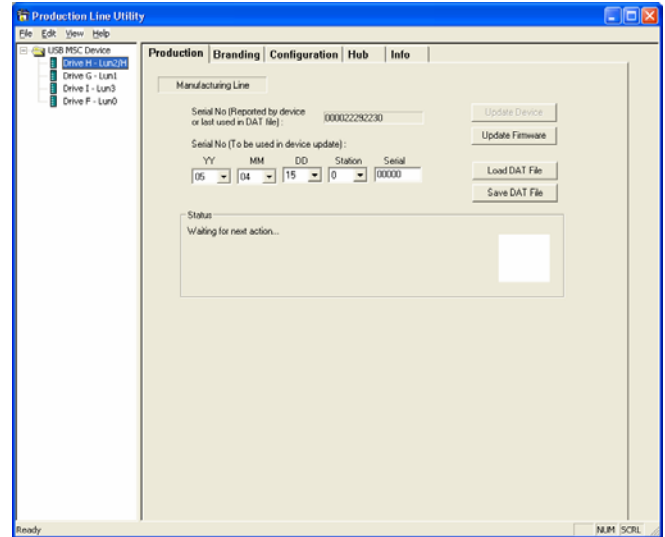
- 
1. When the application is run with no SMSC devices plugged in, all controls should be disabled.
  2. While the application is running, the controls should be dynamically enabled and disabled as the device is plugged and unplugged.
  3. The button controls on all tabs except the "Production" tab will always be disabled.
  4. The "Update Device" button will be enabled only after loading a DAT file.
  5. If the "Update Firmware" button is clicked, the utility removes reference to a previously loaded DAT file and disables the "Update Device" button.
  6. Any changes made to the YY-MM-DD-S-Sno controls will be lost if user switches to a different tab and returns to the "Production" tab. The changes should be saved immediately to a DAT file before switching tabs. If the user needs to switch tabs to make changes in those tabs, then those changes should be made first and lastly switch to the "Production" tab. Now the user can make necessary changes to the formatted serial number controls and can be saved to a DAT file that will include all the changes done on the other tabs as well. However, NOTE THAT THIS IS THE BEHAVIOR ONLY WHEN A DAT FILE HAS NOT BEEN LOADED ALREADY AND THE BUTTON "Update Device" IS DISABLED.
  7. When a DAT file is loaded and the "Update Device" button is enabled, following behavior is to be expected:
    - a. Changes made to serial number controls in "Production" tab will be lost if user changes tabs. However, these changes will be active for the user to save to a DAT file or update to the device immediately after the changes are made and before switching tabs.
    - b. Changes made to controls on other tabs will be lost and can never be saved to a DAT file or updated to a device.
    - c. Changes made to serial number controls, though available for an immediate update to a device, will be lost after the update completes if those changes don't reflect the current date (YY-MM-DD).

**Button Behavior:**

- 
1. Update Device
    - a. Enabled only after a DAT file is loaded.
    - b. Disabled whenever a firmware update is done.
    - c. When clicked, app reads the serial number from the YY-MM-DD-S-Sno controls and embeds this serial number into the DAT file that is loaded in memory to write to the device.
    - d. After every update, the DAT file is updated to reflect the last serial number used to write to the device and also automatically increments the serial number.
  2. Update Firmware
    - a. Prompts for a DFU file (only the first time) and uses this file to update the device's firmware.
    - b. Always ignores any changes done to the controls in all tabs. The utility always reads the descriptor data from the device and embeds this into the DFU file image before writing the DFU file image to the device. Thus, after a firmware update, the device's descriptor data should be the same as it was before the update.
    - c. If a DAT file was previously loaded, clicking this button would unload the DAT file and disable the "Update Device" button.
  3. Load DAT File
    - a. Loads a DAT file into memory.
    - b. Enables the "Update Device" button.
    - c. Any changes done to controls on all tabs are lost while switching tabs.
    - d. Changes done to YY-MM-DD-S-Sno controls are available for saving to a DAT file or writing to device only immediately after the changes are made.
    - e. When a DAT file is loaded, the YY-MM-DD-S-Sno controls are set to reflect current date, same station number digit as in the DAT file and either a default value of "00000" or DAT file's last 5 digits of serial number value incremented by one. The default value of "00000" is used whenever the DAT file's YY-MM-DD digits do not match the current date.
  4. Save DAT File
    - a. Saves the values from the controls to a DAT file.
    - b. Refer to earlier sections to find out when changes to controls are lost.
    - c. After saving to DAT file, this DOES NOT automatically load that DAT file into memory.

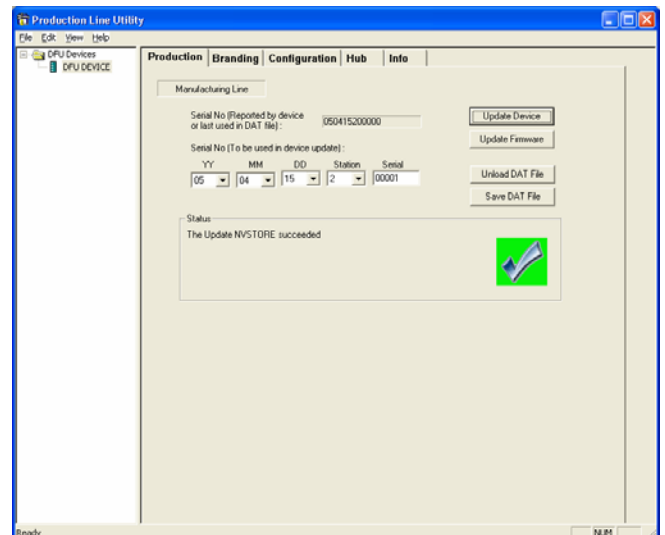
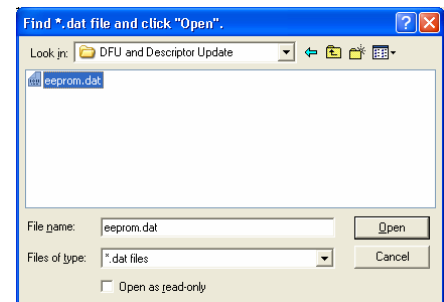
## Setting Up the PLDU Application

1. First attach a USB2602 device to the host. To start the PLDU application, simply double click “PLDU.exe” executable.
2. After the main program dialog opens the production tab displays four options:
  - a. Update Device—Updates NVStore descriptor data such as VID/PID, Manufacturer and Product ID strings from the “EEPROM.DAT” file. Note – this option is not available until a DAT file is loaded.
  - b. Update Firmware—Updates the device firmware using a DFU update file with the .dfu extension.
  - c. Load DAT file—Loads a DAT file into memory
  - d. Save DAT file—Saves the values from the controls to a DAT file



## Using the PLDU to Update Device Descriptors

1. The first operation that should be performed on a USB2602 device coming off the production line is to update its descriptors. To do this, first press “Load DAT file.” The application will prompt you to select the EEPROM.dat file that will be used to program the descriptors. Once the EEPROM.DAT file has been selected, the option to Update Device will be available.
2. Click the “Update Device” button. The PLDU application will swap the mass storage class driver for the SMSC DFU driver.
3. Once the DFU driver swap has completed, the data from the eeprom.dat file that is loaded is programmed into the device. The operation takes about 12 seconds to complete. Provided the programming was successful, the Status box displays a green box with a checkmark and reports success. At this point the user simply detaches the device and reattaches the next device to be programmed. The PLDU automatically updates the EEPROM.DAT file to the next unique serial number.



### **Using the Production Line Test Utility (PLTU)**

**Purpose:** The PLTU application is used to test the basic functionality of USB2602 devices in a production line environment using Windows 2000 (SP3) only. The application creates a subdirectory on the media for each LUN, copies a 'Test File' to the subdirectory, deletes the 'Test File,' and then deletes the subdirectory.

**Features:**

1. Capable of testing 5 devices with 4 LUNs each simultaneously.
2. After testing, the application cleans up the registry entries involving the OEM's VID, PID, Inquiry MFG, and Product strings.
3. Graphical and Text status display of test results.
4. GUI editor to edit and create ini files.

### **Creating the PLTU ini File**

Before using the PLTU you must create or edit an ini file. A sample ini file is shipped with the PLTU application which can be modified for your setup. The ini file should contain the following lines:

OEMVID = **VID**

This is the original equipment manufacturer's VID (Vendor ID) of the device whose descriptor has already been updated. The '**VID**' is specified as a four digit hexadecimal number.

OEMPID = **PID**

This is the original equipment manufacturer's PID (Product ID) of the device whose descriptor has already been updated. The '**PID**' is specified as a four digit hexadecimal number.

INQUIRY\_MFG = **Inquiry MFG String**

This is the string returned by the device as part of the Vendor information in the Inquiry data. This can be of maximum 8 characters.

INQUIRY\_PRODUCT = **Inquiry Product String**

This is part of the string returned by the device Product information Inquiry data. This can be of maximum 5 characters.

TEST\_FILE = **path to Test file**

Specifies the full path to the file that is to be used during file copy tests.

DEV1\_LUN0 = **Drive Letter**

DEV1\_LUN1 = **Drive Letter**

DEV1\_LUN2 = **Drive Letter**

DEV1\_LUN3 = **Drive Letter**

DEV2\_LUN0 = **Drive Letter**

DEV2\_LUN1 = **Drive Letter**

DEV2\_LUN2 = **Drive Letter**

DEV2\_LUN3 = **Drive Letter**

DEV3\_LUN0 = **Drive Letter**

DEV3\_LUN1 = **Drive Letter**

DEV3\_LUN2 = **Drive Letter**

DEV3\_LUN3 = **Drive Letter**

DEV4\_LUN0 = **Drive Letter**

DEV4\_LUN1 = **Drive Letter**

DEV4\_LUN2 = **Drive Letter**

DEV4\_LUN3 = **Drive Letter**

### Creating the PLTU ini File (Cont.)

DEV5\_LUN0 = *Drive Letter*

DEV5\_LUN1 = *Drive Letter*

DEV5\_LUN2 = *Drive Letter*

DEV5\_LUN3 = *Drive Letter*

These lines specify the Drives that are associated with the multiple LUNs of the respective devices to be tested. If the 'Drive Letter' is not specified for a particular LUN, it means that the corresponding LUN of that device is NOT to be tested. If the 'Drive Letter' is not specified for all LUNs for a particular device, it means that the entire device is either NOT present or NOT to be tested.

### A Sample PLTU ini File

```
OEMVID      = 0424
OEMPID      = 2228
INQUIRY_MFG  = SMSC
INQUIRY_PRODUCT = 2228
TEST_FILE    = C:\TEST\1MEG.R01
```

```
DEV1_LUN0 = F
DEV1_LUN1 = G
DEV1_LUN2 = H
DEV1_LUN3 = I
```

```
DEV2_LUN0 = J
DEV2_LUN1 = K
DEV2_LUN2 = L
DEV2_LUN3 = M
```

```
DEV3_LUN0 = N
DEV3_LUN1 = O
DEV3_LUN2 = P
DEV3_LUN3 = Q
```

```
DEV4_LUN0 = R
DEV4_LUN1 = S
DEV4_LUN2 = T
DEV4_LUN3 = U
```

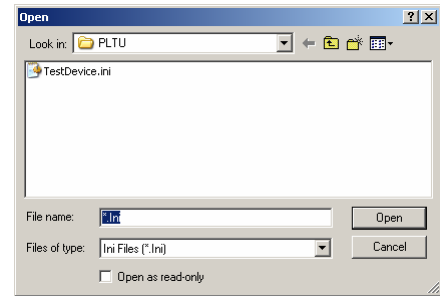
```
DEV5_LUN0 =
DEV5_LUN1 =
DEV5_LUN2 =
DEV5_LUN3 =
```

#### **NOTE:**

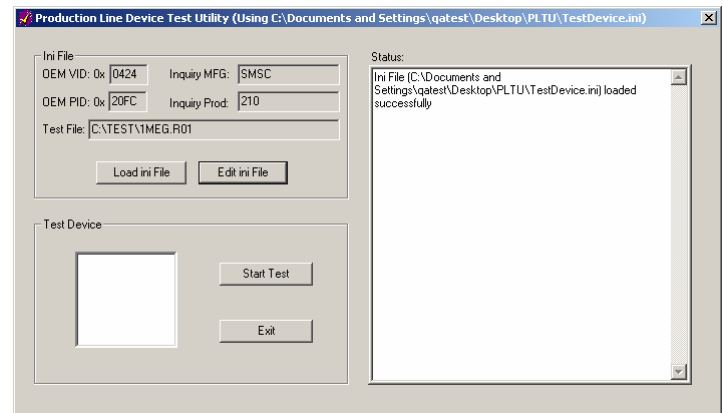
There can be spaces before and after the '=' sign, but the total number of characters for an entire line (including spaces) should be less than 255.

### Setting Up the PLTU Application

1. First attach a USB2602 device to the host. To start the PLTU application, simply double click “TestDevice.exe” executable. The application will prompt you to select the location of the ini file when it is first started.

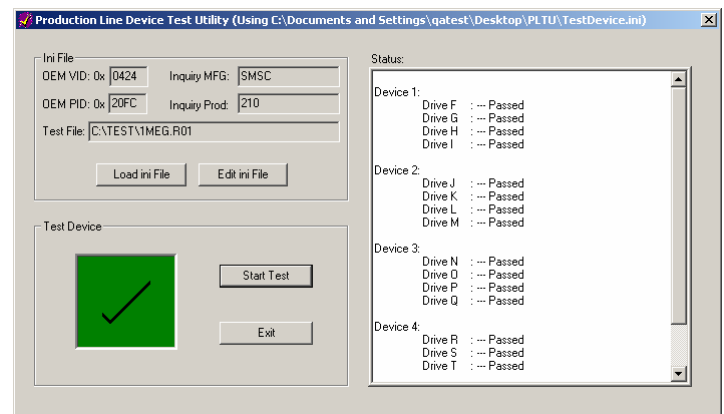


2. Provided the ini file contains the correct path to the key files on the local machine, the main program dialog opens. The station is now ready to begin testing devices. At this point you should attach the devices to be tested and ensure that they have good media with sufficient free space to hold the file being used for testing.



### Using the PLTU to Test Multiple Devices

1. Once all of the devices have been attached, the user simply presses the “Start Test” button to begin testing devices in accordance with the contents of the ini file being used. After the testing has completed, the user receives a graphical representation of the test results in the form of a green box with a black checkmark to indicate “PASS,” or a red box with a black “X” to indicate “FAIL.”



2. Once the test has completed, the user should remove all of the tested devices and then attach the next set of devices to be tested. Once all of the devices are attached and enumerated (as indicated by the presence of drive icons in Windows Explorer), the user repeats step 1 to test the next set of devices.



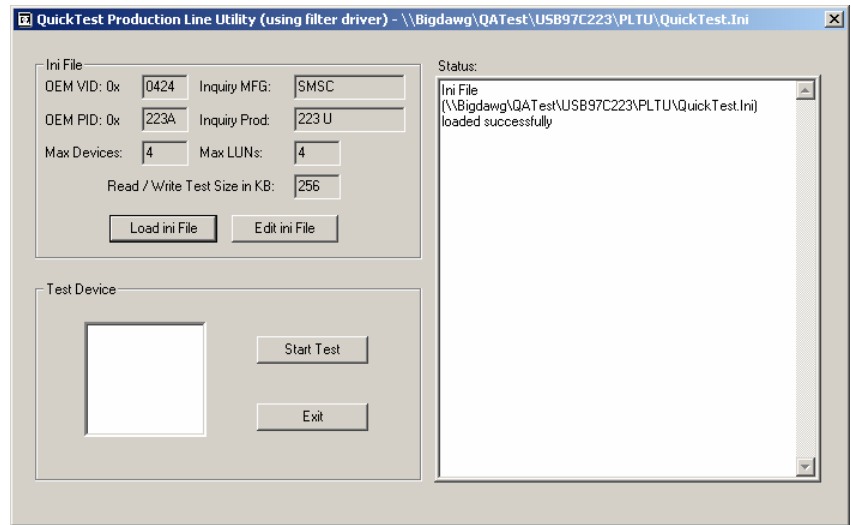
**Known Issues with the USB2602 Production Line Utilities**

Issue:	Workaround:	Status:
The PLDU and PLTU applications are designed to be used with Windows 2000 (SP3) host systems using the Microsoft mass storage class driver. While the applications may work with other operating systems, only Windows 2000 (SP3) is supported.	N/A	N/A
Some EHCI host controller drivers such as Orange Micros do not work properly with the DFU driver swapping performed by the PLTU application.	We highly recommend that you use the Microsoft supplied EHCI drivers for the test systems running the PLTU applications.	N/A
The PLTU does not distinguish between general device write failures and media specific write failures. This means that the test will fail if no media present in the drive, media is full, media is unformatted, media is corrupt, media is write protected, etc... Under such circumstances, the test results do not reflect the status of the device, but rather the failure of the media. Hence, it is recommended that the test is performed again on the device with known good media.	Only use known good media to perform the PLTU testing.	N/A
Due to caching by the OS, the IO transfer may not be fully completed before the test results are displayed by the application. It is recommended that the user wait for 5 to 10 seconds before disconnecting the devices.	Wait 5-10 seconds after completion of the PLTU tests before removing the devices from the host.	N/A
User may experience errors when running applications if certain files used by applications are marked read only.	Make sure that all files used with these utilities are not marked read only.	N/A

### Using the QuickTest Production Line Read/Write Test Utility

The QuickTest utility is a streamlined version of the full Production Line Test Utility discussed previously. QuickTest can test a maximum of (4) USB2602 devices at a time, with a maximum of 4 LUNs each. The testing procedure is very simple, involving only these 4 steps:

1. Writes to media on each LUN starting from LBA 1024
2. Reads from media on each LUN starting from LBA 1024
3. Compares the data read against the data written to the media
4. Updates the status for each LUN in the application



The testing is performed on all the LUNs of the device serially. However, tests on multiple devices are performed simultaneously using multiple threads. The QuickTest utility requires the presence of the SMSC password filter driver to send BULK-ONLY commands, totally by-passing the native file system. On Windows 2000 systems, Service Pack 3 should be installed.

QuickTest.exe requires the SMSC password filter driver (Smscpwd.inf) to be installed in order to function properly. This driver is no longer installed by the card reader installer and must be installed manually before running QuickTest. In order to install the password filter drivers, copy Smscpwd.inf and Smscpwd.sys to your system. Open the device manager and double click on the USB Mass Storage Device entry in the Universal Serial Bus Controllers section. In the driver tab, select update driver. The wizard will assist in installing this driver. When given the choice, specify to have the wizard display a list of known drivers for this device. Choose "Have Disk" and browse to where you copied the smscpwd.inf file and select it. This should give you the option to install "USB Mass Storage Device with Password Protection (WinMe)."

### **Limitations of the QuickTest Utility:**

1. Does not distinguish between general device write failures and media specific write failures. This means that the test will fail if no media is present in the drive, the media is full, unformatted, corrupt, write protected, etc... Under such circumstances, the test results do not reflect the status of the device. Hence, it is recommended that the test is performed again on the device with known good media.
2. The time taken to complete the tests depend on the following:
  - Test size–This can be from 64KB to 5000KB. The bigger the size, the more time it will take to complete the tests.
  - Number of devices connected–The field "Max Devices" specifies how many devices to test at once (should be  $1 \leq N \leq 4$ ). However, it is not necessary that the actual number of devices connected be equal to the number specified in the "Max Devices" field. For example, the "Max Devices" field can specify 4 but the actual number of devices connected may be  $<4$  or  $>4$ . However, the utility will either test only the actual number of devices connected or the "Max Devices," whichever is less. Though tests on multiple devices are performed simultaneously, the time taken for the tests to complete on multiple devices will be a little more than that for a single device.

**Using the Windows XP Special Memory Stick Format Registry Key**

Windows XP has the capability to apply a Sony factory format on Memory Stick cards by adding a special key to the registry:

```
[HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\PerHwIdStorage\
USBSTOR#DiskSMSC____2602_U_HS-MS____] "DeviceGroup"="MemoryStick"
```

This key has to be customized to match the inquiry data returned from the device. The inquiry data is made up of the first 8 characters of the Manufacturer String, followed by the first 5 characters on the Product String. In the example registry key above, the strings are:

Manufacturer String = "SMSC" (Note that SMSC is followed by four spaces denoted by underscores to make up the 8 characters.)

Product String = "2602 USB2602" (Note that only the first 5 characters, including the space, are used.)

This registry key works for Windows XP only. It will not work for Windows 2000 or any other operating system. Once the registry key has been added, when a user formats a Memory Stick card using Windows, the Sony factory FAT format will be applied, including the creation of the "MEMSTICK.IND" hidden file.

### **Using the KillReg Utility**

KillReg is a DOS based application to stop a device and clean its related registry entries during an automated uninstallation process. KillReg is designed to be called from a Windows Installer script. It is used during installation and uninstallation of USB97C210/223/2224/2228/2602 devices under all Windows operating systems to remove the device entries from the registry. This allows the SMSC Win2K or Windows native driver to be loaded if the device has previously been installed without a driver, or with an incorrect driver. KillReg is also used during the uninstallation process to completely remove the registry entries for a particular device.

#### **Requirements:**

KillReg requires an ini file to be present in the Windows directory. The name of this ini file should be passed as command line argument to the application from the installer script.

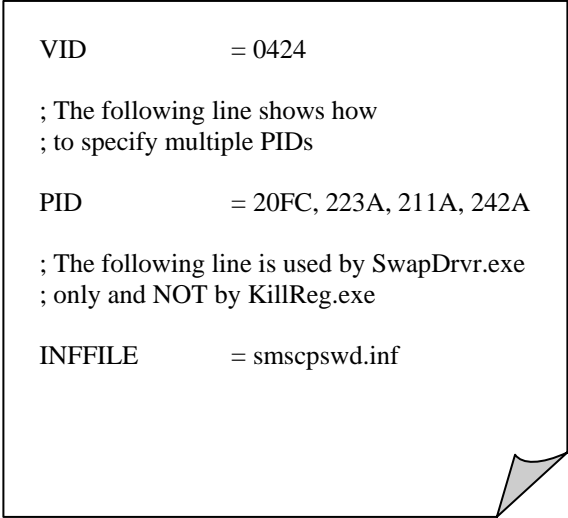
#### **INI File Requirements:**

1. The ini file should be in the Windows directory.
2. The ini file should contain the following lines;

```
VID = VID  
PID = PID1[,PID2,PID3,...,PID30]
```

where VID and PID are represented as 4 digit hexadecimal numbers.

#### **A Sample ini File:**



```
VID          = 0424  
  
; The following line shows how  
; to specify multiple PIDs  
  
PID          = 20FC, 223A, 211A, 242A  
  
; The following line is used by SwapDrvr.exe  
; only and NOT by KillReg.exe  
  
INFFILE      = smscpswd.inf
```

#### **NOTE:**

1. The ini file is also used by the application "SwapDrvr.exe," which will expect the line specifying the INFFILE. KillReg ignores this line.
2. Multiple PIDs separated by a comma can be specified to uninstall all the PIDs associated with a single VID.

### **Using the Swapdrv Utility**

Swapdrv is a DOS based application used by a Windows installer to load the password filter driver in Windows XP. Unfortunately, SwapDrv does not work with Windows 98 and Me. The only USB2602 application that requires the password filter driver be loaded when running XP is the QuickTest production line test utility. If you are not using that utility or do not want to include it in your installer, you can skip this section.

#### **Requirements:**

1. The device should be connected while this application is invoked from a Windows installer. The application will prompt the user to connect the device during run time.
2. Swapdrv needs an ini file to be present in the Windows directory. The name of this ini file should be passed as command line argument to the application from the installer script.
3. The installer application should have already placed the required INF and SYS files in their correct locations.

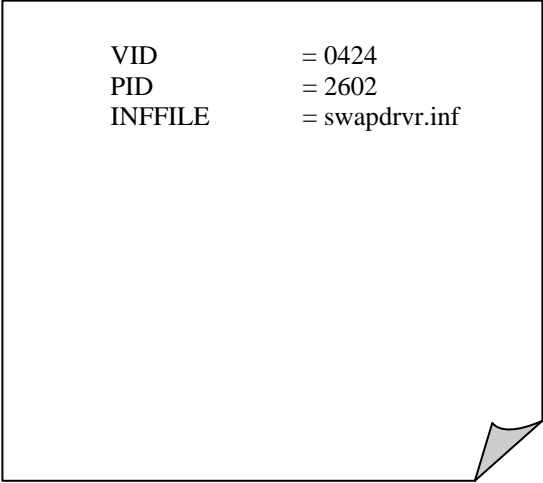
#### **INI File Requirements:**

1. The ini file should be in the Windows directory.
2. The ini file should contain the following lines;

VID = VID  
PID = PID  
INFFILE = Inf file name

where VID and PID are represented as 4 digit hexadecimal numbers.

#### **A Sample ini File:**



```
VID          = 0424
PID          = 2602
INFFILE      = swapdrv.inf
```

## **Using the USB2602 with Linux**

Versions 2.4.20 and greater of the Linux kernel provide native support for multi-LUN USB mass storage class devices like the USB2602. Some brands of Linux such as SuSe 8.2 require little or no user setup. Simply plug in your USB2602 device, and icons will appear, provided there is media in the card reader slots. Other brands of Linux such as Redhat require the user to configure the kernel in order to enable multi-LUN support in the mass storage class driver. The procedure for doing that is:

### **Requirement:**

RedHat Linux 9.0 with kernel 2.4.20 or greater

### **Steps:**

1. Install RedHat Linux 9.0 on the host system.
2. Login to the system as 'root'.
3. Open a terminal window.
4. Plug the multi-LUN card reader into the host.
5. At the shell prompt, type 'cat /proc/scsi/scsi'.
6. If the screen shows only one LUN, type 'lsmod'.
7. If 'usb-storage' does not exist, type 'insmod usb-storage'.
8. If 'usb-storage' exists, type 'cdrecord -scanbus'. It will display  
scsibus0:  

```

0,0,0) 'SMSC          ' '223 U HS-CF' 'X.XX' Removable Disk
0,1,01) *
0,2,02) *
0,3,03) *
0,4,04) *
0,5,05) *
0,6,06) *
0,7,07) *
```
9. Create a batch file with the following calls:  

```

'echo "set-single-device 0 0 0 0">/proc/scsi/scsi
'echo "set-single-device 0 0 0 1">/proc/scsi/scsi
'echo "set-single-device 0 0 0 2">/proc/scsi/scsi
'echo "set-single-device 0 0 0 3">/proc/scsi/scsi
'cat /proc/scsi/scsi'
```
10. After running the batch file, the screen should display:

Attached devices:

Host: scsi0 Channel: 00 ID: 00 LUN: 00

Vendor: SMSC Model: 223 U HS-CF

Type: Direct-Access

Rev: X.XX

ANSI SCSI revision: 02

Attached devices:

Host: scsi0 Channel: 00 ID: 00 LUN: 01

Vendor: SMSC Model: 223 U HS-MS

Type: Direct-Access

Rev: X.XX

ANSI SCSI revision: 02

Attached devices:

Host: scsi0 Channel: 00 ID: 00 LUN: 02

Vendor: SMSC Model: 223 U HS-SM

Type: Direct-Access

Rev: X.XX

ANSI SCSI revision: 02

Attached devices:

Host: scsi0 Channel: 00 ID: 00 LUN: 03

Vendor: SMSC Model: 223 U HS-SD/MMC

Type: Direct-Access

Rev: X.XX

ANSI SCSI revision: 02

11. Now multi-LUN support is enabled and you should be able to mount and access all media normally.

**Media Tested with the USB2602**

The following flash media cards were used during the development and testing of the USB2602.

<b>Compact Flash</b>	Sony 64MB Sony 128MB Sony 256MB	<b>Secure Digital</b>	Transcend 256MB Transcend 512MB Transcend 1GB Transcend 2GB
CompUSA 16MB CompUSA 64MB Efilm Pro 640MB Lexar 32MB Lexar 64MB Lexar 256MB Lexar 512MB (24x) Lexar 512MB (80x) Lexar 1GB Lexar 2GB Memorex 32MB Memorex 64MB Memorex 256MB PNY 64MB PNY 256MB PNY 512MB PQI 16MB PQI 64MB SanDisk 32MB Sandisk 512MB SanDisk 1GB SanDisk Extreme 1GB SanDisk Ultra 128MB Transcend 8GB (120X) Viking 32MB	<b>High Speed Memory Stick</b> Sony 16MB Sony 32MB Sony 128MB	Infineon 64MB Infineon 128MB Kingston 64MB Lexar 32MB Lexar 64MB Lexar 128MB Lexar 256MB Lexar 512MB Lexar 1GB Memorex 32MB PNY 512MB PQI 64MB PQI 128MB PQI 256MB SanDisk 32MB SanDisk 64MB SanDisk 128MB SanDisk Extreme 256MB SanDisk 512MB SanDisk 2GB SanDisk Ultra II 2GB SimpleTech 128MB Transcend 4GB	<b>MMC 4.2</b> Phison 4GB
	<b>Memory Stick Pro</b> Sony 256MB Sony 512MB Sony 1GB SanDisk 256MB SanDisk 512MB SanDisk Ultra II 512MB SanDisk Ultra II 1GB		<b>MMC Micro</b> Transcend 128MB Transcend 256MB Transcend 512MB
	<b>Memory Stick Duo</b> Sony 16MB Sony 32MB Sony 64MB Sony 128MB		<b>RS-MMC</b> SanDisk 128MB Transcend 128MB Transcend 256MB Transcend 512MB
	<b>Mini Secure Digital</b> Panasonic 32MB Panasonic 64MB Panasonic 128MB SanDisk 32MB SanDisk 64MB Toshiba 32MB	<b>MMC</b> Lexar 32MB Lexar 64MB PQI 32MB PQI 64MB PQI 128MB PQI 256MB SanDisk 32MB SanDisk 64MB SimpleTech 64MB SimpleTech 128MB	<b>Smart Media</b> Fuji Film 16MB Kingston 64MB I-O Data 32MB I-O Data 64MB I-O Data 128MB Lexar 32MB Lexar 64MB Lexar 128MB Memorex 32MB Memorex 64MB Memorex 128MB PNY 128MB Samsung 32MB SanDisk 32MB SanDisk 64MB SanDisk 128MB Viking 64MB
<b>MicroDrive</b>			<b>xD Picture Card</b> Fuji 64MB Fuji 128MB Fuji 512MB Olympus 64MB Olympus 128MB Olympus 256MB Olympus 512MB
<b>Memory Stick</b>	<b>Micro Secure Digital</b> SanDisk 128MB SanDisk 256MB Sandisk 512MB Transcend 128MB Transcend 256MB	<b>High Speed MMC</b> CompUSA 256MB CompUSA 512MB Connect 512MB PQI 512MB Samsung 128MB Sandisk 512MB Transcend 128MB	
Lexar 16MB Lexar 32MB Lexar 64MB Lexar 128MB Lexar 256MB PQI 128MB SanDisk 16MB SanDisk 32MB SanDisk 128MB Sony 8MB Sony 16MB Sony 32MB	<b>High Speed Secure Digital</b> Panasonic 512MB Panasonic 1GB		
	<b>High Capacity Secure Digital</b> Sandisk SD-HC 4GB Toshiba SD-HC 8GB		

NAND Flash:

NAND ID	Size (Bits)	Size (Bytes)
0x6E	8M	1M
0xE8	8M	1M
0xEC	8M	1M
0x64	16M	2M
0xEA	16M	2M
0xF1	16M	2M
0x6B	32M	4M
0xE3	32M	4M
0xE5	32M	4M
0xE6	64M	8M
0x73	128M	16M
0x75	256M	32M
0x76	512M	64M
0x79	1G	128M
0x71	2G	256M
0xDC	4G	512M
0xD3	8G	1G
0xD5	16G	2G



**USB2602 Performance Benchmarks**

The measurements were performed using Atto Windows Benchmark 2.02, Transfer Size set to 512KB, Total Length set to 4MB, on a Windows XP (SP2) system with an ICH8 south bridge. (Intel D950, 3.4GHz, 1GB DDR). All benchmarks were measured on new (out of the box) media. Please note that the benchmark performance of flash cards varies widely from manufacturer to manufacturer, and the performance of all manufacturers' cards degrade with use. In order to duplicate the results below, you must use brand new media and test on a similarly configured host.

Full Speed (USB1.1)	Reads	Writes
<i>Compact Flash</i>	1043 KB/s	932 KB/s
<i>Memory Stick</i>	909 KB/s	550 KB/s
<i>High Speed Memory Stick</i>	811 KB/s	652 KB/s
<i>Memory Stick Pro</i>	1031 KB/s	902 KB/s
<i>Smart Media</i>	977 KB/s	537 KB/s
<i>XD Card</i>	818 KB/s	437 KB/s
<i>Secure Digital</i>	1039 KB/s	945 KB/s
<i>High Speed Secure Digital</i>	913 KB/s	906 KB/s
<i>Multimedia Card</i>	996 KB/s	374 KB/s
<i>High Speed Multimedia Card</i>	882 KB/s	811 KB/s

**Media Used for Testing:**

SanDisk Extreme IV 8GB  
 Lexar Media 128MB  
 Sony MagicGate 128MB  
 Sandisk Extreme III 1GB  
 Memorex 128MB  
 Fuji xD-Picture Card 512MB  
 SanDisk Ultra II 1GB  
 Panasonic Pro High Speed 512MB  
 Transcend 512MB MMCmicro  
 Transcend 256MB HS-MMC

High Speed (USB2.0)	Reads	Writes
<i>Compact Flash</i>	10275 KB/s	8370 KB/s
<i>Memory Stick</i>	1681 KB/s	730 KB/s
<i>High Speed Memory Stick</i>	5435 KB/s	936 KB/s
<i>Memory Stick Pro</i>	13784 KB/s	11705 KB/s
<i>Smart Media</i>	4152 KB/s	730 KB/s
<i>XD Card</i>	4064 KB/s	528 KB/s
<i>Secure Digital</i>	9030 KB/s	7557 KB/s
<i>High Speed Secure Digital</i>	8677 KB/s	7342 KB/s
<i>Multimedia Card</i>	7374 KB/s	6194 KB/s
<i>High Speed Multimedia Card</i>	7608 KB/s	6155 KB/s

**Media Used for Testing:**

SanDisk Extreme IV 8GB  
 Lexar Media 128MB  
 Sony MagicGate 128MB  
 Sandisk Extreme III 1GB  
 Memorex 128MB  
 Fuji xD-Picture Card 512MB  
 SanDisk Ultra II 1GB  
 Panasonic Pro High Speed 512MB  
 Transcend 512MB MMCmicro  
 Transcend 256MB HS-MMC

**GPIO Assignment Table**

The following is a table of GPIO assignments for the USB2602. Please note that multi-function GPIO operations are determined by attribute settings. Please refer to the software release notes for detail on configuration settings.

<b>Name</b>	<b>Description</b>	<b>Function</b>	
GPIO0	<i>Not available due to pin count</i>		
GPIO1	Flash Media Activity LED	Media Activity LED/xD Door detect	
GPIO2	EE_CS	EE_CS	
GPIO3	V_BUS	V_BUS	
GPIO4	EE_DIN/EE_DOUT	EE_DIN&DOUT/xD Card Identification	
GPIO5	SD Card Detect	SD Card Detect	
GPIO6	A16 ( external ROM only ) /ROMEN	ROMEN/A16	
GPIO7	EE_CLK/Unconfigured LED	EE_CLK/Uncfg LED	
GPIO8	MS Power Control	MS Power Control	
GPIO9	CF Power Control	CF Power Control	
GPIO10	SM Power Control	SM Power Control	
GPIO11	SD Power Control	SD Power Control	
GPIO12	MS Activity	MS Activity/Media Activity LED	
GPIO13	CF Activity	CF Activity	
GPIO14	SM Activity	SM Activity	
GPIO15	SD/MMC Activity	SD/MMC Activity	

**Known Firmware Related Issues****General:**

Issue: Workaround: Status:

--	--	--

**CF Devices:**

Issue: Workaround: Status:

No known issues.		
------------------	--	--

**MS Devices:**

Issue: Workaround: Status:

When High Speed Magic Gate Memory Stick media is formatted with a FAT file system on a MacOS 10.X host, the media becomes unreadable on machines with Windows operating systems, but will continue to work normally with Macs.	None.	We believe this is a Magic Gate security protocol issue. We will continue to investigate and provide a fix in a future release of the USB2602 firmware if possible.
--	-------	---

**SM Devices:**

Issue: Workaround: Status:

Writes to 2MB Smart Media cards are not supported.	None.	2MB Smart Media cards can be read by the USB2602, but writes are not supported. These cards are considered obsolete and there are no plans to implement support for them in the future.
--	-------	---

**SD/MMC Devices:**

Issue: Workaround: Status:

No known issues.		
------------------	--	--

**xD Devices:**

Issue: Workaround: Status:

xD Compliancy will fail if the C3 option is enabled	Disable C3 or insert media in the CF, Memory Stick, or SD card socket.	Resolved as xD test setup issue only
---	--	--------------------------------------

### Issues Not Related to Firmware

Issue:	Workaround:	Status:
Due to the write caching functionality of Windows, data corruption can sometimes occur if the media is removed improperly.	Before removing any piece of media, you should right click the drive icon in Windows Explorer and select "Eject" from the context menu. This will force the operating system to perform a write of any cached data.	Limitation of the OS.
Reading or writing multiple media types simultaneously will generally happen at the slowest media rate.	This is a limitation of the OS. If writes to a slow media type like MS are made while reading from a fast media type like CF or SM, then the read will slow to approximately the rate of the write. This is because the OS must process each command separately. It is not a limitation of the firmware.	Limitation of the OS.
If the USB2602 board doesn't have a properly programmed serial number, only one drive appears in Windows Explorer.	Program a unique serial number into the board using the "USBDM" utility.	
Surprise removal of the USB cable during a write to any media type under MacOS sometimes causes the host to become unresponsive.	Reboot the host.	This appears to be a bug with the operating systems. All mass storage class devices tested have displayed this behavior.
Occasionally, surprise removal of the USB cable during writes to any media type under Windows XP results in the failure of the device to re-enumerate after being reattached.	Reboot the host.	This appears to be a bug in Windows XP. No mass storage class USB devices will enumerate once the host is in this state.
Windows 2000 does not immediately report that media is write protected when attempting to perform a full format. The format will appear to progress to completion, but at the end of the operation reports that the media is write-protected.	None.	This is normal behavior for Windows 2000. This occurs for all USB write protectable devices when attempting to perform a full format.
Prematurely attempting to access a drive after resuming from suspend sometimes results in a device I/O error in Win2K. This is a known issue at Microsoft. (Reference Microsoft Knowledge Base article Q323754).	Obtain and install the updated Usbhub.sys file from the hotfix that is described in Microsoft Knowledge Base article Q306455.	N/A
DFU for Mac 10.X does not work when the device is connected to a 1.1 USB host controller.	Attach device to a 2.0 host controller when using DFU on Mac OS 10.X.	Under investigation. May be fixed in future release
When MSPro media is inserted and the device is enumerated drive icons won't come up until media is ready to be read. Per MSPro spec larger media could takes 10 seconds to be ready.	None.	
Under Windows 2000 SP2 or below, only one drive icon appears.	Windows 2000 SP2 and below does not provide native support for multi LUN mass storage class devices like the USB2602. Either use the SMSC Windows 2000 driver, or upgrade your OS to Service Pack 3 or higher. (This is a free update.)	Use the SMSC Windows 2000 driver, or upgrade to Windows 2000 SP3 or higher.
Some Memory Sticks can not be formatted with the Sony Format utility if "Physical Device Display" is not checked.	Check the "Physical Device Display" if the MS does not have the option to format. After the first format, this box may not need to be checked for future formats.	This is a limitation of the Sony Format tool, and is mentioned in the applications help topics.
When the card reader attribute byte that forces the device to report itself as USB version 1.1 in the bcdUSB device descriptor is set, Windows XP still displays the dialog message "A HI Speed USB device is connected to a Full Speed Hub" when the 2602 is connected at Full Speed.	Setting Byte 2, bit 4 of the card reader attribute bytes results in the card reader portion of the USB2602 device reporting itself as a USB 1.1 device. However, this attribute does not affect the hub portion so the device is still seen as a USB 2.0 device.	Currently under investigation. May be fixed in a future release