AVR273: USB Mass Storage Implementation

8-bit AVR® Microcontrollers

Application Note

Features

- **Bulk-Only Transport Protocol**
- **Supported by all Microsoft O/S from Windows® 98SE and later**
- **Supported by Linux Kernel 2.4 or later and Mac OS 9/x or later.**
- **Complete solution based on DataFlash memory.**
- **Can support different memories with the suitable drivers (NF, SD, MMC...)**
- **Runs on any AVR USB microcontroller**

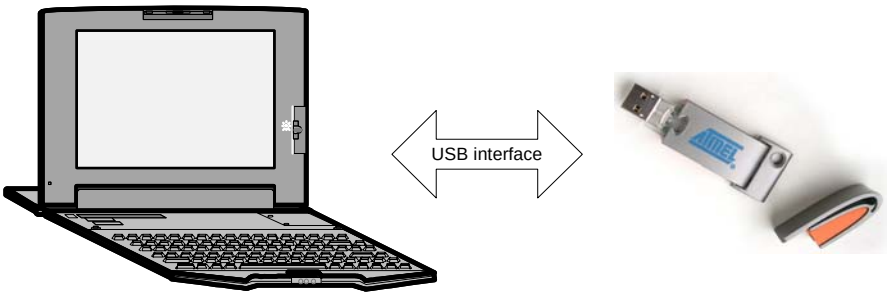
1. Introduction

The floppy disk is over, too slow, too fragile and small capacity. The CD-ROM is not convenient to exchange data (generally not rewritable) and it is not convenient for travelling. The USB key offers you the flexibility and the small size of the floppy disk and the big capacity of the CD-ROM.

Atmel offers a complete solution based on Mass Storage class with an Atmel DataFlash as target. This ensures a full duplex file transfer between the device and the PC.

The aim of this document is to describe how to start and implement a USB application based on the Mass Storage (Bulk only) class to transfer data between a PC and user equipment.

A familiarity with the USB firmware architecture (Doc 7603, Included in the USB CD-ROM & Atmel website) and the Mass Storage specification (http://www.usb.org) is assumed.



Rev. 7631A–USB–03/06

2. Hardware Requirements

The Mass Storage application requires the following hardware:

1. AVR USB evaluation board (STK525) or AT90USBKey Demo board
2. AT90USB microcontroller with default factory configuration (including USB bootloader)
3. USB cable (Standard A to Mini B)
4. PC running on Windows (98SE, ME, 2000, XP) with USB 1.1 or 2.0 host

3. Software Requirement

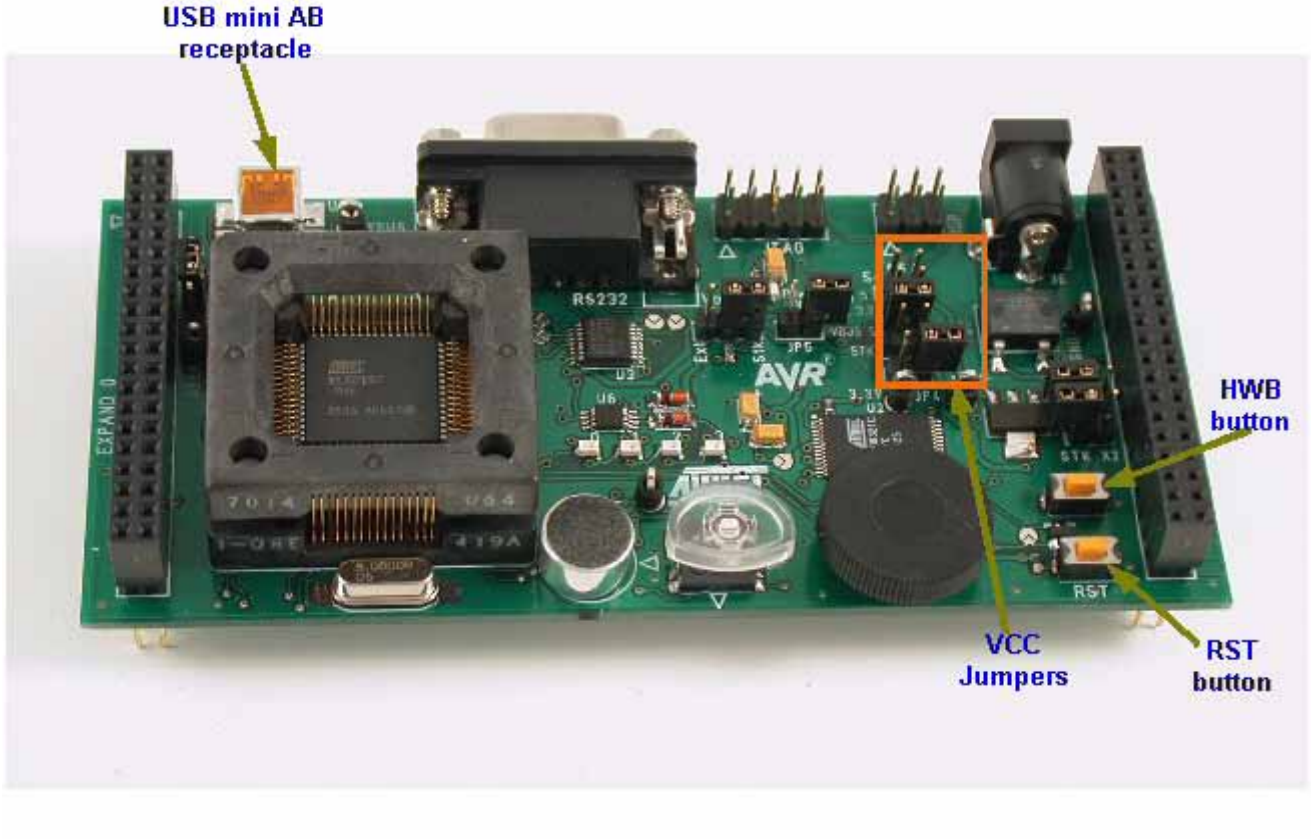
The software needed for this application includes:

1. FLIP software (Device Firmware Upgrade tool)
2. ms_df_stk525.a90 or ms_df_usbkey.a90 (included in USB CD-ROM)

4. Hardware Default Settings

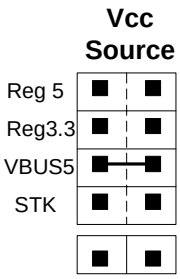
The applications are bus powered, no external power supply is required. The STK525 board must be configured as below:

Figure 4-1. STK525 Board



All the jumpers should be opened, only the Vcc Source jumper VBUS5 should be set as below:

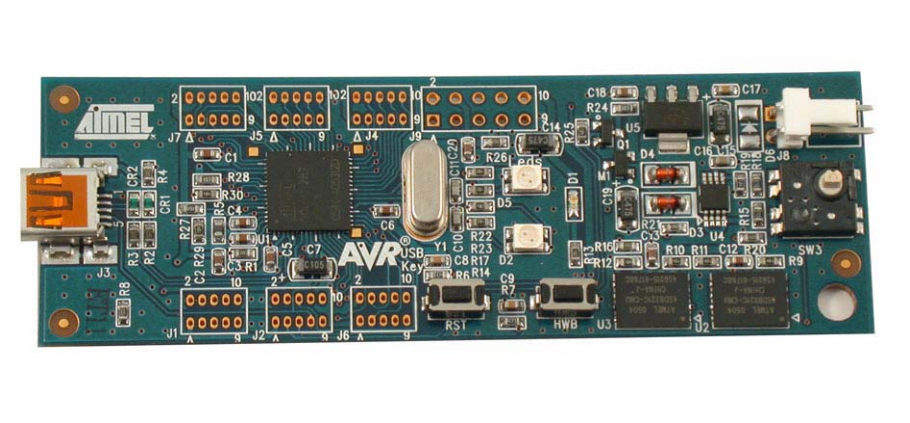
Figure 4-2. Vcc Jumpers



The microcontroller must be properly placed on its socket. Please refer to STK525 Hardware User's Guide

The AT90USBKey board does not required a specific configuration.

Figure 4-3. AT90USBKey



5. Device Firmware Upgrade

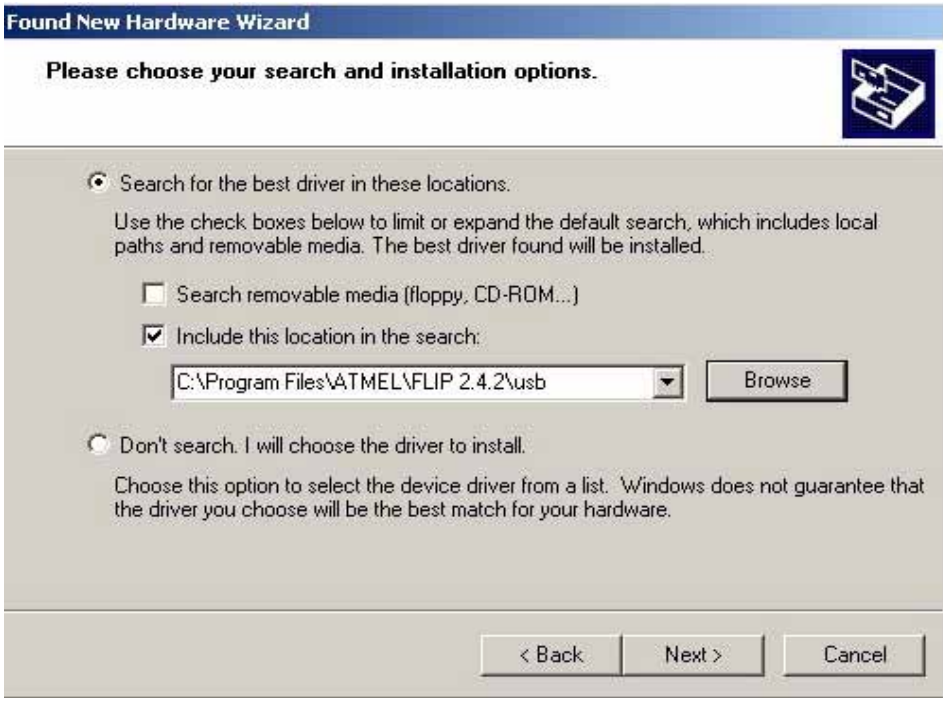
The first thing to do before starting the demo is to load the HEX file into the on-chip Flash memory of the microcontroller. The "Flip" software is the tool used to upgrade the firmware (available freely from the USB CD-ROM or Atmel website).

The following steps should be completed to allow the device starting DFU (Device Firmware Upgrade)mode and load the HEX file:

1. Install Flip software (Flip version 3.0 or above is required).
2. Push the RST (Reset) button
3. Connect the board to the PC using the USB cable (Standard A to Mini B).
4. Push the HWB (Hardware Bootloader) button
5. Release the RST button
6. Release the HWB button
7. If your hardware conditions explained above are correct, a new device detection wizard will be displayed fi you are using Flip for the first time. Please follow the instructions (the INF file is located in the USB subdirectory from Flip installation: "install path:\ATMEL\FLIP\FLIPx.x.x\usb").

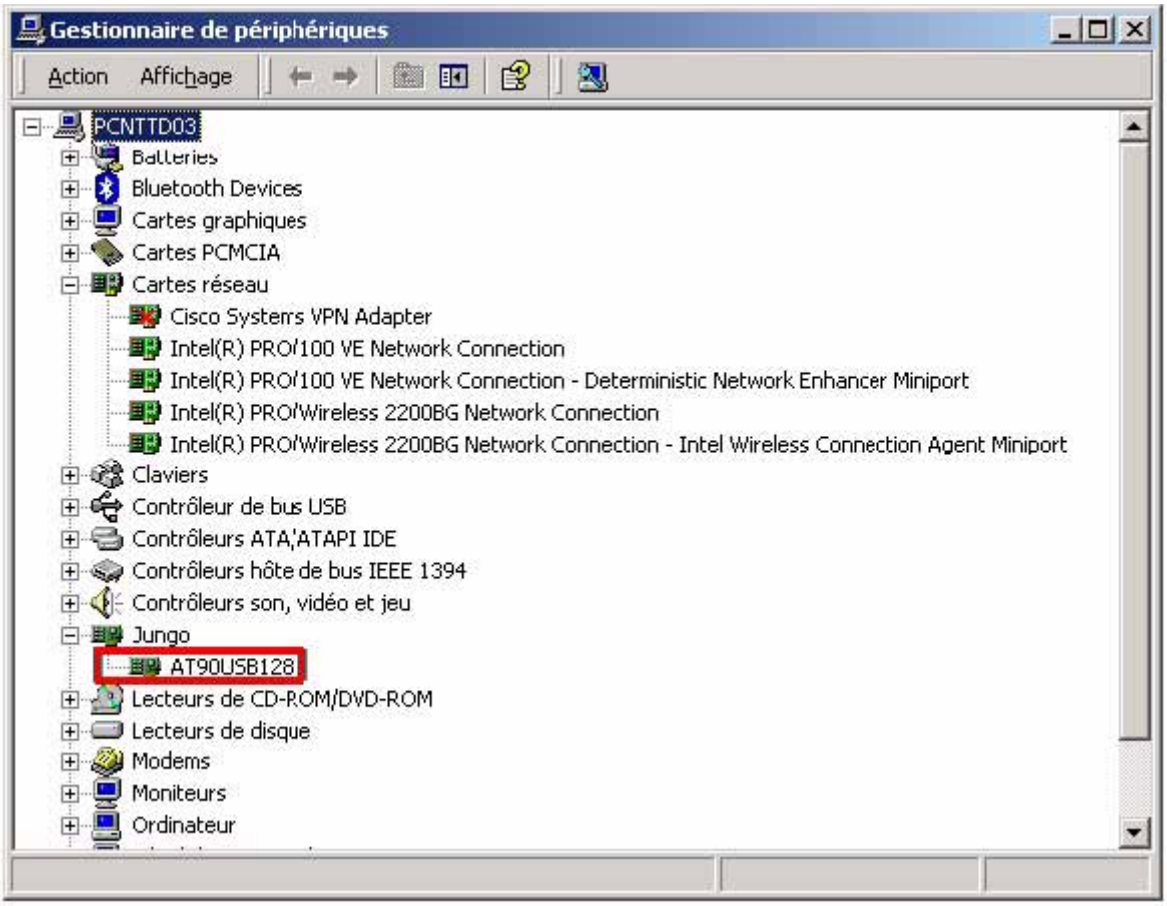
Figure 5-1. New Device Detection Wizard

Figure 5-2. Driver Location



8. Check the Device Manager, and you should see the same icon (Jungo® icon) as shown in the figure below. If not start again from the step 2.

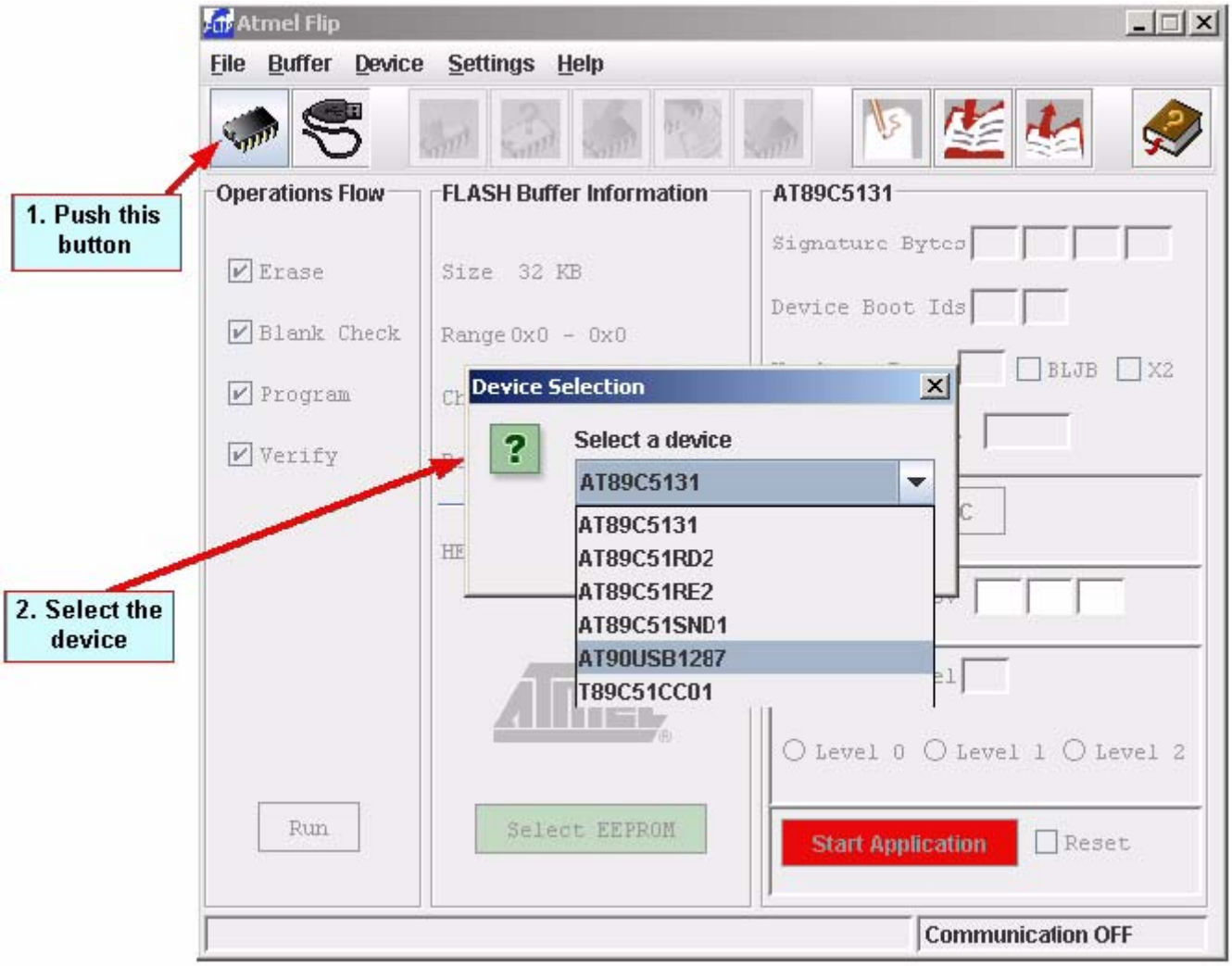
Figure 5-3. Device Manager



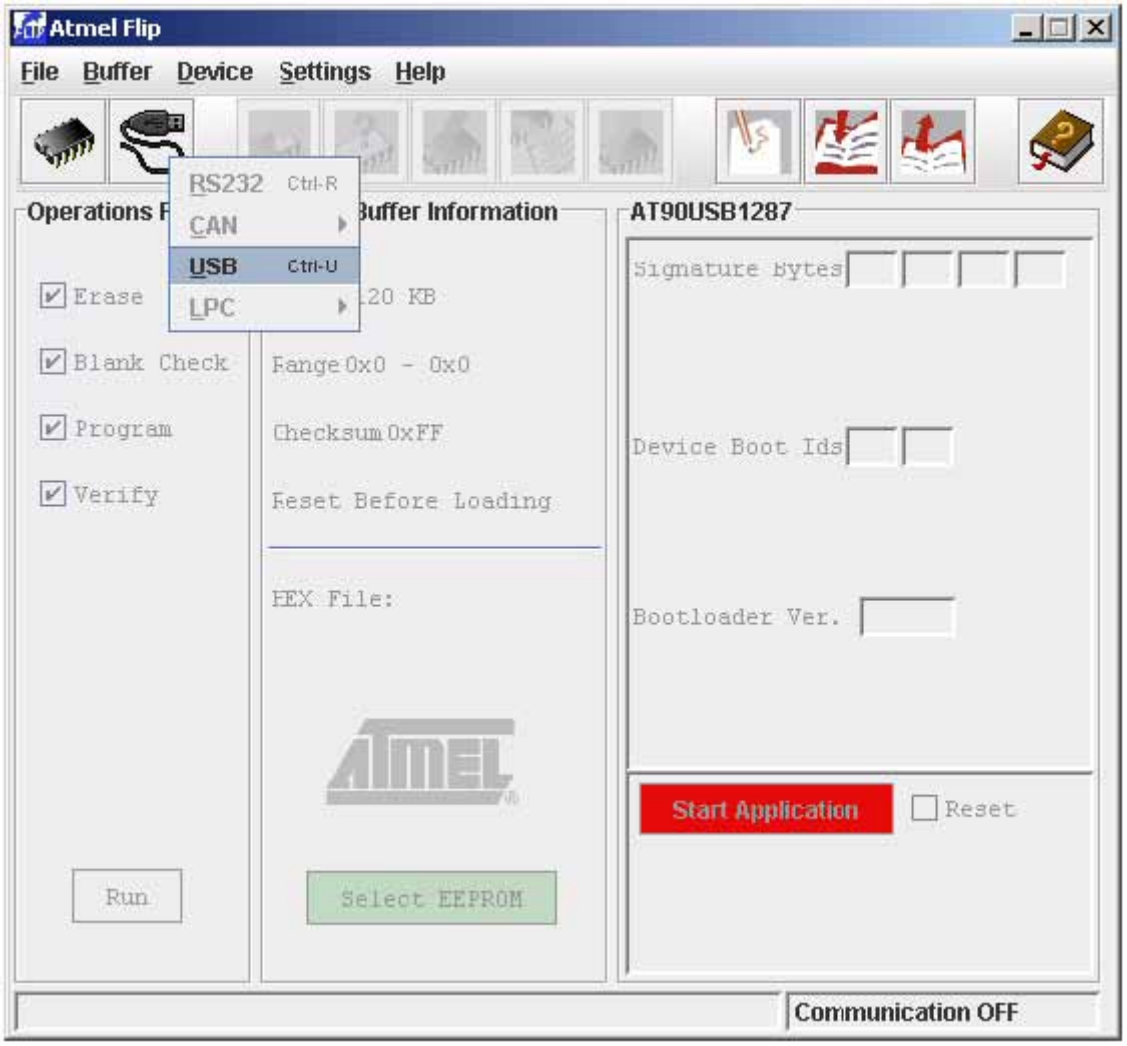
Once your device is in DFU mode, launch the Flip software and follow the instructions explained below, Figure 5-4.

1. Select AT90USB device

Figure 5-4. Device Selection

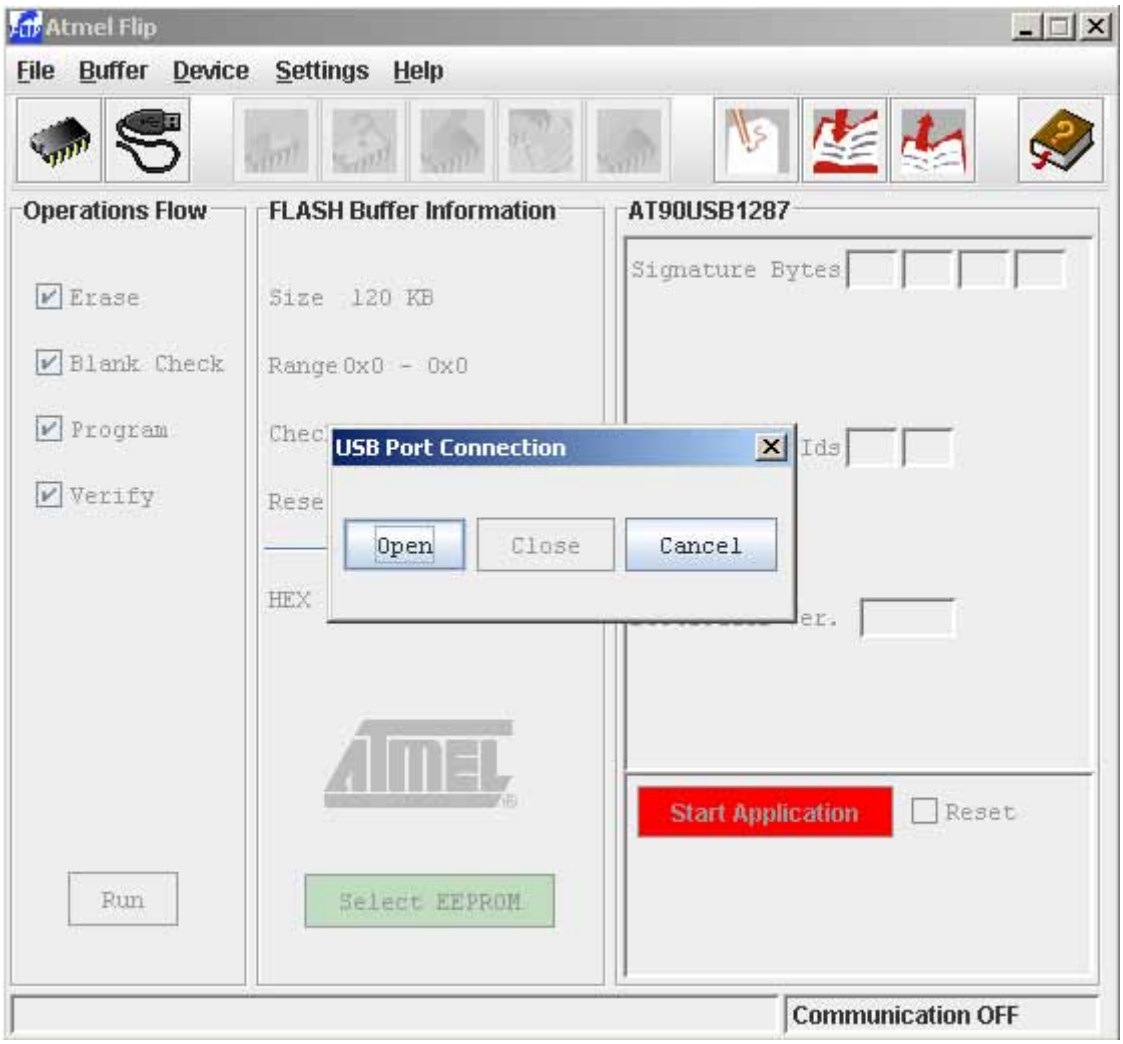

1. Push this button
2. Select the device
Atmel Flip
File Buffer Device Settings Help
Operations Flow
Erase
Blank Check
Program
Verify
Run
FLASH Buffer Information
Size 32 KB
Range 0x0 - 0x0
Select EEPROM
AT89C5131
Signature Bytes
Device Boot Ids
BLJB
X2
Level 0
Level 1
Level 2
Start Application
Reset
Communication OFF
Device Selection
Select a device
AT89C5131
AT89C5131
AT89C51RD2
AT89C51RE2
AT89C51SND1
AT90USB1287
T89C51CC01

2. Select the USB as communication mode
   a. USB Communication Mode



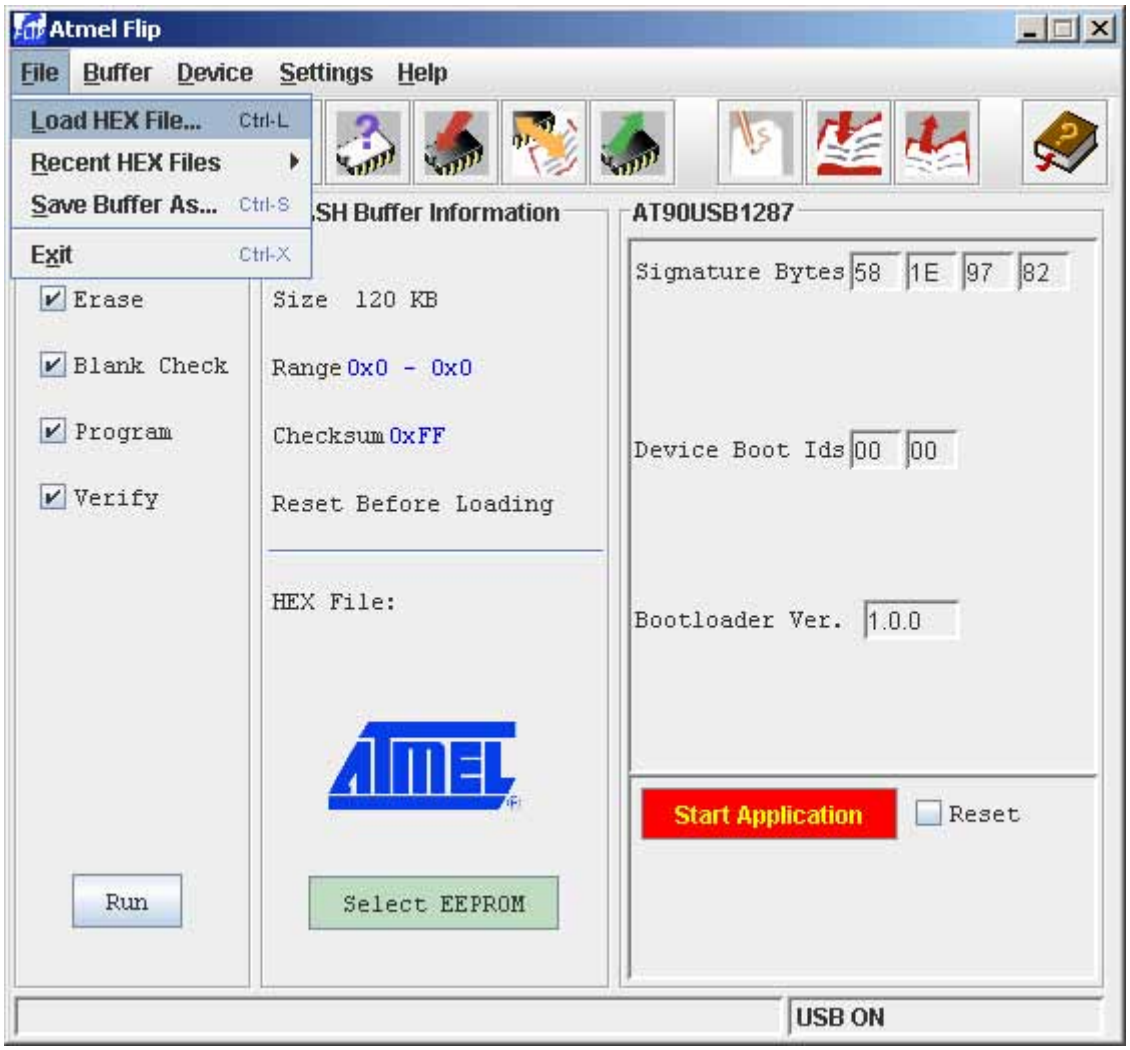
3. Open the communication

Figure 5-5. Open the USB Communication



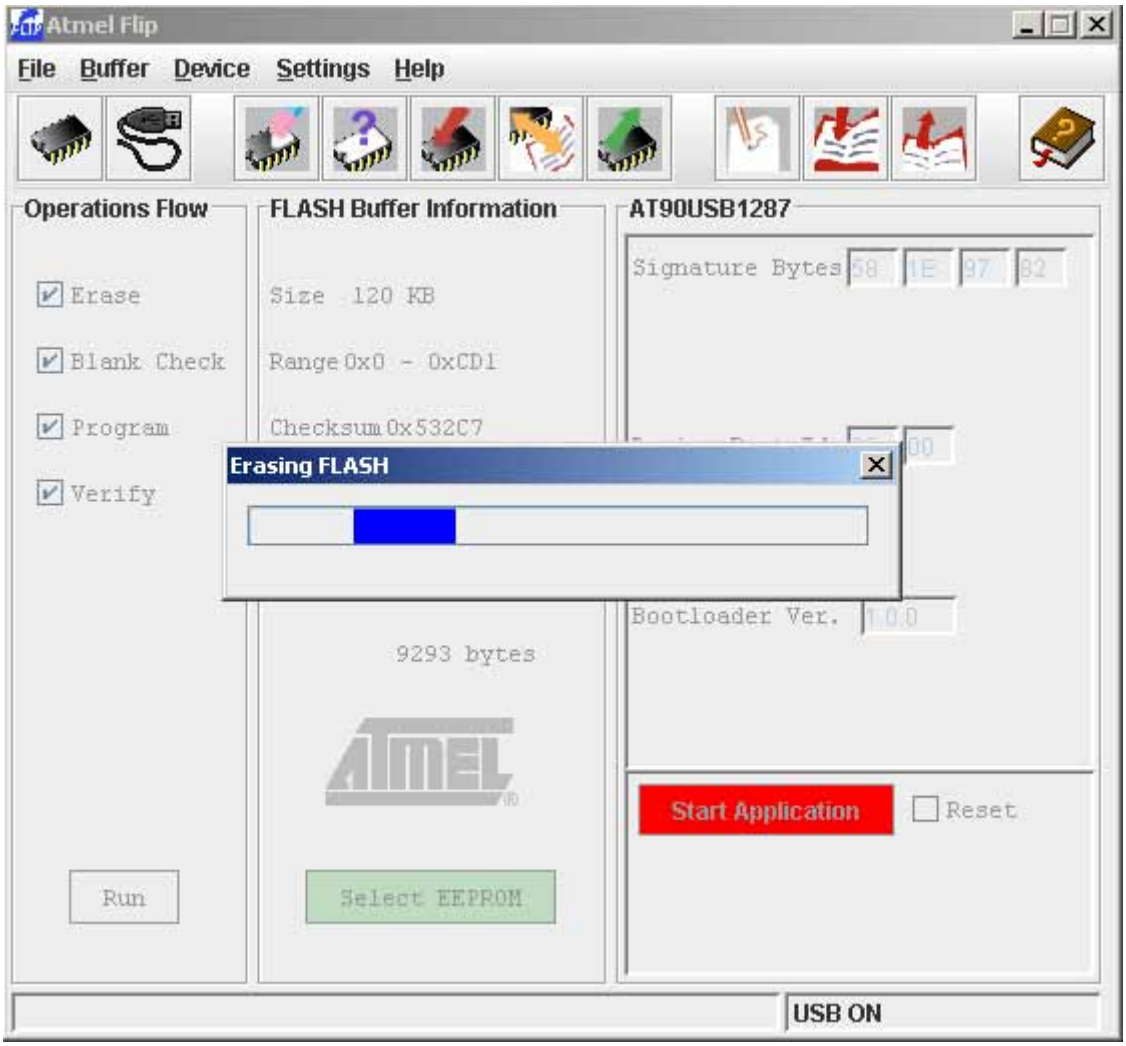
4. Choose the HEX file to load (the HEX file is including in USB CD-ROM: usb_hid_generic.hex

Figure 5-6. HEX File to Load



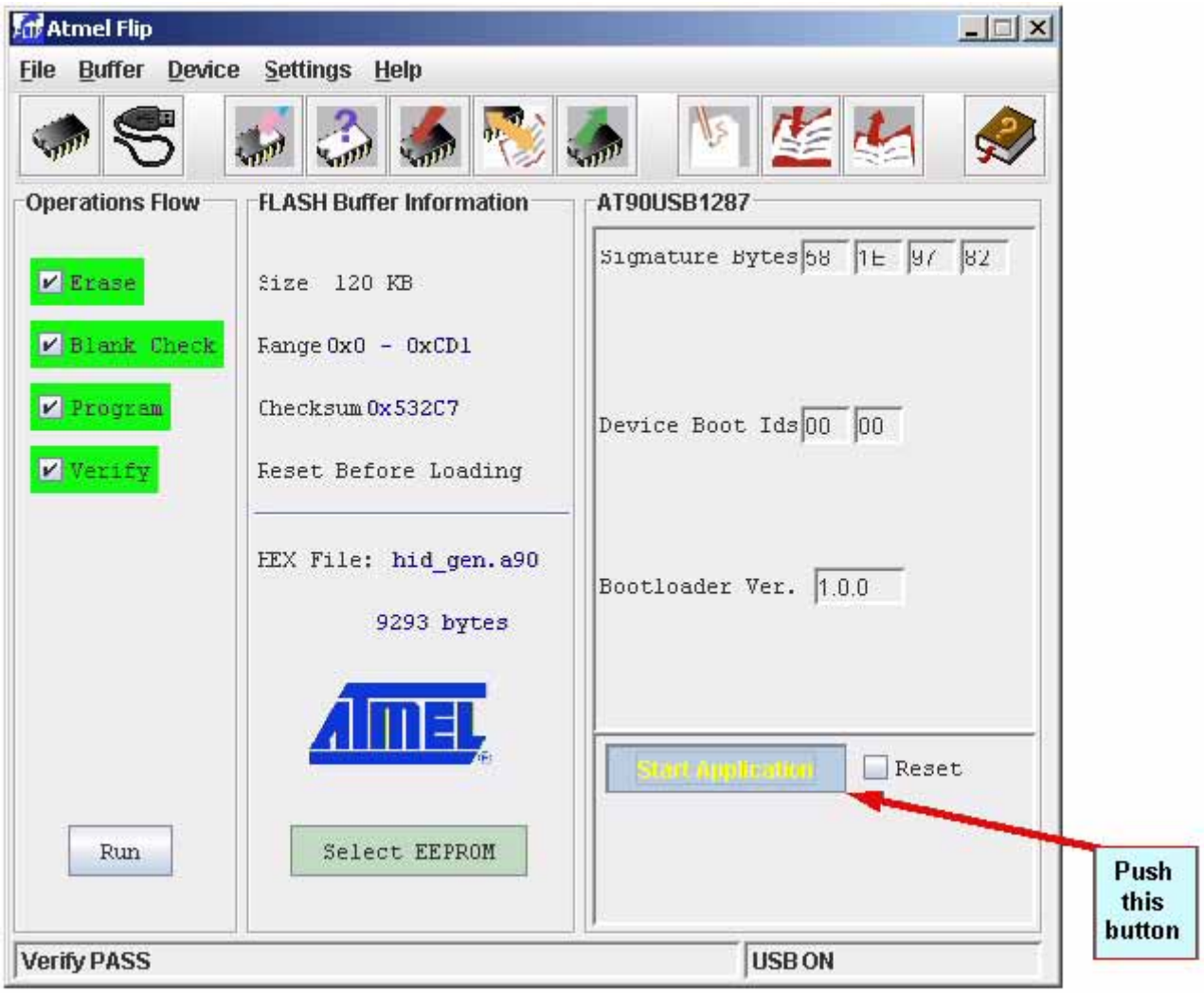
5. Load the HEX file (*Check Erase, Program* and *Verify*, then Push *Run* button)

Figure 5-7. HEX File Loading



6. Start the application

Figure 5-8. Start Application



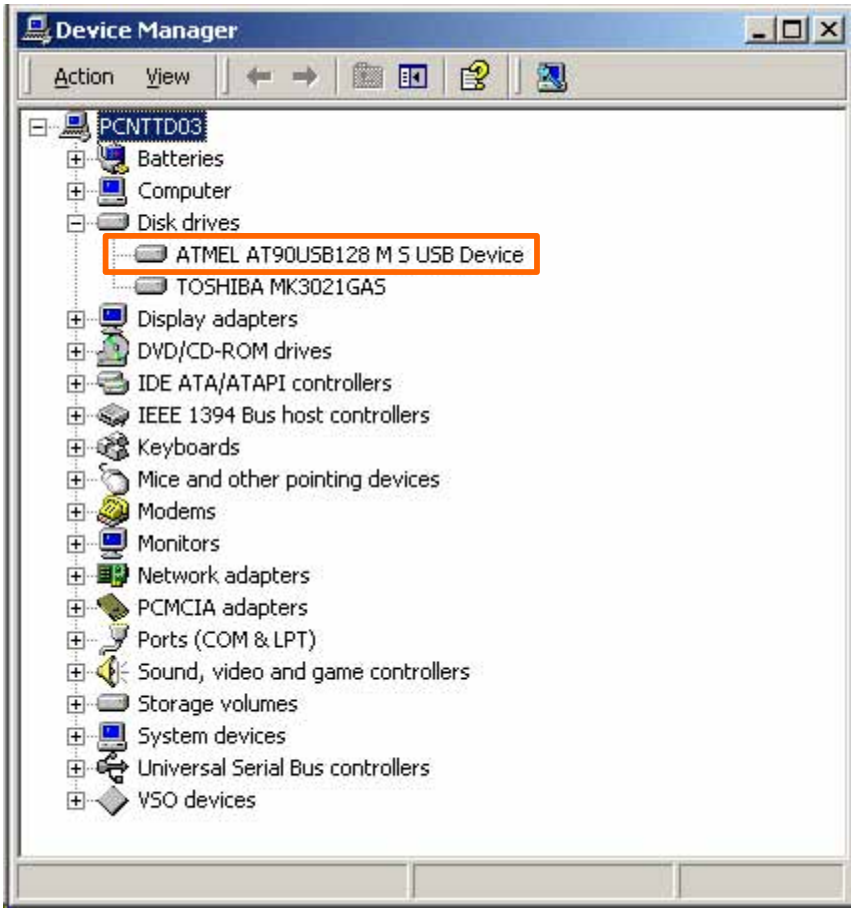
Note: The AT90USB bootloader will detach and jump into the user application when “Start Application” button is pressed.

6. Quick Start

Once your device is programmed with ms_df_stk525.a90 (for the STK525) or ms_df_usbkey.a90 (for the AT90USBKey) file, you can start using your kit as an USB key. Check that your device has enumerated as Mass Storage device (see Figure 6-1.), then launch the PC explorer, a new removable disk has appeared. Now you can start transferring files between the PC and your board.

Note: For the first use, the PC will ask you to format the removable disk.

Figure 6-1. Mass Storage Enumeration



7. Application Overview

The Mass Storage application is a simple file transfer application between the USB host and the starter kit or demonstartion board.

The USB data exchange for this application is based on the SCSI (Small Computer System Interface) commands which use two bulk endpoints (one IN and one OUT) to perform the status and data transfer. The endpoint 0 (control endpoint) is used only to perform the enumeration process, the errors management and to determine the LUN value.

In other words, the Mass Storage application is a set of SCSI commands send by the host to manage the file transfer.

The Mass Storage class allows one device to manage several storage units at the same time thanks to the LUN (Logic Unit Number).

Figure 7-1. Mass Storage Application Overview

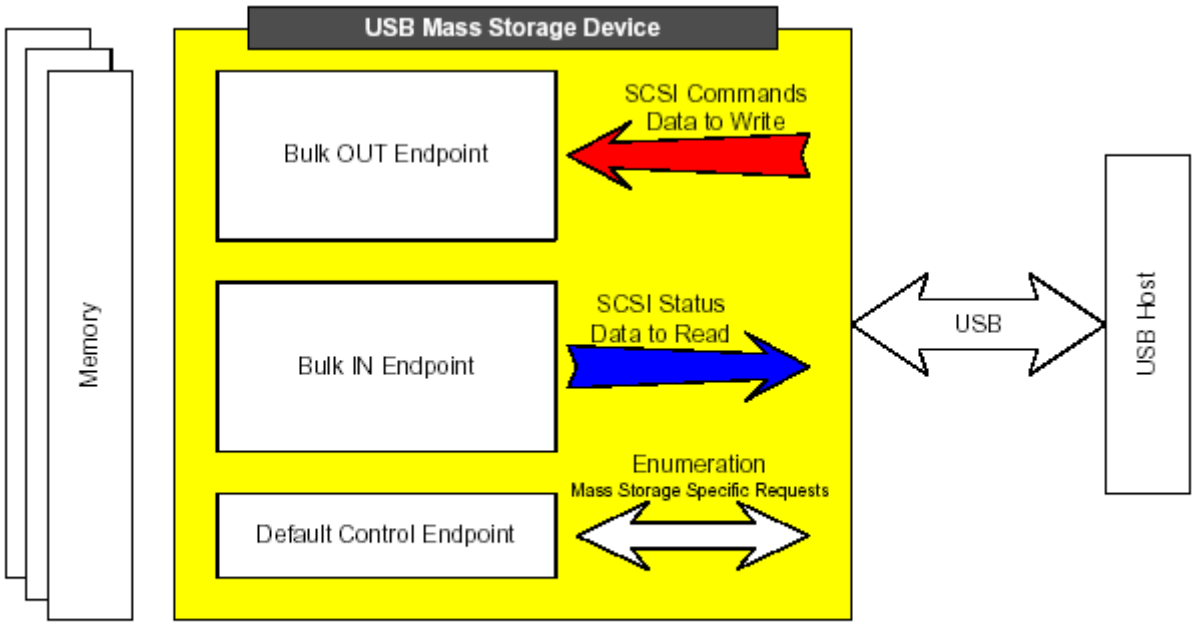


Figure 2. A Mass Storage Device seen by a USB Host

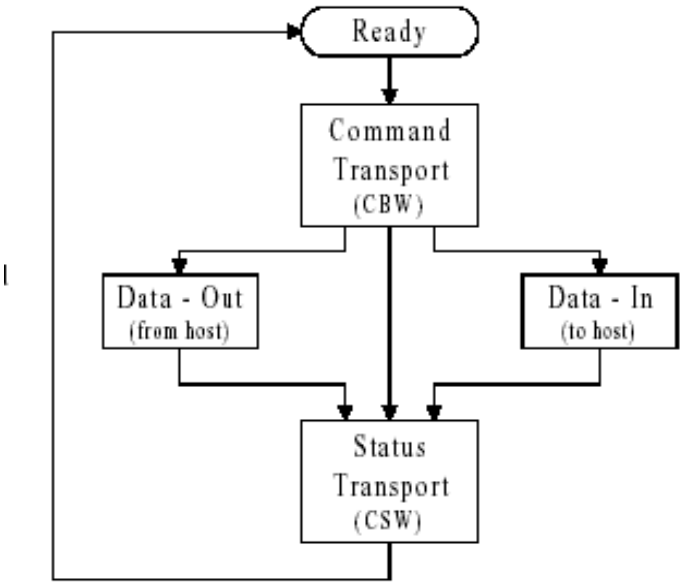
The standard enumeration process (USB chapter 9 support) is performed through the default control endpoint. This process consists of a set of parameters sent by the device to the host to identify the device class and load the appropriate drivers. These parameters are called the descriptors.

The SCSI commands are performed through both endpoints (IN or OUT). Each SCSI command is decoded and transmitted to the appropriate Storage Unit through a command set (Read, Write, is memory present, is memory write protected,...).

The memory answers are converted in SCSI status before being wrapped in USB CSW (Command Status Wrapper) and sent to the USB Host controller.

As the USB bus is a single master bus (the USB Host), each data transfer is initiated by the USB Host, following a specific Command-Data-Status flow (see figure below)

Figure 7-2. Command/Data/Status Flow

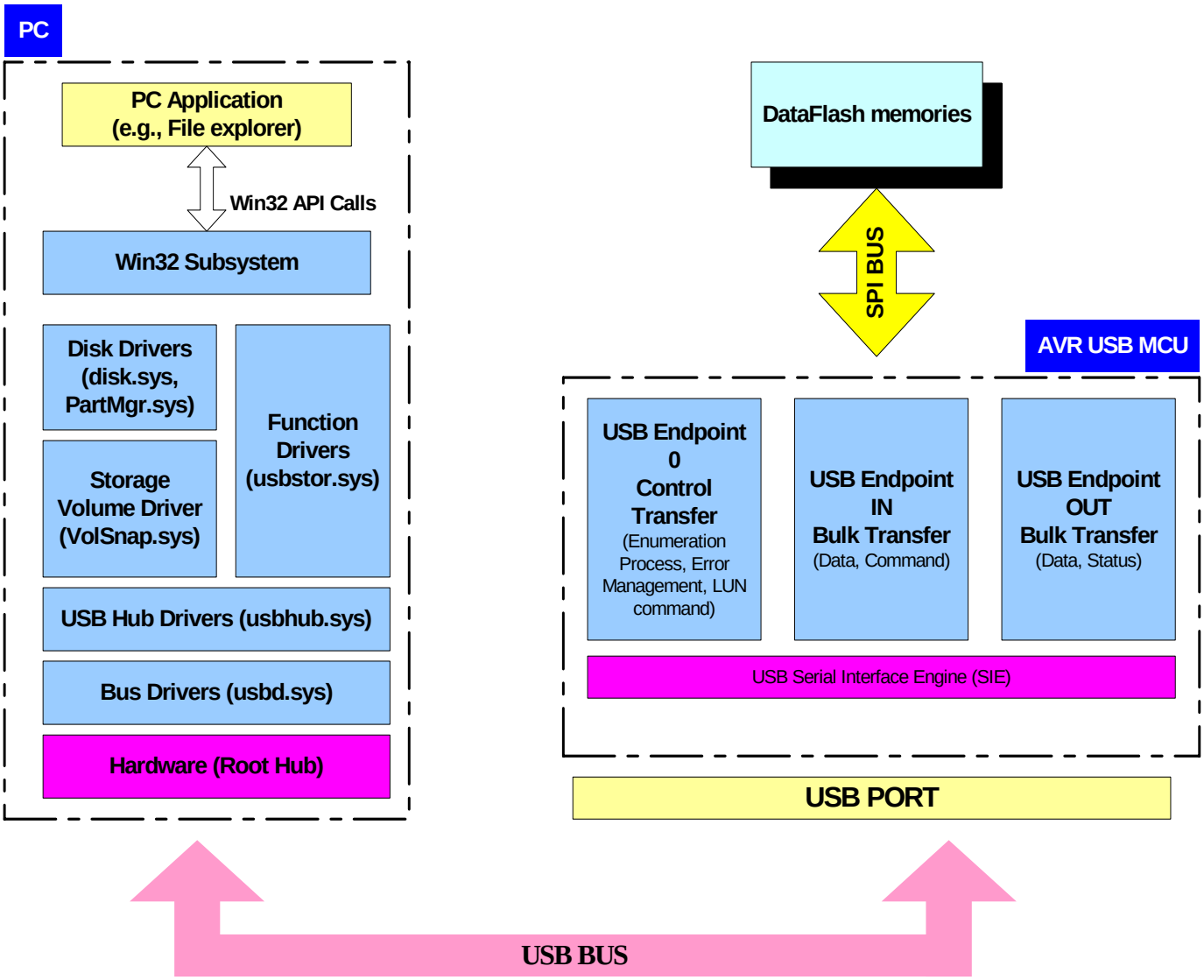


The CBW (Command Block Wrapper) contains some USB information such as the addressed LUN , the length of the SCSI command, and of course, it also contains the SCSI command for the memory.

The CSW (Command Status Wrapper) contains the SCSI status. If the status is GOOD, the Host will send the next following command. If the status is different from GOOD (FAILED, PHASE ERROR,...), the Host will ask for more information regarding the error by sending a REQUEST SENSE command.

The figure below showns an overview of the solution provided by Atmel which targets DataFlash memories: one for STK525 and two for the AT90USBKey. Physical memories can be mapped on the same logical unit and interleaved to reduce the apparent write access time. The maximum size per logical unit is limited to...

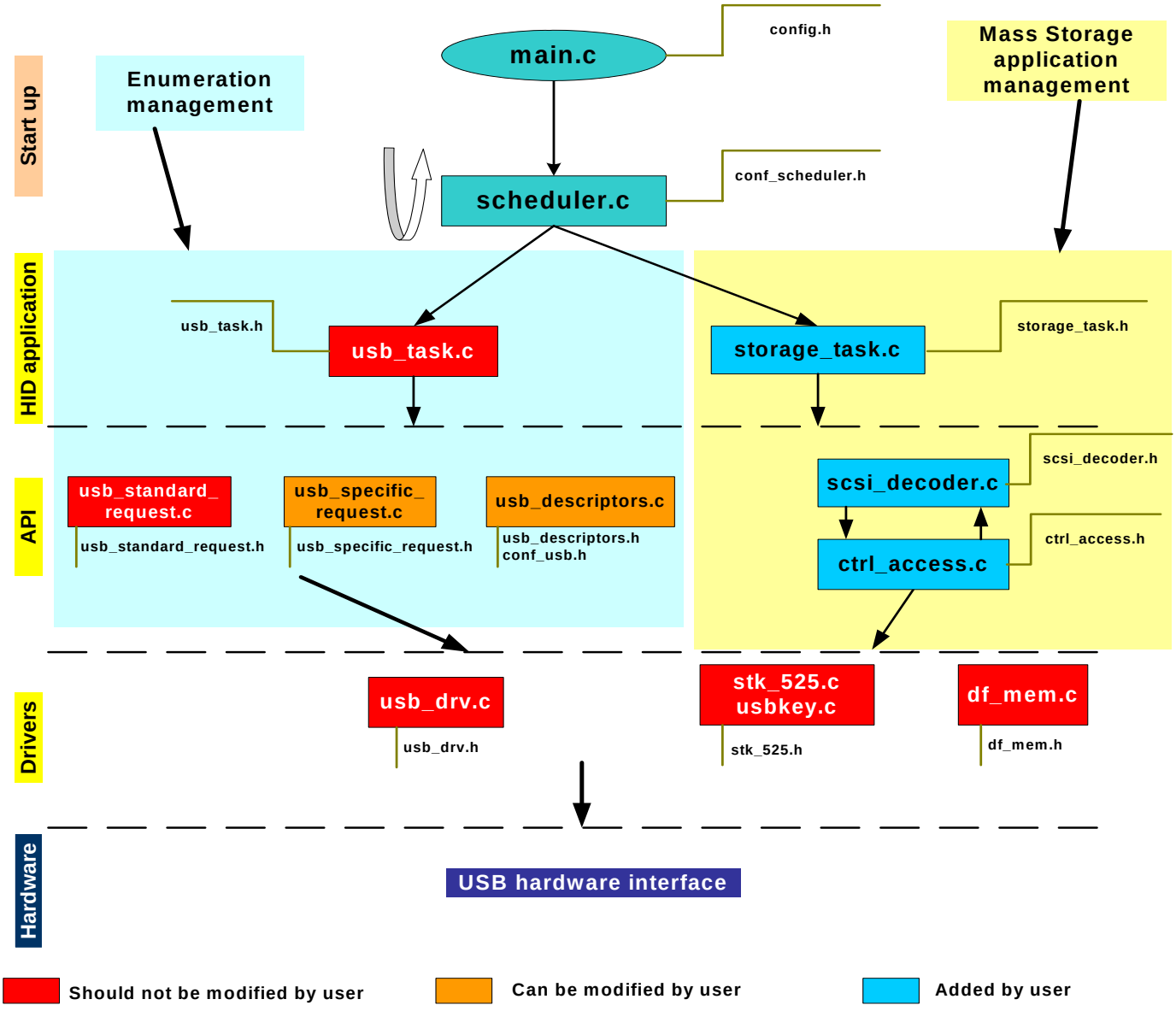
Figure 7-3. Atmel Mass Storage Solution


PC
PC Application
(e.g., File explorer)
Win32 API Calls
Win32 Subsystem
Disk Drivers
(disk.sys,
PartMgr.sys)
Storage
Volume Driver
(VolSnap.sys)
Function
Drivers
(usbstor.sys)
USB Hub Drivers (usbhub.sys)
Bus Drivers (usbd.sys)
Hardware (Root Hub)
DataFlash memories
SPI BUS
AVR USB MCU
USB Endpoint
0
Control
Transfer
(Enumeration
Process, Error
Management, LUN
command)
USB Endpoint
IN
Bulk Transfer
(Data, Command)
USB Endpoint
OUT
Bulk Transfer
(Data, Status)
USB Serial Interface Engine (SIE)
USB PORT
USB BUS

8. Firmware

As explained in the USB Firmware Architecture document (Doc 7603, included in the USB CD-ROM) all USB firmware packages are based on the same architecture (please refer to this document for more details).

Figure 8-1. Mass Storage Firmware Architecture

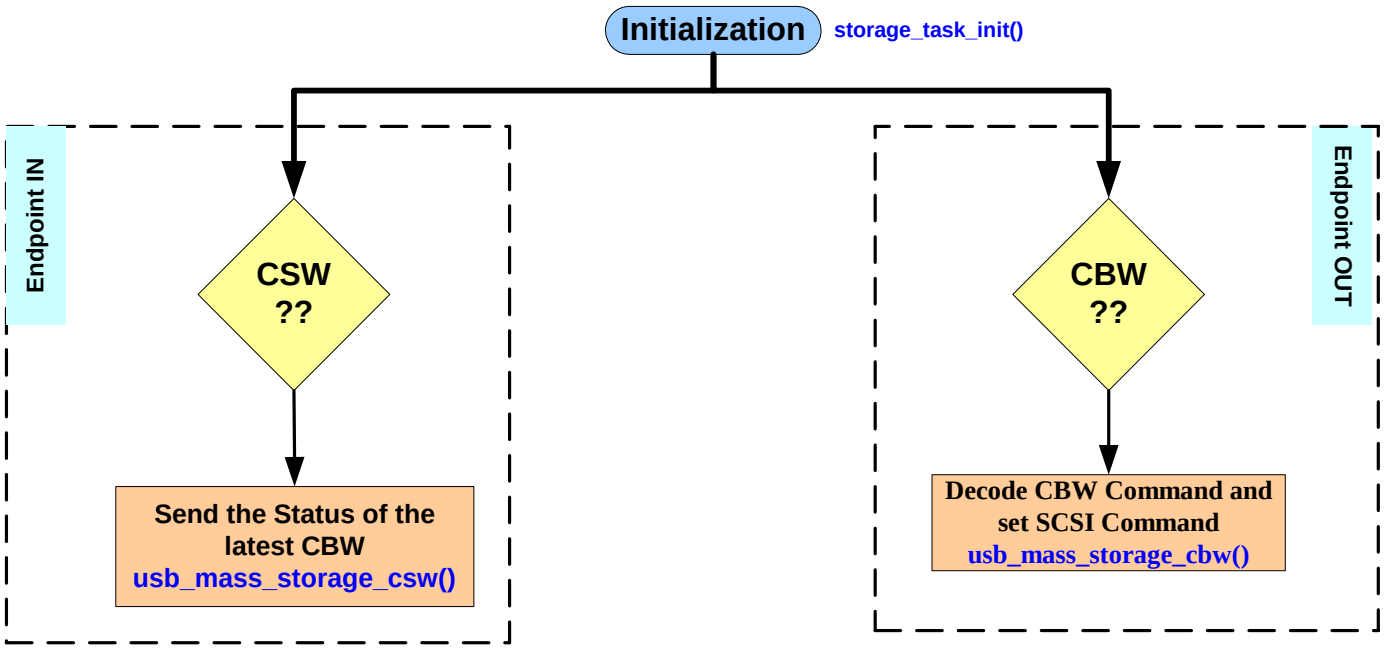


This section is dedicated to the Mass Storage module only.To customize this firmware, you have just to modify the memories drivers, the rest can be used as is. Find hereunder the explanation of the files related to the Mass Storage module:

8.1 storage_task.c

This file contains the functions to initialize the prameters of the hardware used by the application (spi, DataFlash, Leds) and to manage the commands sent by the host (Command Block Wrapper, Command Status Wrapper).

Figure 8-2. Mass Storage task



8.1.1 storage_task_init

This function performs the initialization of the device parameters and hardware resources.

8.1.2 usb_mass_storage_cbw

This function decodes the CBW (Command Block Wrapper) and store the SCSI command.

8.1.3 usb_mass_storage_csw

This function sends the status (CSW: Command Status Wrapper) of the last CBW.

8.2 stk_525.c/usbkey.c

This file contains all the routines to manage the board resources (Joystick, potentiometer, Temperature sensor, LEDs...). The user should not modify this file when using the STK525 or the AT90USBKey board. Otherwise he has to build his own hadware management file.

8.3 Memory management

Each memory is interfaced to the Atmel firmware by a specific memory driver.

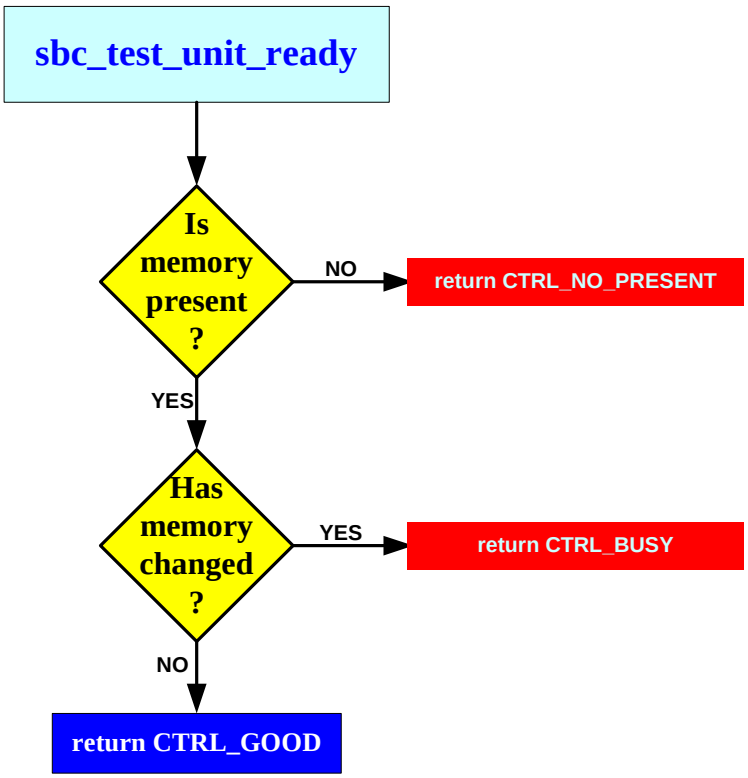
The following functions have to be implemented in order to support a memory with the USB Mass Storage Device firmware. In order to support a new memory, the developper has to write the memory driver according to this memory interface. Some functions only return the status of the memory (present, write protected, total capacity and if the memory can be removed). The other functions are used to read or write into the memory. The functions read_10 and write_10 open the memory at a specific location. The functions usb_read and usb_write manage the data transfer between the USB Controller and the memory. Most of these functions returns a Ctrl_status byte that could be:

- CTRL_GOOD: function is PASS and another command can be sent
- CTRL_FAIL: there is a FAIL in the command execution
- CTRL_NO_PRESENT: the memory is not present
- CTRL_BUSY: the current memory is not initialized or its status has changed

8.3.1 sbc_test_unit_ready

This function returns the memory state.

Figure 8-3. sbc_test_unit_ready

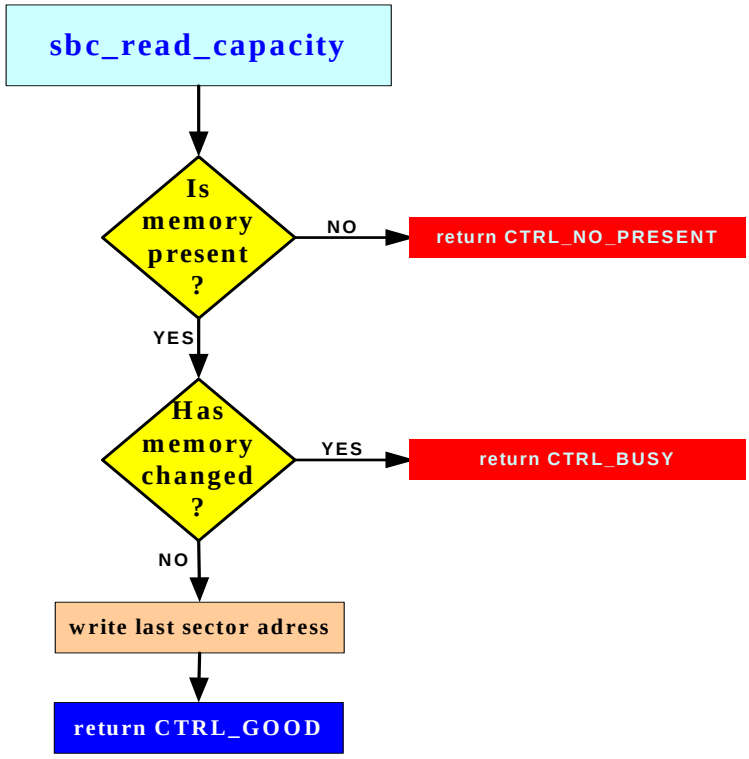


8.3.2 sbc_read_capacity

This function returns the address of the last valid sector, stored in u32_nb_sector. The sector size is fixed to 512 Bytes for OS compatibility.

For example, a memory of 16KBytes returns ((16 x 1024)/512) -1) = 31

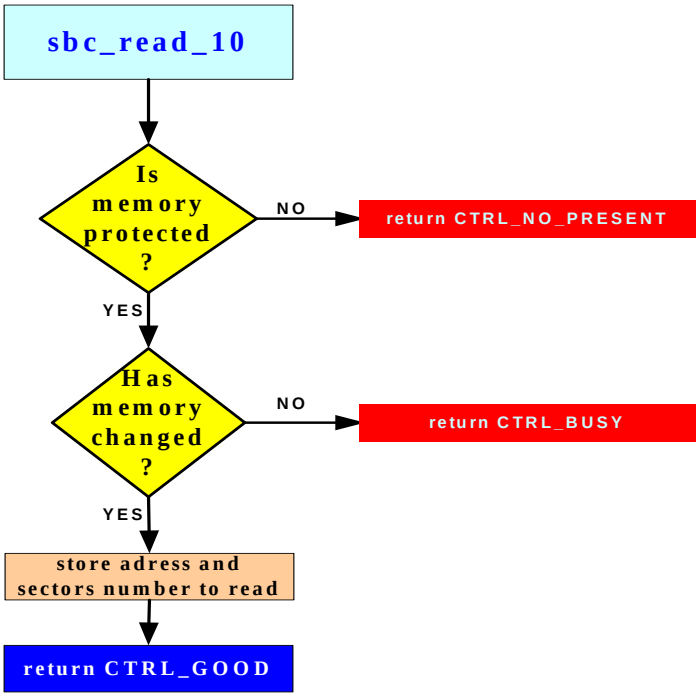
Figure 8-4. sbc_read_capacity



8.3.3 sbc_read_10

This function sets the sector address (addr) and the number of consecutive sector (512Bytes each) to read.

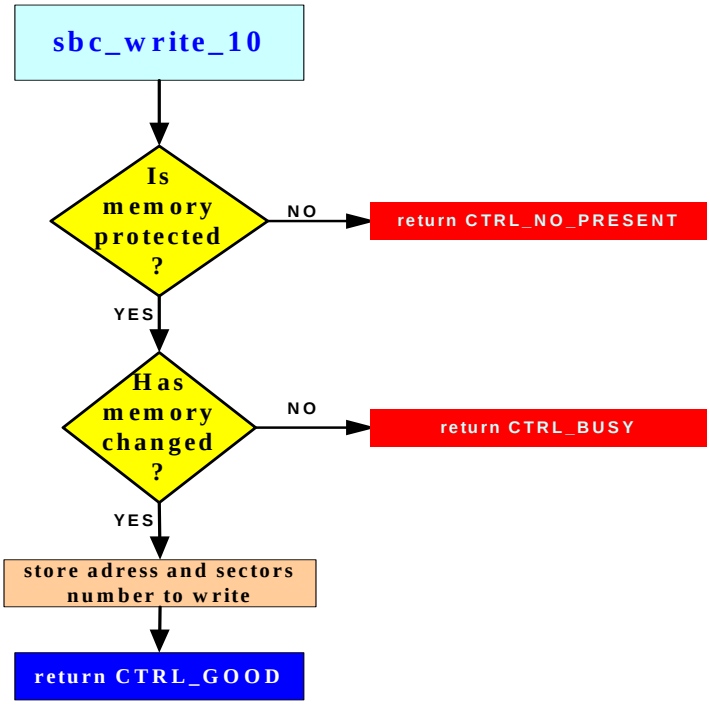
Figure 8-5. sbc_read_10



8.3.4 sbc_write_10

This function sets the sector address (addr) and the number of consecutive sector (512Bytes each) to write.

Figure 8-6. sbc_write_10



8.3.5 mem_wr_protect

This function returns FALSE if the memory is not write protected and TRUE if the memory is write protected.

Figure 8-7. mem_wr_protect removed.

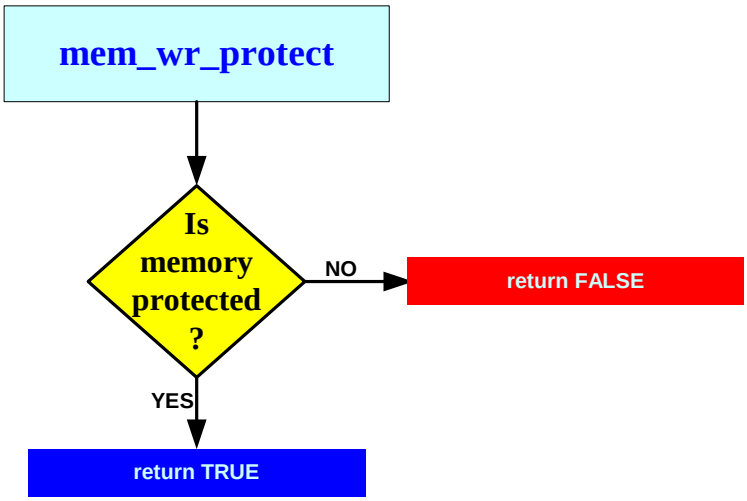
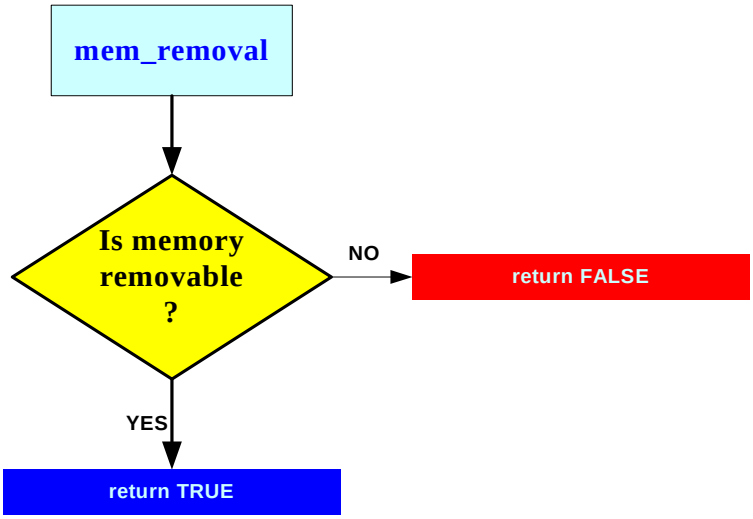


Figure 8-8. mem_removal



8.4 Integration of new memory

The integration of a memory on the USB Mass Storage stack is performed in *conf_access.h*. The corresponding LUN has to be first set to ENABLE and the corresponding functions have to be defined.

The USB Mass Storage stack supports up to 8 different LUN.

Here is an example with the DataFlash memory sets as LUN_3:

```
// Active the Logical Unit
#define  LUN_0                  DISABLE   // On-Chip flash vitual memory
#define  LUN_1                  DISABLE   // NF 2KB
#define  LUN_2                  DISABLE   // NF 512B
#define  LUN_3                  ENABLE    // Data Flash
#define  LUN_4                  DISABLE
#define  LUN_5                  DISABLE
#define  LUN_6                  DISABLE
#define  LUN_7                  DISABLE
// LUN 3 DEFINE
#if (LUN_3 == ENABLE)
#define  DF_MEM                                 ENABLE
#else
#defineDF_MEM                                 DISABLE
#endif
#define  LUN_3_INCLUDE                          "lib_mem\df\df_mem.h"
#define  Lun_3_test_unit_ready()                df_test_unit_ready()
#define  Lun_3_read_capacity(nb_sect)           df_read_capacity(nb_sect)
#define  Lun_3_wr_protect()                     df_wr_protect()
#define  Lun_3_removal()                        df_removal()
#define  Lun_3_read_10(ad, sec)                 df_read_10(ad, sec)
#define  Lun_3_usb_read()                       df_usb_read()
#define  Lun_3_write_10(ad, sec)                df_write_10(ad, sec)
#define  Lun_3_usb_write()                      df_usb_write()
```

9. PC Software

.The Mass Storage device does not require a PC software. However a PC drivers are needed for Windows 98SE, this drivers are delivered by Atmel with the Mass Storage package.

10. Limitation

11. Related Documents

AVR USB Datasheet (doc 7593)

USB Firmware Architecture (doc 7603)

USB Mass Storage class specification

Atmel Corporation

2325 Orchard Parkway
San Jose, CA 95131, USA
Tel: 1(408) 441-0311
Fax: 1(408) 487-2600

Regional Headquarters

Europe

Atmel Sarl
Route des Arsenaux 41
Case Postale 80
CH-1705 Fribourg
Switzerland
Tel: (41) 26-426-5555
Fax: (41) 26-426-5500

Asia

Room 1219
Chinachem Golden Plaza
77 Mody Road Tsimshatsui
East Kowloon
Hong Kong
Tel: (852) 2721-9778
Fax: (852) 2722-1369

Japan

9F, Tonetsu Shinkawa Bldg.
1-24-8 Shinkawa
Chuo-ku, Tokyo 104-0033
Japan
Tel: (81) 3-3523-3551
Fax: (81) 3-3523-7581

Atmel Operations

Memory

2325 Orchard Parkway
San Jose, CA 95131, USA
Tel: 1(408) 441-0311
Fax: 1(408) 436-4314

Microcontrollers

2325 Orchard Parkway
San Jose, CA 95131, USA
Tel: 1(408) 441-0311
Fax: 1(408) 436-4314

La Chantrerie
BP 70602
44306 Nantes Cedex 3, France
Tel: (33) 2-40-18-18-18
Fax: (33) 2-40-18-19-60

ASIC/ASSP/Smart Cards

Zone Industrielle
13106 Rousset Cedex, France
Tel: (33) 4-42-53-60-00
Fax: (33) 4-42-53-60-01

1150 East Cheyenne Mtn. Blvd.
Colorado Springs, CO 80906, USA
Tel: 1(719) 576-3300
Fax: 1(719) 540-1759

Scottish Enterprise Technology Park
Maxwell Building
East Kilbride G75 0QR, Scotland
Tel: (44) 1355-803-000
Fax: (44) 1355-242-743

RF/Automotive

Theresienstrasse 2
Postfach 3535
74025 Heilbronn, Germany
Tel: (49) 71-31-67-0
Fax: (49) 71-31-67-2340

1150 East Cheyenne Mtn. Blvd.
Colorado Springs, CO 80906, USA
Tel: 1(719) 576-3300
Fax: 1(719) 540-1759

Biometrics/Imaging/Hi-Rel MPU/ High Speed Converters/RF Datacom

Avenue de Rochepleine
BP 123
38521 Saint-Egreve Cedex, France
Tel: (33) 4-76-58-30-00
Fax: (33) 4-76-58-34-80

Literature Requests

www.atmel.com/literature

7631A–USB–03/06