AVR32758: AVR32 UC3 USB Host Mass Storage Bootloader

32-bit AVR® Microcontroller

Application Note

Features

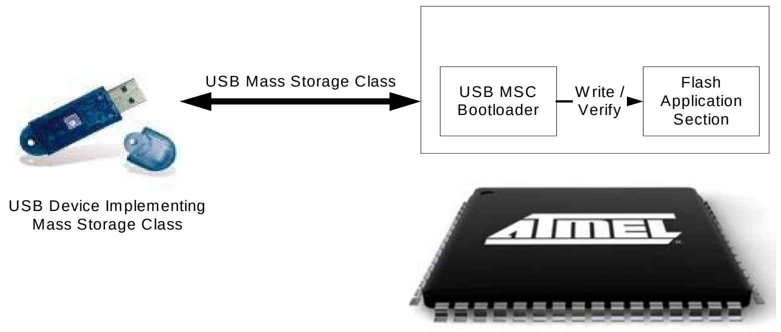
- **In-System Programming (ISP) and Field Upgrade**
  - **Configurable Start Condition with an Upgrade Firmware File 'avr32fwupgrade.uc3' Located on a USB Device Implementing the USB Mass Storage Class**
- **USB Protocol**
  - **Based on the USB Host Mass Storage Class (MSC)**
- **USB MSC bootloader is additional to the pre-programmed USB Device Firmware Upgrade (DFU) bootloader**
- **No computer required**

1. Description

This USB host mass storage class bootloader allows to perform In-System Programming of AVR®32 UC3 parts from a USB device implementing the Mass Storage class (USB drive), without:

- removing the part from the system
- any external programming interface other than USB.

Figure 1-1. Physical Environment


USB Mass Storage Class
USB MSC Bootloader
Write / Verify
Flash Application Section
USB Device Implementing Mass Storage Class

The condition used to request the start of the ISP is the presence of a specific upgrade file located on a USB device implementing the mass storage class.

This document describes the USB host mass storage class bootloader functionalities and its usage in various contexts.

Note that all the AT32UC3 devices with a USB interface are shipped with a pre-programmed USB Device Firmware Upgrade (DFU) bootloader. This DFU bootloader behaves as a USB DFU device and is used to upgrade the firmware of the AVR32 UC3 from a computer (acting as a USB DFU host).

The UC3 USB MSC bootloader (acting as a USB host) comes in addition to the first bootloader (acting as USB device), but is not pre-programmed on the shipped part. Both bootloaders can be present and used on the same part.

7818B–AVR32–07/09

In the rest of this document, we will suppose that the USB DFU bootloader is always present and located at the beginning of the internal Flash memory. Consequently, the USB MSC bootloader is located in the memory map above the USB DFU bootloader.

For more information on the USB DFU bootloader, please refer to the AVR32 UC3 USB DFU Bootloader datasheet.

This application note comes with a software package avr32758.zip which contains project examples and script to program the bootloaders.

2. Related Parts

This documentation applies to the following AT32UC3 parts:

- AT32UC3A0512
- AT32UC3A0256
- AT32UC3A0128
- AT32UC3A1512
- AT32UC3A1256
- AT32UC3A1128
- AT32UC3B0256
- AT32UC3B0128
- AT32UC3B064
- AT32UC3B1256
- AT32UC3B1128
- AT32UC3B164

The bootloader needs to be re-compiled:

- For each AT32UC3x series (AT32UC3A, AT32UC3B) due to differences in the MCU peripheral memory map. The functionalities are however the same between series.
- For each hardware board, since the oscillator value and the USB VBOF pin multiplexing value may be different.

The provided software example are by default configured with the largest flash size part (i.e AT32UC3A0512) with the series evaluation kit (i.e. EVK1100 for the UC3A).

3. Related Items

- AT32UC3A Series Datasheet:
http://www.atmel.com/dyn/resources/prod_documents/doc32058.pdf
- AT32UC3B Series Datasheet:
http://www.atmel.com/dyn/resources/prod_documents/doc32059.pdf
- AVR32 UC3 Software Framework:
http://www.atmel.com/dyn/products/tools_card.asp?tool_id=4192
- AVR32 Studio:
http://www.atmel.com/dyn/products/tools_card.asp?tool_id=4116
- AVR32 GNU toolchain:
http://www.atmel.com/dyn/products/tools_card.asp?tool_id=4118
- AVR32 UC3 USB DFU Bootloader datasheet:
http://www.atmel.com/dyn/resources/prod_documents/doc7745.pdf

4. Abbreviations

• ISP:	In-System Programming
• BOD:	Brown-Out Detector
• USB:	Universal Serial Bus
• DFU:	Device Firmware Upgrade
• MSC:	Mass Storage Class
• Udrive:	USB MSC Device

5. Inner Working

The USB MSC bootloader manages the USB communication protocol and performs erase, program and verify operations of the on-chip memories.

The **USB DFU bootloader** (USB device only) is located at the beginning of the on-chip Flash. The USB DFU bootloader size is 8kB and is protected against erasure by the BOOTPROT fuses.

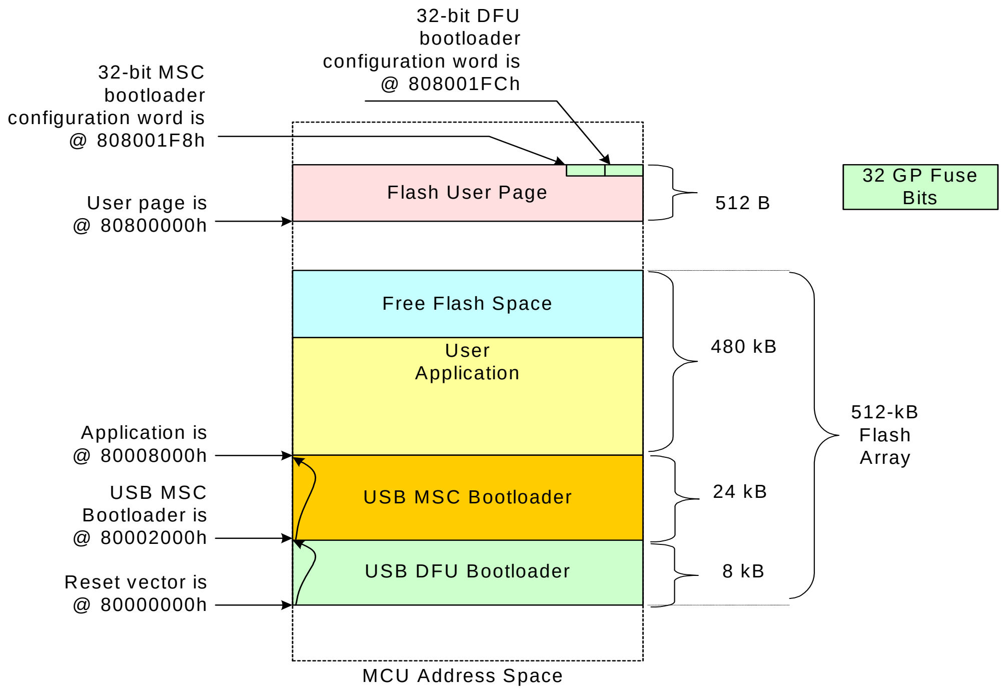
The **USB MSC bootloader** (USB host only) is located above the USB DFU bootloader in the on-chip Flash as detailed in the Figure 5-1, page 4. The USB MSC bootloader is not protected by the BOOTPROT fuse, however the programming script, that programs the two bootloaders, is using the Flash lock bits during the programming to protect the two bootloaders.

5.1 Memory Layout

An AT32UC3 part having the bootloader programmed resets as any other part at 80000000h. The first bootloader execution begins here:

- The USB DFU bootloader first performs its boot process to know whether it should start the USB DFU ISP or another program located at 80002000h. If the tested conditions indicate that the USB DFU ISP should be started, then execution continues in the USB DFU bootloader area, i.e. between 80000000h and 80002000h, else the execution resumes at 80002000h and starts the USB MSC bootloader or the user's application whether the USB MSC bootloader is programmed or not.
- The USB MSC bootloader performs its own boot process to know if it should start the USB MSC ISP or the application located at 80008000h. If the tested conditions indicate that the USB MSC ISP should be started, then execution continues in the USB MSC bootloader area, i.e. between 80002000h and 80008000h, else the bootloader launches the application at 80008000h.

Figure 5-1. AT32UC3A0512 Non-Volatile Memory Layout with USB DFU Bootloader and USB MSC Bootloader


32-bit DFU bootloader configuration word is @ 808001FCh
32-bit MSC bootloader configuration word is @ 808001F8h
Flash User Page
512 B
32 GP Fuse Bits
User page is @ 80800000h
Free Flash Space
User Application
480 kB
512-kB Flash Array
Application is @ 80008000h
USB MSC Bootloader
24 kB
USB MSC Bootloader is @ 80002000h
USB DFU Bootloader
8 kB
Reset vector is @ 80000000h
MCU Address Space

5.2 USB DFU Bootloader Configuration

The conditions tested by the USB DFU boot process are configured by the general-purpose fuse bits located outside of the MCU address space and by a 32-bit configuration word located at the end of the Flash User page.

The DFU bootloader has a configuration which determines the behavior of the boot process and of the ISP. This configuration is non-volatile and is stored Flash User page (see Figure 5-1). The USB DFU configuration word is located in the Flash user page at 808001FCh.

Note:

1. Refer to the AT32UC3 datasheets (see Section 3, page 2) for further information about the Flash User page.
2. See the USB DFU Bootloader datasheet (see Section 3, page 2) for further information about the DFU configuration.

5.3 USB MSC Bootloader Configuration

A 32-bit word located near the end of the Flash user page, before the USB DFU configuration word (see Section 5.3.1, page 4), is used internally by the boot process

5.3.1 Programming the User Page USB MSC Word

The USB MSC bootloader uses one word located in the Flash user page at 808001F8h for its boot process.

The USB MSC bootloader will start after a reset when this word is set to a specific value by the user application. Refer to Section 6.2.1.1, page 7 for further information.

5.3.2 Boot Process

After being started by the USB DFU bootloader, the USB MSC bootloader starts its boot process and checks the following condition:

- A USB key implementing the USB mass storage device class (compatible with, full speed or high speed USB keys) is plugged in,

Note: The Udrive should be previously formatted with one of the following file system: FAT12, FAT16 or FAT32 and contain only one single file named “avr32fwupgrade.uc3” (default name), generated by the script described in Section 6.2.3, page 9.

Once the new user application firmware is programmed in the on-chip Flash memory (in the range [80008000h-end of the Flash array]), the USB MSC bootloader updates the Flash user page configuration word (to avoid restart at next start-up) and resets the part.

6. Using The USB MSC Bootloader

6.1 Building a USB MSC Bootloader Compliant Application

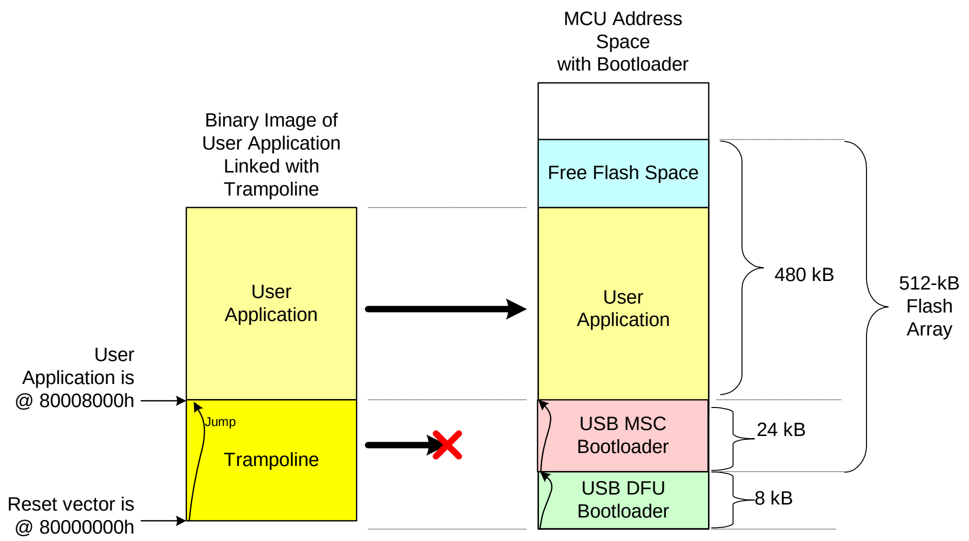
An example is provided in the software package, refer to Section 7.2 “Projects” on page 10.

6.1.1 Taking care of the bootloader overlay

In order to avoid that application code being placed inside the bootloaders area, a trampoline code must be placed at the reset vector (80000000h) that simply jumps to the beginning of the user application (80008000h). This trampoline code ensures that the application code starts at 80008000h and is located above this address.

When programming the user application with the USB MSC bootloader, the USB MSC bootloader takes into consideration the whole binary image including the trampoline. Since the trampoline cannot overwrite the USB DFU and the USB MSC bootloaders, it is not programmed and thus, only the user application (starting at location 80008000h) is programmed.

Figure 6-1. User Application Programming on AT32UC3A0512 with the USB MSC Bootloader


MCU Address Space with Bootloader
Binary Image of User Application Linked with Trampoline
Free Flash Space
User Application
User Application
480 kB
512-kB Flash Array
User Application is @ 80008000h
Jump
Trampoline
USB MSC Bootloader
24 kB
USB DFU Bootloader
8 kB
Reset vector is @ 80000000h

The trampoline and configuration files for GCC and IAR® compilers can be found into:
`\SERVICES\USB\CLASS\MASS_STORAGE\EXAMPLES\ISP\BOOT`

6.1.1.1 AVR32 Studio

- Add the `trampoline.S` and the `conf_isp.h` to the project assembly source file.
- The entry point must be set to the `_trampoline` section. Add the "`-Wl,-e,_trampoline`" to the AVR32/GNU C Linker / Miscellaneous section of the project properties.

6.1.1.2 GCC Compiler

- Add the `trampoline.S` and the `conf_isp.h` to the project assembly source file.
- The entry point must be set to the `_trampoline` section. In the config.mk, simply add `-e,_trampoline` to the linker option `LD_EXTRA_FLAGS`.

6.1.1.3 IAR Compiler

- Add the `trampoline.s82` and the `conf_isp.h` to the project source file.
- In the project option, linker->Config field, specify "override the default program entry" with Entry label: `__trampoline`.

6.1.2 Starting the USB MSC bootloader from the application

To allow upgrade of the existing application in the field, the application must provide the way to restart the bootloader.
To activate the ISP, the user application must set the configuration word located in the Flash user page at 808001F8h to a specific value (`ISP_FORCE_VALUE`, see Section 6.2.1.1, page 7).

These C code lines program the USB MSC configuration word in the Flash user page to start the USB MSC bootloader after the next reset:

```
Disable_global_interrupt();
// Write at destination (AVR32_FLASHC_USER_PAGE + ISP_FORCE_OFFSET) the value
// ISP_FORCE_VALUE. Size of ISP_FORCE_VALUE is 4 bytes.
flashc_memset32(AVR32_FLASHC_USER_PAGE+ISP_FORCE_OFFSET, ISP_FORCE_VALUE, 4, TRUE);
wdt_enable(17777);
while (1);          // wait WDT time-out to reset and start the MSC bootloader
```

Note: The code above supposes that the user application uses the software drivers from the UC3 Software Framework (DRIVERS/FLASHC/flashc.c, DRIVERS/WDT/wdt.c and dependency files).

Refer to the Section 7.2.2 "User Project" on page 10 for example of user project.

6.2 Implementing the USB MSC Bootloader

The USB MSC bootloader can be programmed to the UC3 part using the USB DFU bootloader through its USB interface or using a JTAG programmer (AVR ONE!, JTAGICE mkII) through its JTAG interface.

This section is a step-by-step guide of the USB MSC bootloader:

- How to build the USB MSC bootloader,
- How to program the USB MSC bootloader,
- How to activate the USB MSC bootloader,
- How to build a USB MSC bootloader compliant application,
- How to build the application upgrade file,
- How to setup the Udrive for UC3 firmware upgrade,
- How to program the user firmware with the USB MSC bootloader.

6.2.1 Building the USB MSC Bootloader .hex File

An example of USB MSC bootloader project is provided in the software package, refer to Section 7.2 “Projects” on page 10.

6.2.1.1 Static Configuration

The static configuration of the USB MSC bootloader is done in the conf_isp.h file located in: `SERVICES\USB\CLASS\MASS_STORAGE\EXAMPLES\ISP\CONF`.

The following configuration items are available:

- ISP version. Default value is 1000000h (stands for 1.0.0 build number 0).
- Offset of the USB MSC configuration word in the Flash user page (see Section 5.3.1, page 4). Default value is 1F8h (i.e. it is located at address 1F8h of the page).
- Value of the USB MSC configuration word in the Flash user page to start the USB MSC bootloader (ISP_FORCE_VALUE). Default value is:
  `('M' << 24 | 'S' << 16 | 'I' << 8 | 'F')`.
- Firmware upgrade file name. Default value is “`avr32fwupgrade.uc3`”.
- Program start offset: the offset of the application start in the Flash. Default value is 8000h (i.e. the address is 80008000h).

6.2.1.2 Hardware Considerations

In order to work, the ISP requires that either an external clock or a crystal is mounted on Osc0. Osc1 can be used instead of Osc0, but in this case, the user has to change the `ISP_OSC` preprocessor definition to `1` in
`SERVICES/USB/CLASS/MASS_STORAGE/EXAMPLES/ISP/AT32UC3A0512/GCC/config.mk`
for GCC or in the
`SERVICES/USB/CLASS/MASS_STORAGE/EXAMPLES/ISP/AT32UC3A0512/GCC//IAR/isp.eww`
workspace project options for IAR.

Note: By default, the USB DFU bootloader preprogrammed on all UC3 parts is configured to run on OSC0.

The default hardware pin connection is set for the EVK1100 board for UC3A family and the EVK1101 board for UC3B family.

Board definition can be changed in
`SERVICES/USB/CLASS/MASS_STORAGE/EXAMPLES/ISP/AT32UC3x_EVK110x/GCC/config.mk` for GCC
or in the
`SERVICES/USB/CLASS/MASS_STORAGE/EXAMPLES/ISP/AT32UC3x_EVK110x/IAR/isp.eww`
workspace project options for IAR.

6.2.1.3 Projects

The GCC and IAR project files are stored in
\SERVICES\USB\CLASS\MASS_STORAGE\EXAMPLES\ISP.

Refer to Section 8.3 “Building the UC3A USB MSC project” on page 13.

6.2.2 Programming the USB MSC Bootloader

A script example is provided in the software package, refer to Section 7.4 “Script to Program the Bootloaders” on page 10.

The following command script is an example to program through JTAG:

- the released USB DFU bootloader (in Flash array),

- the USB MSC bootloader (in Flash array),
- the user application firmware (in Flash array),
- the USB DFU ISP configuration word (in the Flash user page),
- the general-purpose fuse bits,
- the security bit.

And start the user application.

6.2.2.1 Performing a Flash Chip Erase

The following lines erase the Flash area, the Flash user page, the general-purpose fuse bits and the security bit.

```
echo.
echo Performing a JTAG Chip Erase command.
avr32program chiperase
```

6.2.2.2 Programming the .hex Files in the Flash Array

The following lines program:

- the USB DFU bootloader: `at32uc3a-isp-1.0.2.hex` (located in \SERVICES\USB\CLASS\DFU\EXAMPLES\ISP\AT32UC3A\Releases\AT32UC3A-ISP-1.0.2\).
- the USB MSC bootloader: `at32uc3a0512-ms-isp-1.0.0.hex`
- The user application: `Release\Exe\user_application.hex`

```
echo.
echo Programming MCU memory from `at32uc3a-isp-1.0.2.hex', `at32uc3a0512-ms-isp-
1.0.0.hex' and `user_application.hex'.
srec_cat ^
\SERVICES\USB\CLASS\DFU\EXAMPLES\ISP\AT32UC3A\Releases\AT32UC3A-ISP-1.0.2\at32uc3a-
isp-1.0.2.hex -intel ^
  -crop 0x80000000 0x80002000 ^
  -offset -0x80000000 ^
  ..\at32uc3a0512-ms-isp-1.0.0.hex -intel ^
  -crop 0x80002000 0x80008000 ^
  -offset -0x80000000 ^
  Release\Exe\user_application.hex -intel ^
  -crop 0x80008000 0x80080000 ^
  -offset -0x80000000 ^
  -o at32uc3a-isp-1.0.2_at32uc3a0512-ms-isp-1.0.0_user_application.bin -binary
avr32program program -finternal@0x80000000,512Kb -cxtal -v -O0x80000000 -Fbin
at32uc3a-isp-1.0.2_at32uc3a0512-ms-isp-1.0.0_user_application.bin
del at32uc3a-isp-1.0.2_at32uc3a0512-ms-isp-1.0.0_user_application.bin
```

6.2.2.3 Programming the USB DFU Configuration Word in the Flash User Page

The following lines program the USB DFU configuration word in the Flash user page (at 808001FCh):

```
echo.
echo Programming DFU ISP configuration word from `at32uc3a-isp_cfg-1.0.2.bin'.
avr32program program -finternal@0x80000000,512Kb -cxtal -e -v -O0x808001FC -Fbin
\SERVICES\USB\CLASS\DFU\EXAMPLES\ISP\AT32UC3A\Releases\AT32UC3A-ISP-1.0.2\at32uc3a-
isp_cfg-1.0.2.bin
```

6.2.2.4 Programming the general-purpose fuse bits

The following lines program the general-purpose fuse bits (set to 7C07FFFEh) used by the USB DFU bootloader:

```
echo.
echo Programming general-purpose fuse bits.
avr32program writefuses -finternal@0x80000000,512Kb gp=0x7C07FFFE
```

Note: The general-purpose fuses configure the USB DFU behavior.

6.2.2.5 Setting the Security bit

The following lines set the Security bit:

```
echo.
echo Setting Security bit.
avr32program secureflash
```

6.2.2.6 Starting the User Application

The following lines start the bootloader:

```
echo.
echo Starting the Bootloader.
avr32program run -R
```

6.2.3 Generating the Firmware Upgrade File "avr32fwupgrade.uc3"

To generate the firmware upgrade file "*avr32fwupgrade.uc3*" from a user application .hex or .bin file, execute the following script with the user application hex or binary file (generated in Section 6.2.1.3, page 7) as input:
gen_uc3.cmd for Windows® machine or gen_uc3.sh for Unix® machine.

1. The scripts require the SRecord utility software to be installed (http://srecord.sourceforge.net/. For Windows machine, the .exe path must be setup in the PATH variable).
2. Refer to Section , page 9 for more information on the user application memory layout constraints.

Refer to Section 7.3 "Generate the .uc3 upgrade file" on page 10 for location of this script.

6.2.4 Setting-up the Udrive for UC3 Firmware Upgrade

- Format a Udrive in either FAT12, FAT16 or FAT32.
- Copy the generated "avr32fwupgrade.uc3" file (see Section 6.2.3, page 9).
- Safely detach the Udrive.

6.2.5 Programming an Application Firmware

6.2.5.1 Programming

The USB MSC ISP is activated when the Udrive, previously set-up as described in Section 6.2.4, page 9, is connected to the USB port.

If the boot condition is met, the USB MSC boot process starts and programs the user application into the Flash array.
Once the programming is done and verified, configuration word is changed and then the application starts at the location 80008000h.

6.2.5.2 Upgrading

To upgrade an existing application, The application must provide a way to restart the bootloader as detailed in Section 6.1.2, page 6. Then upgrade is done like in Section 6.2.5.1, page 9.

7. Software Package

This section details the content of the software package avr32758.zip delivered with this application note.

7.1 Tools

/tools folder
For Windows machine, this folder contains the `srec_cat.exe` tool that needs to be copied to a folder included in your system path.

7.2 Projects

/avr32_studio folder
This folder contains some AVR32 Studio projects
Note: These projects require AVR32 Studio V2.1 or higher to be properly compiled.

7.2.1 USB MSC bootloader

– `uc3a0_1-msc-bootloader.zip` file
  This file contains the USB MSC bootloader project for the UC3A0/1 on EVK1100.
– `uc3b0_1-msc-bootloader.zip` file
  This file contains the USB MSC bootloader project for the UC3B0/1 on EVK1101.

7.2.2 User Project

– `evk1100-start-msc-bl-example.zip` file
  This file contains the project of an EVK1100 example showing how to launch the USB MSC bootloader from the user application.
– `evk1101-start-msc-bl-example.zip` file
  This file contains the project of an EVK1101 example showing how to launch the USB MSC bootloader from the user application.
– `evk1105-start-msc-bl-example.zip` file
  This file contains the project of an EVK1105 example showing how to launch the USB MSC bootloader from the user application.

7.3 Generate the .uc3 upgrade file

- **/uc3-file-gen** folder
  This folder contains the scripts that generate the avr32fwupgrade.uc3 upgrade file.
  – `gen_uc3.cmd` file
    DOS® script file.
    Usage: *`gen_uc3 {<user_hexfile>|<user_binfile>}`*
  – `gen_uc3.sh` file
    UNIX script file.
    Usage: *`gen_uc3 {<user_hexfile>|<user_binfile>}`*

7.4 Script to Program the Bootloaders

- **/msc-bl-prog** folder
  This folder contains the scripts that program the user application code with the requested bootloaders.
  – `dfuprogram-uc3a-ms_bl-user_appli.cmd` file:
    `dfuprogram-uc3b-ms_bl-user_appli.cmd` file:
    DOS script file using the USB DFU bootloader.

This script programs the MSC bootloader and the user application. The USB DFU bootloader must be present.
Usage: *`dfuprogram-ms_bl-user_appli {<user_hexfile>}`*

– `dfuprogram-uc3a-ms_bl-user_appli.sh` file:
`dfuprogram-uc3b-ms_bl-user_appli.sh` file:
UNIX script file using the USB DFU bootloader.
This script programs the USB MSC bootloader and the user application.
Usage: *`dfuprogram-ms_bl-user_appli {<user_hexfile>}`*

– `jtagprogram-uc3a-dfu_bl-ms_bl-user_appli.cmd` file:
`jtagprogram-uc3b-dfu_bl-ms_bl-user_appli.cmd` file:
DOS script file using a JTAG programmer.
This script programs the DFU bootloader, the MSC bootloader and the user application.
Usage: *`jtagprogram-dfu_bl-ms_bl-user_appli {<user_hexfile>}`*

– `jtagprogram-uc3a-dfu_bl-ms_bl-user_appli.sh` file
`jtagprogram-uc3b-dfu_bl-ms_bl-user_appli.sh` file
UNIX script file using a JTAG programmer.
This script programs the DFU bootloader, the MSC bootloader and the user application.
Usage: *`jtagprogram-dfu_bl-ms_bl-user_appli {<user_hexfile>}`*

8. Getting Started

8.1 Example with EVK1100

This section details a quick way to test the MSC bootloader functionality. The step by step procedure is as follow:

1. Program the MSC bootloader
2. Start the MSC bootloader
3. Generate the upgrade file
4. Upgrade the application

8.1.1 Programming the MSC Bootloader

8.1.1.1 Using DFU

Program with USB the USB MSC bootloader + the EVK1100 example application with the programming script: `jtagprogram-uc3a-dfu_bl-ms_bl-user_appli`.
Type:

```
msc-bl-prog/dfuprogram-uc3a-ms_bl-user_appli evk1100-start-msc-bl-example.hex
```

Note: To launch the DFU bootloader, press the joystick center and power cycle the board.

8.1.1.2 Using JTAG

Program with JTAG the USB DFU bootloader + USB MSC bootloader + the EVK1100 example application with the programming script: `jtagprogram-uc3a-dfu_bl-ms_bl-user_appli`.
Type:

```
msc-bl-prog/jtagprogram-uc3a-dfu_bl-ms_bl-user_appli evk1100-start-msc-bl-
example.hex
```

8.1.2 Starting the USB MSC Bootloader

Now the EVK1100 is programmed with the DFU bootloader, the MSC bootloader, and a user application that allows user to launch the MSC bootloader.
Reset the board to start the application. LED0 blinks slowly.

Press EVK1100 PB0 button, it will start the USB MSC bootloader.

8.1.3 Generating the Firmware Upgrade File

Use the gen_uc3 script to generate a new version of the EVK1100 user project.
Type:

```
gen_uc3 evk1100-blink-example.hex
```

Copy the `avr32fwupgrade.uc3` in the empty root directory of a Udrive.

Note: Don't forget to install the `srec_cat` tool as detailed in Section "Tools", page 10.

Note: Any user code can be used at this step. To generate the .hex file, refer to Section 8.2.3, page 13.

8.1.4 Upgrading the Firmware Using the USB MSC Bootloader

Once the USB MSC bootloader is running, plug the Udrive on the EVK1100 USB connector. At the end of programming, the new application will start running on the EVK1100.
If the blink-example is used the LEDs will blink sequentially.

8.2 Creating a MSC Bootloader Compliant Project

8.2.1 Starting with an existing Project

Modify the conf_isp.h file by changing the `PROGRAM_START_OFFSET` definition:
From:

```
#define PROGRAM_START_OFFSET          0x00002000
```

to:

```
#define PROGRAM_START_OFFSET          0x00008000
```

This will force the user application to start above the bootloaders.

8.2.2 Starting with the Start-Example Project

Import the project example. In AVR32 Studio, select File -> New -> import -> Existing archive into workspace -> select archive file:
`/avr32_studio/evk1100-start-msc-bl-example.zip`

This will create a project ready to use as base for your application.

8.2.3 Creating the .hex file

8.2.3.1 From a Shell

To create a .hex file from a .elf file use the following command line:

```
avr32-objcopy -O ihex in_file.elf out_file.hex
```

8.2.3.2 From AVR32 Studio

In AVR32 Studio the post build option may be filled with following command line:

```
avr32-objcopy -O ihex ${ProjName}.elf ${ProjName}.hex
```

8.3 Building the UC3A USB MSC project

8.3.1 From AVR32 Studio

Import the USB MSC bootloader project for EVK1100. In AVR32 Studio, select File -> New -> import -> existing archive into workspace -> select archive file:
`/avr32_studio/uc3a0_1-msc-bootloader.zip`.
Build the project, it will generate the at32uc3a0512-ms-isp-beta.elf file.

8.3.2 From Software Framework with GCC

Go to:
/SERVICES/USB/CLASS/MASS_STORAGE/EXAMPLES/ISP/AT32UC3A0512_EVK1100/GCC

Type:

```
make rebuild
```

This will generate the uc3a0512-isp.elf file.

8.3.3 From Software Framework with IAR

Go to:
/SERVICES/USB/CLASS/MASS_STORAGE/EXAMPLES/ISP/AT32UC3A0512_EVK1100

Open the isp.eww IAR project file.

9. Frequently Asked Questions

Q: I do not want to use the USB DFU bootloader and I only want the USB MSC bootloader. How do I do that?

Remove the USB DFU bootloader with JTAGICE mkII by unprotecting and erasing the MCU Flash array with a command `avr32program chiperase`.

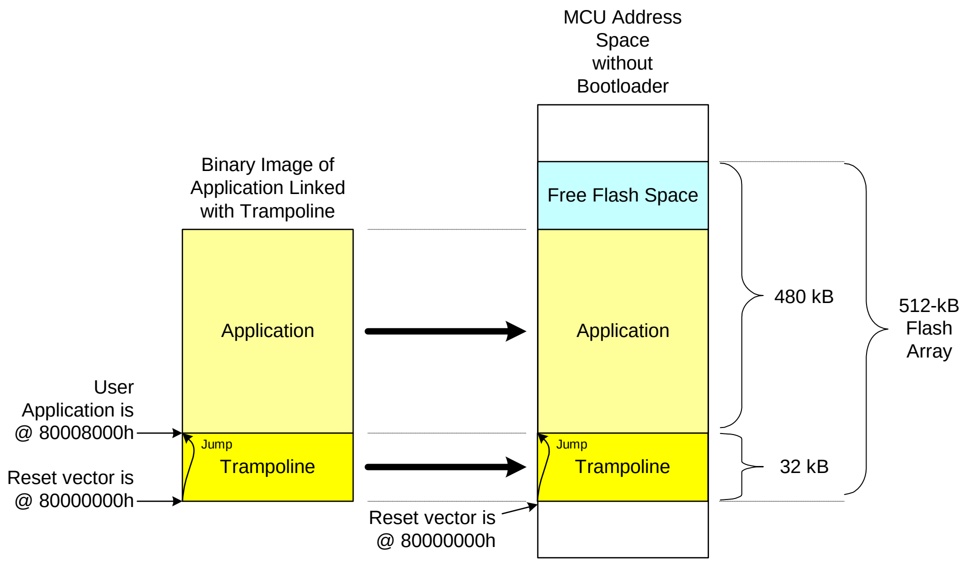
Now change the USB MSC bootloader start address (80002000h to be changed in 8000000h) by changing the following file in `SERVICES\USB\CLASS\MASS_STORAGE\EXAMPLES\ISP\BOOT\`:

- For GCC compiler: boot.S file: the line " .org 0x00002000" must be removed.
- For IAR compiler: boot.s82 file: the line " ORG 0x00002000" must be removed.

The user program Flash start offset is still set to 8000h (user application starts at 80008000h). The user program needs to be aligned on a Flash region boundary because of the Flash lock mechanism put in place by the USB MSC bootloader during the ISP process. It is also possible to use the BOOTPROT fuse to protect the USB MSC bootloader here.

When programming the user application with a JTAGICE mkII, the whole binary image including the trampoline and the application is copied to the Flash array. Consequently, when MCU execution is then started, the trampoline executes at the reset vector at 80000000h and jumps to the application at 80008000h.

Figure 9-1. Application Programming on AT32UC3A0512 with JTAGICE mkII



MCU Address Space without Bootloader
Binary Image of Application Linked with Trampoline
Free Flash Space
Application
Application
User Application is @ 80008000h
Reset vector is @ 80000000h
Jump
Trampoline
Jump
Trampoline
Reset vector is @ 80000000h
480 kB
32 kB
512-kB Flash Array

The following figures show how to program a user application with the USB MSC bootloader in two steps:

- Generate the avr32fwupgrade.uc3 file
- Program the avr32fwupgrade.uc3 file with the USB MSC bootloader

Figure 9-2. Generating the avr32firmware.uc3 file

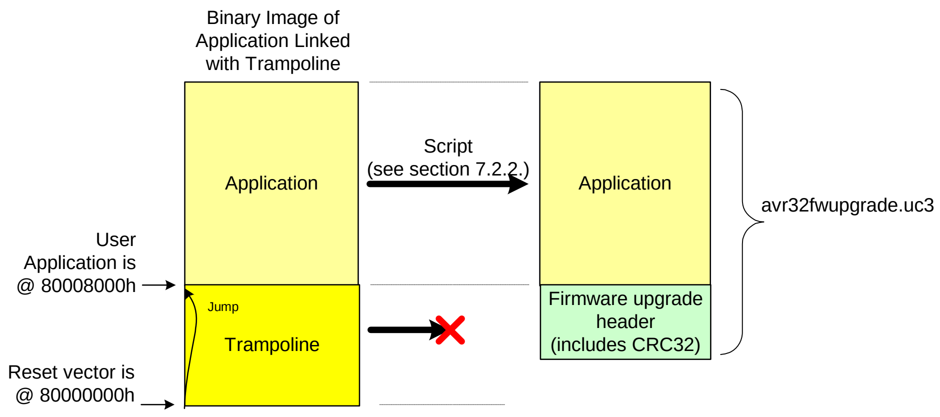
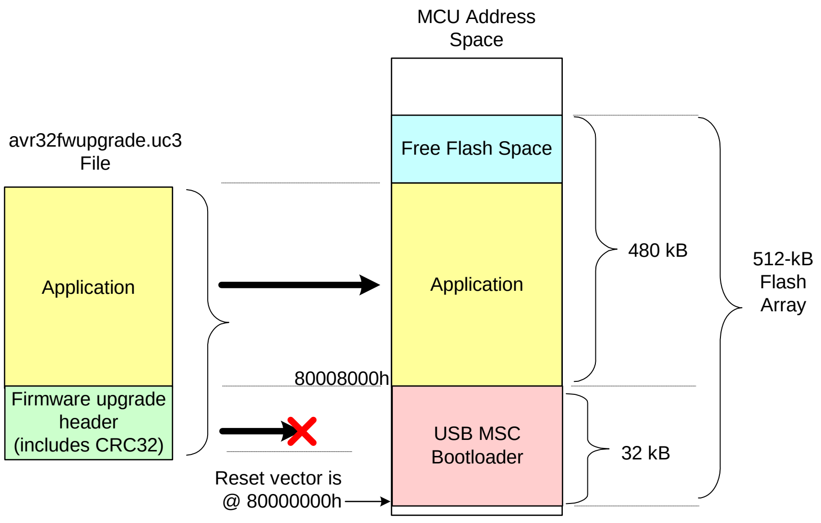

Binary Image of Application Linked with Trampoline
Application
Jump
Trampoline
User Application is @ 80008000h
Reset vector is @ 80000000h
Script (see section 7.2.2.)
Application
Firmware upgrade header (includes CRC32)
avr32fwupgrade.uc3

Figure 9-3. Application Programming on AT32UC3A0512 with the USB MSC Bootloader



Headquarters

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

International

Atmel Asia
Unit 1-5 & 16, 19/F
BEA Tower, Millennium City 5
418 Kwun Tong Road
Kwun Tong, Kowloon
Hong Kong
Tel: (852) 2245-6100
Fax: (852) 2722-1369

Atmel Europe
Le Krebs
8, Rue Jean-Pierre Timbaud
BP 309
78054 Saint-Quentin-en-Yvelines Cedex
France
Tel: (33) 1-30-60-70-00
Fax: (33) 1-30-60-71-11

Atmel 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

Product Contact

Web Site
www.atmel.com

Technical Support
avr32@atmel.com

Sales Contact
www.atmel.com/contacts

Literature Requests
www.atmel.com/literature