AN4854
Application note

In-application programming with SD cards

Introduction

An important requirement for many systems based on Flash memory is the ability to update the firmware installed in the end product. This feature is referred to as in-application programming (IAP). The purpose of this application note is to provide general guidelines for creating an IAP application.

The microcontroller can run user-specific firmware to perform IAP of the embedded Flash memory. This feature makes it possible to use several communication protocols (such as CAN, USART, USB or SD cards) for the reprogramming process.

SD card is the case described in this document, based on the X-CUBE-IAP-SD firmware.

IAP using SDMMC is very interesting because it's a "standalone IAP", that is one in which the user does not need to use a host computer, but only an SD card to upgrade the target STM32 device.

Table 1. Glossary

Term	Meaning
CAN	Controller area network
IAP	In-application programming
ICP	In-circuit programming
JTAG	Joint Test Action Group
SWD	Serial wire debugger
USART	Universal synchronous/asynchronous receiver/transmitter
SD card	Secure digital card (non-volatile)
SDMMC	Secure digital multi media card (non-volatile)

Contents

List of tables

List of figures

1 IAP overview

1.1 Principle

To program the IAP driver to the Flash memory base address, use ICP, either with the JTAG/SWD interface (using the chosen development toolchain), or with the factory-embedded bootloader in the System memory area.

The IAP driver can be used to:

- download a binary file (.bin) from an SD card to the internal Flash memory of STM32 microcontrollers
- upload all the content of the STM32 internal Flash memory into a binary file
- execute the user program.

1.2 IAP driver description

The IAP driver contains the following set of source files:

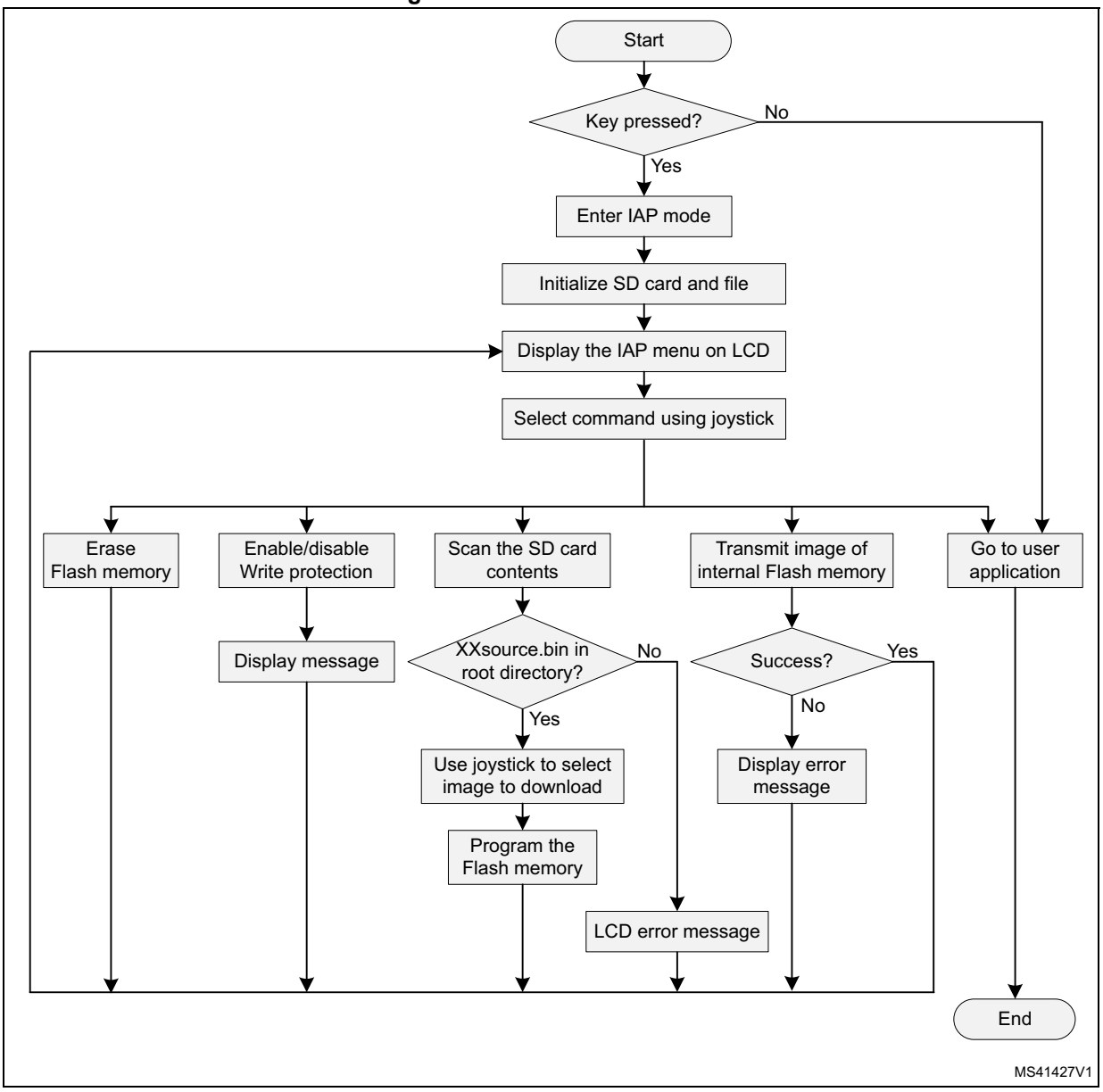
- main.c: contains the SD card and file system initialization data. The SD card is used if the user wants to enter IAP mode or if the program will execute the user code.
- stm32xxxx_it.c: this file includes the interrupt handlers for the application.
- command.c: this file includes IAP commands (download, upload, jump and lock/unlock)
- stm32xxxx_flash_if.c: this file provides a medium layer access to the STM32 embedded Flash memory driver.
- memory_card.c: this file implements SD card initialization.

The user can choose to either go to the user application or to execute the IAP for reprogramming purposes by pressing a tamper-button connected to a pin:

- tamper-button not pressed the at reset: switch to the user application
- tamper-button pressed at reset: displays the IAP main menu.

The IAP flowchart is shown in *Figure 1*.

Figure 1. IAP driver flowchart



2 IAP driver menu

After pressing the tamper-button at reset, the user can run the IAP driver to reprogram the STM32 internal Flash memory.

The driver is tested on STM32072B-EVAL and STM32L476G-EVAL boards. Although the physical communication interface is different in the two devices (SPI vs. SDMMC), thanks to STM32Cube (covering lower layers) the implementation is almost identical, making it very easy to port the technique to other STM32 products, the only difference being access to Flash memory.

To display the IAP menu user must press the tamper-button: the LCD will then display the text shown in *Figure 2*. The available options are described in *Table 2*.

Figure 2. LCD IAP menu

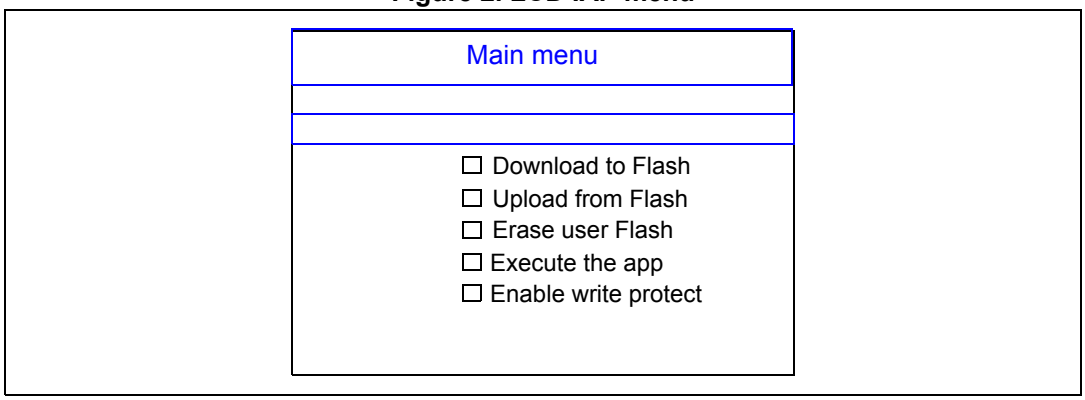


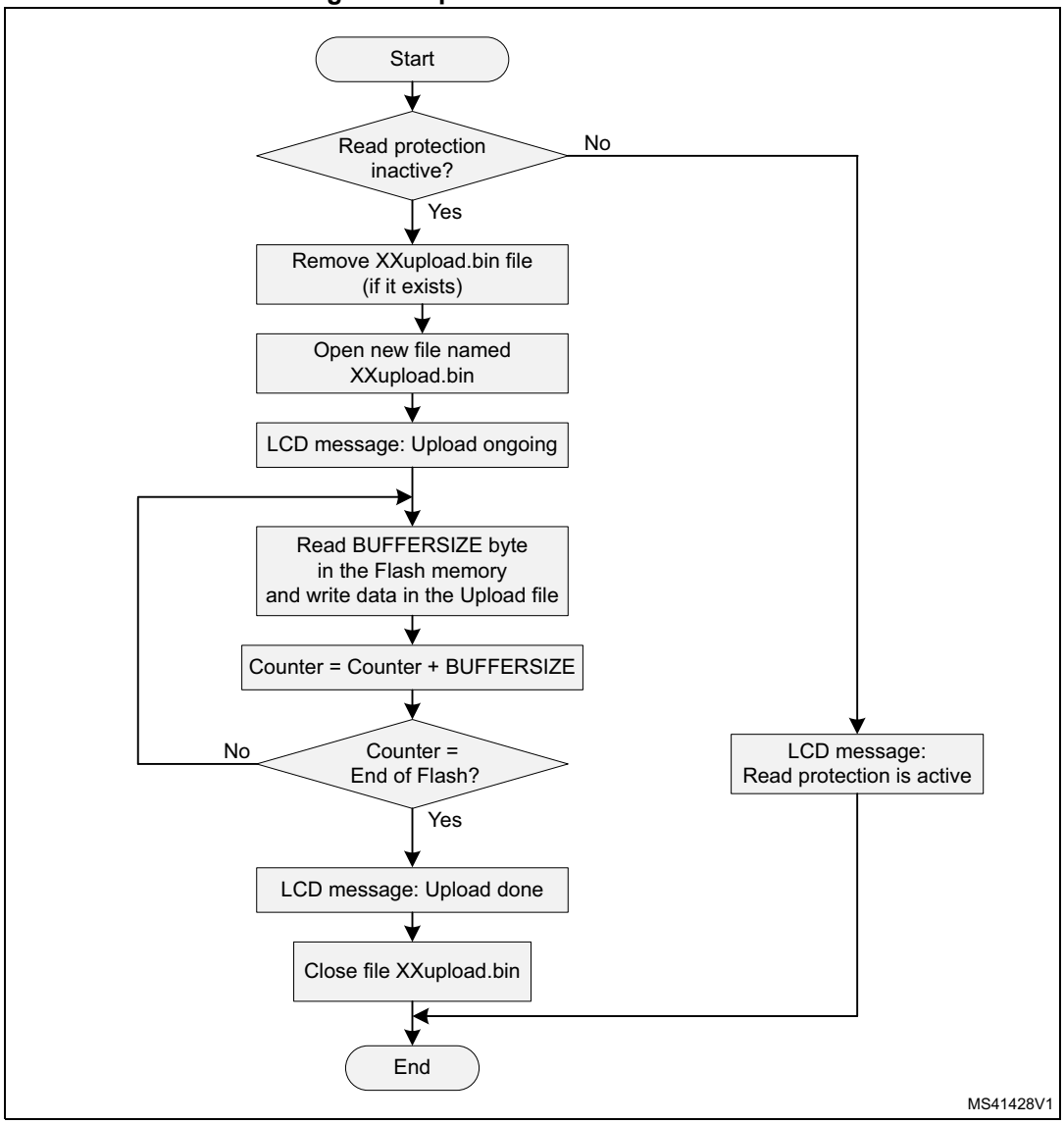
Table 2. Menu options

Option	Description
Download to Flash	Reads the selected .bin image from the thumb drive and writes it to the embedded Flash memory
Upload from Flash	Reads the whole embedded Flash memory and saves the content in an upload .bin file in the thumb drive.
Erase user Flash	Erases the Flash memory (except the IAP application), at address – 0x0800B000 for STM32F0xx – 0x0800E000 for STM32L4xx
Execute the app	Executes the uploaded application.
Enable write project	Enables/disables Flash memory write protection.

2.1 Upload command

To upload a copy of the internal Flash memory refer to the flowchart in *Figure 3*.

Figure 3. Upload command flowchart

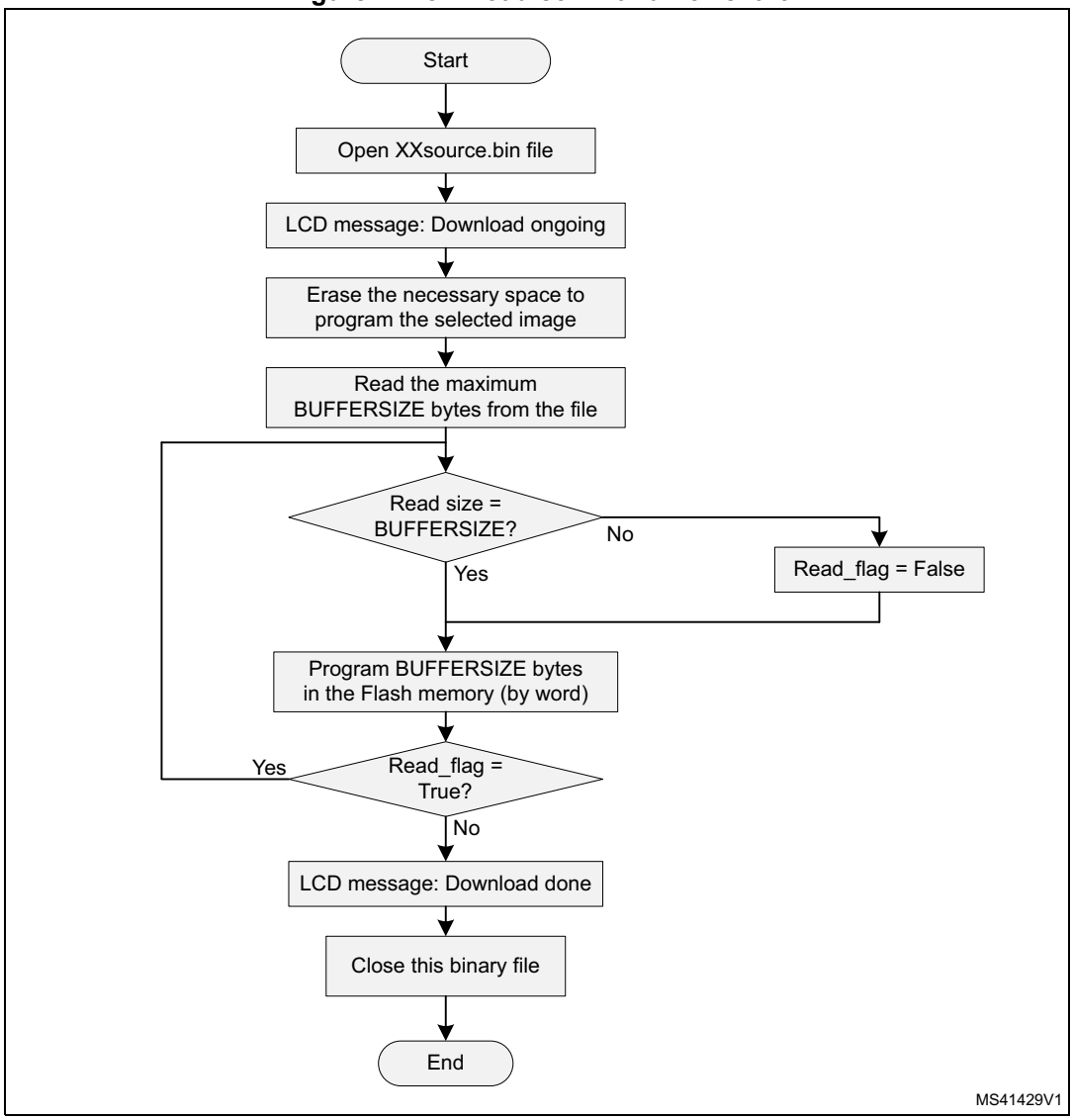


*Note: BUFFERSIZE is a user define in "memory_card.h" file that can be modified at compilation. BUFFERSIZE =4 * x, where x=[1,8192], limited by firmware.*
When the user selects the Upload command, the old "XXupload.bin" file will be removed and replaced by a new one, containing the new Flash memory data.

2.2 Download command

To download a binary file from the SD card to the STM32 embedded Flash memory refer to the flowchart shown in *Figure 4*.

Figure 4. Download command flowchart



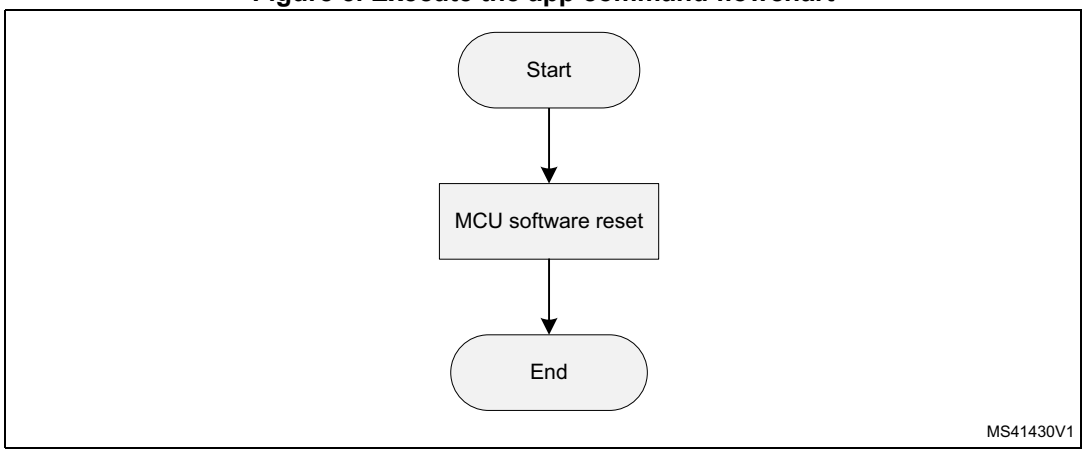
*Note: BUFFERSIZE is a define in "memory_card.h" file that can be modified by the user. BUFFERSIZE = 4 * x, where x=[1,8192], limited by the firmware.*

2.3 Execute the app command

Once the new program has been loaded, user can jump to execute this image, which must be defined from Flash address 0x0800B000 (STM32F0xx) or 0x0800E000 (STM32L4xx). Otherwise, user must adapt the firmware to jump to another address.

The flowchart of this command is shown in *Figure 5*.

Figure 5. Execute the app command flowchart



Note: After selecting Jump command, joystick-button should not be pressed.

2.4 Erase command

If no write protection is detected, the whole user Flash memory will be erased, otherwise a warning message will be displayed.

2.5 Enable/Disable write protect command

Depending the memory protection status the "Disable write protect" or the "Enable write protect" command is displayed. When selected, the disabled protection will be enabled, and vice versa. If a problem is detected, a warning message will be displayed.

The system has to be restarted to have the change applicable.

To get back to the menu the tamper-button must be pressed.

3 User program condition

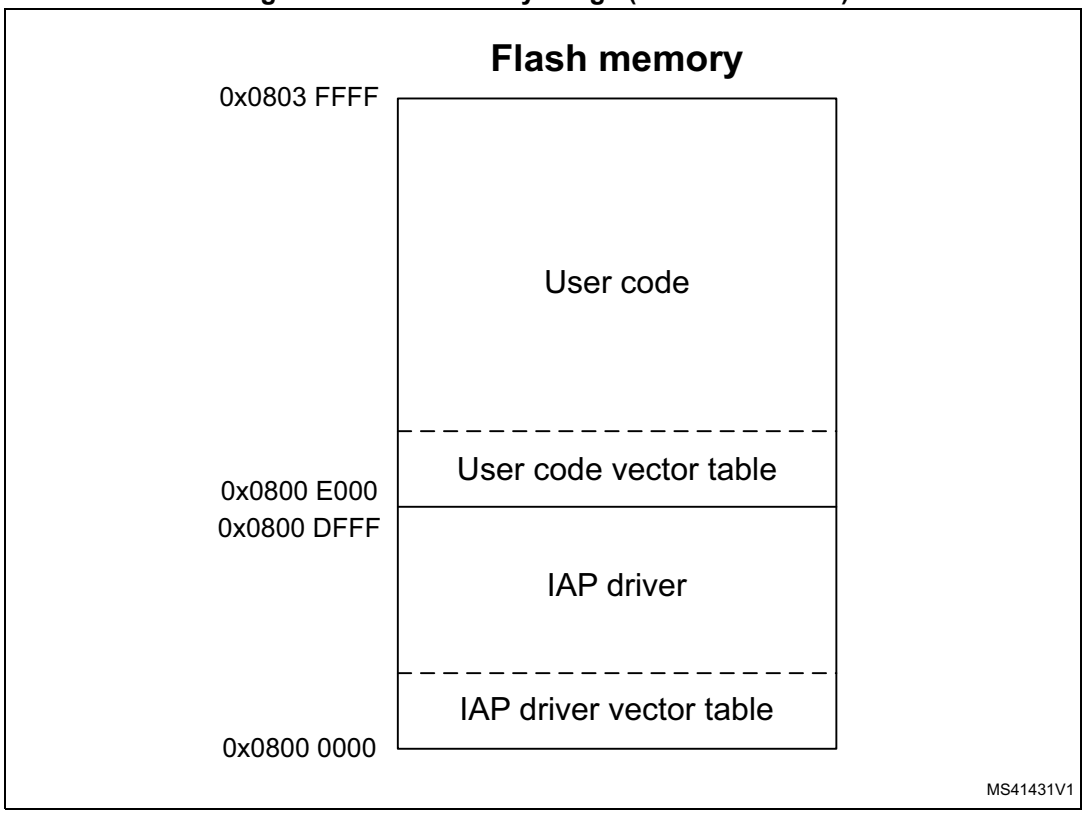
3.1 STM32L4 implementation

The user application to be loaded into the Flash memory using IAP should be built with the following configuration settings:

1. Set the program load address at 0x0800 E000, using the toolchain linker file
2. Relocate the vector table at address 0x0800 E000, changing the value of SCB VTOR.

Note: An example application program to be loaded with the IAP application is provided with preconfigured projects.

Figure 6. Flash memory usage (STM32L4 Series)



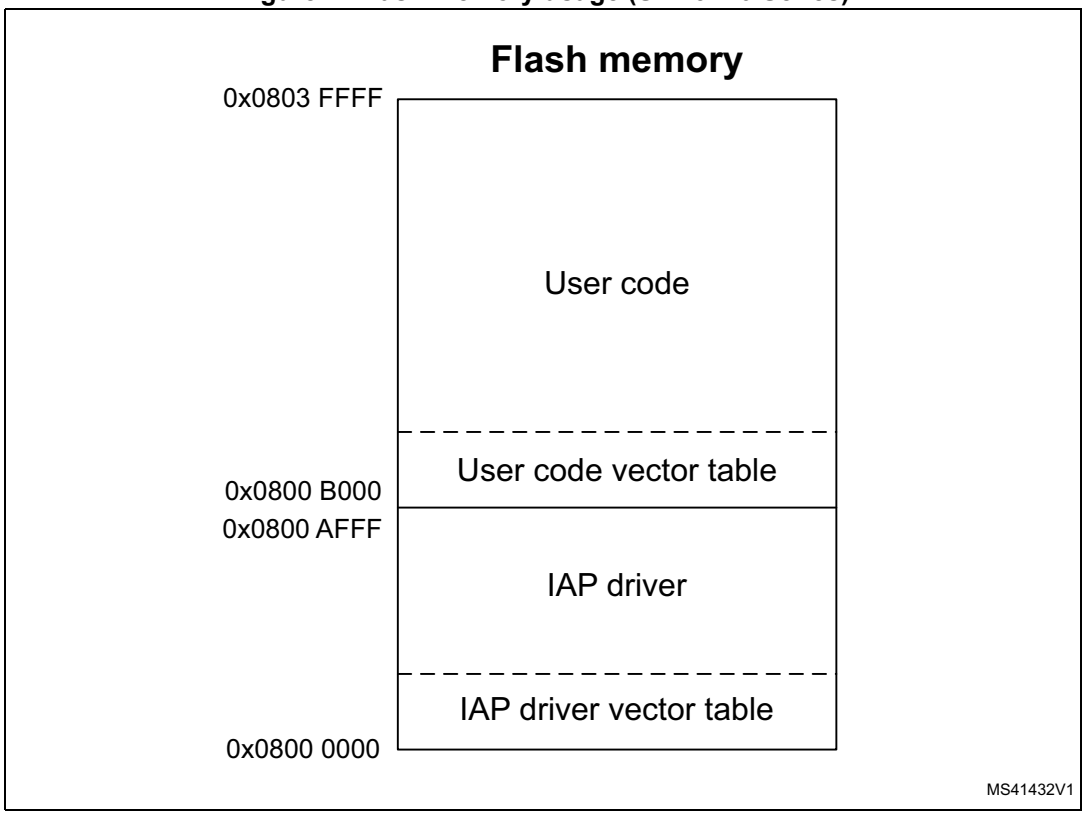
Note: User must use the High code optimization, and verify that the IAP driver size fits into the IAP driver memory.

3.2 STM32F0 implementation

The user application to be loaded into the Flash memory using IAP should be built with these configuration settings:

1. Set the program load address to 0x0800 B000 in the toolchain linker file.
2. Copy the vector table from the Flash (mapped at the base of the application load address 0x0800B000) to the base address of the SRAM at 0x20000000.
3. Remap SRAM at address 0x00000000, using __HAL_SYSCFG_REMAPMEMORY_SRAM() macro
4. Then, once an interrupt occurs, the Cortex®-M0 processor will fetch the interrupt handler start address from the relocated vector table in SRAM, then it will jump to execute the interrupt handler located in the Flash.

Figure 7. Flash memory usage (STM32F0 Series)



4 Porting to other STM32 families

The described implementation uses different physical SD card interfaces, namely SPI and SDMMC for, respectively, STM32F0 and STM32L4.

Porting the firmware to other STM evaluation boards is an easy task. This is because the Cube interface is almost identical for all families, it takes care of lower layers and hides the details of implementation from the user. Based on the microcontroller Flash memory configuration, the user has to modify the “stm32YYxx_flash_if” files, and, of course, modify the communication interface (display, user button and joystick) accordingly.

If the firmware has to be used on other boards, the user must also take care of the communication with the card, this means to reuse files stm32l476g_eval_sd.c or stm32072b_eval_sd.c that call the lower HAL layer.

5 Conclusion

The method and algorithm described in this application note offer an IAP where software is stored on a common SD card. This solution doesn’t need an host computer to upgrade the firmware, and has the advantage of being easily portable.

6 Revision history

Table 3. Document revision history

Date	Revision	Changes
04-Jul-2016	1	Initial release.