AN3354
Application note

STM32F105/107 in-application programming using a USB host

1 Introduction

An important requirement for most Flash-memory-based systems is the ability to update firmware installed in the end product. This ability 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 STM32F105/107 microcontroller can run user-specific firmware to perform IAP of the microcontroller-embedded Flash memory. This feature allows the use of any type of communication protocol for the reprogramming process (such as CAN, USART, USB). USB Host Mass storage is the example used in this application note.

This kind of IAP using USB Host is very interesting because it is a standalone IAP in which the user does not need to use a host computer to perform the firmware upgrade. The user only needs a Flash disk 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 (JTAG) debugger
SWD	Serial wire debugging.
USART	Universal synchronous/asynchronous receiver/transmitter
USB	Universal serial bus

2 IAP overview

2.1 Principle

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

The IAP driver uses the USB Host to:

- Download a binary file (.bin) from a Flash disk (thumb drive) to the STM32F105/107's internal Flash memory.
- Upload all the STM32F105/107's internal Flash memory content into a binary file.
- Execute the user program.

Note: This application note is based on the STM32F105/107xx USB Host library. For more details about the USB Host stack and a mass storage demonstration, please refer to the STM32F105/107xx USB Host library user manual (UM1021).

2.2 IAP driver description

The IAP driver contains the following set of source files:

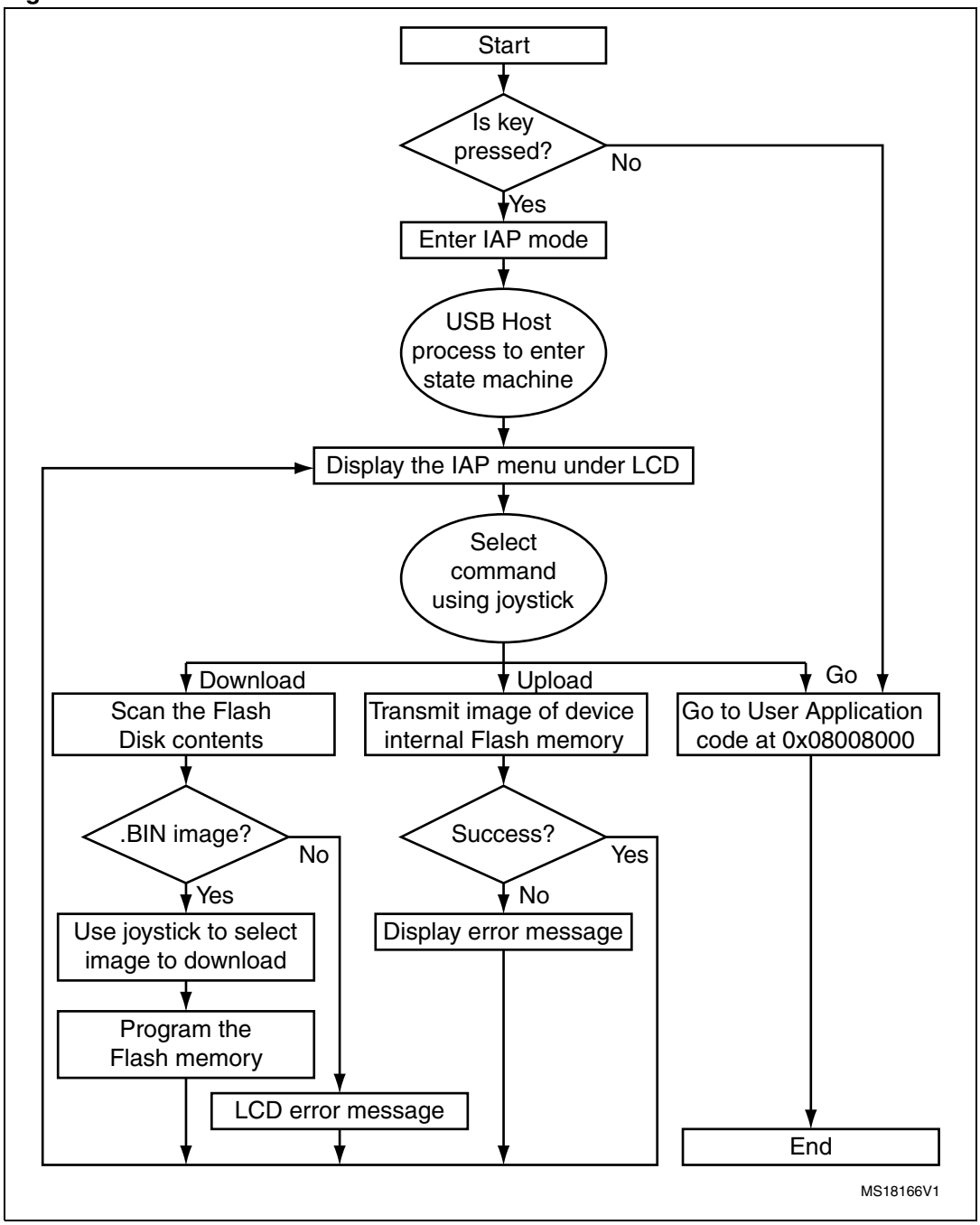
- `main.c`: contains the USB initialization data. The USB Host state machine is then executed if the user wants to enter IAP mode or the program will execute the user code.
- `stm32f10x_it.c`: contains the interrupt handlers for the application.
- `command.c`: contains the IAP commands (download, upload and jump commands).
- `flash_layer.c`: provides a medium layer access to the STM32 embedded Flash driver.
- `usb_bsp.c`: implements the board support package for the USB host library.
- `usbh_usr.c`: includes the USB host library user callbacks and LCD control data.

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

- If the Key-button is not pressed at reset, the device switches to the user application.
- If the Key-button is pressed at reset, the device displays the IAP main menu.

Figure 1 illustrates the IAP flowchart.

Figure 1. Flowchart of the IAP driver

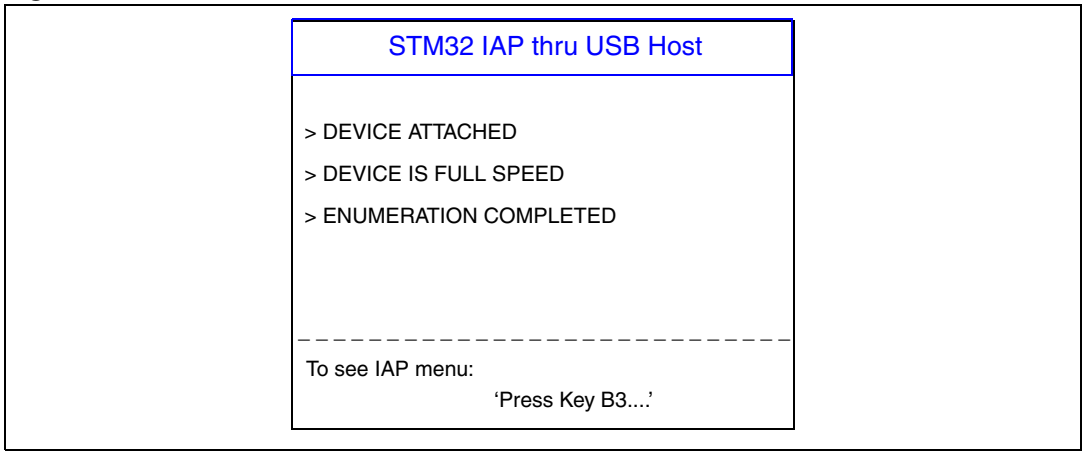


3 IAP driver menu

After pressing the Key-button at reset, the user can run the IAP driver to reprogram the STM32F device's internal Flash memory.

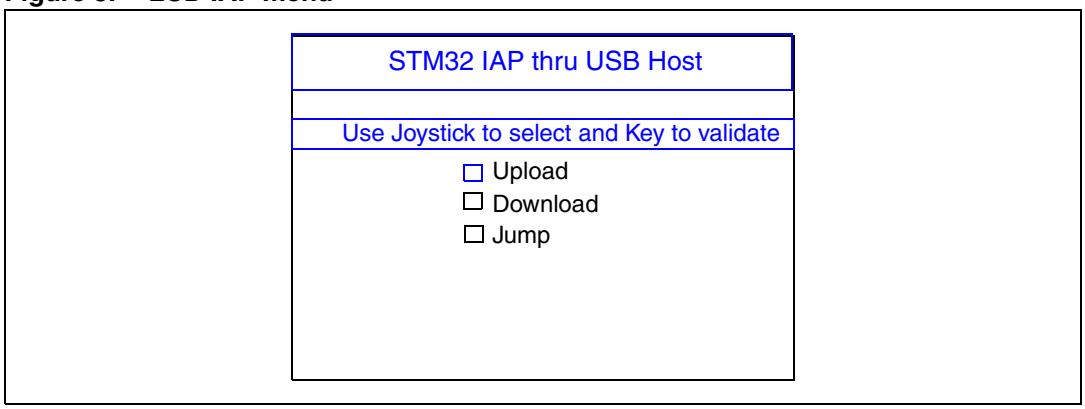
At this stage, the LCD displays the following text.

Figure 2. LCD IAP initialization



To display the IAP menu, press the Key-button. The LCD displays the following text.

Figure 3. LCD IAP menu



To select an IAP menu option, use the Joystick UP/DOWN button. To validate the selection, press the Key button.

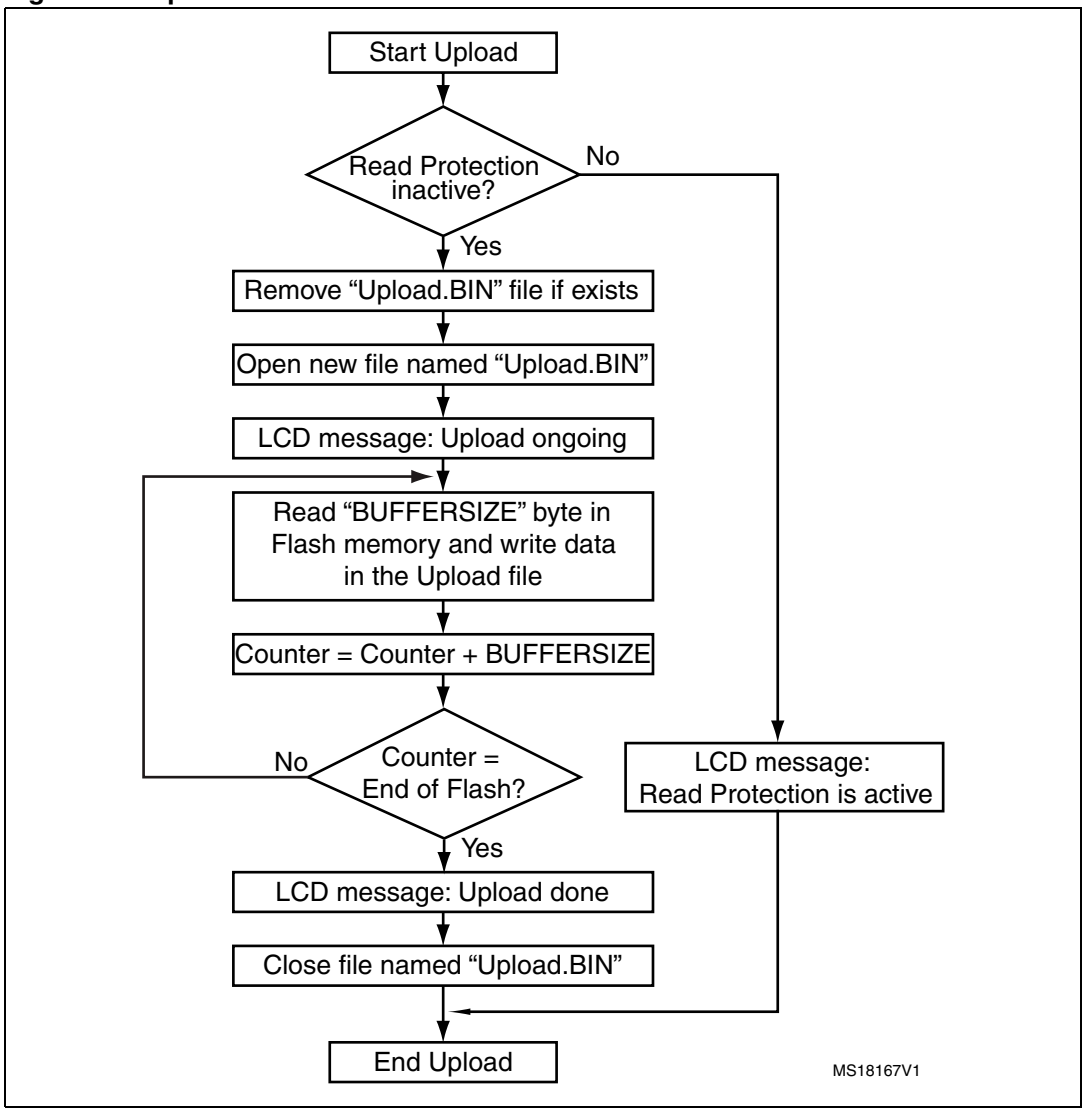
Table 2. Menu options

Option	Description
Upload	Reads the entire embedded Flash memory and saves the contents in an upload.bin file in the thumb drive.
Download	Reads the selected .bin image from the thumb drive and writes it to the embedded Flash memory.
Jump	Executes the user code at address 0x08008000.

3.1 Upload command

Figure 4 illustrates how to upload a copy of the internal Flash memory.

Figure 4. Upload command flowchart



*Note: 1 BUFFERSIZE is a user-defined variable in the “usbh_usr.h” file that can be modified at compilation. BUFFERSIZE = 4 * x; where x = [1,8192] limited by firmware.*

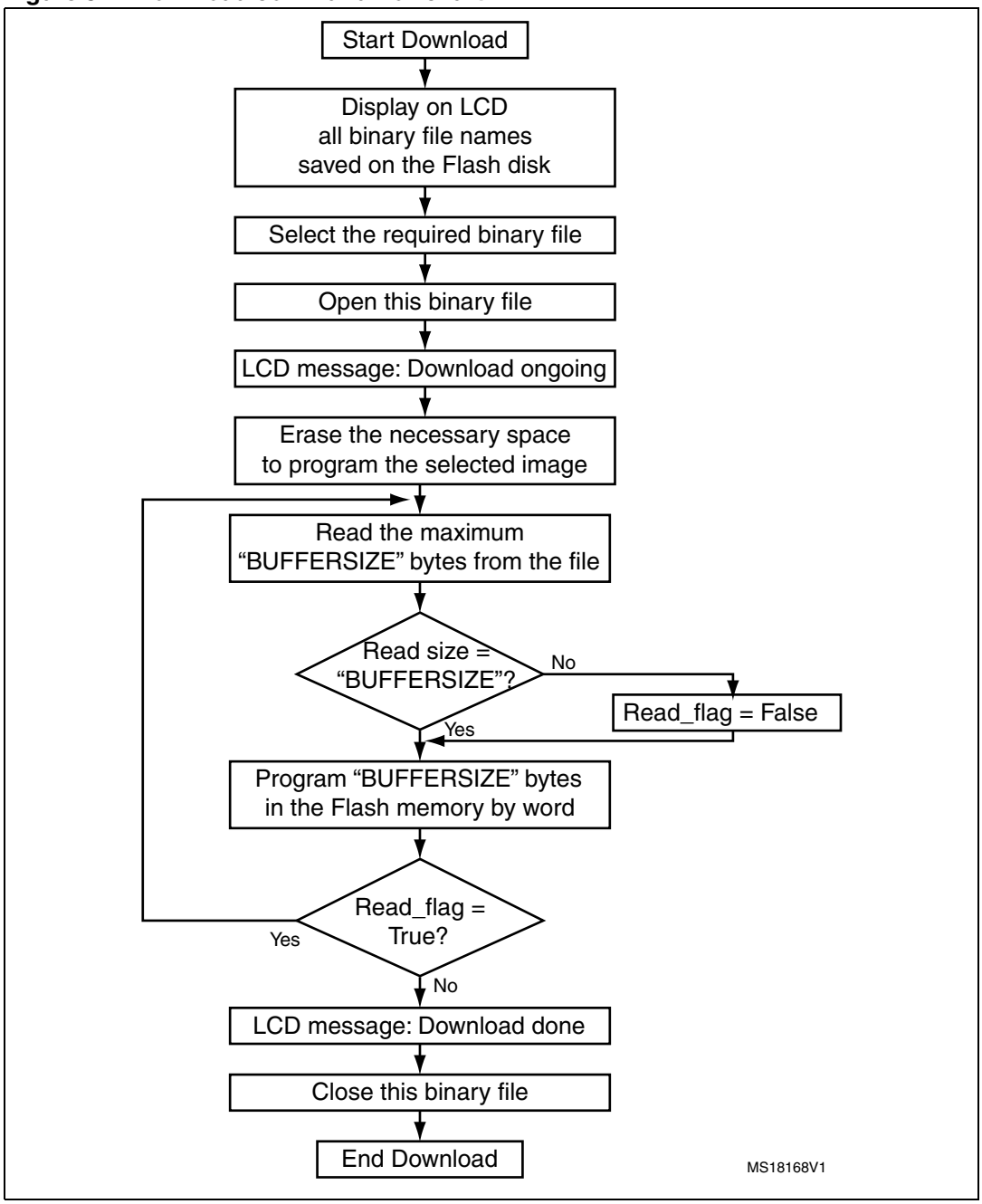
*2 With BUFFERSIZE = 4 * 8192 = 32 Kbytes, uploading of all Flash memory takes about 2 seconds.*

3 When the user selects the Upload command, the old “UPLOAD.BIN” file will be deleted and replaced by a new one that contains the new Flash memory data.

3.2 Download command

To download a binary file from the flash pen drive to the internal STM32F105/107's embedded flash memory the following flowchart is applied:

Figure 5. Download command flowchart



*Note: 1 BUFFERSIZE is a user-defined variable in the "usbh_usr.h" file that can be modified at compilation. BUFFERSIZE = 4 * x; where x = [1,8192] limited by firmware.*

*2 With BUFFERSIZE = 4 * 8192 = 32 Kbytes and image size = 14.6 Kbytes, the download operation takes about 0.5 seconds (erase time included).*

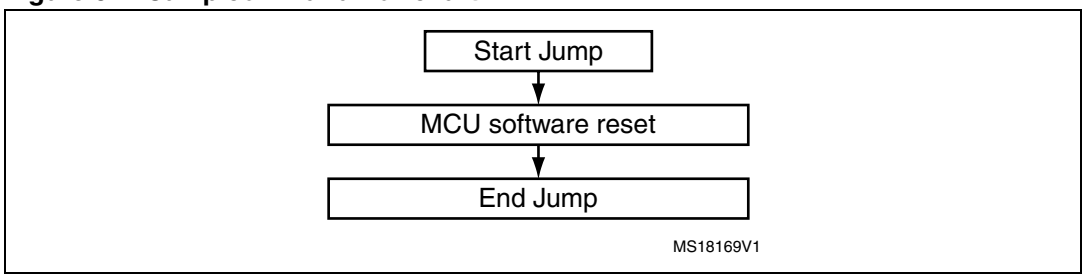
*3 With BUFFERSIZE = 4 * 8192 = 32 Kbytes and image size = 190 Kbytes, the download operation takes about 7.8 seconds (erase time included).*

3.3 Jump command

Once the new program has been loaded, user can jump to execute this image which must be defined from this flash address: 0x08008000. Other wise, user must adapt the firmware to jump to an other address.

The flowchart of this command is:

Figure 6. Jump command flowchart



Note: After selecting the Jump command, the "Key" button should not be pressed.

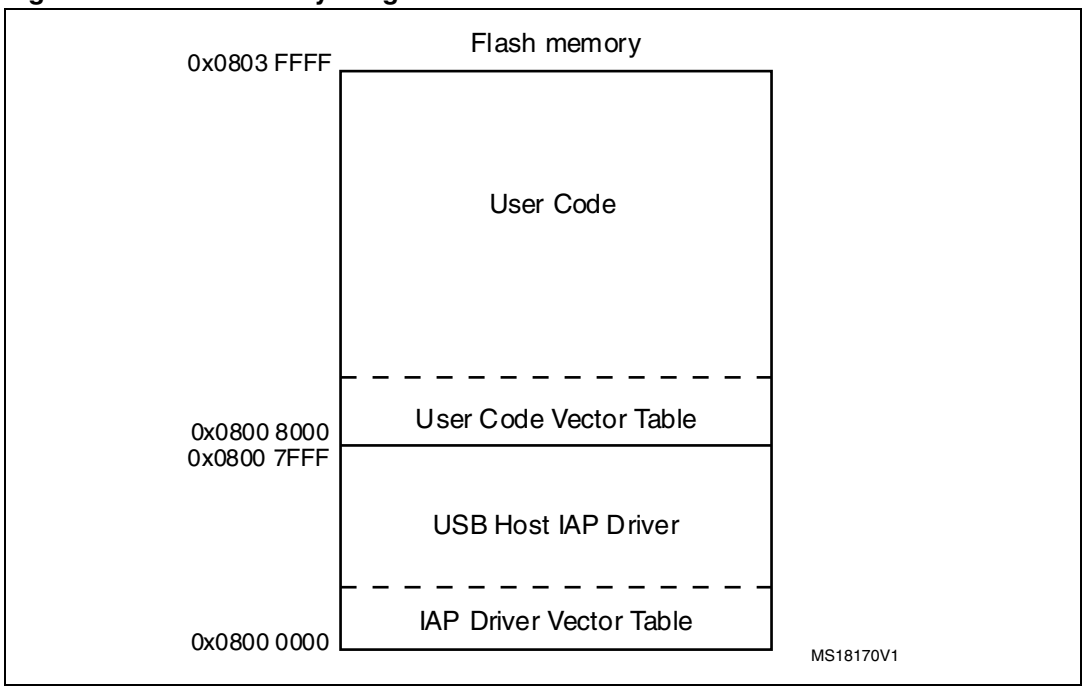
4 User program condition

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 8000 in the toolchain linker file.
2. Relocate the vector table to address 0x0800 8000 using the `NVIC_SetVectorTable` function or the `VECT_TAB_OFFSET` definition inside the `system_stm32f10x.c` file.

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

Figure 7. Flash memory usage



Note: The user must use high code optimization and verify that the IAP driver size is less than 32 Kbytes.

5 Revision history

Table 3. Document revision history

Date	Revision	Changes
31-Mar-2011	1	Initial release.