AN4657
Application note

STM32 in-application programming (IAP) using the USART

Introduction

An important requirement for most Flash-memory-based systems is the ability to update firmware when 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 on STM32 microcontrollers.

An STM32 microcontroller can run user-specific firmware to perform IAP of the Flash memory embedded in the microcontroller. This feature can use any communication interface available and supported by the product. The USART using the Ymodem protocol is the example taken in this application note.

The X-CUBE-IAP-USART firmware package is delivered with this document and contains the source code of IAP examples for STM32 microcontrollers. It is available from *www.st.com*.

Contents

List of tables

List of figures

1 IAP overview

1.1 Principle

When a reset occurs, the program counter is set to execute the IAP driver. It should be a compact code that checks a specific condition, for example that a combination of keys is pressed. When this condition is met, the IAP driver code either executes a branch that updates the user application or directly (usually by default) executes this user application.

The user application needs to be separated from the IAP driver. The most practical solution is to place the IAP driver code at the beginning of program memory and the user code at the beginning of the next free Flash memory block, sector or page, which allows independent memory protection to be configured on both areas. In the example given in this application note, the IAP is performed through the USART rather than a more advanced communication interface, thus allowing to minimize memory footprint.

The user application is likely to have independent stack and interrupt vector (both are recommended but neither is mandatory) (see *Figure 1*). When the IAP driver directly launches the user application:

1. The IAP driver sets the main stack pointer to the application address.
2. The next instruction executes the jump (unconditional branching) to the application.
3. The application then sets its own interrupt vector table as active.

Figure 1. Flash memory map
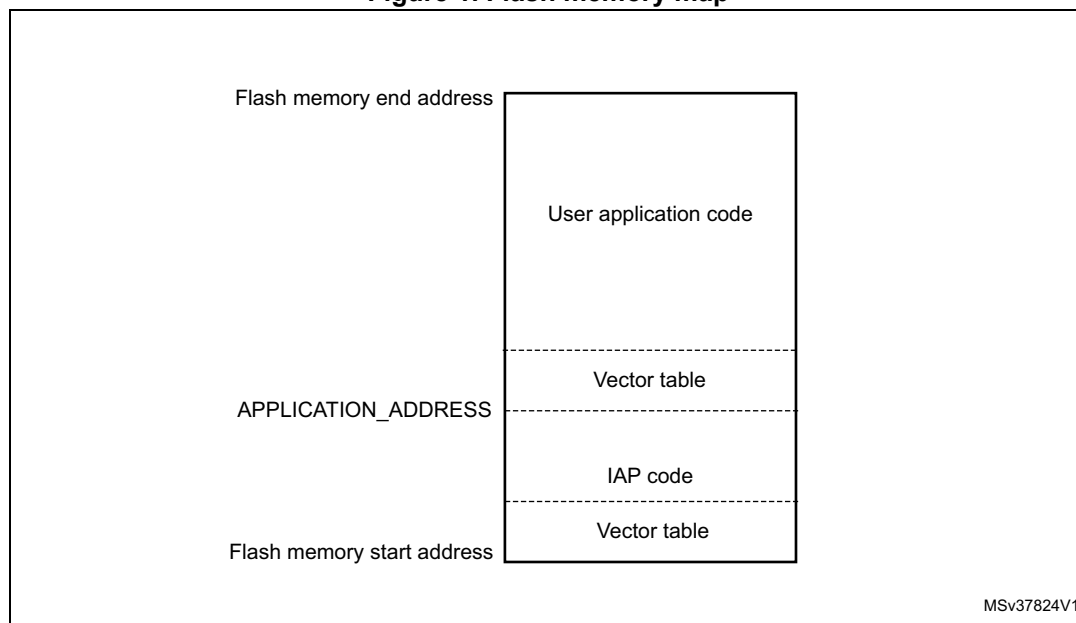
1.2 IAP driver example description

The IAP driver contains the following set of source files:

- *main.c*: where the USART initialization and RCC configuration are set. A main menu is then executed from the *menu.c* file.
- *menu.c*: it contains the main menu routine. The main menu gives the options of downloading a new binary file, uploading internal Flash memory, executing the binary file already loaded and managing the write protection of the pages where the user loads his binary file.
- *flash_if.c*: it contains write, erase and configure write protection of the internal Flash memory functions.
- *common.c*: it contains functions related to read/write from/to USART peripheral
- *ymodem.c*: it is used to receive/send the data from/to the terminal emulation application using the Ymodem protocol[a]. In the vent of a failure when receiving the data, the "Failed to receive the file" error message is displayed. If the data is successfully received, it is programmed into the internal Flash memory from the appropriate address. A comparison between internal RAM contents and internal Flash memory contents is performed to check the data integrity. If there is any data discrepancy, the "Verification failed" error message is displayed. Other error messages are also displayed when the image file size is greater than the allowed memory space and when the user aborts the task.
- STM32Cube hardware abstraction layer files.

1.3 IAP driver flowchart

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

- Not pressing the push-button at reset switches to the user application if it is present in the memory.
- Pressing the push-button at reset displays the IAP main menu (see *Figure 4*).

Refer to *readme.txt*, provided with the example code, for more details on your board implementation.

The IAP application flowchart is shown in *Figure 2*.

---

a. The Ymodem protocol sends data by 1024-byte blocks. An error check is performed in the data blocks transmitted to compare the transmitted and received data. Blocks unsuccessfully received are acknowledged with an NAK (Negative Acknowledgment). For more details about the Ymodem protocol, refer to the existing documentation.

Figure 2. IAP flowchart

MSv37825V1
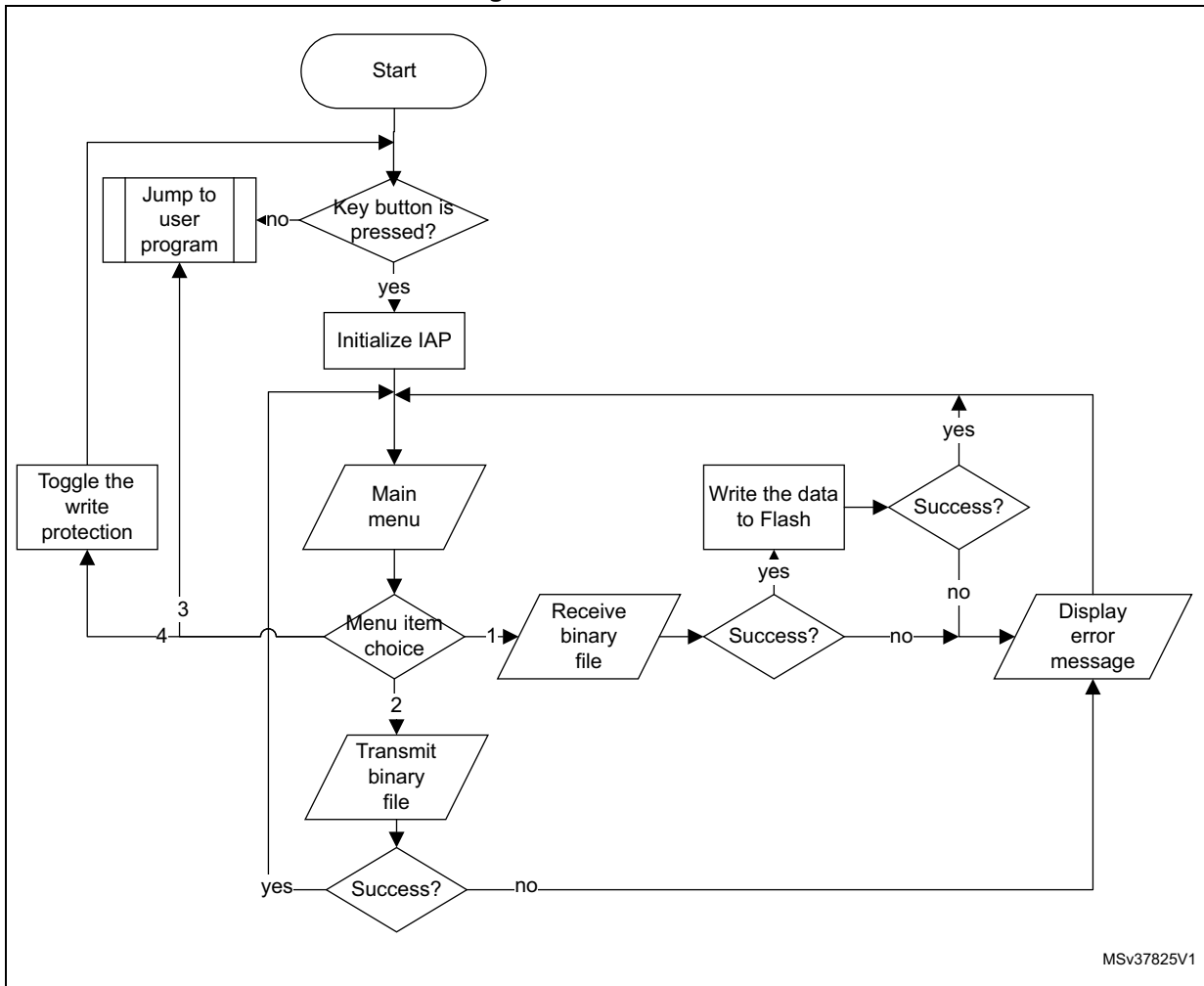
2 Running the IAP driver

The IAP driver must be programmed via the JTAG/SWD interface, starting from the Flash memory base address. This can be performed either through the development toolchain of your choice or the factory-embedded bootloader located the System memory area.

The IAP driver uses the USART to:

- download a binary file from the terminal emulator to the STM32 internal Flash memory.
- upload the STM32 internal Flash memory content (starting from the defined user application address) into a binary file.

2.1 Tera Term configuration

To use the IAP, the user must have a PC running terminal emulator that supports the Ymodem protocol.

This application note takes as an example the Tera Term popular open-source emulator. Tera Term COM port must be configured as shown in *Figure 3*. To do this, select **Setup** followed by **Serial port** from the application menu bar.

Figure 3. COM port settings
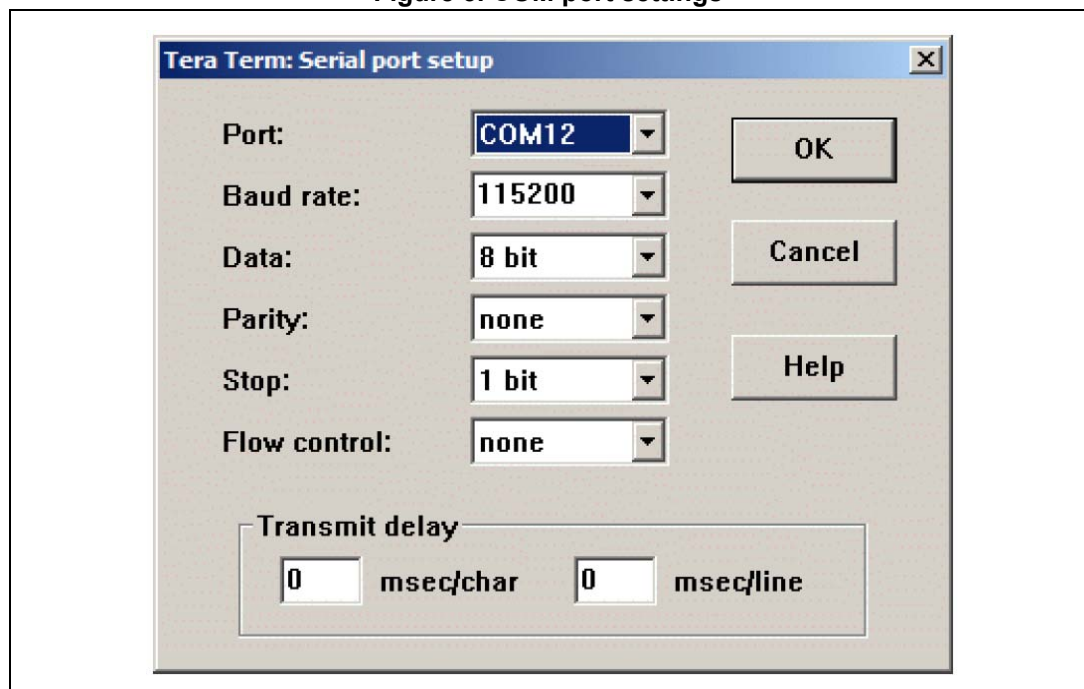
Note: *The baud rate value of 115200 bps is used in the firmware examples.*

Care must be taken when selecting the system clock frequency. To guarantee successful communication via the USART, the system clock frequency in the end application must be such that a baud rate equal to 115200 bps can be generated.

The COM port number depends on the user PC configuration. If no physical COM port is available in your development environment, use a virtual COM port such as the one provided by the ST-LINK/V2.1 interface.

## 2.2 Executing the IAP driver

In the example given in this application note, pressing the push-button at reset launches the IAP driver.

By pressing the push-button at reset, the user can run the IAP driver to reprogram the STM32 microcontroller internal Flash memory. Refer to the *readme.txt* file provided within the firmware package and to your board documentation for implementation details.

3 IAP driver menu

Running the IAP displays the following menu in the Tera Term window.

Figure 4. IAP driver menu
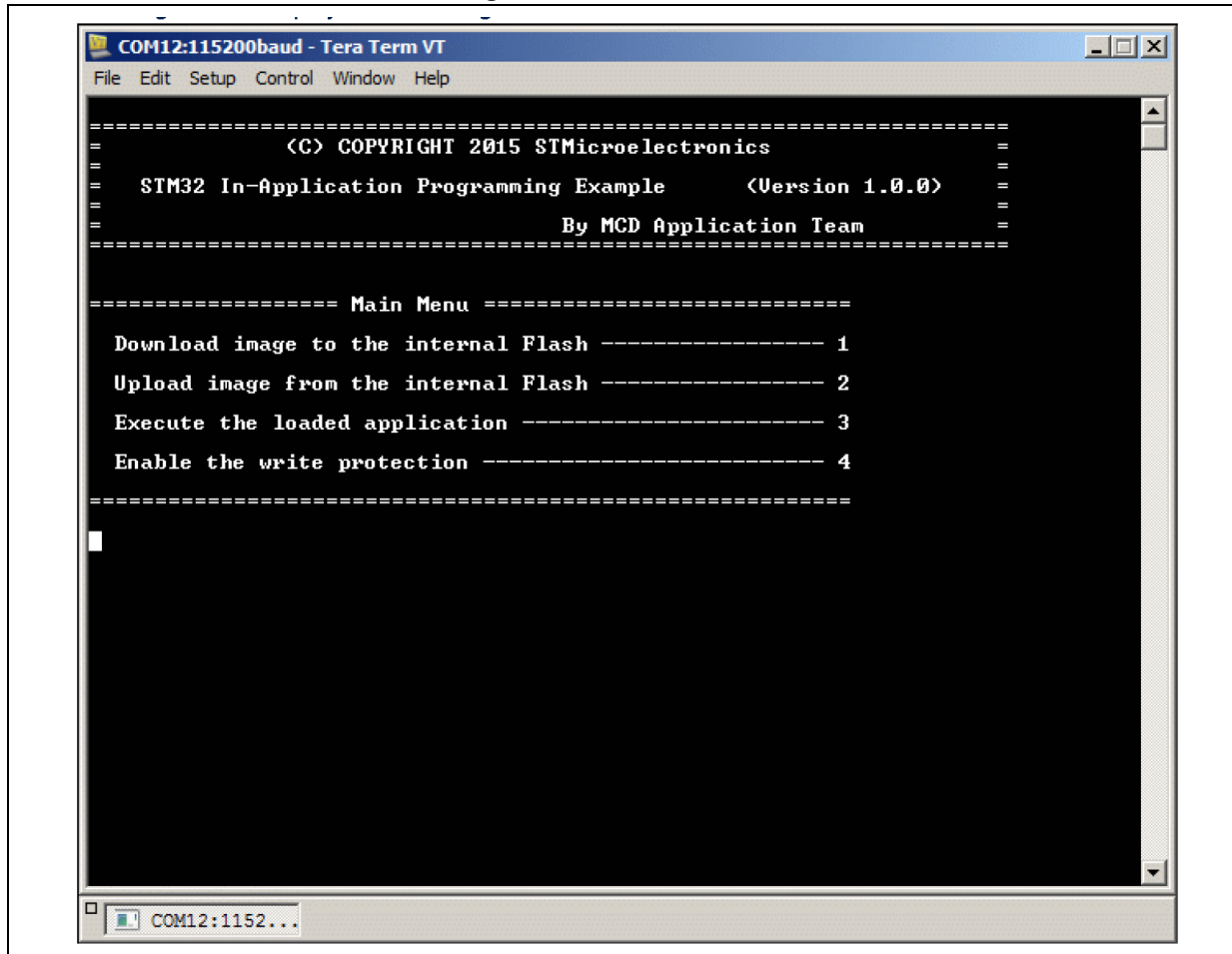
3.1 Downloading an image into the internal Flash memory

Follow the sequence below to download a binary file via Tera Term into the STM32 internal Flash memory:

1. Press **1** on the keyboard to select the **Download image to the internal Flash** menu option.
2. Select **File**/**Transfer/YMODEM/Send** from the Tera Term menu.
3. In Tera Term **YMODEM Send** dialog window, select the binary file of the application.
4. Click the **Open** button.
5. The IAP driver then loads the binary file into the internal Flash memory starting from the defined base address and displays the binary file name and size in the Tera Term window.

3.2 Uploading an image from the internal Flash memory

Follow the sequence below to upload a copy of the internal Flash memory starting from the user application address:

1. Press **2** on the keyboard to select **Upload image from the internal Flash** menu option.
2. Select **File/Transfer/YMODEM/Receive** from the Tera Term menu.
3. A file, containing a copy of the internal Flash memory content, is saved in the last used folder.
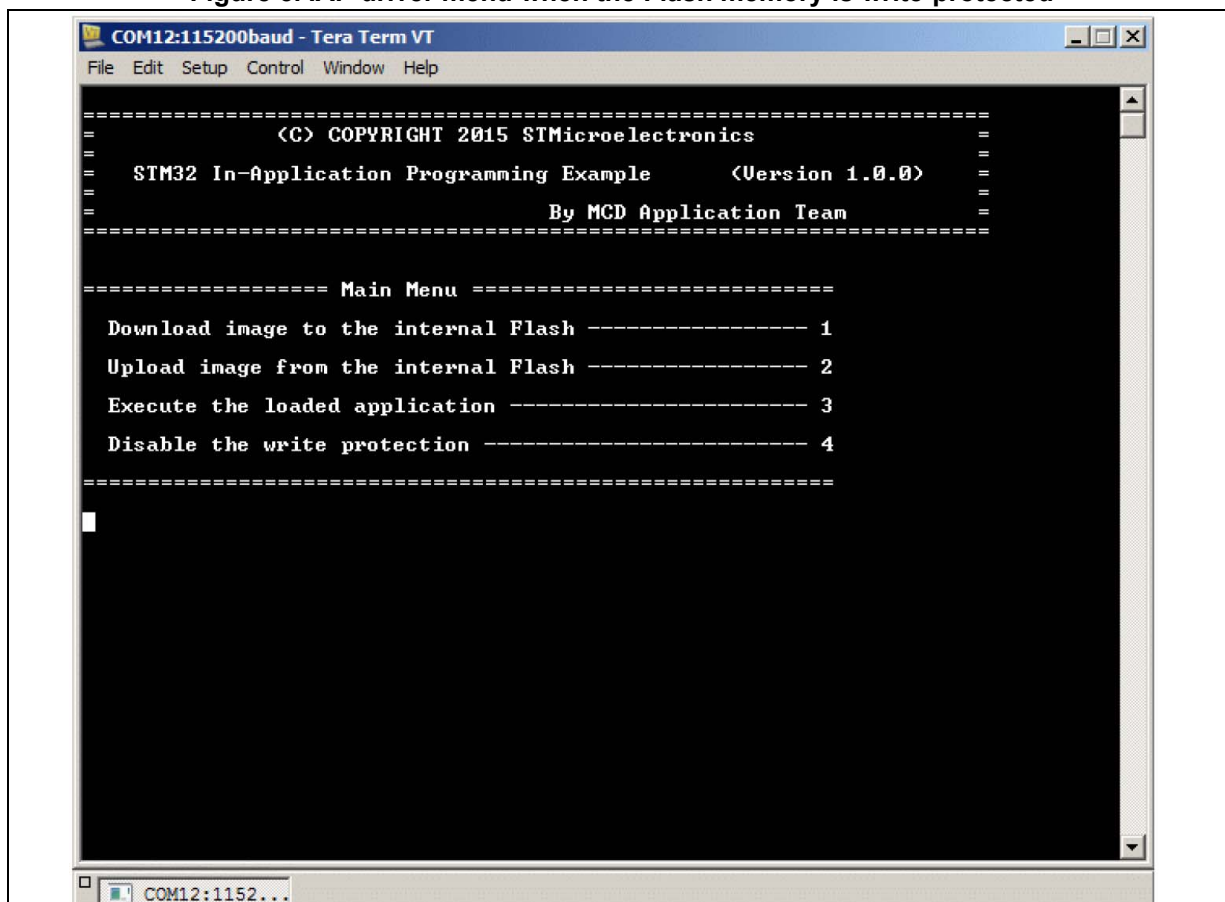
3.3 Executing the new program

Once the new program has been loaded, press **3** on the keyboard to select the **Execute the loaded application program** menu option and jump to the application code.

3.4 Configuring the write protection

When the IAP starts, it checks the Flash memory pages where the user program is to be loaded to see if any are write-protected. If it is the case, the menu shown in *Figure 5* is displayed.

Figure 5. IAP driver menu when the Flash memory is write-protected

Prior to downloading the new program, the write protection must be disabled. To do so, press **4 Disable the write protection** on the keyboard. The write protection is then disabled and a system reset is generated to reload the new option byte values. After resuming from reset, the menu shown in *Figure 4* is displayed if the key push-button is pressed.

When a new user application is installed, the write protection can be enabled again to prevent accidental corruption of the code. To toggle the write protection, select menu option **4** again.

Note: *In this example, the read protection is not supported, so the user has to verify that the Flash memory is not read-protected. Removing the read protection requires a mass erase of the non-volatile memory content.*

4 IAP implementation summary

The IAP firmware package comes with:

- source files and pre-configured projects for the IAP program (*under IAP_Main sub-directory)*
- source files and pre-configured projects that build the application to be loaded into Flash memory using the IAP (under *IAP_Binary_Template sub-directory*).

The *readme.txt* files provided within code package describes step by step how to execute this IAP application and contains microcontroller specific implementation details.

5 User program conditions

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

1. Using your toolchain linker settings, set the program load address as configured in the IAP project code.
2. Relocate the vector table to APPLICATION_ADDRESS, for example by modifying the value of the constant VECT_TAB_OFFSET defined in the project source files.

An example application program to be loaded with the IAP is provided with pre-configured projects. Refer to the project *readme.txt* file for details.

6   Revision history

Table 1. Document revision history

Date	Revision	Changes
27-Apr-2015	1	Initial release.

IMPORTANT NOTICE – PLEASE READ CAREFULLY