UM1709
User manual

STM32Cube Ethernet IAP example

Introduction

The In-Application Programming (IAP) is a way to program the flash memory while code execution from the same flash. It provides the possibility to load an application code using high speed communication protocols.

This user manual is intended for developers who use STM32Cube firmware on STM32 microcontrollers. It provides a full description of how to implement In-Application Programming (IAP) using Ethernet communication.

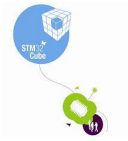
This document is applicable to all STM32 devices featuring an Ethernet peripheral; however for simplicity reason, the STM32F2x7xx and STM32F4x7/9xx microcontrollers and STM32CubeF2 and STM32CubeF4 are used as reference platform.

Two possible solutions are provided on top of the LwIP TCP/IP stack:

- IAP using TFTP (Trivial File Transfer Protocol)
- IAP using HTTP (Hypertext Transfer Protocol)

Note: In this document, STM32Cube refers to STM32CubeF2 and STM32CubeF4, STM32F4xx to STM32F4x7xx and STM32F4x9xx microcontrollers, and STM324xx-EVAL to the STM324xG-EVAL and STM324x9I-EVAL evaluation boards.

The screenshots and file names provided in this document correspond to application examples running on STM32F4 microcontrollers. However, they are also applicable to STM32F2x7xx.



Contents

List of tables

List of figures

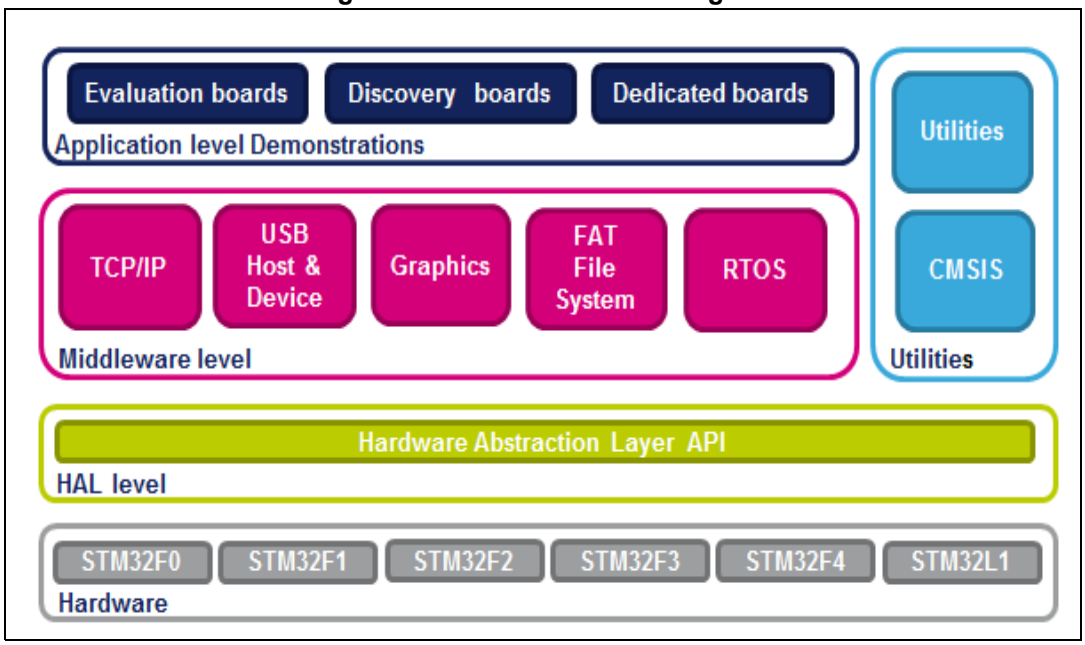
1 STM32Cube overview

The STMCube™ initiative was originated by STMicroelectronics to ease developers' life by reducing development efforts, time and cost. STM32Cube covers the STM32 portfolio.

STM32Cube Version 1.x includes:

- The STM32CubeMX, a graphical software configuration tool that allows the generation of C initialization code using graphical wizards.
- A comprehensive embedded software platform, delivered per series (such as STM32CubeF2 for STM32F2 series)
  - The STM32Cube HAL, an STM32 abstraction layer embedded software, ensuring maximized portability across STM32 portfolio
  - A consistent set of middleware components such as RTOS, USB, STMTouch, FatFS and Graphics
  - All embedded software utilities coming with a full set of examples.

Figure 1. STM32Cube block diagram



2 IAP overview

2.1 Theory of operation

In-Application Programming (IAP) is a means of upgrading firmware in the field using the MCU communication interfaces such as UART, USB, CAN and Ethernet.

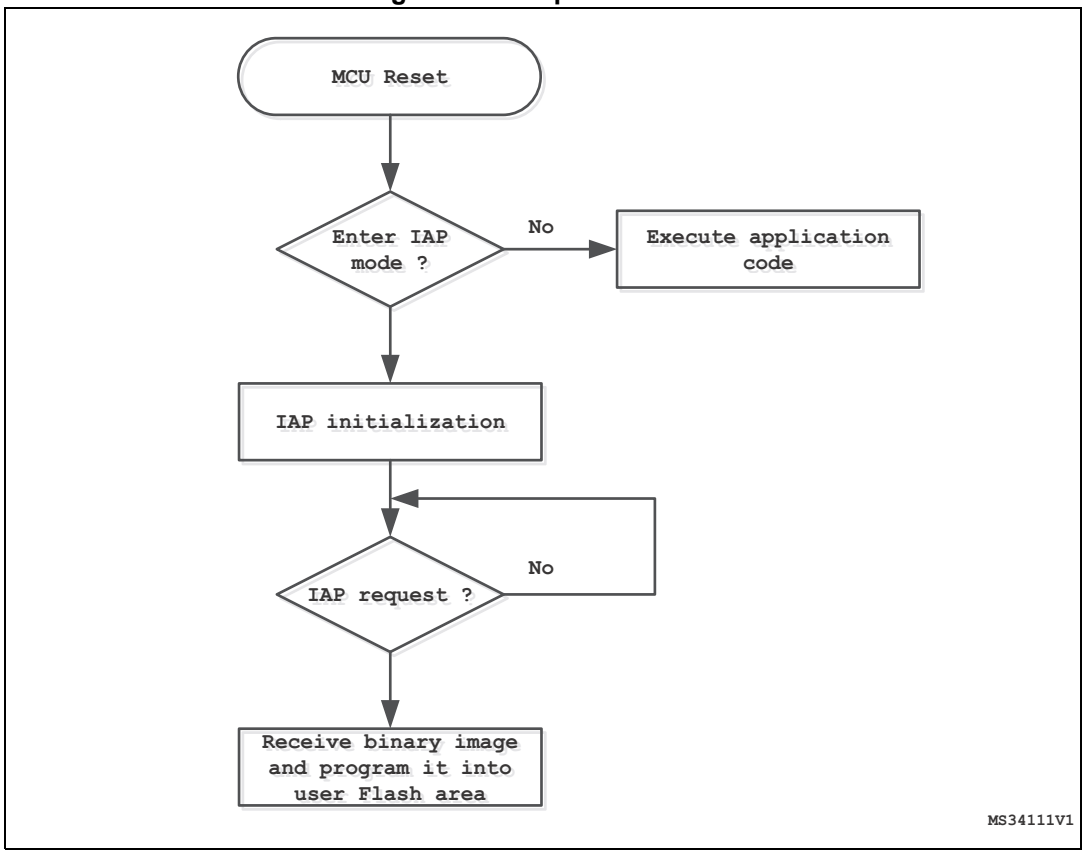
When you boot the microcontroller, you can choose to put it in either:

- IAP mode in order to execute the IAP code.
- Normal mode in order to execute the application code.

Both the IAP code and the application code are in the embedded Flash memory of the microcontroller. The IAP code is usually stored in the first pages of the MCU Flash, and the user application code occupies the remaining Flash area.

Figure 2 illustrates the IAP operation flow:

Figure 2. IAP operation flow



2.2 IAP using the MCU Ethernet interface

When it is available, Ethernet is often the preferred interface for implementing IAP capability in an embedded application. The advantages are:

- High speed communication interface (10/100 Mbit/s)
- Remote programming through the network (LAN or WAN)
- Standardized application protocols such as FTP, TFTP, HTTP on top of the TCP/IP stack that can be used for implementing the IAP

2.3 Implementing IAP over the Ethernet

This user manual describes two solutions that implement IAP for the STM32F4xx using the Ethernet communication peripheral:

- IAP using TFTP (Trivial File Transfer Protocol)
- IAP using HTTP (Hypertext Transfer Protocol)

Both solutions run on top of the LwIP stack, which is a light-weight implementation of the TCP/IP protocol suite.

2.3.1 IAP method using TFTP

The IAP method using TFTP is widely used in embedded applications that require a firmware upgrade capability (for example, in embedded Linux bootloaders).

TFTP is a simple file transfer protocol that works on top of the UDP transport layer. It is intended to be used in a LAN environment. It is based on a client/server architecture, where a client requests a file transfer (read or write operation) from a file server.

In this case the server only processes write requests from a PC TFTP client, so a simple TFTP server is implemented on top of the LwIP stack.

2.3.2 IAP method using HTTP

A firmware upgrade using the HTTP protocol is less common than with TFTP, but it can be a useful solution when remote programming over the Internet is needed. In this case, the TCP transport protocol is needed to ensure optimum operation.

HTTP works on top of TCP, and offers a way of sending a binary file from a Web client (Mozilla Firefox or Microsoft Internet Explorer) using HTML Forms. This is called HTTP File-upload (RFC 1867).

The following sections of this document provide details about the implementation of both IAP methods, and an explanation of how to use the software.

3 IAP using TFTP

3.1 TFTP overview

TFTP is a simple file transfer protocol that works on top of UDP. A file transfer is initiated from a TFTP client, that sends a Read or Write request to a TFTP server. When the server acknowledges the request, the file data transfer starts. The data is sent in fixed size blocks (for example in blocks of 512 bytes).

Each transferred data block must be acknowledged by the recipient before the next block can be sent. The acknowledge mechanism is based on the block number sent with each data block. A data block with less than the fixed block size indicates the termination of the file transfer.

Figure 3 describes the format of the various TFTP packets:

Figure 3. TFTP packets

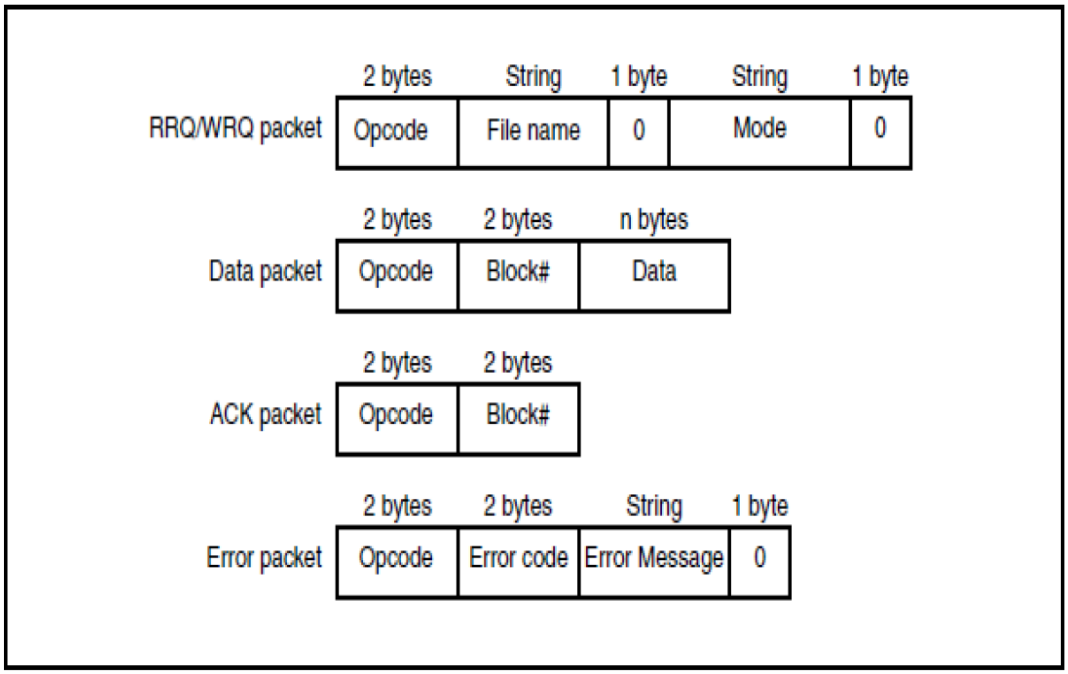


Table 1 lists the TFTP opcodes.

Table 1. TFTP opcode

Opcodes	Operation
0x1	Read request (RRQ)
0x2	Write request (WRQ)
0x3	Data
0x4	Acknowledgment (ACK)
0x5	Error

3.2 Implementing IAP using TFTP for STM32F4xx

This IAP implementation consists of a TFTP server on top of the LwIP TCP/IP stack.

This server responds to file write requests received from a remote TFTP client (PC).

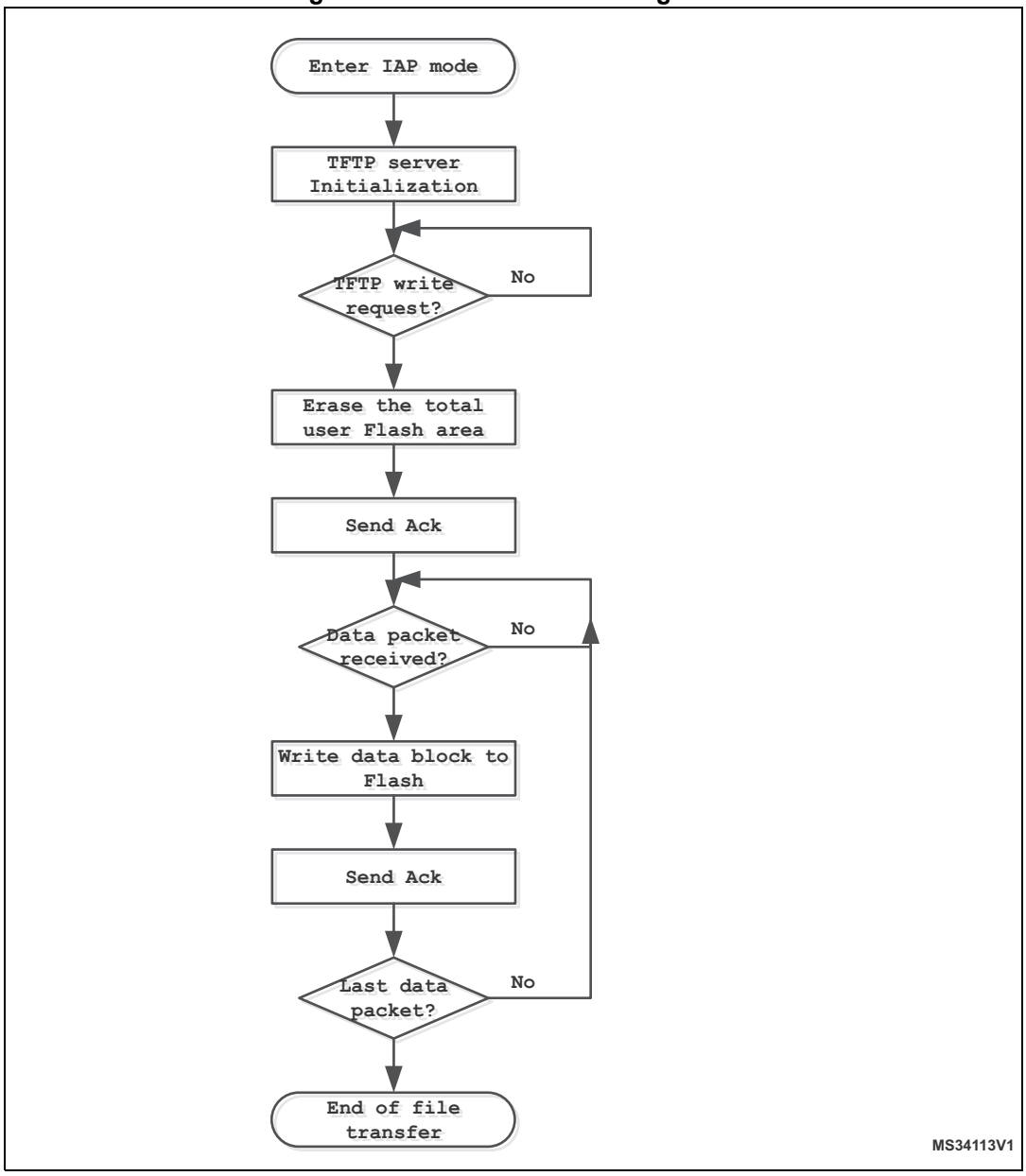
TFTP read requests are ignored.

Instead of writing received files to a file system, which is normally what TFTP is used for, the server writes the received data blocks into the MCU Flash (in the user Flash area).

Note: In this implementation, the data block size is fixed to 512 bytes.

Figure 4 provides an overview of the IAP operation using TFTP.

Figure 4. Flowchart of IAP using TFTP

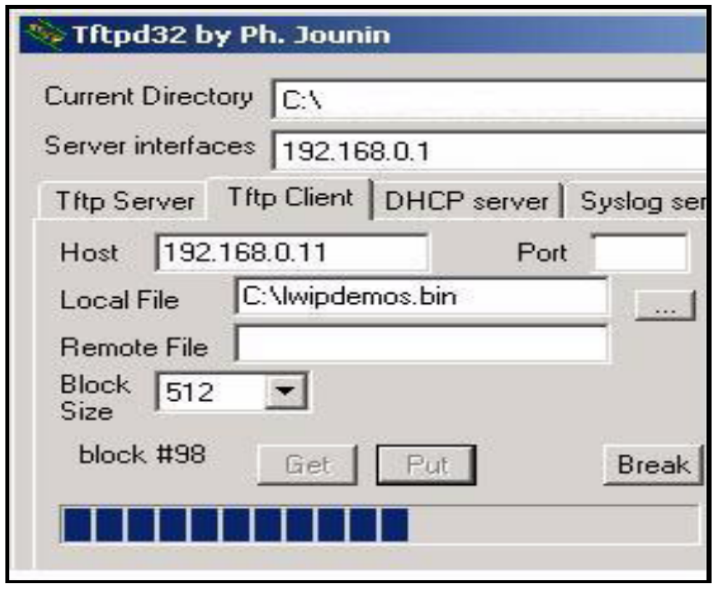


3.3 Using the firmware

In order to test the IAP through TFTP, follow these steps:

1. Make sure the jumper settings in the evaluation board are set correctly (see *Section 5.2*)
2. In the main.h file, uncomment the option `#define USE_IAP_TFTP`. Also, depending on your needs, you can uncomment/comment other options such as `#define USE_DHCP` or `#define USE_LCD`.
3. Recompile the firmware. Using the generated map file, be sure that there is no overlap between the IAP code area (starting from address 0x0) and the user Flash area starting from address: `USER_FLASH_FIRST_PAGE_ADDRESS` defined in main.h.
4. Program the firmware in the STM32F4xx Flash and run it.
5. To enter IAP mode, press and then release the Reset button while keeping the Key button pressed.
6. If `USE_LCD` is defined in main.h file then the LCD screen displays a message indicating that IAP mode has been entered. Also if DHCP is used (`USE_DHCP` defined in main.h), a message is displayed on the LCD screen indicating the success or failure of DHCP IP address allocation.
7. After IP address assignment (either static or dynamic address), the user can start the IAP process.
8. On the PC side, open the TFTP client (for example TFTPD32), and configure the TFTP server address (host address in TFTPD32).
9. Browse for a binary image to load in the STM32F4xx Flash (a binary image is provided as examples in the /project/binary folder).
10. Start a file write request by clicking the **Put** button in the TFTPD32 utility.
11. When LCD is enabled, the progress of the IAP operation is shown on the LCD.
12. At the end of IAP operation, you can reset the evaluation board and run the application that you have just programmed in the STM32F4xx Flash.

Figure 5. TFTPD32 dialog box



4 IAP using HTTP

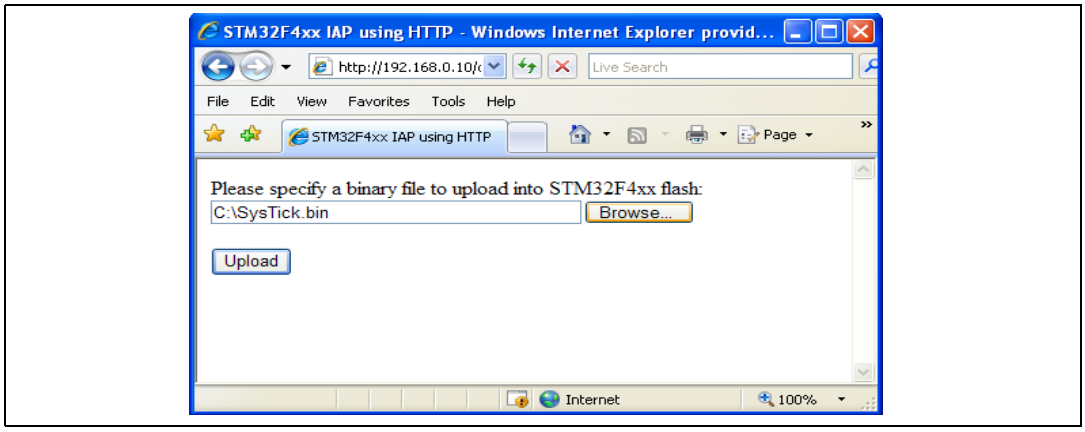
4.1 HTTP file upload overview

File upload using HTTP is defined in RFC1867. This method of uploading files is based on HTML forms. To send raw binary data, the HTML `POST` method is used instead of `GET`.

The following is an example of HTML code for implementing form-based file upload:

```
<form action ="/upload.cgi" enctype="multipart/form-data" method="post">
 <p>Specify a binary file to upload into STM32F4xx Flash:
  <br>
  <input type="file" name="datafile" size="40">
 </p>
 <div>
  <input type="submit" value="Upload">
    </div></form>
```

Figure 6. Browser view of the file upload HTML form



Press **Browse** button to select a binary file to upload, and then the **Upload** button to send it.

Depending on the file size, the data is sent in consecutive TCP segments to the web server.

Note: Before sending the file data, the web client sends HTTP header data that contains information such as the file name and the content length, some of which must be parsed by the web server.

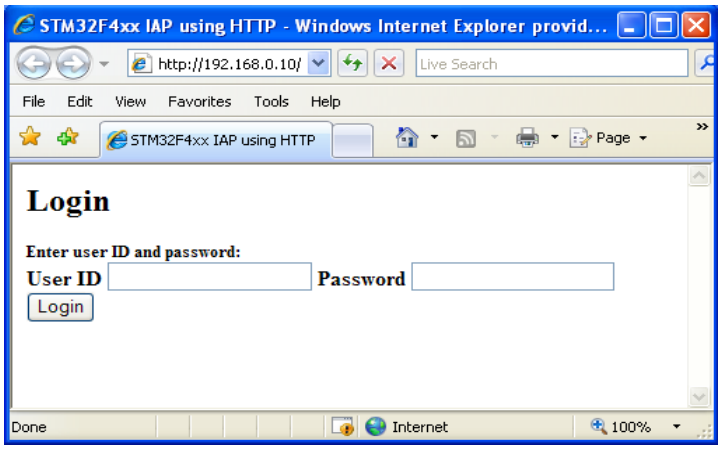
Web clients do not always have the same HTTP header format. The http web server must handle these differences.

4.2 Implementing IAP using HTTP for STM32F4xx

This IAP implementation consists of an HTTP Web server on top of the LwIP stack.

When typing the STM32 IP address in a browser, a login web page is shown (*Figure 7*). This login web page restricts access to the IAP file upload to authorized users.

Figure 7. Login web page



Enter a correct **User ID** and **Password** (pre-defined in main.h file) and click the **Login** button. A file upload web page is then loaded (see *Figure 6*).

Note:

1. The default **User ID** is: **user** and **Password** is **stm32**.
2. If the **User ID** or **Password** is incorrect, the login web page is reloaded. After a successful login, browse to select the binary file to be loaded into the STM32 Flash.
3. Make sure the binary file size does not exceed the total size of the STM32 user Flash area.
4. When clicking the **Upload** button (see *Figure 6*), a `POST` request is sent to the server. At this moment the server starts erasing all the user Flash area and waits for the binary file raw data. The received data is then written into the user Flash area.
5. Note that the total length of the data to be received is extracted from the HTTP header data sent at the beginning of the transfer.
6. At the end of IAP operation, a web page indicates the success of IAP operation, displaying a button which allows you to reset the MCU.

Figure 8. File upload done

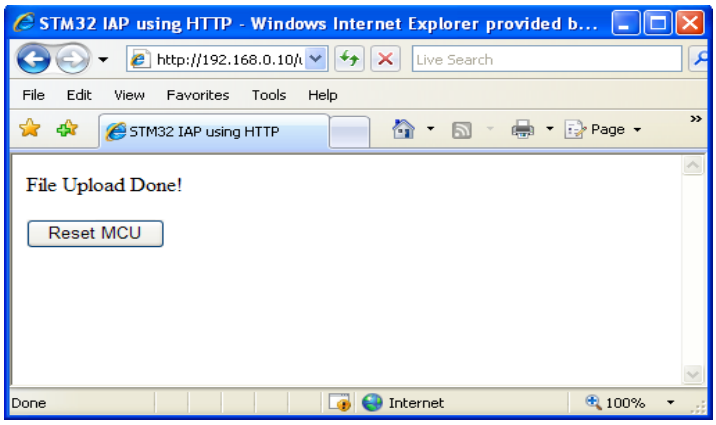
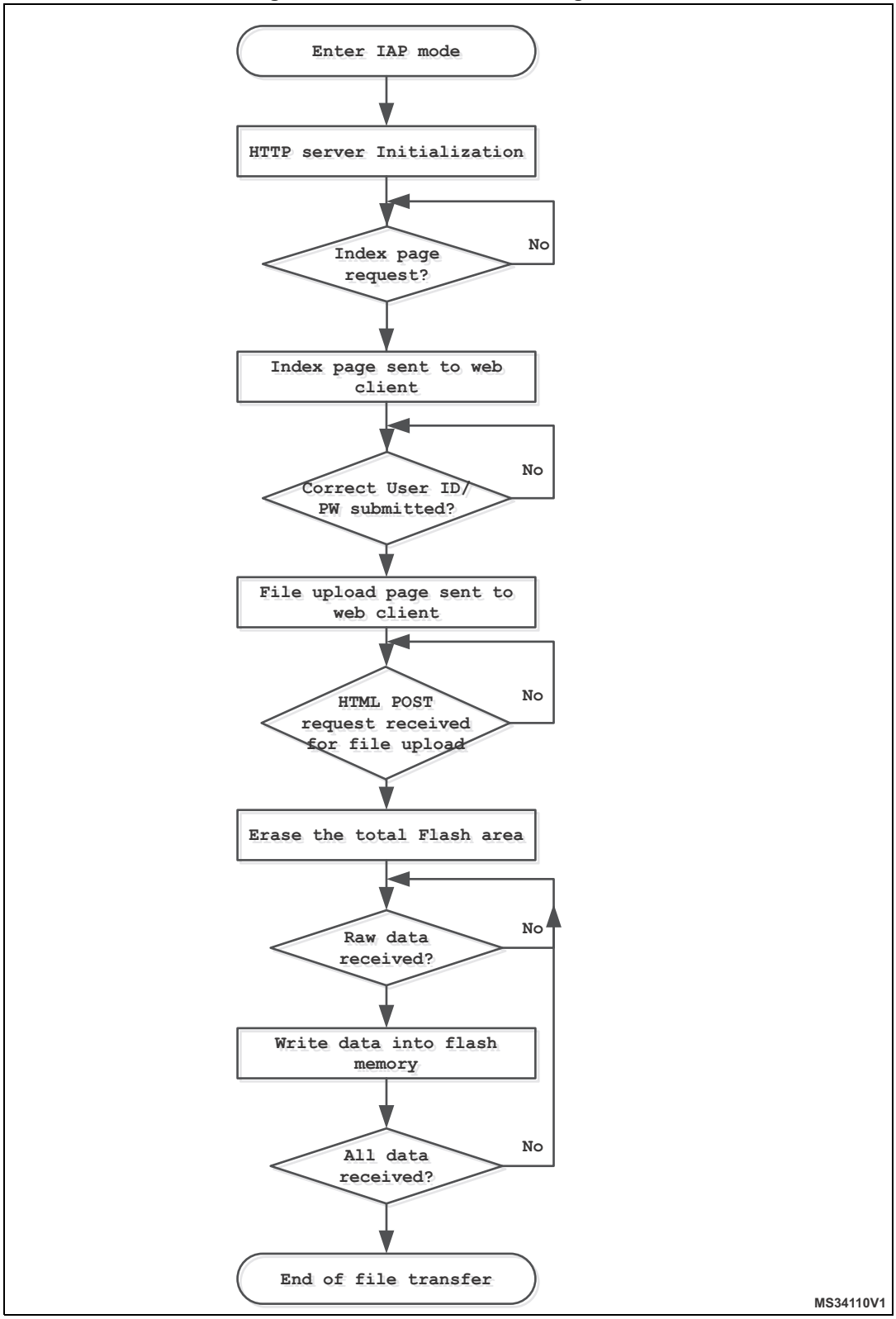


Figure 9 summarizes the IAP method using HTTP.

Figure 9. Flowchart of IAP using HTTP



4.3 Using the firmware

In order to test the IAP using HTTP, follow these steps:

1. Make sure the jumpers on the evaluation board are set correctly (see *Section 5.2*).
2. In the main.h file, uncomment the option `USE_IAP_HTTP`, also depending on your needs you can uncomment/comment other options like `USE_DHCP` or `USE_LCD`.
3. Recompile the firmware. Using the generated map file, make sure there is no overlap between the IAP code area (starting from address 0x0) and the user Flash area starting from address: `USER_FLASH_FIRST_PAGE_ADDRESS` (defined in main.h).
4. Program the firmware into STM32F4xx Flash and run it.
5. To enter IAP mode, press then release the Reset button while keeping the Key button pressed.
6. If USE_LCD is defined in main.h file then the LCD screen displays a message indicating that IAP mode has been entered. Also in the case of using DHCP (`USE_DHCP` defined in main.h), a message is displayed on the LCD screen indicating the success or failure of DHCP IP address allocation.
7. After IP address assignment (either static or dynamic address), the user can start the IAP process.
8. Open a web client (Mozilla Firefox or Microsoft Internet Explorer) and enter the STM32 IP address.
9. A login web page will be shown. In the User ID field enter “user” and in the Password field enter “stm32” then press the Login button.
10. The fileupload.html web page is then loaded. Browse for a binary image to be loaded into STM32 Flash then press the Upload button in order to start the IAP process.
11. If LCD is enabled, the progress of the IAP operation is shown on LCD.
12. At the end of the IAP operation, a new web page is loaded indicating the success of the file upload operation.
13. Press the RESET MCU button to reset the MCU and run the application just programmed in the STM32F4xx Flash.

Note:

1. If there is a connection issue when LCD is enabled, an error message displays on the LCD screen indicating the connection failure.
2. The software was tested with the following Web clients: Microsoft Internet Explorer 8 and Mozilla Firefox 24.

5 Environment

5.1 Application settings

5.1.1 PHY interface configuration

The Ethernet peripheral is interfaced with an external PHY to provide physical layer communication. The PHY registers definition and defines are located under the HAL configuration file “stm32f4xx_hal_conf.h”.

The PHY operates following two modes MII and RMII; to select the required mode user has to fill the “MediaInterface” parameter of “Init” structure when initializing the Ethernet peripheral

Note: The RMII mode is not supported when using the STM324x9I-EVAL board.

In the RMII mode with STM324xG-EVAL, you have to provide the 50 MHz clock by soldering a 50 MHz oscillator (ref SM7745HEV-50.0M or equivalent) on the U3 footprint located under CN3 and also by removing the jumper from JP5. This oscillator is not provided with the board.

5.1.2 MAC and IP address settings

The default MAC address is set to: 00:00:00:00:00:02. To change this address, modify the six bytes defined in the *stm32f4xx_hal_conf.h* file.

The default IP address is set to: 192.168.0.10. To change this address, modify the six bytes defined in the *main.h* file.

5.2 Evaluation boards settings

5.2.1 STM324x9I-EVAL settings

In order to run the software on the STM324x9I-EVAL board, please configure it as shown in *Table 2*.

Table 2. STM324x9I-EVALjumper configurations

Jumper	MII mode configuration
JP6	1-2: provide 25MHz clock by external crystal 2-3: provide 25MHz clock by MCO at PA8

5.2.2 STM324xG-EVAL settings

In order to run the software on the STM324xG-EVAL board, please configure it as shown in *Table 3*.

Table 3. STM324xG-EVAL jumper configurations

Jumper	MII mode configuration	RMII mode configuration
JP5	1-2: provide 25MHz clock by external crystal 2-3: provide 25MHz clock by MCO at PA8	Not fitted
JP6	2-3: MII interface mode is enabled.	1-2: RMII interface mode is enabled.
JP8	Open: MII interface mode is selected.	Closed: RMII interface mode is selected.

5.2.3 STM322xG-EVAL settings

To run the software on the STM322xG-EVAL board, configure it as shown in *Table 4*.

Table 4. STM322xG-EVAL jumper configurations

Jumper	MII mode configuration	RMII mode configuration
JP5	1-2: provide 25MHz clock by external crystal 2-3: provide 25MHz clock by MCO at PA8	Not fitted
JP6	2-3: MII interface mode is enabled.	1-2: RMII interface mode is enabled.
JP8	Open: MII interface mode is selected.	Closed: RMII interface mode is selected.

5.3 Firmware file organization

Table 5 describes the project source files:

Table 5. Files organization

File name	Description
main.c	Main application file
main.h	Main configuration file
httpserver.c /.h	HTTP server implementation
tftpserver.c /.h	TFTP server implementation
flash_if.c /.h	High level Flash access functions
stm32f4xx_it.c /.h	Interrupt handler
fsdata.c	HTML files as a ROM file system
lwipopts.h	LwIP configuration options
ethernetif.c/.h	interface between LwIP and Ethernet driver
stm32f4xx_hal_conf	HAL configuration file

Note: The table does not show files used from the STM32Cube HAL and BSP libraries and the LwIP stack.

5.4 Building an image for IAP

In order to build an image for IAP (to be loaded using the IAP software), make sure that:

1. The firmware is compiled and linked to run starting from the start address of the user Flash area (this address should be the same address as the one defined by USER_FLASH_FIRST_PAGE_ADDRESS in main.h).
2. The vector table start address is configured as the start address of the user Flash area. The vector table base offset is configured by modifying the value of the constant VECT_TAB_OFFSET defined in system_stm32f4xx.c file. For example, to set the vector table base location at 0x08020000: #define VECT_TAB_OFFSET 0x20000
3. The compiled software size does not exceed the total user Flash area.

6 Conclusion

The aim of this user manual is to explain the Ethernet In-Application Programming (IAP) using the STM32Cube HAL drivers for the STM32F4xx microcontrollers.

Two solutions are provided to support HTTP and TFTP protocols; both of them are based on the LwIP stack as a middleware component for TCP/IP communication.

Appendix A FAQ

A.1 How to choose between static or dynamic (DHCP) IP address allocation

When the macro `#define USE_DHCP` located in “main.h” is commented, a static IP address is assigned to the STM32 microcontroller (by default 192.168.0.10, this value can be modified from “main.h” file).

If the macro `#define USE_DHCP` is uncommented, the DHCP protocol is enabled, and the STM32 will act as a DHCP client

A.2 How the application behaves when the Ethernet cable is disconnected

When the cable is disconnected the Ethernet peripheral stops both transmission and reception traffics, also the network interface will be set down. If an LCD controller is used a message is displayed to inform user that the cable is not connected, else the Red LED of the evaluation board will turn on.

When the user re-connects the cable, the Ethernet traffic will resume and network interface will be set up. If an LCD controller is used a message is displayed to inform user the new IP address either with static or dynamic allocation, else the Yellow LED of the evaluation board will turn on.

A.3 How to port the application on a different hardware

When another hardware platform is used, you have to check the GPIO configuration into the HAL_ETH_MspInit() function for the Ethernet peripheral, also HAL_PPP_MspInit() or HAL_MspInit() if the application needs more PPP peripheral.

7 Revision history

Table 6. Document revision history

Date	Revision	Changes
28-Mar-2014	1	Initial release
05-Feb-2015	2	Updated *Section : Introduction* and *Section 1: STM32Cube overview*