



STM32F2x7 In-Application Programming (IAP) over Ethernet

Introduction

This application note is intended for developers using the STM32F2x7 microcontroller. It provides implementation solutions for In-Application Programming (IAP) using the STM32F2x7 Ethernet communications interface.

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)

Contents

1	IAP overview	5
1.1	Theory of operation	5
1.2	IAP using the MCU Ethernet interface	5
1.3	Implementing IAP over the Ethernet on the STM32F2x7	6
1.3.1	IAP method using TFTP	6
1.3.2	IAP method using HTTP	6
2	IAP using TFTP	7
2.1	TFTP overview	7
2.2	Implementing IAP for the STM32F2x7 using TFTP	7
2.3	Using the software	9
3	IAP using HTTP	10
3.1	HTTP file upload overview	10
3.2	Implementing IAP using HTTP on the STM32F2x7	12
3.3	Using the software	14
3.4	Known limitations	14
3.4.1	Extra bytes added to binary file	14
4	Environment	15
4.1	MAC and IP address settings	15
4.2	Jumper settings on the STM322xG-EVAL board	15
4.3	Software file organization	16
4.4	Code size measurements	16
4.5	Building an image for IAP	17
5	Revision history	18

List of tables

Table 1. TFTP opcodes 7

Table 2. Jumper configurations 15

Table 3. File organization 16

Table 4. Code size vs. configuration options 16

Table 5. Document revision history 18

List of figures

Figure 1. IAP operation flow. 5

Figure 2. TFTP packets 7

Figure 3. Flowchart of IAP using TFTP 8

Figure 4. TFTP32 dialog box. 9

Figure 5. Browser view of the file upload HTML form 10

Figure 6. IE8 HTTP header format. 11

Figure 7. Mozilla Firefox HTTP header format. 11

Figure 8. Login web page 12

Figure 9. File upload done web page. 12

Figure 10. Flowchart of IAP using HTTP 13



1 IAP overview

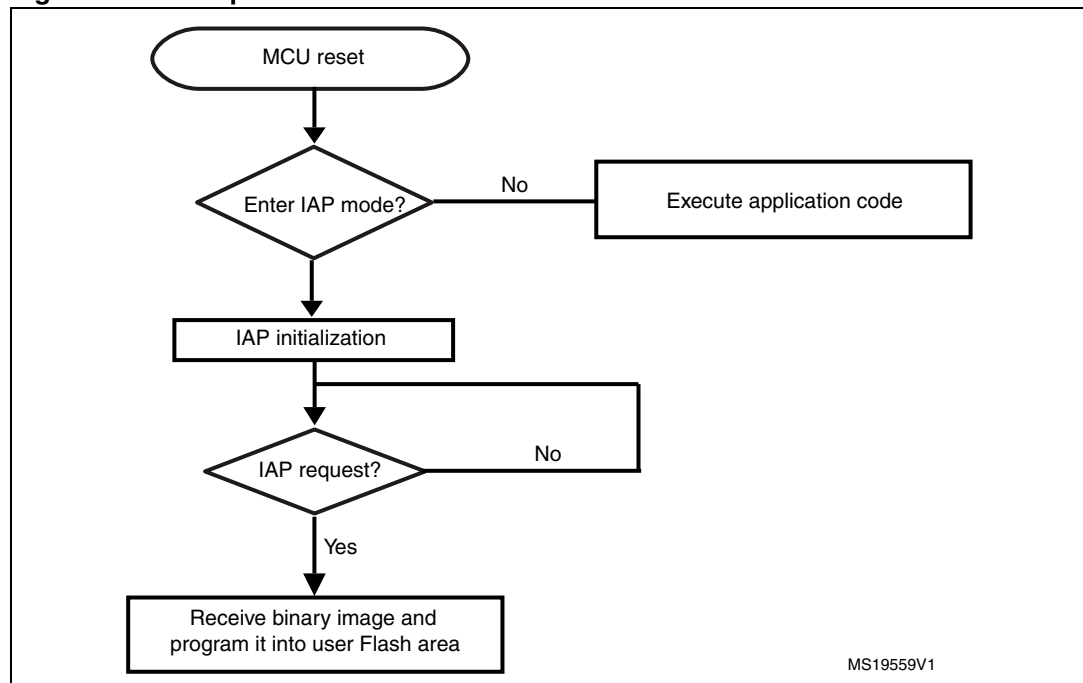
1.1 Theory of operation

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

When you boot the microcontroller, you can choose to put it in IAP mode in order to execute the IAP code or, in normal mode, start 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 1 illustrates the IAP operation flow:

Figure 1. IAP operation flow



1.2 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

1.3 Implementing IAP over the Ethernet on the STM32F2x7

This application note describes two solutions that implement IAP for the STM32F2x7 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 (v1.3.2), which is a light-weight implementation of the TCP/IP protocol suite.

1.3.1 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 on 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.

1.3.2 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.

2 IAP using TFTP

2.1 TFTP overview

TFTP is a simple file transfer protocol that works on top of UDP. A file transfer is initiated from a TFTP client, which 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 2 describes the format of the various TFTP packets:

Figure 2. TFTP packets

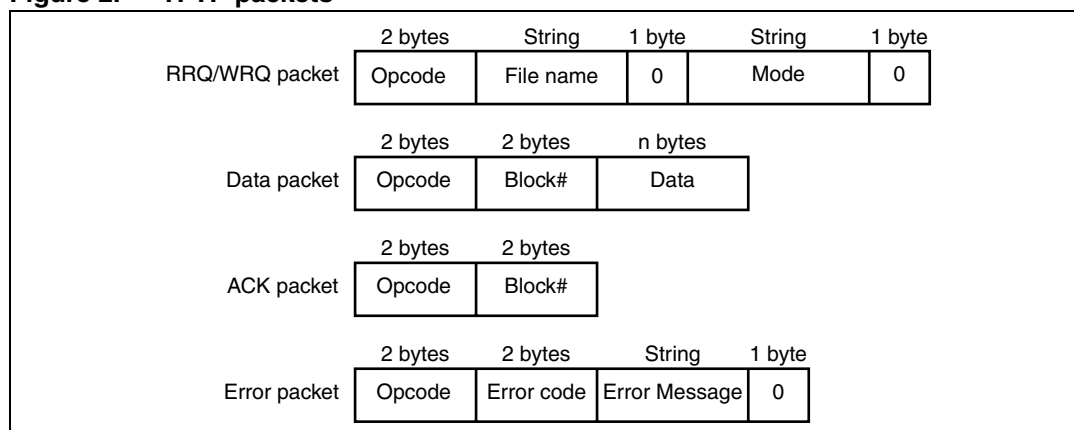


Table 1 lists the TFTP opcodes.

Table 1. TFTP opcodes

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

2.2 Implementing IAP for the STM32F2x7 using TFTP

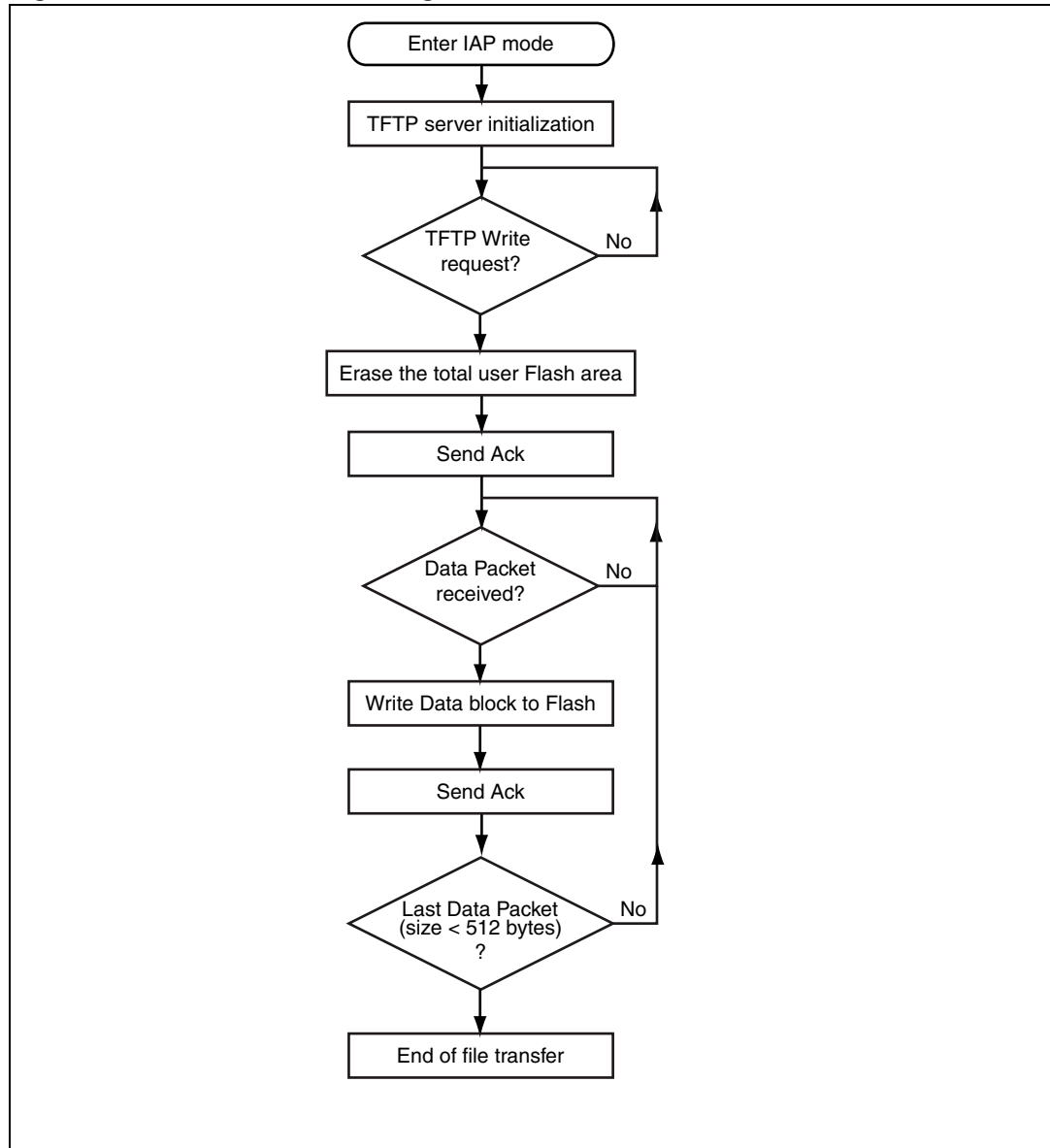
This IAP implementation consists of a TFTP server on top of the LwIP TCP/IP stack. This server responds to file WRITE requests received from a remote TFTP client (PC). TFTP READ requests are ignored.

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 3](#) provides an overview of the IAP operation using TFTP.

Figure 3. Flowchart of IAP using TFTP



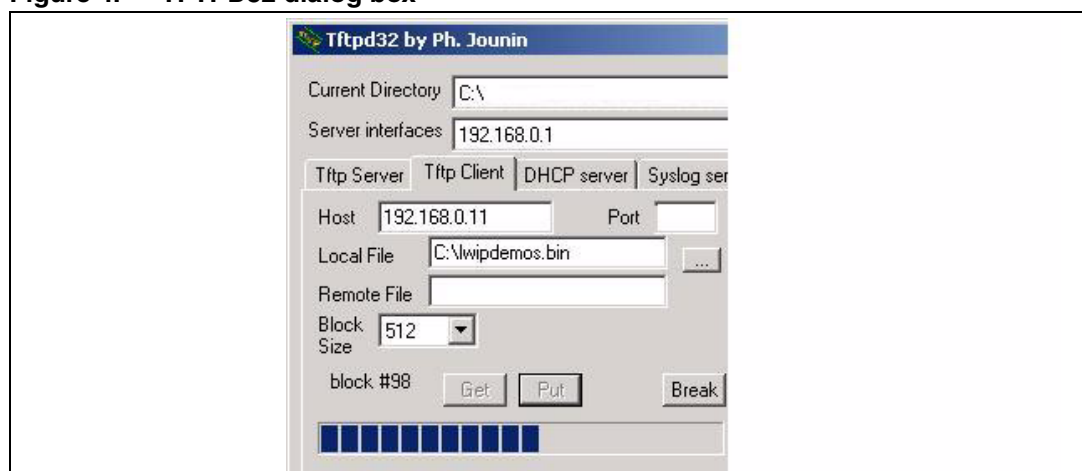
2.3 Using the software

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

1. Make sure the jumper settings in the STM32xG-EVAL board are set correctly (see [Table 2](#)).
2. In the main.h file, uncomment the option "USE_IAP_TFTP". Also, depending on your needs, you can uncomment/comment other options such as "USE_DHCP" or "USE_LCD".
3. Recompile the software. 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 software in the STM32 Flash and run it.
5. To enter the 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 TFTP32), and configure the TFTP server address (host address in TFTP32).
9. Browse for a binary image to load in the STM32 Flash (two binary images are provided as examples in the /project/binary folder).
10. Start a file write request by clicking the "Put" button in the TFTP32 utility.
11. When USE_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 STM32 Flash.

Note: *If there is a connection issue when USE_LCD is enabled, an error message displays on the LCD screen indicating the connection failure.*

Figure 4. TFTP32 dialog box



3 IAP using HTTP

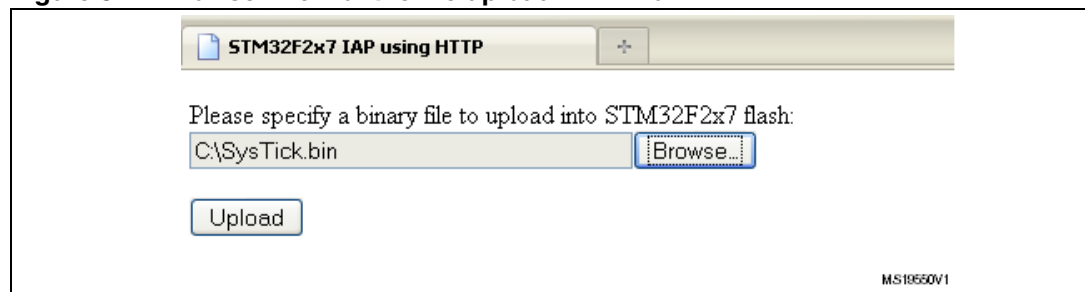
3.1 HTTP file upload overview

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 the 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>Please specify a binary file to upload into STM32F2x7 flash:
    <br>
    <input type="file" name="datafile" size="40">
  </p>
  <div>
    <input type="submit" value="Upload">
  </div></form>
```

Figure 5. Browser view of the file upload HTML form



Browse to select a binary file to upload, and then press 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. [Figure 6](#) shows Internet Explorer HTTP header formatting for a POST request. [Figure 7](#) shows the Mozilla Firefox HTTP header formatting.

The http Web server must handle these different formats.

Figure 6. IE8 HTTP header format

```

Hypertext Transfer Protocol
POST /upload.cgi HTTP/1.1\r\n
Accept: image/gif, image/jpeg, image/pjpeg, image/pjpeg, application/x-shockwave-flash, appl
Referer: http://192.168.0.12/checklogin.cgi\r\n
Accept-Language: en-us\r\n
User-Agent: Mozilla/4.0 (compatible; MSIE 8.0; windows NT 5.1; Trident/4.0; .NET CLR 1.1.432
Content-Type: multipart/form-data; boundary=-----7db28061402a2\r\n
Accept-Encoding: gzip, deflate\r\n
Host: 192.168.0.12\r\n
Content-Length: 1965\r\n
[Content length: 1965]
Connection: Keep-Alive\r\n
Cache-Control: no-cache\r\n
\r\n
MIME Multipart Media Encapsulation, Type: multipart/form-data, Boundary: "-----
[Type: multipart/form-data]
First boundary: -----7db28061402a2\r\n
Encapsulated multipart part: (application/octet-stream)
Content-Disposition: form-data; name="datafile"; filename="STM322xG_EVAL_SysTick.bin"\r\n
Content-Type: application/octet-stream\r\n\r\n
Media Type
Last boundary: \r\n-----7db28061402a2--\r\n

```

Figure 7. Mozilla Firefox HTTP header format

```

Hypertext Transfer Protocol
POST /upload.cgi HTTP/1.1\r\n
Host: 192.168.0.13\r\n
User-Agent: Mozilla/5.0 (windows; U; windows NT 5.1; en-US; rv:1.9.2.3) Gecko/20100401 Firef
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8\r\n
Accept-Language: en-us,en;q=0.5\r\n
Accept-Encoding: gzip,deflate\r\n
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7\r\n
Keep-Alive: 115\r\n
Connection: keep-alive\r\n
Referer: http://192.168.0.13/checklogin.cgi\r\n
Content-Type: multipart/form-data; boundary=-----114782935826962\r\n
Content-Length: 1969\r\n
[Content length: 1969]
\r\n
MIME Multipart Media Encapsulation, Type: multipart/form-data, Boundary: "-----
[Type: multipart/form-data]
First boundary: -----114782935826962\r\n
Encapsulated multipart part: (application/octet-stream)
Content-Disposition: form-data; name="datafile"; filename="STM322xG_EVAL_SysTick.bin"\r\n
Content-Type: application/octet-stream\r\n\r\n
Media Type
Last boundary: \r\n-----114782935826962--\r\n

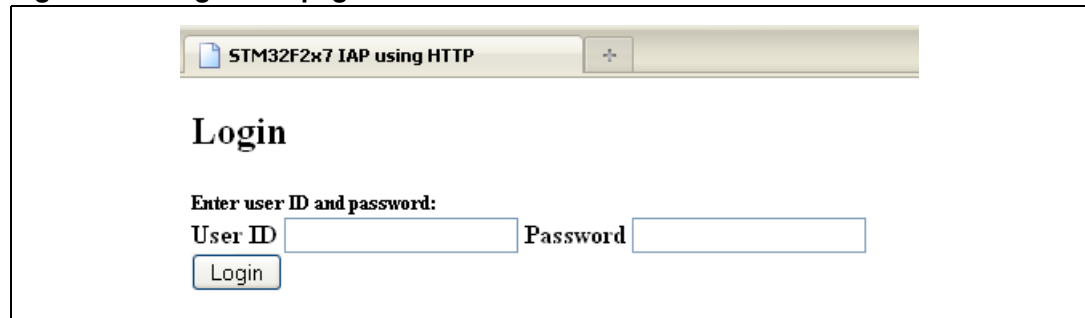
```

3.2 Implementing IAP using HTTP on the STM32F2x7

This IAP implementation consists of a 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 8](#)). This login Web page restricts access to the IAP file upload to authorized users.

Figure 8. Login web page



Enter a correct userID and password (pre-defined in main.h file) and click the Login button. A file upload Web page is then loaded (see [Figure 5](#)).

- Note:*
- 1 The default userID is: “**user**” and password is “**stm32**”.
 - 2 If the userID 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.

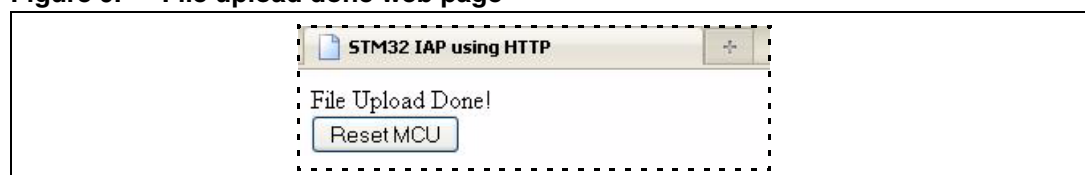
- Note:*
- Make sure that the binary file size does not exceed the total size of the STM32 user Flash area.

When clicking the Upload button (see [Figure 5](#)), 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.

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.

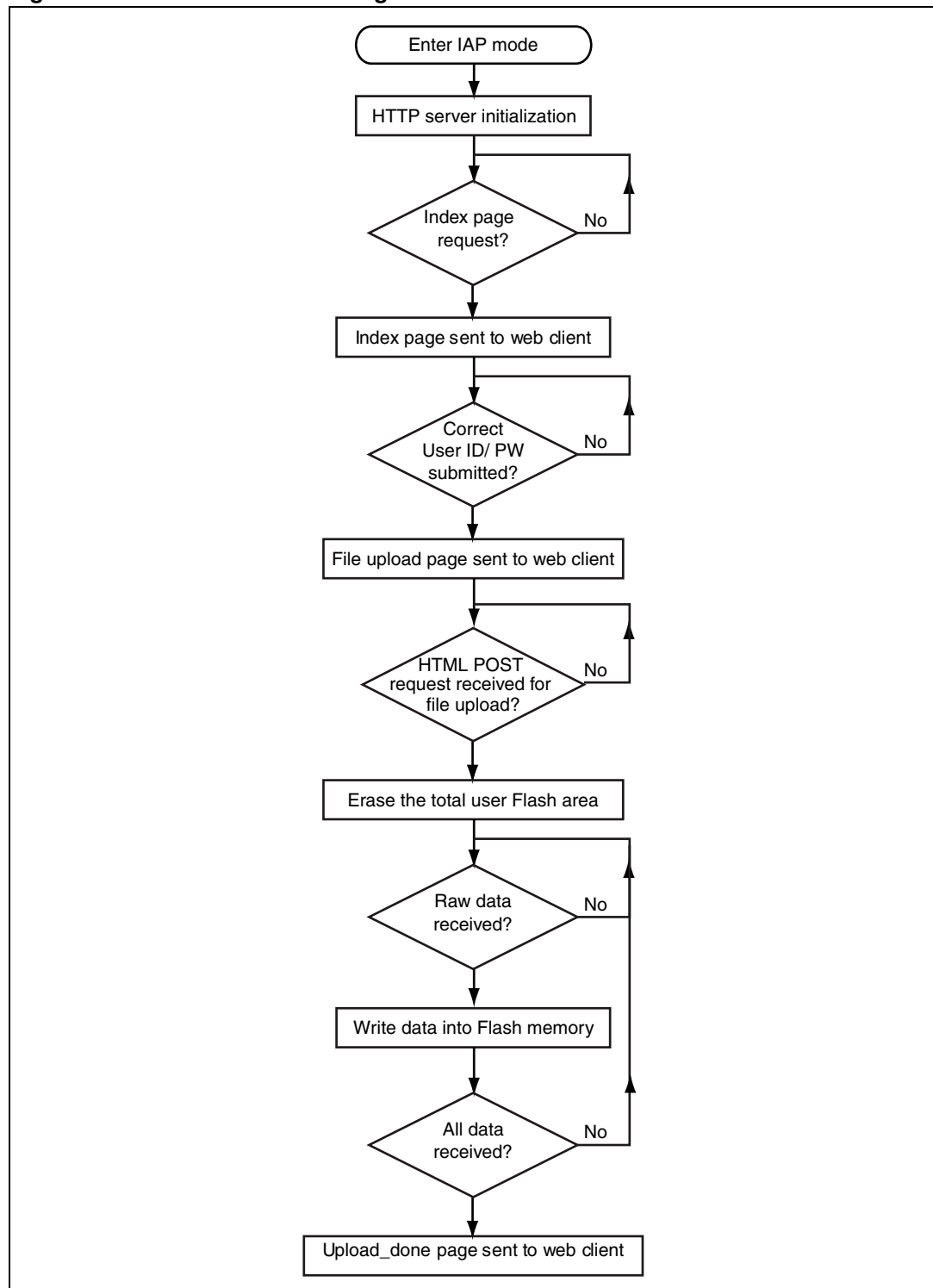
At the end of IAP operation, a Web page indicates the success of IAP operation, displaying a button which allows you to reset the MCU.

Figure 9. File upload done web page



[Figure 10](#) summarizes the IAP method using HTTP.

Figure 10. Flowchart of IAP using HTTP



3.3 Using the software

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

1. Make sure the jumper settings in the STM322xG-EVAL board are set correctly (see [Table 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 software. Using the generated map file, make 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 software into STM32 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 UserID 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. When USE_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. You can press the "RESET MCU" button to reset the MCU and run the application that you have just programmed in the STM32 Flash.

- Note:*
- 1 If there is a connection issue when USE_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 3.6.

3.4 Known limitations

3.4.1 Extra bytes added to binary file

A random boundary tag (no larger than 72 bytes according to RFC 1521) is added to the end of the uploaded binary file by the internet browser (Microsoft Internet Explorer or Mozilla Firefox). In the current version of the IAP software, this boundary tag is not removed and is stored in the Flash if sufficient space is available. If there is not enough space, extra bytes are not written in the Flash and no error is returned.

4 Environment

4.1 MAC and IP address settings

The MAC and IP address are defined in the *main.h* file.

The default MAC address is fixed to: 00:00:00:00:00:02

The IP address can be set either as a static address or as a dynamic address, assigned by a DHCP server. The default static IP address is: 192.168.0.10

You can select DHCP mode by enabling `USE_DHCP` in the *main.h* file.

Note that if you choose to configure the IP address by DHCP and the application does not find a DHCP server on the network to which it is already connected, the IP address is then automatically set to the static address (192.168.0.10).

4.2 Jumper settings on the STM322xG-EVAL board

In order to run the software, configure the STM322xG-EVAL board as shown in [Table 2](#).

Select between MII and RMII configuration in the *main.h* file in the project\inc folder.

For example, to select the RMII mode:

```
//#define MII_MODE
#define RMII_MODE
```

For MII mode, the PHY clock can be taken from the external crystal or from the STM32 via the MCO pin if both `MII_MODE` and `PHY_CLOCK_MCO` are defined in the *main.h* file.

- Note:**
- 1 In RMII mode, it is not possible to use MCO to output the 50 MHz clock to PHY due to the PLL limitation explained in chapter 2.6.5 of STM32F20x & STM32F21x Errata sheet (ES0005). In such a case, it is possible 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 on JP5. This oscillator is not provided with the board. For more details, please refer to STM3220G-EVAL evaluation board User manual UM1057.
 - 2 Throughout this document, "STM322xG-EVAL board" refers to STM3220G-EVAL and STM3221G-EVAL boards.

Table 2. Jumper configurations

Jumper number	MI I mode configuration	RMII mode configuration
JP5	1-2: 25 MHz clock driven by external crystal 2-3: 25 MHz clock driven 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

4.3 Software file organization

[Table 3](#) describes the project source files:

Table 3. File 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
netconf.c/.h	High level Ethernet interface functions
stm32f2x7_eth_bsp.c/.h	STM32F2x7 Ethernet hardware configuration
stm32f2xx_it.c/.h	Interrupt handler
fsdata.c	HTML files as a ROM file system
lwipopts.h	LwIP configuration options

Note: The table does not show the files used from the standard firmware library and the LwIP stack.

4.4 Code size measurements

[Table 4](#) provides code size measurements made with different configuration options in the main.h file.

Table 4. Code size vs. configuration options

Code size (bytes)	USE_IAP_TFTP	USE_IAP_HTTP	USE_LCD	USE_DHCP
13752	X			
21736	X		X	
27264	X		X	X
25408		X		
32512		X	X	
38440		X	X	X
39928	X	X	X	X

Note: The software is compiled using IAR EWARM v6.10, with high optimization for code size.

4.5 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 software is compiled/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 in one of two ways:
 - a) In the application code, relocate the vector table at the application load address using the “NVIC_SetVectorTable” function from the `misc.h/c` driver.
For example, set the Vector Table base location at `0x08010000`:
`NVIC_SetVectorTable(NVIC_VectTab_FLASH, 0x10000);`
 - b) By modifying the value of the constant “VECT_TAB_OFFSET” defined in `system_stm32f2xx.c` file.
For example, set the Vector Table base location at `0x08010000`:
`#define VECT_TAB_OFFSET 0x10000`
3. The compiled software size does not exceed the total user Flash area.

A software example with preconfigured projects to build application to be loaded with the IAP is included in *STM32F2xx in-application programming using the USART* (AN3374).

5 Revision history

Table 5. Document revision history

Date	Revision	Changes
27-May-2011	1	Initial release.
19-Oct-2011	2	Updated Section 4.2: Jumper settings on the STM322xG-EVAL board and Table 2: Jumper configurations .

Please Read Carefully:

Information in this document is provided solely in connection with ST products. STMicroelectronics NV and its subsidiaries ("ST") reserve the right to make changes, corrections, modifications or improvements, to this document, and the products and services described herein at any time, without notice.

All ST products are sold pursuant to ST's terms and conditions of sale.

Purchasers are solely responsible for the choice, selection and use of the ST products and services described herein, and ST assumes no liability whatsoever relating to the choice, selection or use of the ST products and services described herein.

No license, express or implied, by estoppel or otherwise, to any intellectual property rights is granted under this document. If any part of this document refers to any third party products or services it shall not be deemed a license grant by ST for the use of such third party products or services, or any intellectual property contained therein or considered as a warranty covering the use in any manner whatsoever of such third party products or services or any intellectual property contained therein.

UNLESS OTHERWISE SET FORTH IN ST'S TERMS AND CONDITIONS OF SALE ST DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY WITH RESPECT TO THE USE AND/OR SALE OF ST PRODUCTS INCLUDING WITHOUT LIMITATION IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE (AND THEIR EQUIVALENTS UNDER THE LAWS OF ANY JURISDICTION), OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

UNLESS EXPRESSLY APPROVED IN WRITING BY TWO AUTHORIZED ST REPRESENTATIVES, ST PRODUCTS ARE NOT RECOMMENDED, AUTHORIZED OR WARRANTED FOR USE IN MILITARY, AIR CRAFT, SPACE, LIFE SAVING, OR LIFE SUSTAINING APPLICATIONS, NOR IN PRODUCTS OR SYSTEMS WHERE FAILURE OR MALFUNCTION MAY RESULT IN PERSONAL INJURY, DEATH, OR SEVERE PROPERTY OR ENVIRONMENTAL DAMAGE. ST PRODUCTS WHICH ARE NOT SPECIFIED AS "AUTOMOTIVE GRADE" MAY ONLY BE USED IN AUTOMOTIVE APPLICATIONS AT USER'S OWN RISK.

Resale of ST products with provisions different from the statements and/or technical features set forth in this document shall immediately void any warranty granted by ST for the ST product or service described herein and shall not create or extend in any manner whatsoever, any liability of ST.

ST and the ST logo are trademarks or registered trademarks of ST in various countries.

Information in this document supersedes and replaces all information previously supplied.

The ST logo is a registered trademark of STMicroelectronics. All other names are the property of their respective owners.

© 2011 STMicroelectronics - All rights reserved

STMicroelectronics group of companies

Australia - Belgium - Brazil - Canada - China - Czech Republic - Finland - France - Germany - Hong Kong - India - Israel - Italy - Japan - Malaysia - Malta - Morocco - Philippines - Singapore - Spain - Sweden - Switzerland - United Kingdom - United States of America

www.st.com

