AN0086
Application Note

AT32F437 IAP using the EMAC

Introduction

This application note introduces how to use AT32F437 Ethernet communication interface to implement in-application programming (IAP).

There are two available solutions based on LwIP TCP/IP stack:

- IAP method using TFTP
- IAP method using HTTP

Applicable products:

Part number	AT32F437 series

Contents

List of tables

List of figures

1 IAP introduction

1.1 Overview

The in-application programming (IAP) is way to upgrade firmware through MCU communication interfaces (such as USART, USB, CAN and Ethernet).

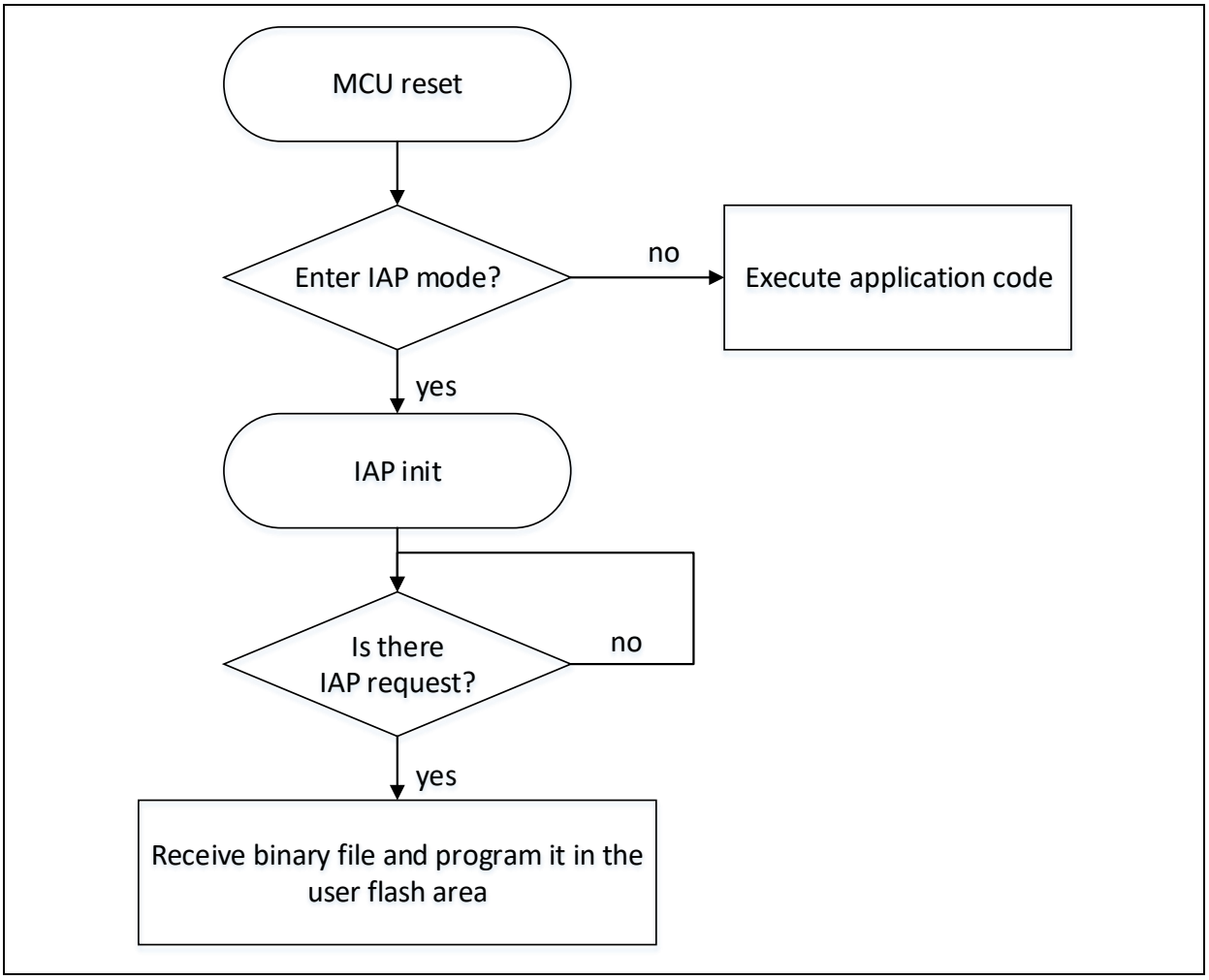
Enable the MCU and set it to one of the following modes:

- IAP mode, for executing IAP code
- Normal mode, for executing application code

Both IAP code and application code are stored in the MCU internal Flash (IAP code stored in the first page of Flash, and application code stored in the rest Flash storage area).

Figure 1 shows the operation flow chart of IAP.

Figure 1 IAP flow chart



1.2 IAP using MCU Ethernet interface

The Ethernet interface is the preferred option for IAP in embedded system. Its advantages are:

- High-speed communication interface (10/100 Mbps)
- Remote programming through the network (LAN or WAN)
- Implement IAP using TCP/IP stack-based FTP, TFTP and HTTP

1.3 IAP using Ethernet for AT32F437 MCUs

This application note introduces two available IAP solutions for AT32F437 MCUs by using Ethernet communication peripherals.

- IAP method using TFTP
- IAP method using HTTP

Both methods are based on LwIP (2.1.2, which is a lightweight TCP/IP stack).

1.3.1 IAP using TFTP

This method is widely used for embedded applications with firmware upgrade function, such as embedded Linux bootloader.

TFTP is a simple file transfer protocol that is executed over the UDP transport layer, which is ideal for LAN environment. It is based on the client/server architecture where the client sends file transfer request to the server (read or write access).

To implement IAP, a simple TFTP server is required on the LwIP stack to handle the write access request from PC TFTP client only.

1.3.2 IAP using HTTP

This method is used for remote programming through Internet, and TCP protocol is required to implement HTTP service.

HTTP runs based on TCP protocol, and it provides a way to send a binary file from a Web client (Mozilla Firefox or Microsoft Internet Explorer) in HTML form format, which is called form-based file upload in HTTP (RFC1867).

This application note introduces how to implement the abovementioned two IAP methods and how to use software.

2 How to implement IAP using TFTP

2.1 TFTP overview

TFTP is a simple file transfer protocol that is executed over the UDP transport layer. File transfer is initiated by the TFTP client by sending read or write access request to TFTP server, and file data transfer starts as soon as the request is acknowledged by the server. The data is transferred in blocks of fixed size (for example, 512 bytes per block).

The nest data block can be sent only after the previous data block is acknowledged by the receiver. This acknowledgement mechanism is realized by the block number sent with each data block. When the data block size is smaller than the fixed block size, it indicates the end of file transfer.

Figure 2 lists multiple TFTP data packets.

Figure 2 TFTP data packets

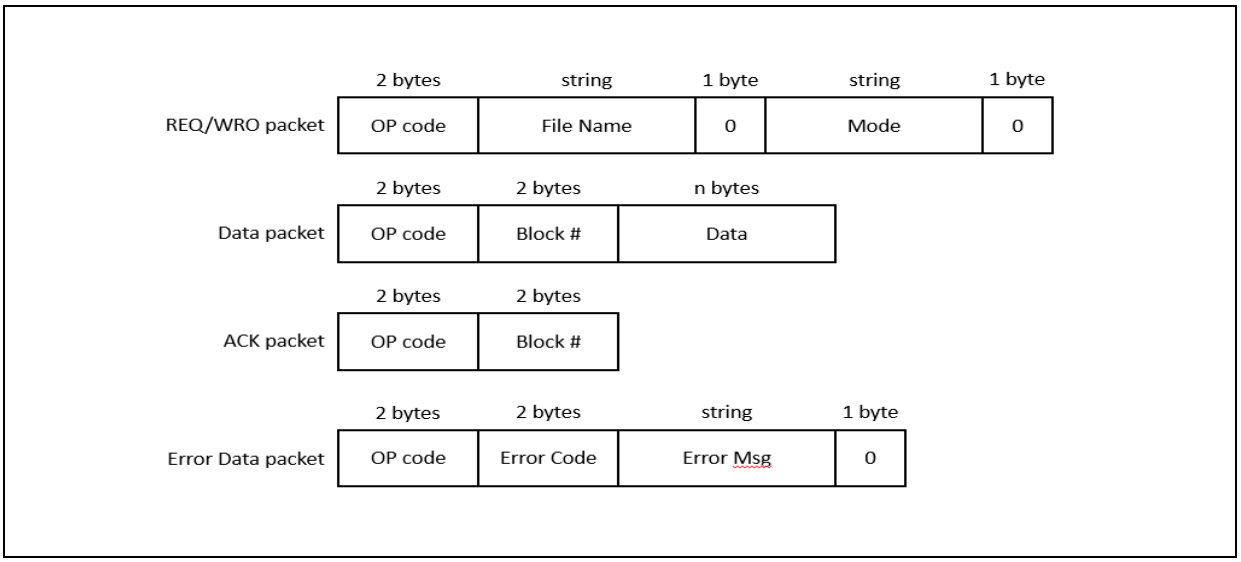


Table 1 lists the TFTP operation codes.

Table 1 TFTP operation codes

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

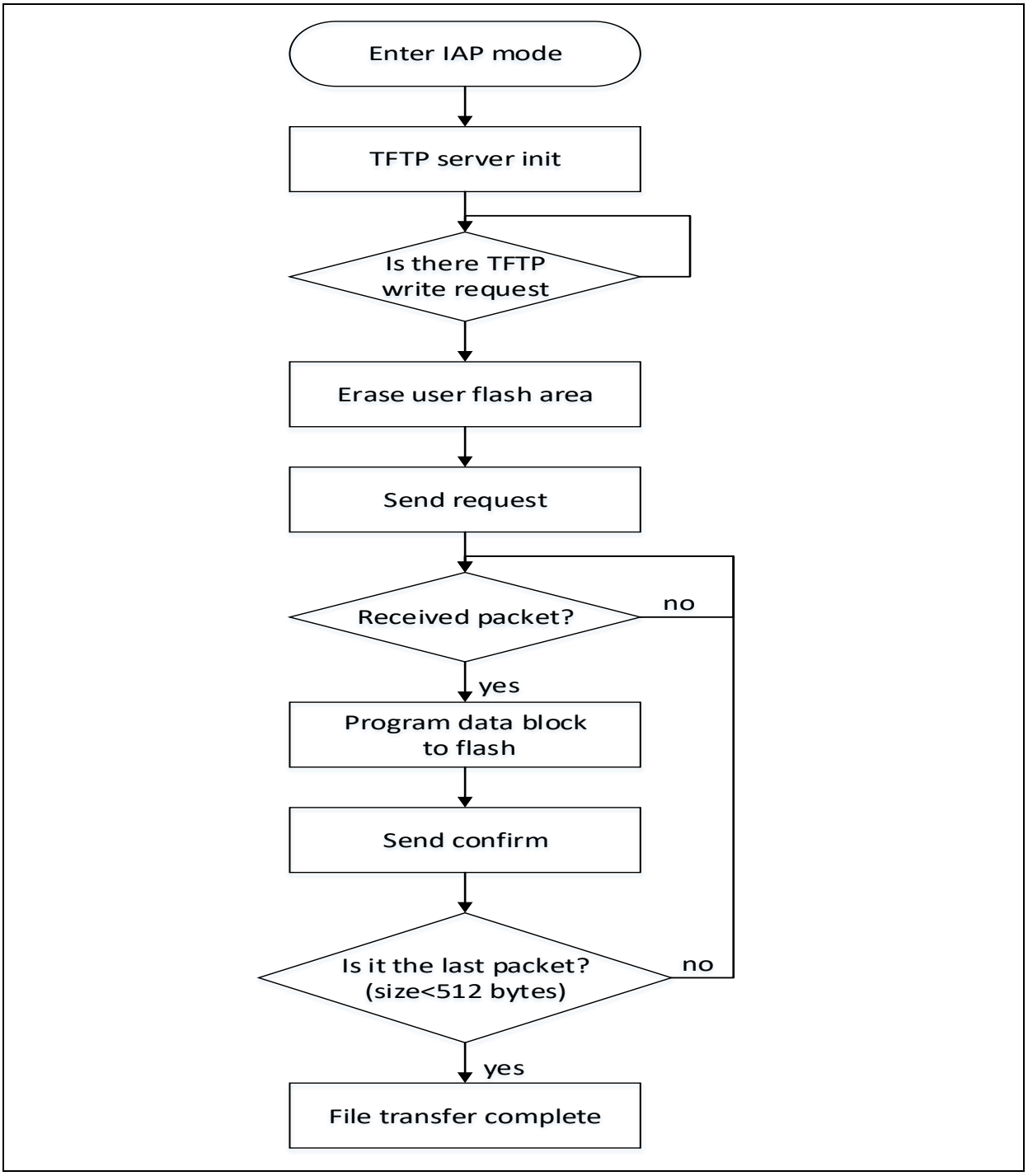
2.2 How to implement IAP using TFTP for AT32F437 MCUs

This IAP is implemented by using the TFTP server based on LwIP TCP/IP stack.
This server responds to the write request sent from remote TFTP client (PC).
TFTP write request is ignored.
TFTP usually writes the received file to the file system, while the server writes the received data blocks to the MCU FLASH.

Note: The data block is fixed to 512 bytes in the whole process.

Figure 3 shows the operation flow of IAP implementation using TFTP.

Figure 3 IAP flow chart (using TFTP)

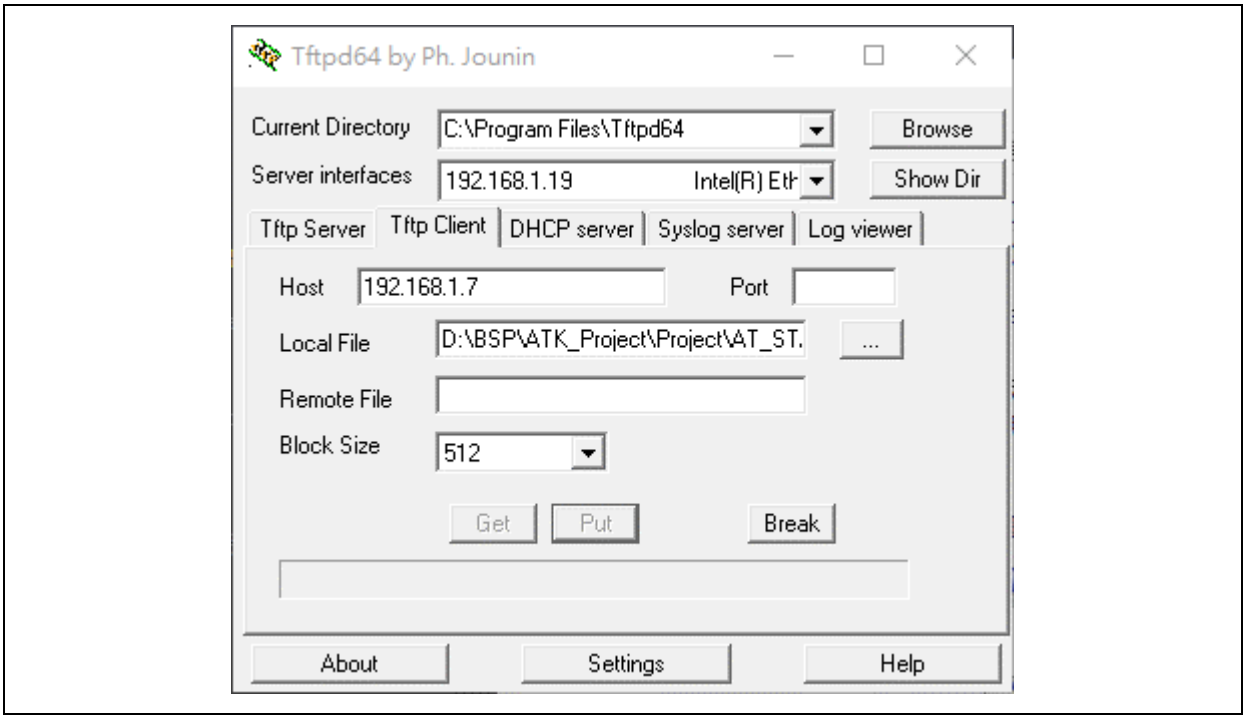


2.3 Application of software

Follow the steps below to test the IAP through TFTP:

1. Open the *iap.h* file, and cancel the comments for USE_IAP_TFTP option;
2. Re-compile the software, and use the generated mapping file to ensure no overlap between IAP code areas (start address: 0x0); the user FLASH area starts from APP_START_SECTOR_ADDR (defined in the *iap.h* file);
3. Write and run software in AT32 FLASH;
4. Press the USER Key on the evaluation board to enter IAP mode;
5. Once the IP is assigned (static or dynamic), the user can start IAP programming;
6. Open the TFTP client (e.g., Tftpd64) on PC, and configure the TFTP server address (host address in Tftpd64);
7. Click the Put button in Tftpd64 application to enable file write request;
8. At the end of IAP programming, reset the evaluation board and run the application in AT32 FLASH.

Figure 4 TFTP64 dialog box



3 How to implement IAP using HTTP

3.1 HTTP file upload

RFC1867 defines form-based file upload in HTTP. The HTTP POST is used for sending the original binary data.

Figure 5 gives an example of HTML code used for form-based file upload.

Figure 5 Form-based file upload interface

Note: Before sending the file data, Web client sends HTTP header file data (including file name, content, length and relevant information), some of which are required to be parsed by Web server.

The HTTP header files used by Web client are usually in different formats. Figure 6 shows the format of HTTP header file of Internet Explorer in POST request; Figure 7 shows the format of HTTP header file in Mozilla Firefox.

HTTP Web server can handle these different file formats.

Figure 6 IE11 HTTP header file format

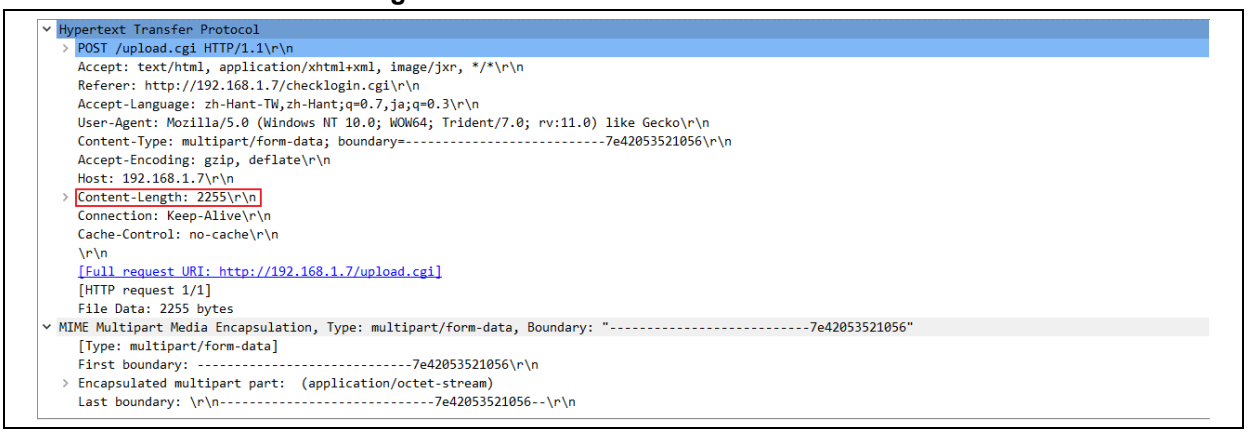
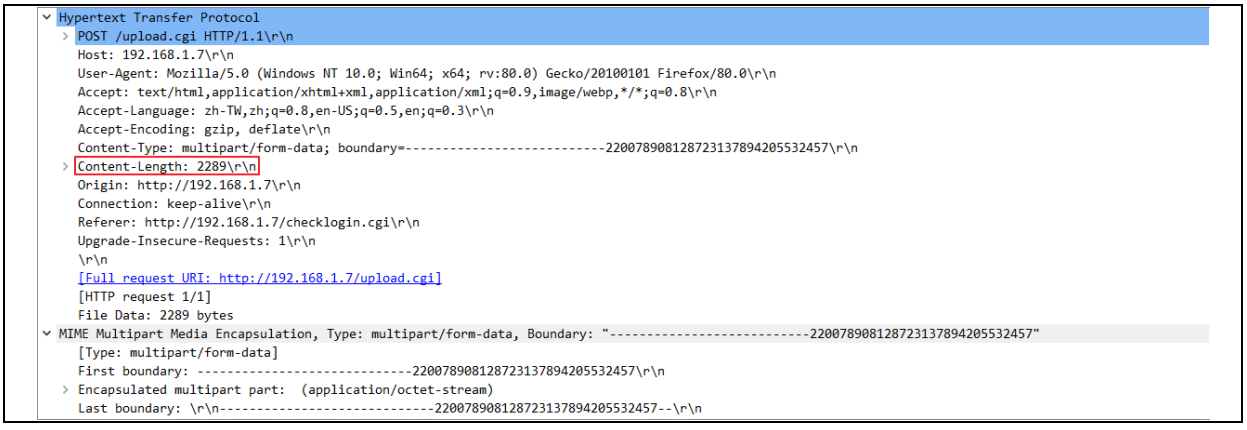


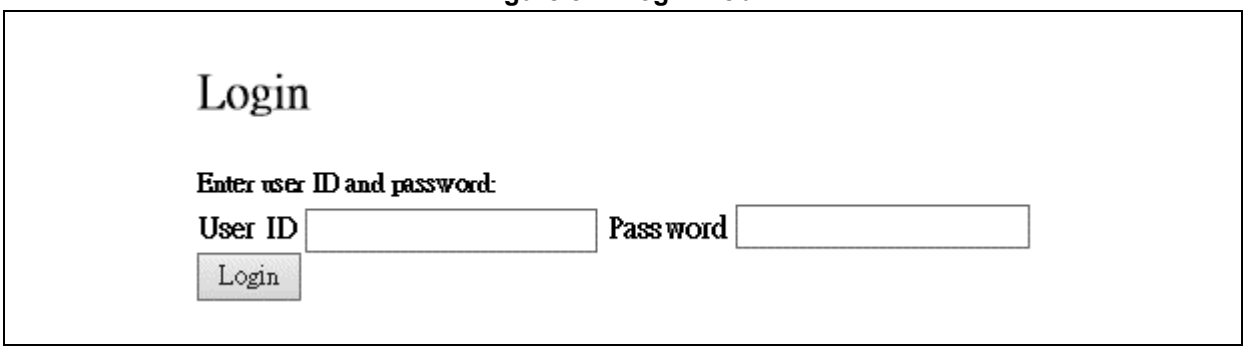
Figure 7 Mozilla Firefox HTTP header file format



3.2 How to implement IAP using HTTP for AT32F437 MCUs

This IAP is implemented by using the HTTP Web server based on LwIP stack.
Enter the AT32 IP address in the browser, and the login web page pops up (Figure 8), allowing authorized users to use the IAP file upload function.

Figure 8 Login web



Note: *1. The default User ID is "user", and the password is "at32".*
2. If the User ID or Password is incorrect, the login Web page will reload.
After successful login, browse and select the binary file to be uploaded to AT32 FLASH.

Note: The binary file size is not allowed to exceed the total user FLASH area.
Click on Upload to send POST request to the server. At this point, the server starts to erase the entire user FLASH area and wait to receive the original data of binary file; then the received data is written to the user FLASH area.
The total length of the to-be-received data is obtained from the HTTP header file that is sent at the beginning of transfer.
After completion of IAP, Web page displays "File Upload Done" and a Reset MCU button.

Figure 9 Web page information

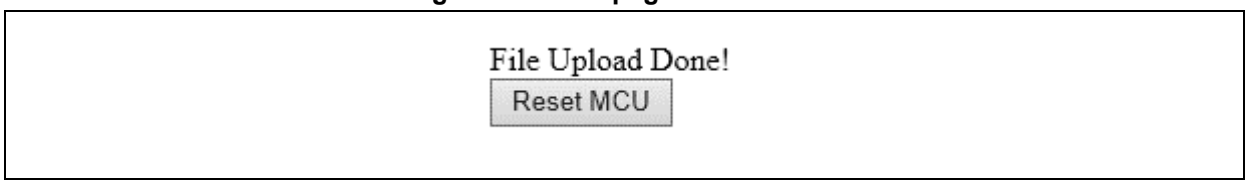
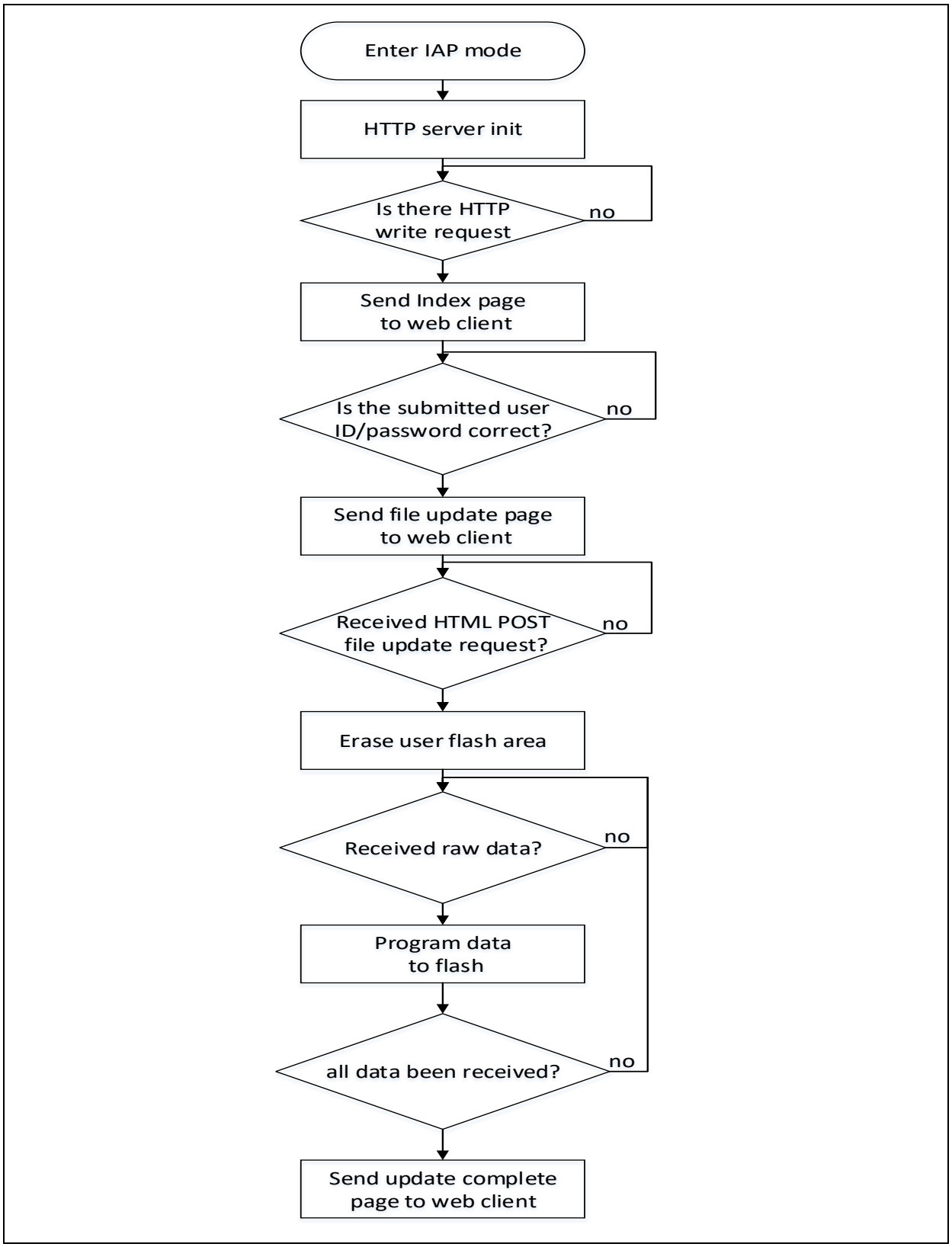


Figure 10 shows the operation flow of IAP using HTTP.

Figure 10 IAP flow chart (using HTTP)



3.3 Application of software

Follow the steps below to test the IAP through HTTP:

1. Open the *iap.h* file, and cancel the comments for USE_IAP_HTTP option;
2. Re-compile the software, and use the generated mapping file to ensure no overlap between IAP code areas (start address: 0x0); the user FLASH area starts from APP_START_SECTOR_ADDR (defined in the *iap.h* file);
3. Write and run software in AT32 FLASH;
4. Press the USER Key on the evaluation board to enter IAP mode;
5. Once the IP is assigned (static or dynamic), the user can start IAP programming;
6. Open the Web client (Mozilla Firefox or Internet Explorer) on PC, and enter the AT32 IP address;
7. The login Web page pops up; enter “user” in the User ID field and “at32” in the Password field, and then click Login;
8. After completion of IP operation, a new Web page is loaded, indicating file upload success;
9. Click the Reset MCU button to reset MCU, and run the application in AT32 FLASH.

Note: Use Microsoft Internet Explorer 11 and Mozilla Firefox 80.0 to test the software.

3.4 Restrictions

3.4.1 Add extra bytes in binary files

The browser (Microsoft Internet Explorer or Mozilla Firefox) will add a random boundary marker to the end of the uploaded binary file (according to RFC1521, this marker is less than 72 bytes). In the latest version of the IAP software, this marker is not deleted but stored in the FLASH when it is available (if there is not available space, extra bytes are not written into Flash, and no error is returned).

4 Environment requirements

4.1 Hardware

1. DM9162 Ethernet module
2. AT-START-F407 evaluation board
3. Ethernet cable

4.2 Software

- utilities\AT32F437_emac_iap_demo\source_code\bootloader, emac iap source program, to run the iap upgrade program

4.3 MAC address and IP address

The MAC address is defined in the *netconf.h* file. The default MAC address is fixed to 00:00:44:45:56:01.

The IP address is defined in the *netconf.h* file. The IP address can be static or dynamic that is assigned by DHCP server. The default static address is 192.168.81.37.

The DHCP mode is selected by enabling LWIP_DHCP in the *lwipopts.h* file.

4.4 Software files

Table 2 Project source files

File name	Description
main.c	Main application file
httpserver.c/.h	HTTP server implementation
tftpserver.c/.h	TFTP server implementation
flash.c/.h	High-level FLASH access function
netconf.c/.h	High-level Ethernet interface function
at32f4xx_EMAC.c/.h	AT32F4xx Ethernet hardware configuration
at32f4xx_it.c/.h	Interrupt handler
fsdata.c	HTML files of ROM file system
lwipopts.h	LwIP configuration option

Note: Files used by standard firmware library and LwIP stack are not listed.

4.5 IAP mapping

Notes on building IAP map (to be loaded using the IAP software):

1. The compile/linkage software must run from the start address of the user Flash area (this address is the same as that defined by APP_START_SECTOR_ADDR in *iap.h*);
2. Configure the vector table start address to be the start address of user Flash area:
   A. In the application code, use misc.h/.c to drive the NVIC_SetVectorTable function to relocate the vector table of the application loading address.
   For example, set the vector table base address to be 0x08010000:
   NVIC_SetVectorTable(NVIC_VectTab_FLASH, 0x10000);
   B. Modify the VECT_TAB_OFFSET value defined in system_at32f4xx.c.
   For example, set the vector table base address to be 0x08010000:
   #define VECT_TAB_OFFSET 0x10000
3. The software after compilation does not exceed the total size of user Flash area.

5 Revision history

Table 3 Document revision history

Date	Version	Revision note
2021.09.17	2.0.0	Initial release.
2023.06.19	2.0.1	Modified document layout.

IMPORTANT NOTICE – PLEASE READ CAREFULLY

Purchasers are solely responsible for the selection and use of ARTERY's products and services; ARTERY assumes no liability for purchasers' selection or use of the products and the relevant services.

No license, express or implied, to any intellectual property right is granted by ARTERY herein regardless of the existence of any previous representation in any forms. If any part of this document involves third party's products or services, it does NOT imply that ARTERY authorizes the use of the third party's products or services, or permits any of the intellectual property, or guarantees any uses of the third party's products or services or intellectual property in any way.

Except as provided in ARTERY's terms and conditions of sale for such products, ARTERY disclaims any express or implied warranty, relating to use and/or sale of the products, including but not restricted to liability or warranties relating to merchantability, fitness for a particular purpose (based on the corresponding legal situation in any unjudicial districts), or infringement of any patent, copyright, or other intellectual property right.

ARTERY's products are not designed for the following purposes, and thus not intended for the following uses: (A) Applications that have specific requirements on safety, for example: life-support applications, active implant devices, or systems that have specific requirements on product function safety; (B) Aviation applications; (C) Aerospace applications or environment; (D) Weapons, and/or (E) Other applications that may cause injuries, deaths or property damages. Since ARTERY products are not intended for the above-mentioned purposes, if purchasers apply ARTERY products to these purposes, purchasers are solely responsible for any consequences or risks caused, even if any written notice is sent to ARTERY by purchasers; in addition, purchasers are solely responsible for the compliance with all statutory and regulatory requirements regarding these uses.

Any inconsistency of the sold ARTERY products with the statement and/or technical features specification described in this document will immediately cause the invalidity of any warranty granted by ARTERY products or services stated in this document by ARTERY, and ARTERY disclaims any responsibility in any form.