ATWINC1510

ATWINC1510 Host Image Download User Guide

Introduction

The host file download feature is an implementation to download the host new image, or any file required by the host. The application example implements the download and update of the host image Over-the-Air (OTA) as follows:

- Initiates the host image download through the ATWINC1510 over-the-air
- Receives the new host image from the ATWINC1510 Flash and bootloads on the host MCU Flash application

The host file download feature provides mechanism using the ATWINC chip, with which it retrieves a file from a remote location (example, web server), stores the downloaded file in the ATWINC's Flash and notifies the application when the file transfer is complete. This feature supports all file types, which provides the ability to download files for the application and also for the host to update its firmware while using the ATWINC to download and store the host image temporarily.

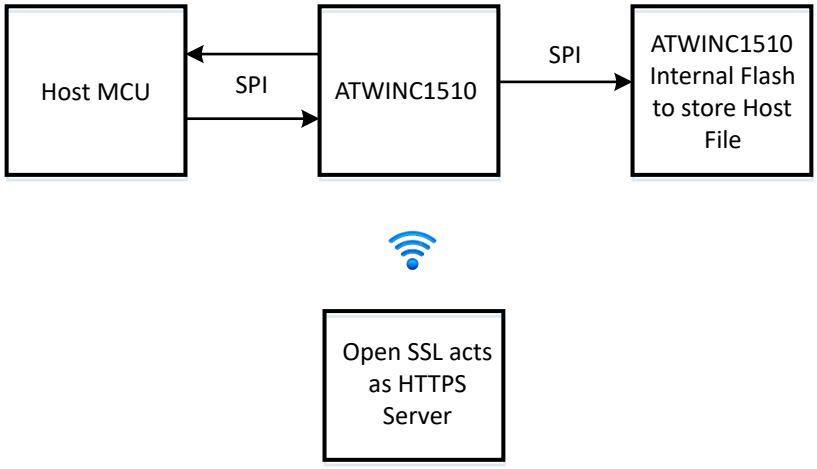
Table of Contents

1. System Overview

The following figure shows the most common setup using the host image download feature. The ATWINC acts as a Station and is connected to an AP which has access to the remote location (server), where the file is stored. When the process completes or interrupts, the ATWINC notifies the application indicating the process status as success or failure.

The application is comprised of components shown in the following figure.

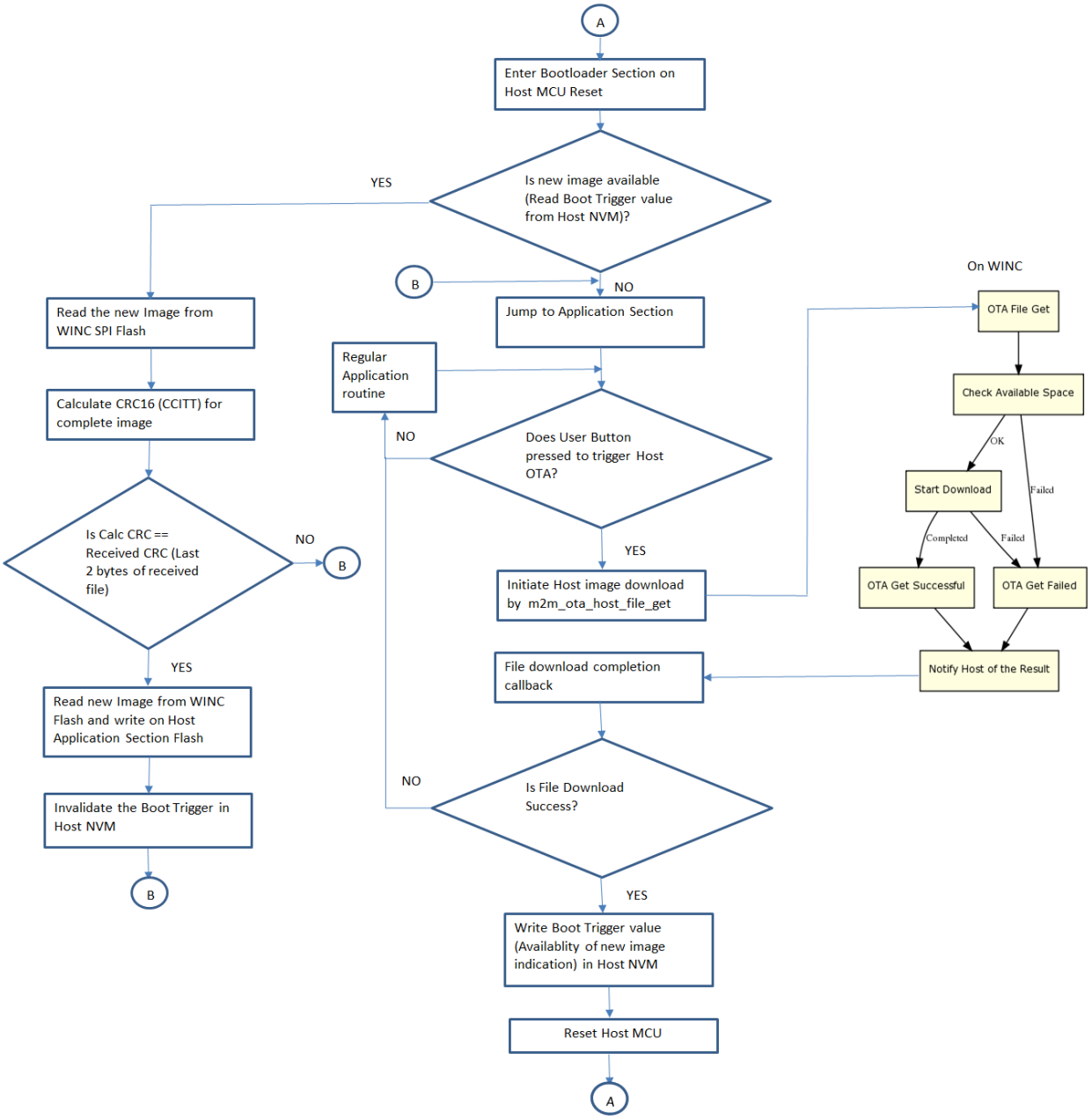
Figure 1-1. System Block Diagram



1.1 Host Image Download Process Flow

The following diagram is the flow of the host image download process.

Figure 1-2. Flow Diagram

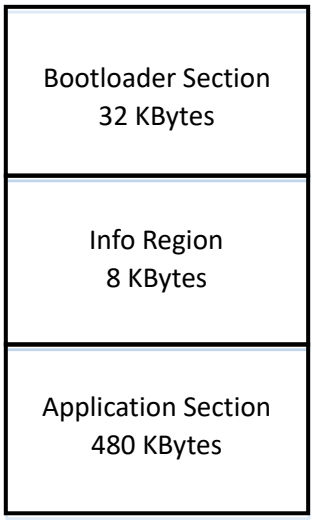


1.2 Memory Split of Host MCU

The 512 KB of host MCU Flash is split into different memory sections as shown in the following figure. These sections can be modified for user host MCU requirement by configuring the size and start address of each section in the `conf_bootloader.h` file.

Note: The following memory split of host MCU Flash is only an example.

Figure 1-3. Memory Split of Host MCU



- Bootloader Section:
  On Reset, the device starts from the bootloader section. The bootloader checks for the availability of new image by reading the boot trigger value from the info region. If a valid image is available, then it reads the image from the ATWINC Flash and loads on the application section. When the new application image received OTA is loaded on the application section, the interrupt vector table is remapped to the application section and jumps to the application start address.
  - Only one application section is available. Therefore, after the data integrity check (calculates the CRC for complete image and compares with received CRC) only a new image is loaded on the application section. The demo example uses CRC16 check mechanism. The user is free to use any checksum mechanism.
  - While flashing a new image on the application section, if there is power failure/Reset, the complete image is loaded again on the application section. This is done as the boot trigger is invalidated only when the complete image is loaded.
- Info Region:
  When the ATWINC notifies the new image, the application code sets the boot trigger and image size into the info region. The information stored in the info region is an 8 byte value, and it is possible to erase as 8 KB block in the SAMG55 host MCU. Therefore, 8 KB is allocated for this section.
- Application Section:
  This section contains the application code, which has the ATWINC driver along with the host OTA feature.
  - The application initiates the host image download by calling `m2m_ota_host_file_get()` API. When the complete image is received, the ATWINC notifies application by calling the callback passed as part of `m2m_ota_host_file_get()` API.
  - During callback, the application writes the info region with boot trigger and image size and then resets the device.

1.3 Prerequisites

- Hardware Prerequisites
  - SAM G55 Xplained Pro Evaluation kit

  - ATWINC1510 extension (For more details, refer to the Software Programming Guide)
    Note: The Host File Download feature is not supported for ATWINC1500 as this variant contains 4 MBits SPI Flash.
  - Micro-USB cable (Micro-A/Micro-B)
- Software Prerequisites
  - Atmel Studio 7.0 (For more details, refer to the User Guide)
  - Advanced Software Framework installed as part of Atmel Studio (for more details, refer to the application notes)
  - OpenSSL - this package can be downloaded from https://www.openssl.org/source/. It is recommended to download the Windows downloader https://slproweb.com/products/Win32OpenSSL.html.
    Note:
    - OpenSSL is used for demo purposes only. The user can choose their own HTTPS server.
    - The demo is tested with the latest Win64 OpenSSL v1.1.0h version on Windows® machine.
  - The SRecord tool is used to merge two images and add CRC16 of the host OTA image. It can be downloaded from http://srecord.sourceforge.net/.
    Note: The SRecord tool is used for demo purposes only. The user can use any tool which performs this operation.

2. OpenSSL Server Setup

In this demo application, OpenSSL is used as a secure HTTPS server. The user can use any HTTPS/ HTTP server of interest.

2.1 Generating Custom Server Certificate

To create an SSL connection, the server and client require an SSL certificate. The following is the procedure to create and sign custom certificates using OpenSSL.

1. After installing OpenSSL, open command prompt and navigate to the directory where OpenSSL is installed (For example: `C:\OpenSSL-Win64\bin`).
2. Enter the following command to generate the server private key and public certificate:

```
openssl req -x509 -newkey rsa:2048 -nodes -keyout key.pem -out cert.pem -days 365 -nodes
```

   This generates `key.pem` and `cert.pem` files in the OpenSSL directory. The `key.pem` file is the server private key and the `cert.pem` file is the server public certificate in PEM (ASCII) format. The `key.pem` and `cert.pem` files must be available on server (OpenSSL) and `cert.pem` file must be available on client (ATWINC) to establish SSL connection between the server and client.

2.2 Downloading Public Server Certificate on ATWINC

The generated public certificate (`cert.pem`) must be downloaded on the ATWINC for SSL connection between the ATWINC (client) and OpenSSL (server).

The most known encodings for the X.509 digital certificates are PEM and DER formats. This implementation of the ATWINC supports only DER format. The following is the procedure to download the public server certificate on the ATWINC.

1. If the certificate is not in DER format, convert the certificate to DER format using the following command:

```
openssl x509 -outform der -in cert.pem -out cert.cer
```

   Note: `.cer` extension is used since the `root_certificate_downloader` script is available as part of the firmware update project that expects the `.cer` extension.
2. Upload the `cert.cer` file on the ATWINC using the firmware update project.
3. Copy and paste `cert.cer` file in `src\firmware\Tools\root_certificate_downloader\binary`.
4. Follow the steps mentioned in the Integrated Serial Flash Memory Download Procedure http://ww1.microchip.com/downloads/en/DeviceDoc/00002378A.pdf
   Note: If the certificate upload fails with “(ERROR) Root Certificate Flash is Full” error, this indicates that the allocated memory in SPI Flash on the ATWINC for certificates is full. Remove one or more certificates from `src\firmware\Tools\root_certificate_downloader\binary` and upload the upload the certificates again.

   The ATWINC is ready for SSL connection with OpenSSL server for receiving the host files.

3. Application Flow

The host file download application consists of bootloader code and main application code, which are described in the following sections.

3.1 Bootloader Code

A small piece of code is added to the main application to provide the ability to download updates, replacing the old firmware of the device. This code is often called as bootloader, as its role is to load a new program while booting. A bootloader always resides in memory to upgrade the device at any time.

The bootloader code implementation is provided as separate project "HOST_FIRMWARE_UPGRADE_BOOTLOADER_EXAMPLE "downloaded from ASF. The bootloader occupies the first part of the Flash memory called as Bootloader section. The size is defined by BOOTLOAD_SIZE which is configurable in `conf_bootloader.h` file.

The bootloader contains NVM driver to write the new image on Host MCU Flash. When the new image is available on ATWINC SPI Flash, it is required to include both the ATWINC1510 driver and SPI driver. The size of the bootloader code is limited by BOOTLOAD_SIZE which must be provided in linker script `(src\ASF\common\components\wifi\winc1500\host_firmware_upgrade\bootloader_example\samg55j19_samg_xplained_pro\samg55j19\gcc\flash.ld)`.

The following is an example where 0x00008000 is the BOOTLOAD_SIZE mentioned in flash.ld.

```
rom (rx) : ORIGIN = 0x00400000, LENGTH = 0x00008000
```

3.2 Main Application Code

The main user application resides on the application section which initiates the host image download. The example main application code, "HOST_FIRMWARE_UPGRADE_HOST_FIRMWARE_UPGRADE_EXAMPLE" can be opened from ASF. The `conf_bootloader.h` file configures the start address and size for all three sections mentioned in memory split section to be maintained same for bootloader code and application code.

The application code is not at the start address of the host MCU Flash, therefore the start address (APP_START) of the application code must be provided in linker script `(\src\ASF\common\components\wifi\winc1500\host_firmware_upgrade\host_firmware_upgrade_example\samg55j19_samg_xplained_pro\samg55j19\gcc\flash.ld)`.

The following is an example where 0x0040A000 is the APP_START address mentioned in flash.ld.

```
rom (rx) : ORIGIN = 0x0040A000, LENGTH = 0x00076000
```

3.2.1 Design and Architecture

The Host File Download feature has a simplistic design. It is assumed that there is no file system and only one file is stored at a time. This avoids a problem such as file fragmentation as a natural process of allocating and freeing memory sections of different sizes. Therefore, if a request for Host File Download is received, the request overwrites the content of the Flash memory reserved for file download (example, previous host OTA image).

If the file size exceeds the Flash storage, the ATWINC sends a process failure notification to the application. If the file download is completed successfully, the ATWINC sends a notification to the host that the download process is successful.

The host MCU then reads the file from the ATWINC's Flash and requests to erase the file when the operations of the host application that are associated with the file are complete.

When a file download is successfully completed, a handler is created and returned to the application, which is used to access the file in the ATWINC. There is only one handler at a time, therefore any older handler becomes invalid. This protects against cases where if a file read operation is attempted while a new file download is in progress, the ATWINC returns an error as the old file is invalidated internally in the ATWINC.

The ATWINC1510 has a section of 512 KB in size and starts at 0x80000 address, which is not used. Therefore, the host file download feature uses the free space. However, the Flash space is only available on the ATWINC1510, which has 8 Mbit of Flash, and the available space may get reduced as new features are added. The ATWINC1500 only has 4 Mbit of Flash memory and therefore it does not include the Host File Download feature.

Out of 512 KB of Flash in the ATWINC1510, the first sector 4 KB of size is used by the ATWINC to store the handler for the host file download feature. Therefore, starting from 0x81000 address, 508 KB size of Flash can be used by application to store the host file.

3.2.2 Application Interface APIs

The following APIs are used to perform the host file download from remote server.

3.2.2.1 NMI_API sint8 m2m_ota_init(tpfOtaUpdateCb pfOtaUpdateCb,tpfOtaNotifCb pfOtaNotifCb)

This API is a synchronous initialization function for the OTA layer which registers the update callback. This API is common for the host file download and the ATWINC firmware download. The following are the parameters of `sint8 m2m_ota_init` API.

- `pfOtaUpdateCb` is an OTA update callback function. For host file download, `pfHFDGetCb` callback is registered when `m2m_ota_host_file_get()` function is called after the completion of download with success status. The `pfOtaUpdateCb()` callback function is called when the download fails due to insufficient Flash memory. For example, when `m2m_ota_host_file_get()` API is called the download fails in the ATWINC1500 where the host file download feature is not supported. Refer to `tenuOtaUpdateStatus enum` for details on failure status.
- `pfOtaNotifCb` is an OTA notify callback function.

3.2.2.2 NMI_API sint8 m2m_ota_host_file_get(unsigned char *pcDownloadUrl, tpfFileGetCb pfHFDGetCb)

This API is called to download a file from a remote location and store the file in the ATWINC's Flash. The following are the parameters of `sint8 m2m_ota_host_file_get` API.

- `pcDownloadUrl` is an URL pointing to the remote file (HTTP/HTTPS only), providing a callback is mandatory.
- `pfHFDGetCb` is a pointer to a callback that are to be executed when the download is complete. In case of download failure, failure status message is printed. Refer to `tenuOtaUpdateStatus enum` for details on failure status. For example, if the device is not able to reach the server (OTA_STATUS_CONNECTION_ERROR) or server does not find the required file (OTA_STATUS_SERVER_ERROR) error status is returned.

3.2.2.3 NMI_API m2m_ota_host_file_read_spi(uint8 u8Handler, uint8 *pu8Buff, uint32 u32Offset, uint32 u32Size)

This API is used to read certain amount of bytes from a file in the ATWINC's Flash using SPI transfer. The following are the parameters of `m2m_ota_host_file_read_spi` API.

- `u8Handler` is a ID of the file to read from the ATWINC SPI Flash. This handler is provided when the file download finish callback is called.
  Note: There is one valid handler at all times and it changes as a new request for "file get" arrives by calling the `m2m_ota_host_file_get` function (when the new handler is passed to the application when the file download is complete). This avoid problems like concurrent requests from the application to download and read a file. Thus ensuring that the application does not read data from the Flash while a file is being downloaded to protect against reading invalid data.
- `pu8Buff` is a pointer to a buffer to store the data being read.
- `u32Size` is the amount of data to read (in Bytes).

It is required to set the ATWINC in Download mode before calling this API. Call, `m2m_wifi_deinit(NULL)` and `m2m_wifi_download_mode()` APIs before `m2m_ota_host_file_read_spi()` API.

Reading the data using this API is faster than by reading via the HIF (see 3.2.2.4 NMI_API sint8 m2m_ota_host_file_read_hif(uint8 u8Handler, uint32 u32Offset, uint32 u32Size, tpfFileReadCb pfHFDReadCb)). However, it blocks and requires the ATWINC to be set into the Download mode where the ATWINC acts as Flash and not as a Wi-Fi® device. After the read operation is complete, reset the ATWINC.

3.2.2.4 NMI_API sint8 m2m_ota_host_file_read_hif(uint8 u8Handler, uint32 u32Offset, uint32 u32Size, tpfFileReadCb pfHFDReadCb)

This API is an alternate for `m2m_ota_host_file_read_spi()` API to read certain amount of bytes from a file in ATWINC's Flash using HIF transfer. The following are the parameters of `sint8 m2m_ota_host_file_read_hif` API.

- `u8Handler` is the ID of the file that is to be read from the ATWINC SPI Flash.
- `u32Offset` is the offset from start of the file to read from the ATWINC SPI Flash (in bytes).
- `u32Size` is the amount of data to read (in bytes).
- `pfHFDReadCb` is a callback to be executed when the read operation is complete.

Using this API, up to 128 bytes of data can be transferred at a time. This limitation reduces the speed of the read due to the extra overhead. The HIF is non-blocking, therefore the application can continue execution as normal and is interrupted only when data is available. It does not require the ATWINC to reset or set in the Download mode, as in the case of reading the file via SPI (`m2m_ota_host_file_read_spi`).

When calling this API while specifying a size more than 128 bytes, the read is limited to the first 128 bytes starting at the read offset. It is recommended that a read for sizes above 128 bytes is performed in multiple steps, using the callback to advance the offset and request another read of 128 bytes (or less) each time.

3.2.2.5 NMI_API sint8 m2m_ota_host_file_erase(uint8 u8Handler, tpfFileEraseCb pfHFDEraseCb)

This API erases the existing file. Although this operation is not required to free up Flash space, it is useful for cases in which the ATWINC stores sensitive information from the host that must be deleted after use. The following are the parameters of `sint8 m2m_ota_host_file_erase` API.

- `u8Handler` is the ID of the file which is to be erased.
- `pfHFDEraseCb` is a pointer that callbacks to execute when the file erase in the ATWINC is complete.

3.2.2.6 NMI_API sint8 m2m_ota_abort(void)

This API requests the interruption of the file download and discards the incomplete file. This API is common for host file download or ATWINC's firmware download.

3.2.3 Concatenation of Bootloader and Main Application Image

The factory image shipped on the host MCU must contain both bootloader and application images. Since bootloader and application are of two different projects, the compilation output provides two images. The bootloader image is stored in the bootloader section and application image is stored in the application section. It is required to merge these two images as one to be programmed on the Host MCU. The user can merge two images using the SRecord package downloaded from http://srecord.sourceforge.net/.

The following is the procedure to merge the bootloader image and application image.

1. After installing SRecord, open CMD prompt and navigate to the directory, where SRecord is installed.
2. Enter the following command to concatenate two images:

```
srec_cat.exe boot.hex -Intel application.hex -Intel -o combined_image.hex -Intel
```

Where,

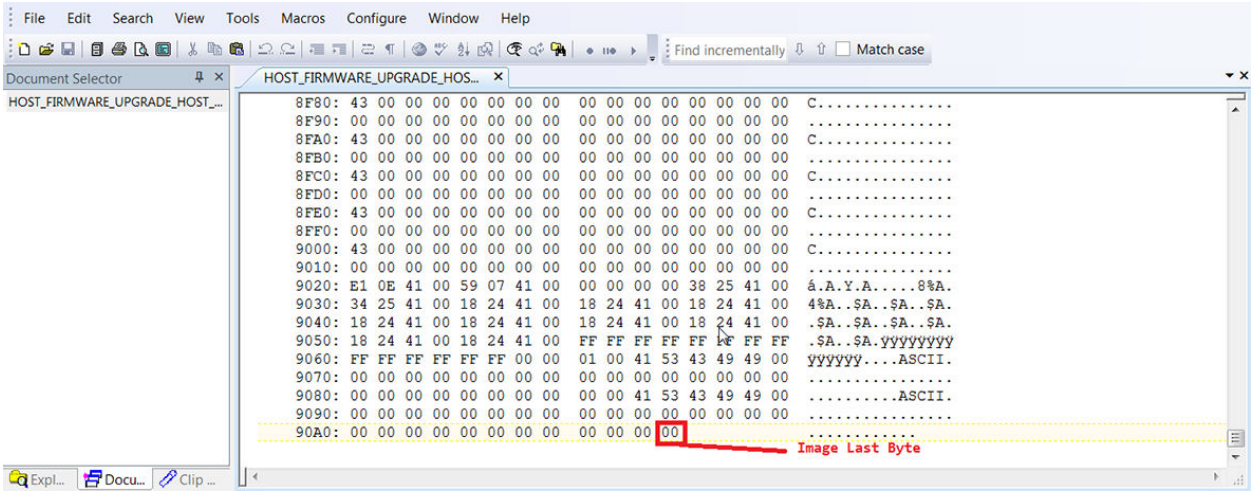
- `boot.hex` is a hex image generated by compiling the bootloader code, which is the actual `.hex` file name of the bootloader code.
- `application.hex` is a hex image generated by compiling the main application code, which is the actual `.hex` file name of the main application code.
- `combined_image.hex` is a concatenated image of the bootloader and application images to program on the host MCU.

3.2.4 Generating Demo Host OTA Upgrade Image

The OTA upgradable image must be in .bin format. For the data integrity check the .bin format must also have the CRC check. The CRC16 for the complete image is calculated and added at the end of the image data (two bytes are left free and CRC is added after that two bytes). `srec_cat` functionality is used for this purpose (http://srecord.sourceforge.net/).

`ota_image_without_crc.bin` can be opened using textpad to see the last byte address as shown:

Figure 3-1. Using Textpad to See the Last Byte Address



The following is an example command to generate CRC16 and append at the address 0x90AE. Here, 0x90AB is the last image byte address, 0x90AC, and 0x90AD are left free, and two byte CRC is added at the addresses 0x90AE, and 0x90AF.

```
srec_cat.exe ota_image_without_crc.bin -binary -Cyclic_Redundancy_Check_16_Little_Endian
0x90AE -o host_mcu_ota.bin -binary
```

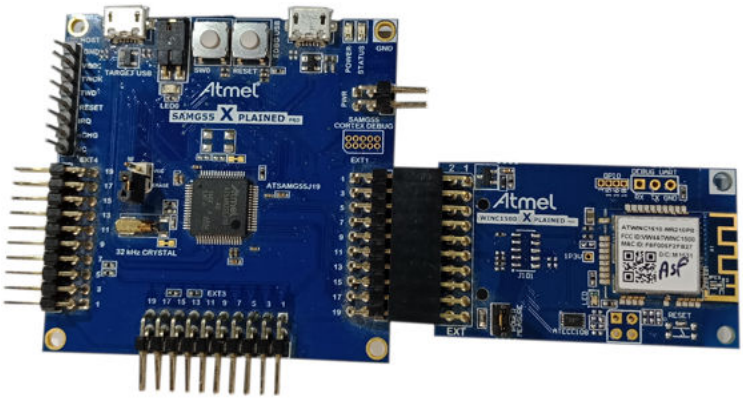
Note: This method is used for demo purposes only. This method of checking the validity of the file is an example and Mirochip does not enforce encapsulation of the file in a particular format. If the user prefers a different format they can use it or if the application already has the hash, then it can download the file and match with the calculated hash. The user can implement their own method of data integrity mechanism.
The OTA image must not contain any bootloader image. It must be the main application code started at APP_START address.

4. Demo Flow

Perform the following steps to run the demo application.

1. Connect SAM G55 XPro (ATWINC1510 XPro is connected on the EXT1 header) to the PC using a micro USB cable, as shown in the following figure.

   Figure 4-1. SAM G55 XPro Connected with ATWINC1510 XPro

2. Follow the steps in Generating Custom Server Certificate and Downloading Public Server Certificate on ATWINC to download the OpenSSL server's public certificate on ATWINC1510 using Firmware Update Project.
3. Connect the PC Wi-Fi with the Wi-Fi router.
4. Navigate to the directory where the OpenSSL is installed (for example: `C:\OpenSSL-Win64\bin`).
   The OpenSSL server certificate and private key generated must be available in the OpenSSL directory.
5. Open the "HOST_FIRMWARE_UPGRADE_BOOTLOADER_EXAMPLE" application from Atmel Studio through *File* > *New* > *Example Project*.
   Note: This example build procedure is developed using the SAM G55 XPro board, which is also valid for other supported hardware platforms.
6. Build the program.
7. Open the "HOST_FIRMWARE_UPGRADE_HOST_FIRMWARE_UPGRADE_EXAMPLE" application from Atmel Studio through *File* > *New* > *Example Project*.
8. Add the following code in `main.h` file to configure the Access Point (AP) information to get connected.

   ```
   /** Wi-Fi Settings */
   #define MAIN_WLAN_SSID "DEMO_AP" /* < Destination SSID */
   #define MAIN_WLAN_AUTH M2M_WIFI_SEC_WPA_PSK /* < Security manner */
   #define MAIN_WLAN_PSK "12345678" /* < Password for Destination SSID */
   ```

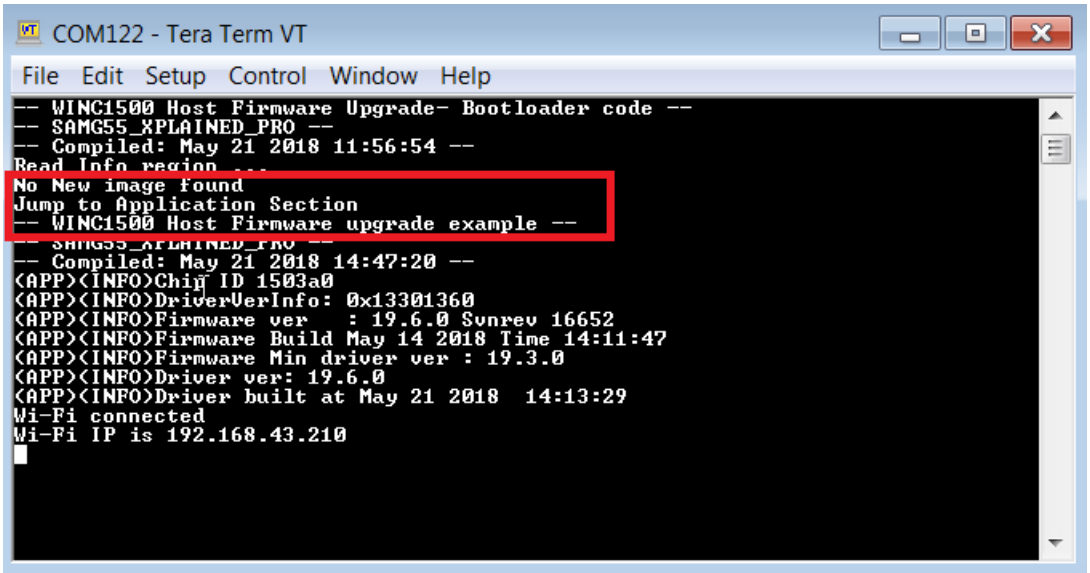
   Note: The OpenSSL server and the ATWINC1510 must be connected to the same network for the OpenSSL server based demo.
9. Configure the HTTPS Server (OpenSSL) IP address, port number and host OTA file name.
   Note: Ensure that the file name generated in Step 16 and the host OTA file name are same.

   ```
   /** Content URI for download. */
   #define MAIN_HOST_OTA_URL https://192.168.43.34:4443/host_mcu_ota.bin
   ```

10. Build the program.

11. Combine the bootloader image generated in Step 6 and main application image generated in Step 10 as per Concatenation of Bootloader and Main Application Image.
12. Download the combined image into the SAM G55 XPRO board.

Figure 4-2. Console Log for Combined Factory Image



13. Start the application. The device enters the bootloader section and finds there is no new upgradable image, and then jumps to the main application and awaits for the user to start the host image download.
14. The OTA image upgraded on the host MCU must be available in the OpenSSL server (for example: `C:\OpenSSL-Win64\bin`). For demo, add a print statement in `main()` application code to differentiate the OTA image from factory image. For example, add the below printf statement in `main()` and compile the project.

```
printf("--New Host OTA Firmware Image--\r\n");
```

Note: The generated image does not have integrity check (CRC).

15. To generate the OTA image with CRC, copy the `.bin` generated in the above step to `srec installer` folder and generate the demo host OTA upgrade image. For more details, see Generating Demo Host OTA Upgrade Image.
16. Copy the generated OTA image to the directory where OpenSSL is installed (for example: `C:\OpenSSL-Win64\bin`).
17. Open the Command prompt and start OpenSSL as HTTPS server, using the following command:

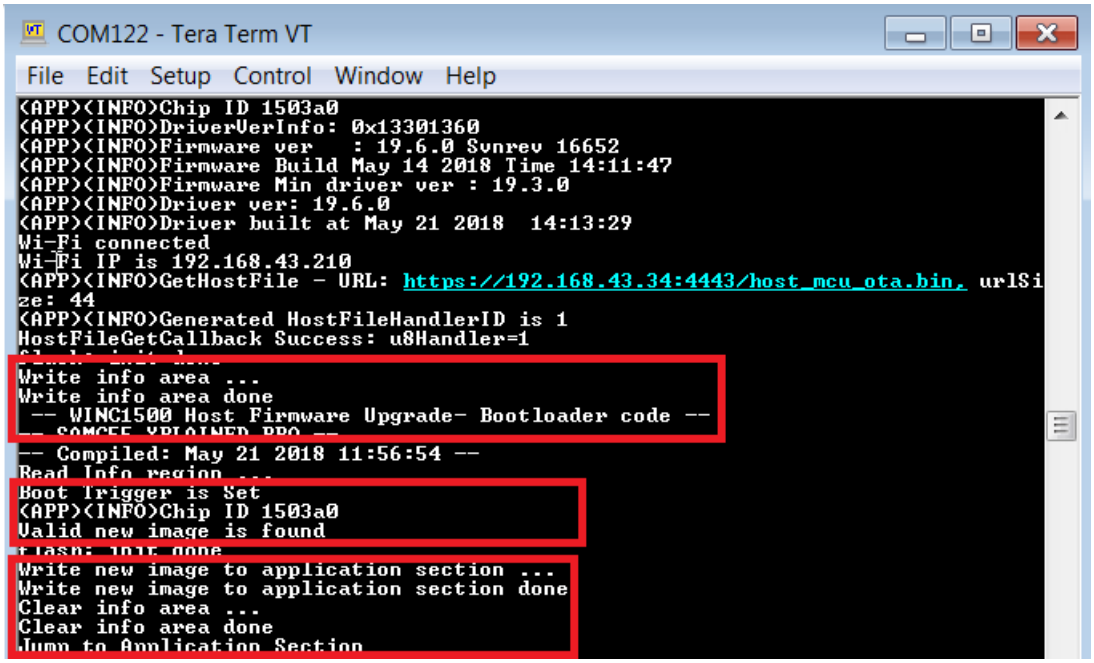
```
openssl s_server -key key.pem -cert cert.pem -accept 4443 -WWW
```

Note:

- The `key.pem` and `cert.pem` files are generated in Generating Custom Server Certificate
- 4443 is the port number

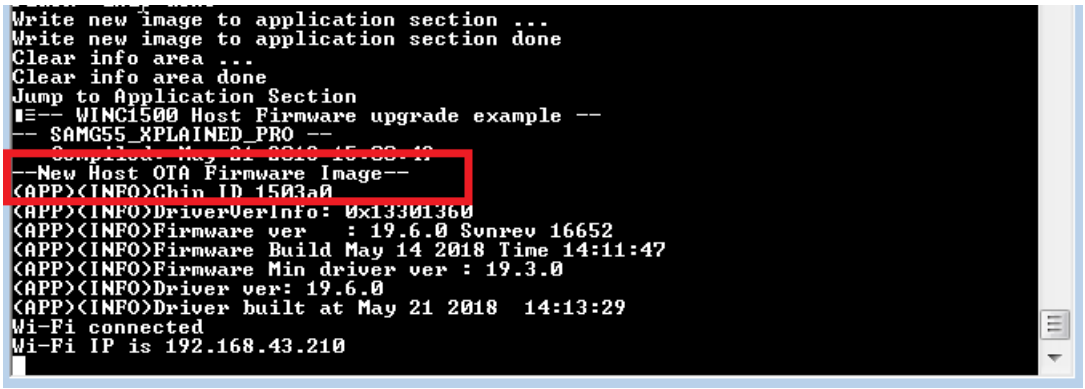
18. Press SW0 button on the SAM G55 XPro board. The ATWINC1510 starts receiving new host image from the OpenSSL server.

Figure 4-3. Host New OTA Image Upgrade Progress



19. When the complete image is received, the host MCU writes the info region and resets. On Reset, the bootloader writes the new image on the application section and the new application starts executing.

Figure 4-4. Switched to Host New OTA Image



5. Host File Download

The APIs provided in 3.2.2 Application Interface APIs are common for host image download and any host file download. The design and architecture is also same for both the cases.

For host file download, `m2m_ota_host_file_read_spi()` or `m2m_ota_host_file_read_hif()` API must be used to read the downloaded file from the ATWINC SPI Flash. The completion of file download is notified through the callback registered in `m2m_ota_host_file_get()` API, which initiates the file read from the ATWINC Flash using `m2m_ota_host_file_read_spi ()` or `m2m_ota_host_file_read_hif()` API by passing required arguments. For more details on application interface APIs, see 3.2.2 Application Interface APIs.

6. Limitations

1. Out of 512 KB of Flash in the ATWINC1510, the first sector 4 KB of size is used by the ATWINC for storing the handler for the host file download feature. Therefore, only 508 KB size of Flash can be used by the application to store the host file.
2. The ATWINC1500 only has 4 Mbit of Flash memory and therefore this feature is not supported for the ATWINC1500.
3. There is no file system and only one file is stored at a time. When the get file is called again, the previously stored file is erased and new file download is initiated.
4. The ATWINC OTA firmware download and the Host OTA file download cannot be started at the same time.
5. The ATWINC interprets the `404 Not Found` error when the application attempts to download a broken or dead link and provides the `OTA_STATUS_SERVER_ERROR` error status. The ATWINC does not interpret if the server responds with some other message for a broken link. The ATWINC downloads the error message into SPI Flash and indicates download success to host and the application checks for the valid file.

7. Document Revision History

Revision	Date	Section	Description
B	02/2019	• 1.3 Prerequisites • 3.2.1 Design and Architecture • 3.2.2.1 NMI_API sint8 m2m_ota_init(tpfOtaUpdateCb pfOtaUpdateCb,tpfOtaNotifCb pfOtaNotifCb) • 6. Limitations	Updated the kit name
A	10/2018	Document	Initial release

The Microchip Web Site

Microchip provides online support via our web site at http://www.microchip.com/. This web site is used as a means to make files and information easily available to customers. Accessible by using your favorite Internet browser, the web site contains the following information:

- **Product Support** – Data sheets and errata, application notes and sample programs, design resources, user's guides and hardware support documents, latest software releases and archived software
- **General Technical Support** – Frequently Asked Questions (FAQ), technical support requests, online discussion groups, Microchip consultant program member listing
- **Business of Microchip** – Product selector and ordering guides, latest Microchip press releases, listing of seminars and events, listings of Microchip sales offices, distributors and factory representatives

Customer Change Notification Service

Microchip's customer notification service helps keep customers current on Microchip products. Subscribers will receive e-mail notification whenever there are changes, updates, revisions or errata related to a specified product family or development tool of interest.

To register, access the Microchip web site at http://www.microchip.com/. Under "Support", click on "Customer Change Notification" and follow the registration instructions.

Customer Support

Users of Microchip products can receive assistance through several channels:

- Distributor or Representative
- Local Sales Office
- Field Application Engineer (FAE)
- Technical Support

Customers should contact their distributor, representative or Field Application Engineer (FAE) for support. Local sales offices are also available to help customers. A listing of sales offices and locations is included in the back of this document.

Technical support is available through the web site at: http://www.microchip.com/support

Microchip Devices Code Protection Feature

Note the following details of the code protection feature on Microchip devices:

- Microchip products meet the specification contained in their particular Microchip Data Sheet.
- Microchip believes that its family of products is one of the most secure families of its kind on the market today, when used in the intended manner and under normal conditions.
- There are dishonest and possibly illegal methods used to breach the code protection feature. All of these methods, to our knowledge, require using the Microchip products in a manner outside the operating specifications contained in Microchip's Data Sheets. Most likely, the person doing so is engaged in theft of intellectual property.
- Microchip is willing to work with the customer who is concerned about the integrity of their code.

- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of their code. Code protection does not mean that we are guaranteeing the product as “unbreakable.”

Code protection is constantly evolving. We at Microchip are committed to continuously improving the code protection features of our products. Attempts to break Microchip’s code protection feature may be a violation of the Digital Millennium Copyright Act. If such acts allow unauthorized access to your software or other copyrighted work, you may have a right to sue for relief under that Act.

Legal Notice

Trademarks

ISBN: 978-1-5224-4112-0

Quality Management System Certified by DNV

ISO/TS 16949

Microchip received ISO/TS-16949:2009 certification for its worldwide headquarters, design and wafer fabrication facilities in Chandler and Tempe, Arizona; Gresham, Oregon and design centers in California and India. The Company's quality system processes and procedures are for its PIC® MCUs and dsPIC® DSCs, KEELOQ® code hopping devices, Serial EEPROMs, microperipherals, nonvolatile memory and analog products. In addition, Microchip's quality system for the design and manufacture of development systems is ISO 9001:2000 certified.

Worldwide Sales and Service

AMERICAS	ASIA/PACIFIC	ASIA/PACIFIC	EUROPE
Corporate Office 2355 West Chandler Blvd. Chandler, AZ 85224-6199 Tel: 480-792-7200 Fax: 480-792-7277 Technical Support: http://www.microchip.com/support Web Address: www.microchip.com **Atlanta** Duluth, GA Tel: 678-957-9614 Fax: 678-957-1455 **Austin, TX** Tel: 512-257-3370 **Boston** Westborough, MA Tel: 774-760-0087 Fax: 774-760-0088 **Chicago** Itasca, IL Tel: 630-285-0071 Fax: 630-285-0075 **Dallas** Addison, TX Tel: 972-818-7423 Fax: 972-818-2924 **Detroit** Novi, MI Tel: 248-848-4000 **Houston, TX** Tel: 281-894-5983 **Indianapolis** Noblesville, IN Tel: 317-773-8323 Fax: 317-773-5453 Tel: 317-536-2380 **Los Angeles** Mission Viejo, CA Tel: 949-462-9523 Fax: 949-462-9608 Tel: 951-273-7800 **Raleigh, NC** Tel: 919-844-7510 **New York, NY** Tel: 631-435-6000 **San Jose, CA** Tel: 408-735-9110 Tel: 408-436-4270 **Canada - Toronto** Tel: 905-695-1980 Fax: 905-695-2078	**Australia - Sydney** Tel: 61-2-9868-6733 **China - Beijing** Tel: 86-10-8569-7000 **China - Chengdu** Tel: 86-28-8665-5511 **China - Chongqing** Tel: 86-23-8980-9588 **China - Dongguan** Tel: 86-769-8702-9880 **China - Guangzhou** Tel: 86-20-8755-8029 **China - Hangzhou** Tel: 86-571-8792-8115 **China - Hong Kong SAR** Tel: 852-2943-5100 **China - Nanjing** Tel: 86-25-8473-2460 **China - Qingdao** Tel: 86-532-8502-7355 **China - Shanghai** Tel: 86-21-3326-8000 **China - Shenyang** Tel: 86-24-2334-2829 **China - Shenzhen** Tel: 86-755-8864-2200 **China - Suzhou** Tel: 86-186-6233-1526 **China - Wuhan** Tel: 86-27-5980-5300 **China - Xian** Tel: 86-29-8833-7252 **China - Xiamen** Tel: 86-592-2388138 **China - Zhuhai** Tel: 86-756-3210040	**India - Bangalore** Tel: 91-80-3090-4444 **India - New Delhi** Tel: 91-11-4160-8631 **India - Pune** Tel: 91-20-4121-0141 **Japan - Osaka** Tel: 81-6-6152-7160 **Japan - Tokyo** Tel: 81-3-6880- 3770 **Korea - Daegu** Tel: 82-53-744-4301 **Korea - Seoul** Tel: 82-2-554-7200 **Malaysia - Kuala Lumpur** Tel: 60-3-7651-7906 **Malaysia - Penang** Tel: 60-4-227-8870 **Philippines - Manila** Tel: 63-2-634-9065 **Singapore** Tel: 65-6334-8870 **Taiwan - Hsin Chu** Tel: 886-3-577-8366 **Taiwan - Kaohsiung** Tel: 886-7-213-7830 **Taiwan - Taipei** Tel: 886-2-2508-8600 **Thailand - Bangkok** Tel: 66-2-694-1351 **Vietnam - Ho Chi Minh** Tel: 84-28-5448-2100	**Austria - Wels** Tel: 43-7242-2244-39 Fax: 43-7242-2244-393 **Denmark - Copenhagen** Tel: 45-4450-2828 Fax: 45-4485-2829 **Finland - Espoo** Tel: 358-9-4520-820 **France - Paris** Tel: 33-1-69-53-63-20 Fax: 33-1-69-30-90-79 **Germany - Garching** Tel: 49-8931-9700 **Germany - Haan** Tel: 49-2129-3766400 **Germany - Heilbronn** Tel: 49-7131-67-3636 **Germany - Karlsruhe** Tel: 49-721-625370 **Germany - Munich** Tel: 49-89-627-144-0 Fax: 49-89-627-144-44 **Germany - Rosenheim** Tel: 49-8031-354-560 **Israel - Ra'anana** Tel: 972-9-744-7705 **Italy - Milan** Tel: 39-0331-742611 Fax: 39-0331-466781 **Italy - Padova** Tel: 39-049-7625286 **Netherlands - Drunen** Tel: 31-416-690399 Fax: 31-416-690340 **Norway - Trondheim** Tel: 47-72884388 **Poland - Warsaw** Tel: 48-22-3325737 **Romania - Bucharest** Tel: 40-21-407-87-50 **Spain - Madrid** Tel: 34-91-708-08-90 Fax: 34-91-708-08-91 **Sweden - Gothenberg** Tel: 46-31-704-60-40 **Sweden - Stockholm** Tel: 46-8-5090-4654 **UK - Wokingham** Tel: 44-118-921-5800 Fax: 44-118-921-5820