



AN1431: SiWx917 SoC Firmware Update

Application Note

This application note describes the firmware update procedure for the SiWx917 SoC (SiWG917). This document covers the firmware load and update process, firmware update mechanisms.

KEY POINTS

- Block Diagram
- Firmware Load, Update, and Update Mechanisms
- Secure Firmware Update

Table of Contents

1. Introduction	3
2. Block Diagram	4
2.1 Firmware File Format (RPS)	4
2.1.1 RPS Header Format.	5
3. SiWG917 Firmware Load and Update Process.	7
4. SiWG917 Firmware Update Mechanisms	9
4.1 Firmware Update via OTA	9
4.1.1 Firmware Update via HTTP/S	10
4.1.2 Firmware Update via M4 as Host	12
4.2 Firmware Update via Bootloader	14
4.2.1 Firmware Update via Kermit	15
4.2.2 Firmware Update via Simplicity Commander Tool or CLI	18
4.2.3 Firmware Update via External Host Interfaces - SPI/SDIO (ISP Mode)	18
5. Secure Firmware Update	21
6. Combined Image	24
6.1 Importance of Secure Over-the-Air Updates with Combined Images	24
7. Guidelines and Recommendations	25
8. References	26
9. Troubleshooting	27
10. Revision History.	28

1. Introduction

The SiWG917 includes two processors: Network Wireless Processors (NWP) and an ARM® Cortex® M4 Processor. All the networking and wireless stacks run on independent threads of the NWP.

In addition, in adherence to the Trusted Execution Environment architecture, the NWP subsystem also acts as the secure processing domain and takes care of secure boot and secure firmware update. The Cortex-M4 is dedicated to peripheral and application-related processing.

Firmware is the software that is embedded into a hardware device. It contains a set of commands that control the behavior of NWP. Whenever a new firmware version is available, it is recommended that you update your devices to the latest version. The complete details about the latest firmware will be available in the Release Notes (shared as part of release package), which will help you to decide whether to update to the new firmware or not. From factory all the parts are shipped without firmware, you need to take the latest NWP firmware and update it on the SiWG917 device. There are two firmware images present in the SiWG917 WiSeConnect release - Lite (For 4 MB flash OPNs) and Standard (For 8 MB Flash OPNs).

The firmware in the SiWG917 device can be updated using the following mechanisms:

1. **Firmware update via Over-The-Air (OTA)** - In this mechanism, the firmware in the device can be updated by the following methods:
 - **HTTP/S:** The firmware is updated by downloading the firmware file from a remote HTTP/S or cloud server over Wi-Fi. The firmware file is directly downloaded to NWP flash location.
 - **M4 as Host:** Using host interfaces – SPI/UART/SDIO to access external storage or a remote TCP server via Wi-Fi or BLE, the firmware is reaching in chunks to M4. The user can choose to send the firmware to NWP Bootloader for upgrade or save in the external flash as per their requirements.
2. **Firmware update via Bootloader** - In this mechanism, the firmware in the device can be updated by the following methods.
 - **Kermit:** Firmware is updated using the Kermit protocol in a serial terminal like Tera Term running in a Windows/Linux PC. The connected to the device through UART interface in ISP mode.
 - **Simplicity Commander Tool/ Command Line Interface (CLI):** Using the Simplicity Commander tool or by using the CLI commands, the firmware is updated.
 - **External Host Interfaces — SPI/SDIO (ISP Mode):** The firmware is updated using the SPI/ SDIO host interfaces while keeping the SiWG917 in ISP mode.

Note: SiWG917 has both the Secure and Non-Secure Firmware update options. The above mechanisms are the same for both secure and non-secure options, except that the firmware does security-related integrity checks before loading the device with the new firmware.

2. Block Diagram

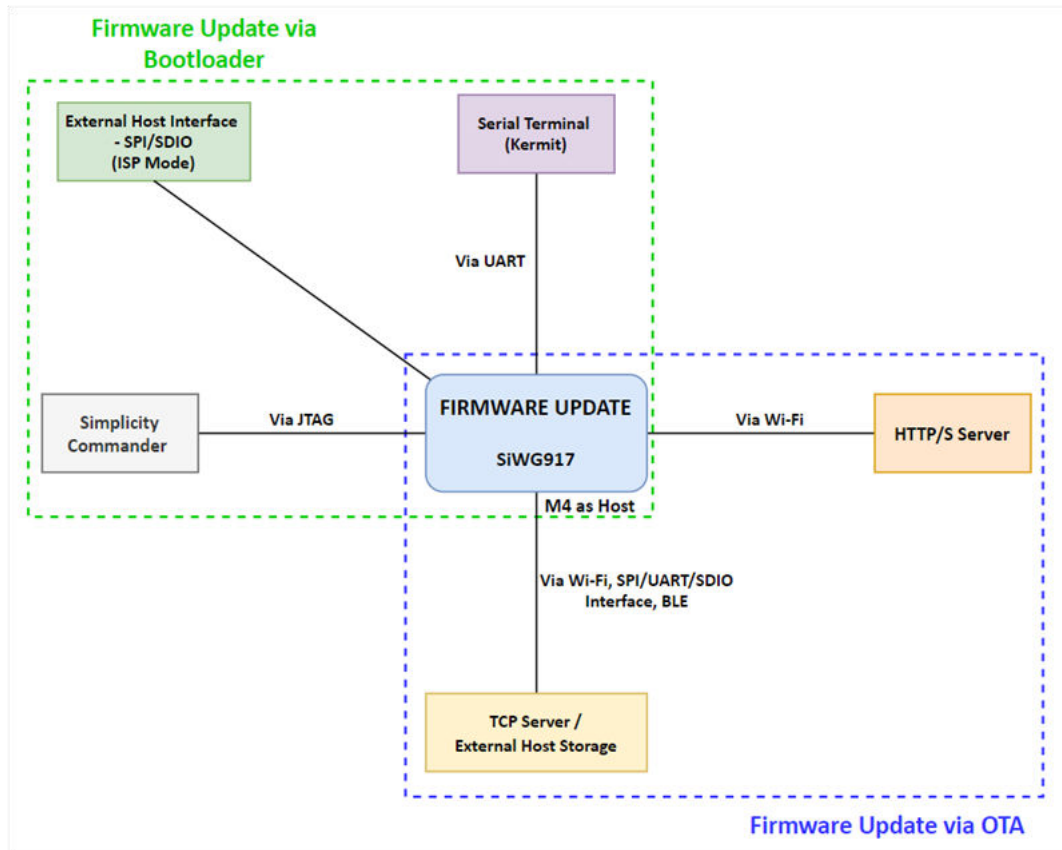


Figure 2.1. Firmware Update Mechanisms

2.1 Firmware File Format (RPS)

The NWP Bootloader uses a proprietary format for NWP and M4 its upgrade images, called RPS . These files have extension .rps. The RPS Format is a binary executable format understood by the Bootloader to perform the required integrity and authenticity checks and load and execute the application.

The Firmware Image in the RPS format includes an RPS header, boot descriptors, the application's binary image, and an optional trailer (digital signature).

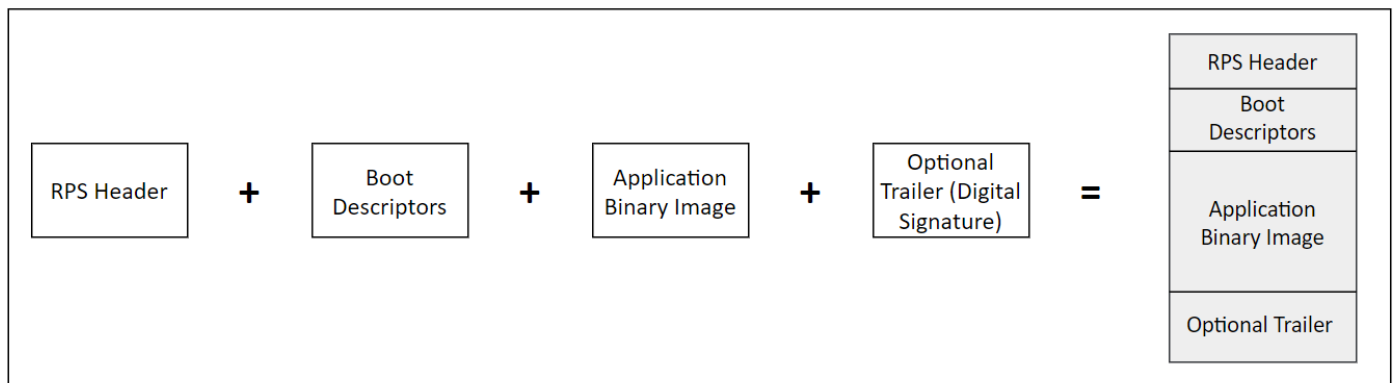


Figure 2.2. RPS Format

2.1.1 RPS Header Format

The following image shows the RPS header format with the fields present in it.

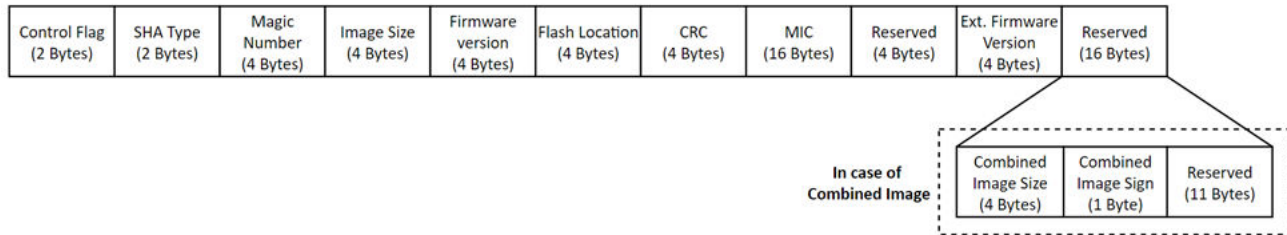


Figure 2.3. RPS Header Format

Table 2.1. RPS Header Fields

S.N.	Field	Size (in bytes)	Description
1	Control flag	2	Bit map which Indicates image information • BIT(0): • 0 - NWP processor image • 1 - MCU image • BIT(1): • 0 - Image is not encrypted • 1 - Image is encrypted • BIT(2): • 0 - CRC based integrity check • 1 - MIC based integrity check • BIT(3): • 0 - Digitally not signed • 1 - Digitally Signed (in this case the Digital signature is attached in the Trailer section of the image)
2	SHA Type	2	Represents the SHA size used to compute the digest for the digital signature • 1 - SHA_256 • 2 - SHA_384 • 3 - SHA_512
3	Magic Word	4	0x900D900D
4	Image Size	4	Size of the image
5	Firmware Version	4	Firmware version (Major number + Minor number + Security version number + Build number)
6	Flash Location	4	Address location in flash where the image is stored
7	CRC	4	CRC of the image
8	MIC	16	MIC of the image
9	Reserved	4	Reserved
10	Ext. Firmware version	4	Additional details of FW version such as patch number, Customer ID, rom_id and chip ID
11	Reserved	16	Reserved

Note: In case of combined image (NWP + M4) – the Reserved (16 Bytes) field present at the end of the RPS header will be replaced with the following.

S.N.	Field	Size (in bytes)	Description
11	Combine Image Size	4	Size of the combined image (NWP + M4)
12	Combined Image Sign	1	Bitmap which indicates whether the combined image is signed. BIT(0): • 0 - Digitally not signed • 1 - Digitally signed
13	Reserved	11	Reserved

3. SiWG917 Firmware Load and Update Process

The following figures show the NWP and M4 firmware load and update processes for the SiWG917 device. The firmware load process loads firmware onto the SiWG917 device for the first time in the case of new devices. Firmware update process updates the SiWG917 device with the latest firmware by replacing the firmware already existing in the device. The following images show the firmware load and update process for a 8 MB common flash SiWG917 device. Following steps need to be followed:

1. To update an existing firmware or a device without a firmware, download the new firmware file to the device's flash memory from the host (MCU/PC) through any host interface or through the OTA process.
2. After reboot, the current firmware is replaced by the new firmware in flash memory, and the device is updated with the new firmware.

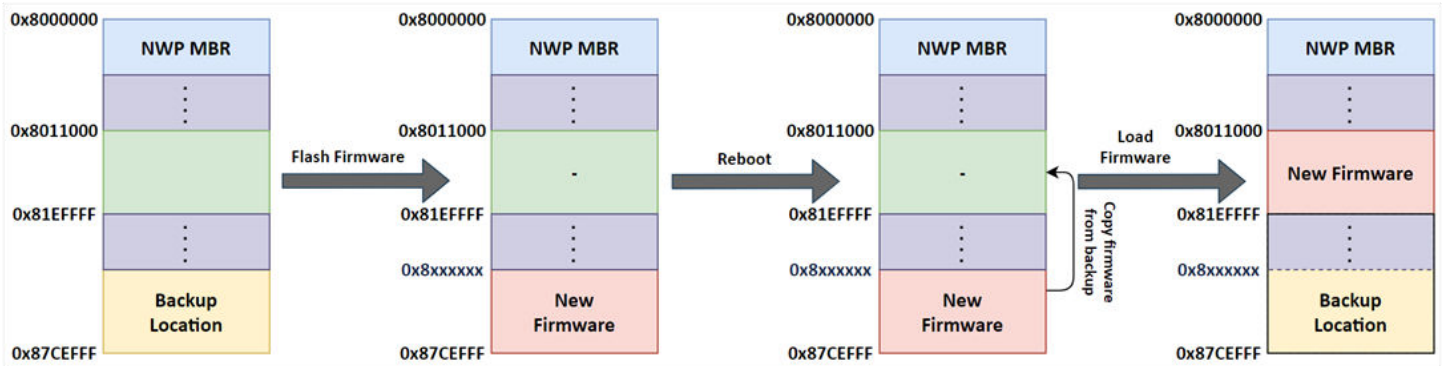


Figure 3.1. NWP Firmware Load Process (on Empty Flash)

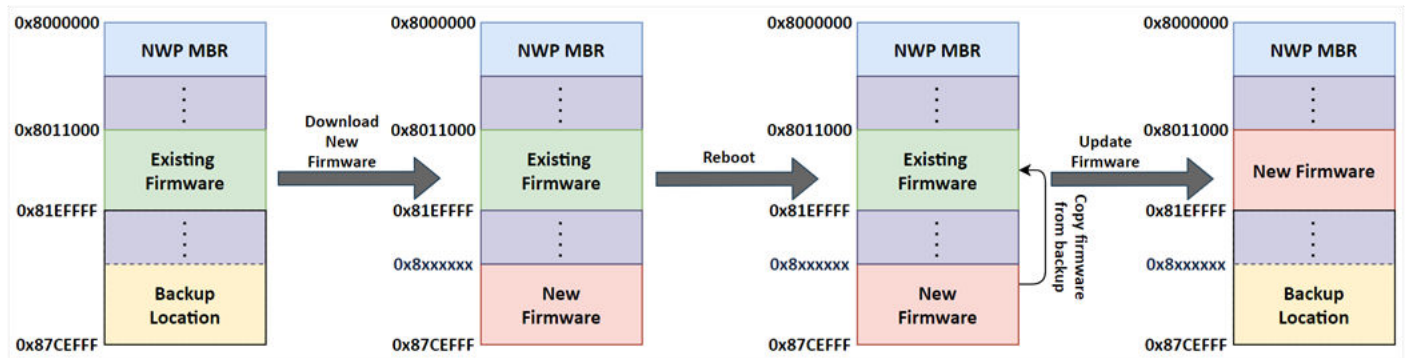


Figure 3.2. NWP Firmware Update Process

A similar process is followed to load the M4 application, the below figure shows the M4 application loading process.

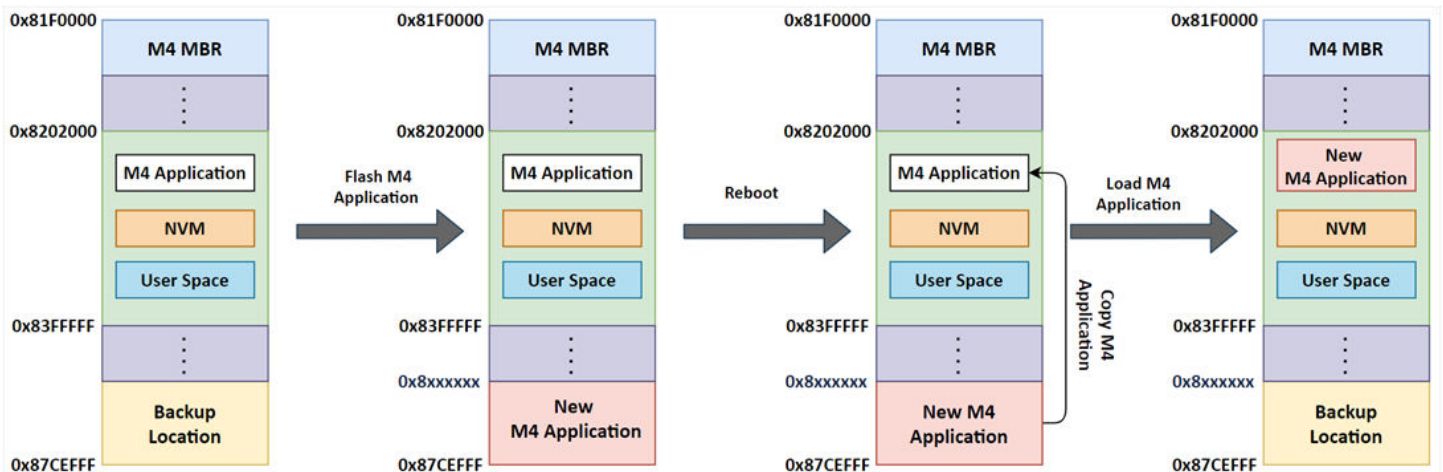


Figure 3.3. M4 Application Update Process

Note:

- The memory addresses shown in the above diagrams are only for reference purposes.
- You do not have control over determining the start address of the backup location.
- The NWP Bootloader is located in the Read Only memory (ROM), you cannot modified it.
- During the SiWG917 firmware upgrade process, the image is initially placed in a backup location. After passing integrity checks, it is then copied to the primary firmware location. There would be no corruption or loss of the firmware image at any step in this process if a power loss occurs. A working firmware image will remain in the primary location, and the upgraded image will stay in the backup location. Upon the next power-up, the bootloader will complete the firmware upgrade process.
- In case of 4MB flash, combined image firmware update is not possible.

4. SiWG917 Firmware Update Mechanisms

Firmware update mechanisms are categorized into two:

1. Firmware Update via OTA
2. Firmware Update via Bootloader

The following subsections provide descriptions for both.

4.1 Firmware Update via OTA

The following steps are executed during firmware update Over The Air:

1. The current M4 application receives and sends the new firmware image (RPS) in backup location.
2. The existing firmware does the integrity check of the new firmware based on the RPS header configuration.
3. If the integrity is verified and valid, then a soft reset is given in the current M4 application.
4. The device boots into the Bootloader.
5. The Bootloader finds that a new firmware is available in the backup location and again does an integrity check using CRC, MIC, or Signature-based check (If security is enabled), depending on the control flags of the RPS and the MBR configuration.
6. After the image is verified, the Bootloader transfers the image from download location to the execution/target location. If the image is encrypted, decryption is performed during the process of transferring and parsing the RPS file.

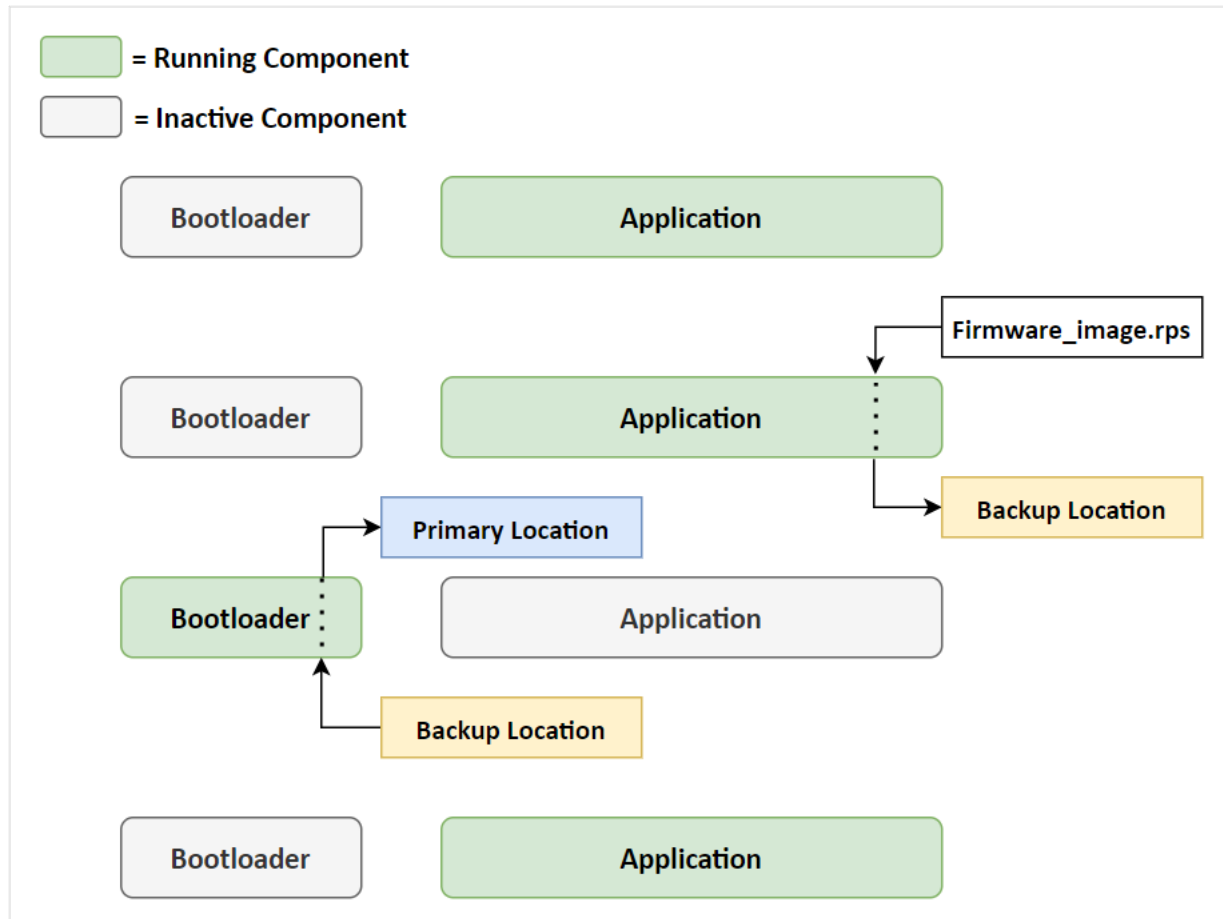


Figure 4.1. Firmware Update via OTA

There are two mechanisms to update Firmware via OTA.

1. Firmware Update via HTTP/S
2. Firmware Update via M4 as Host

The following subsections explain the mechanisms in detail.

4.1.1 Firmware Update via HTTP/S

The SiWG917 connects to access point and configure as a HTTP/S client and establishes connection with an HTTP/S server or the cloud storage server. After a successful HTTP/S connection, the SiWG917 sends a firmware file request to the remote server and the server responds with the firmware file.

The server-transferred firmware file gets loaded/updated in the SiWx91x module flash memory when the respective APIs are called. The following figure shows the firmware update process in the form of an image and flow chart respectively.

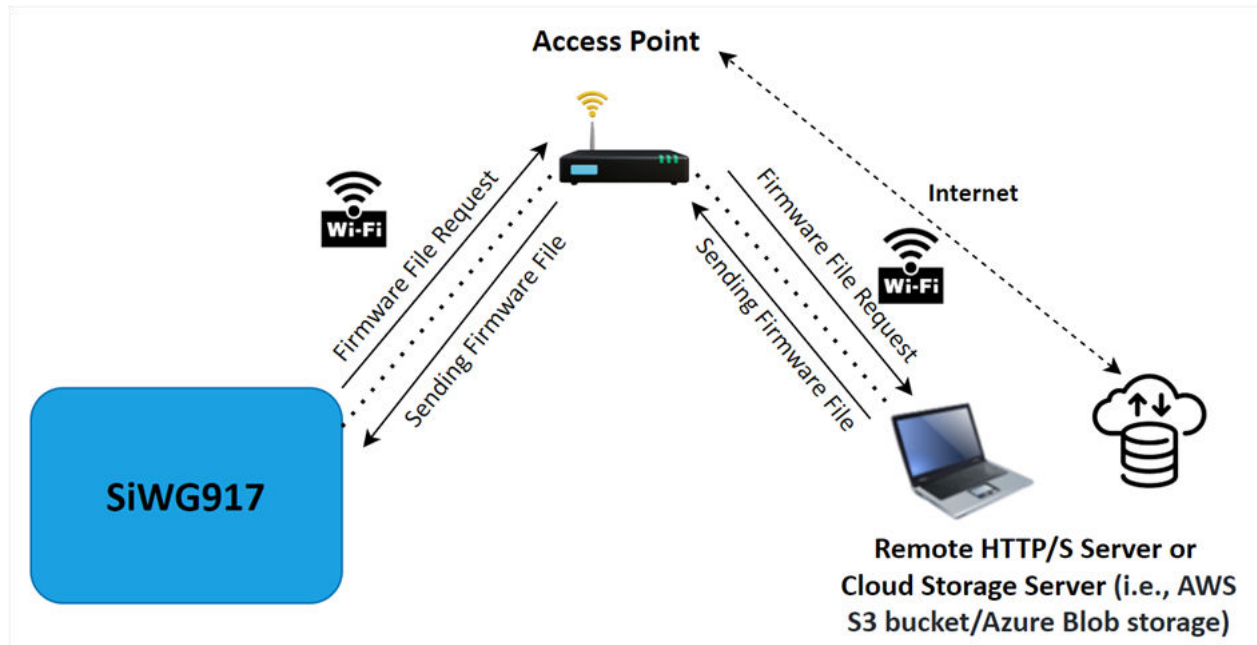


Figure 4.2. Firmware Update via HTTPS (OTA)

Firmware Update APIs

The following APIs are used for downloading and updating the NWP firmware and M4 application.

- **Download Firmware:**

sl_si91x_http_otaf() - the API used for downloading is the same for NWP and M4

- **Update Firmware:**

- For NWP Firmware – First *sl_net_deinit()* and next, *sl_net_init()* should be called
- For M4 Application – *sl_si91x_soc_soft_reset()*

The following flowchart shows the steps involved in the firmware update process via HTTPS.

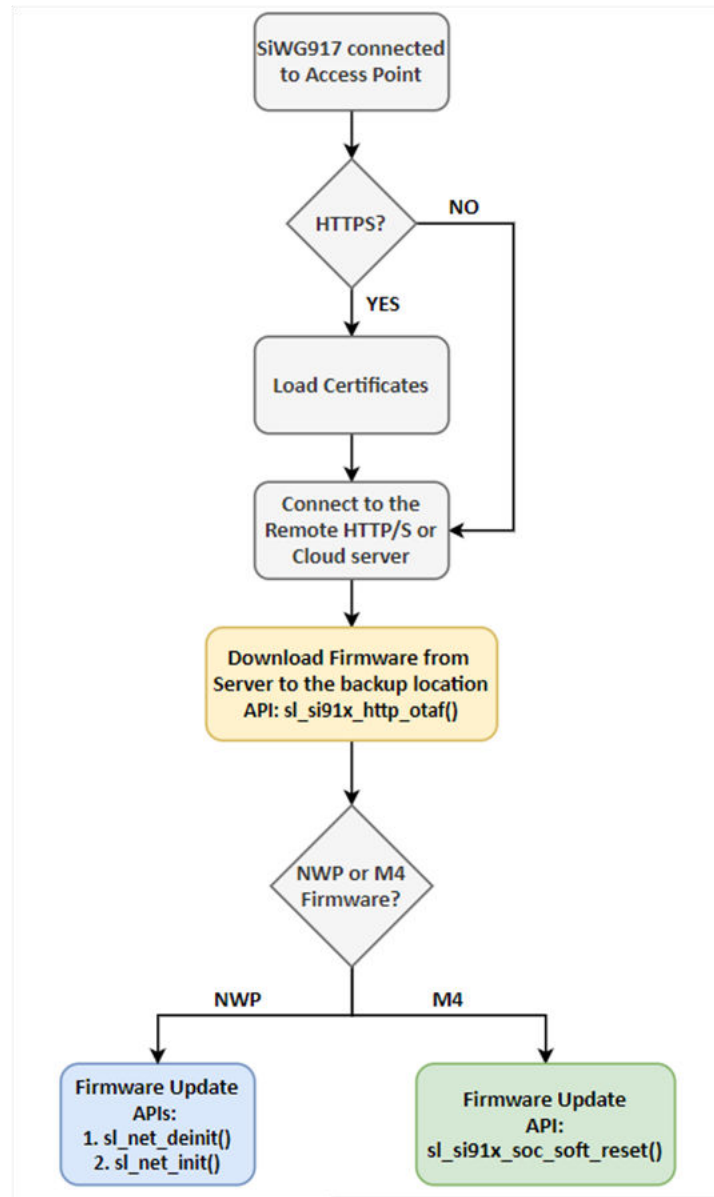


Figure 4.3. Firmware Update via HTTPS (OTA) – Flowchart

4.1.2 Firmware Update via M4 as Host

The SiWG917 connects to a remote TCP server via Wi-Fi or to an external host storage via SPI/UART/SDIO, or BLE to get the firmware file to the M4. The user can update the firmware by calling the required APIs.

Refer to [BLE OTA application](#) for an understanding of the firmware update using BLE.

The following figure shows the firmware update process via M4 as host.

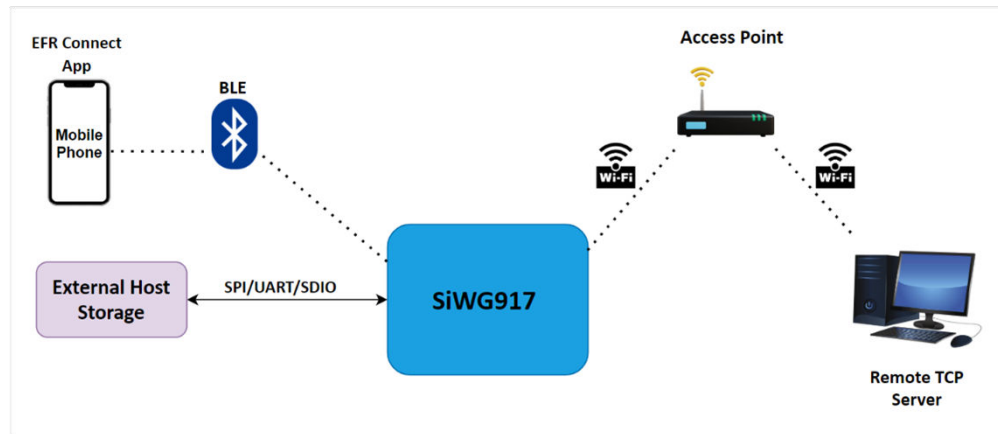


Figure 4.4. Firmware Update via M4 as Host (OTA)

The following flowchart shows the steps involved in the firmware update process via M4 as host.

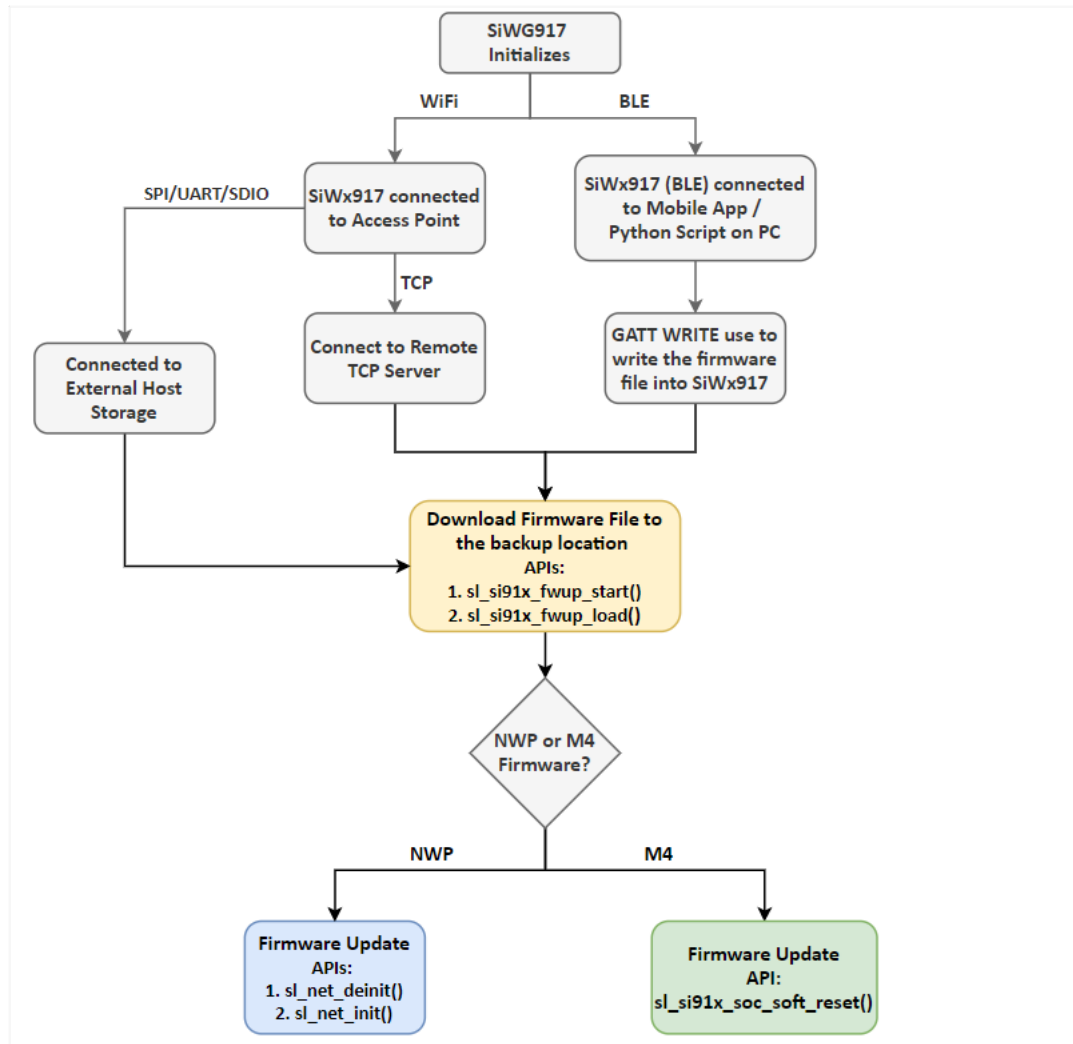


Figure 4.5. Firmware Update via M4 as Host (OTA) – Flowchart

4.2 Firmware Update via Bootloader

The firmware update via Bootloader is a two-stage process where the NWP Bootloader places the application image in a separate download location and then does the integrity check of the received image. The NWP Bootloader then replaces the current application image with the newly received authenticated image.

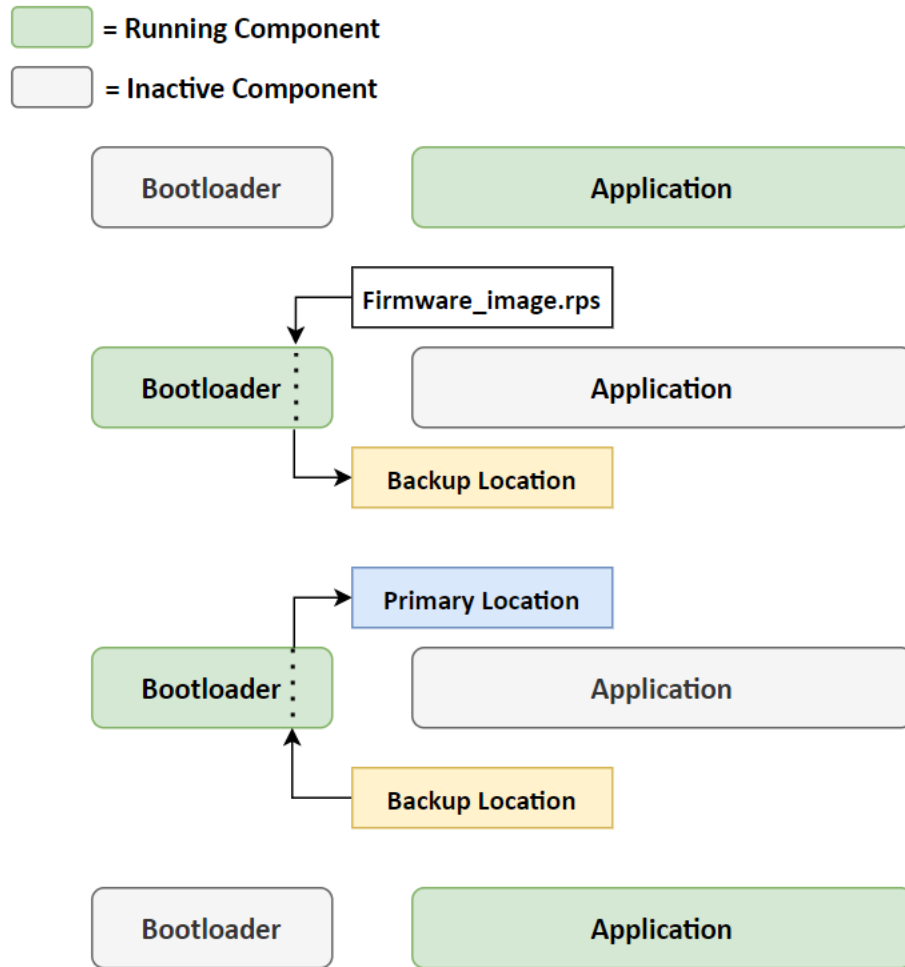


Figure 4.6. Firmware Update via Bootloader

There are two mechanisms to update the Firmware via Bootloader:

1. Firmware Update via Kermit
2. Firmware Update via Simplicity Commander Tool or CLI
3. Firmware Update via External Host Interfaces – SPI/SDIO (ISP Mode)Simplicity Commander Tool or CLI

The following subsections explains the mechanisms in detail.

4.2.1 Firmware Update via Kermit

In updating the firmware via Kermit, the SiWG917 device is connected to the PC using the USB cable. The device is detected as a UART COM Port. In this process, a serial terminal like Tera Term acts like a host to the SiWG917 and gives Bootloader commands to load the firmware.

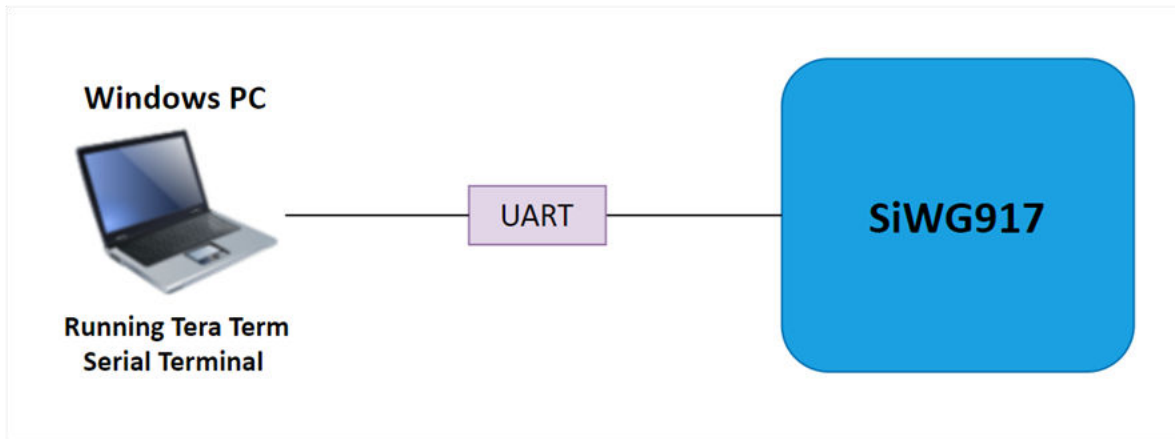


Figure 4.7. Firmware Update via Kermit

Steps for NWP Firmware/M4 Application Update

1. Connect the RX pin of the TTL board (FTDI) to the F9 pin on the main board and the TX pin of the TTL board (FTDI) to the F8 pin.
2. Press the ISP-button, and while holding down the ISP-button, press and release the Reset-button (on the main board) and release the ISP-button, this will enable ISP mode.
3. Open Tera Term and choose COM Serial Port as shown in the following figure.

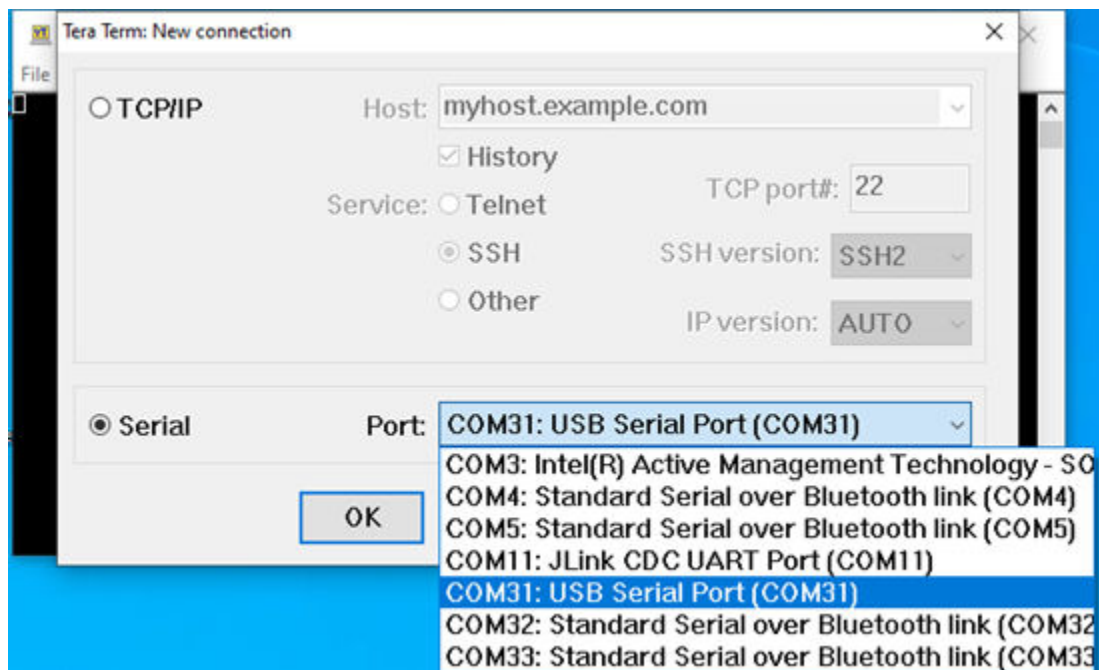


Figure 4.8. Serial COM Port Selection in Tera Term

4. Go to [**Setup** → **Serial port**] and select [**speed**] as **115200**.

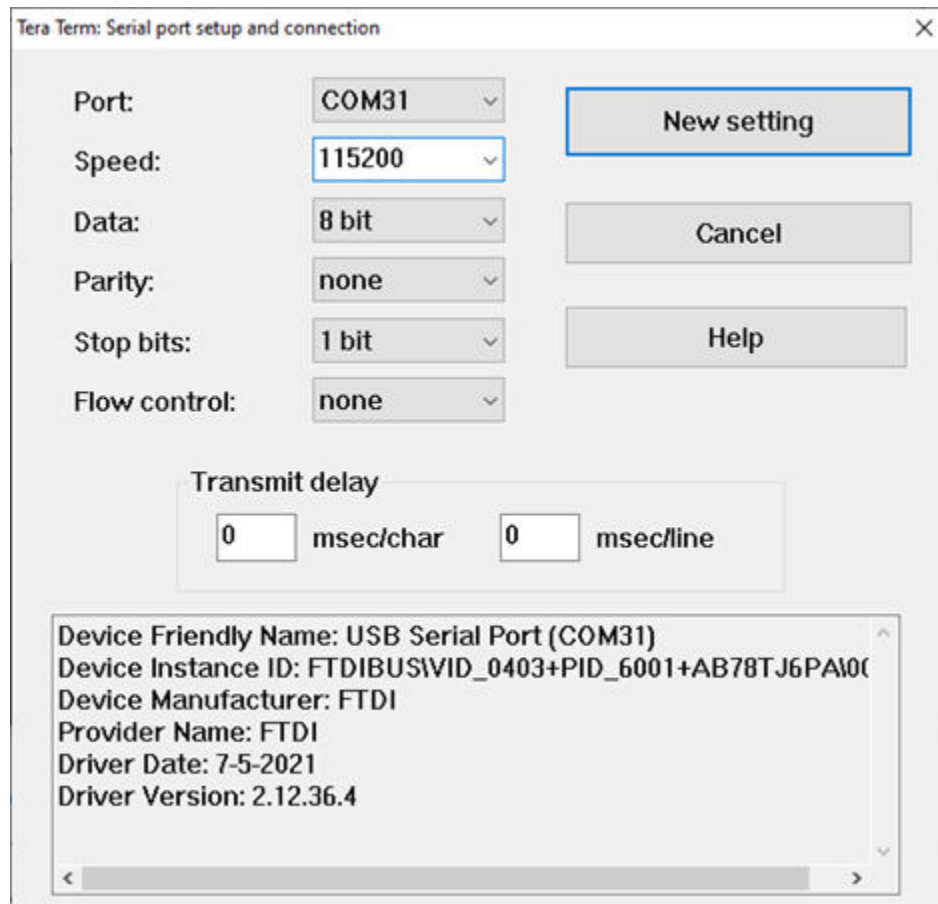


Figure 4.9. Serial Port Speed Selection in Tera Term

5. Press any key for boot message - [U], then type [U] for [**boot menu** → **type B** → **0**] to burn wireless (NWP firmware) (or) [4], [1] to burn MCU (M4) application.

Note: You can load/store only one image in any of the locations for NWP image (0-f) and for M4 (1-f).

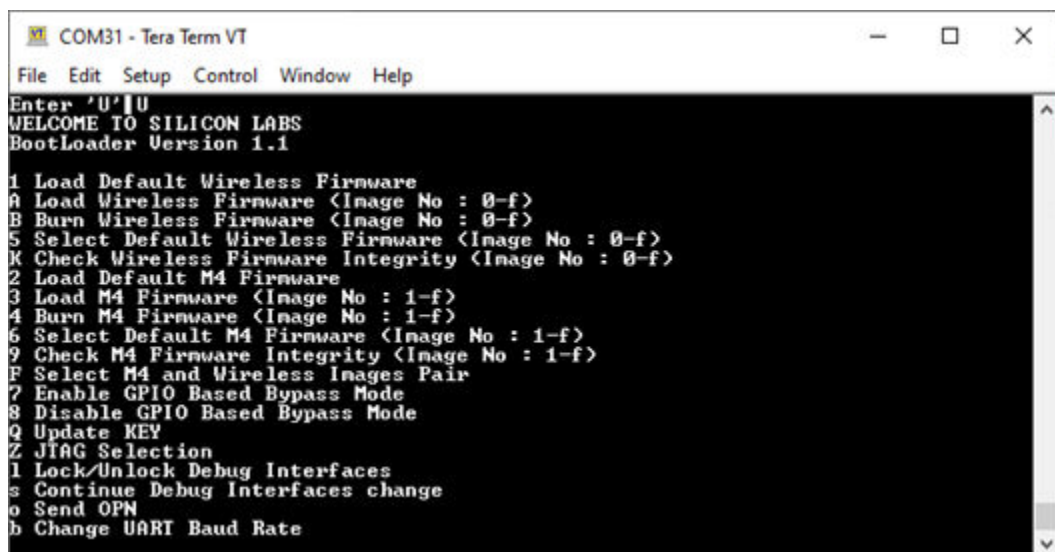


Figure 4.10. SiWG917 Boot Up

6. Go to [**File** → **Transfer** → **Kermit** → **Send..**] and select [**.rps**] or [**.bin**] file, this starts to load the image into the SiWG917 device.

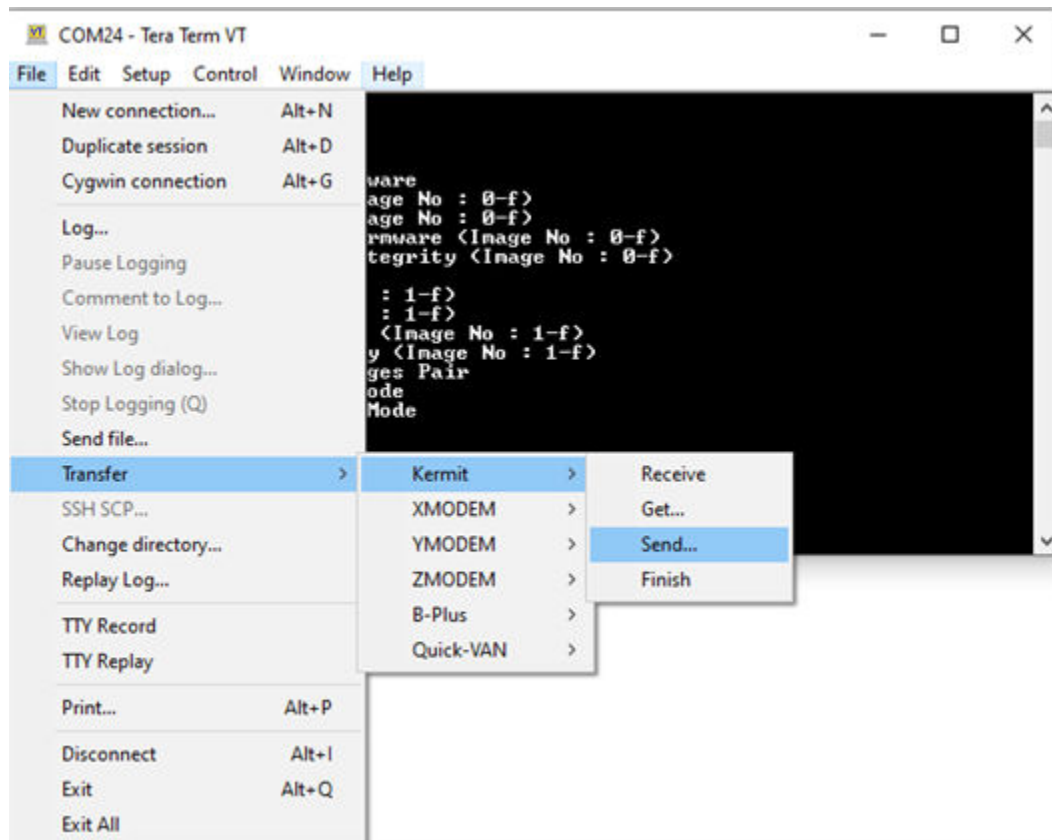


Figure 4.11. Send Firmware File in Tera Term via Kermit

7. Once the upgrade is successful, give [1], the message output [**Loading...**] is displayed. It is now safe to exit from [**Tera Term**].

Note:

1. The time taken to update the NWP and M4 firmware with the baud rate 921600 using the serial terminal (Tera Term) is as follows:
 - NWP firmware (1.6 MB): ~ 85 seconds
 - M4 firmware (104 KB): ~10 seconds
2. To boost the firmware update speed when performing firmware update via Kermit, you can modify as mentioned below

Search for TERATERM.INI File in the Tera Term application folder. Make the below mentioned change, [**KmtLongPacket = OFF --> KmtLongPacket = ON**]

4.2.2 Firmware Update via Simplicity Commander Tool or CLI

Simplicity Commander is a single, all-purpose tool to be used in a production environment. The steps to update the firmware using the Simplicity Commander tool are mentioned in the [Update SiWx91x Connectivity Firmware](#) section in the Getting started with SoC mode. It can also be invoked using a simple Command Line Interface (CLI).

The following figure shows the steps involved in firmware update process using the Simplicity Commander.

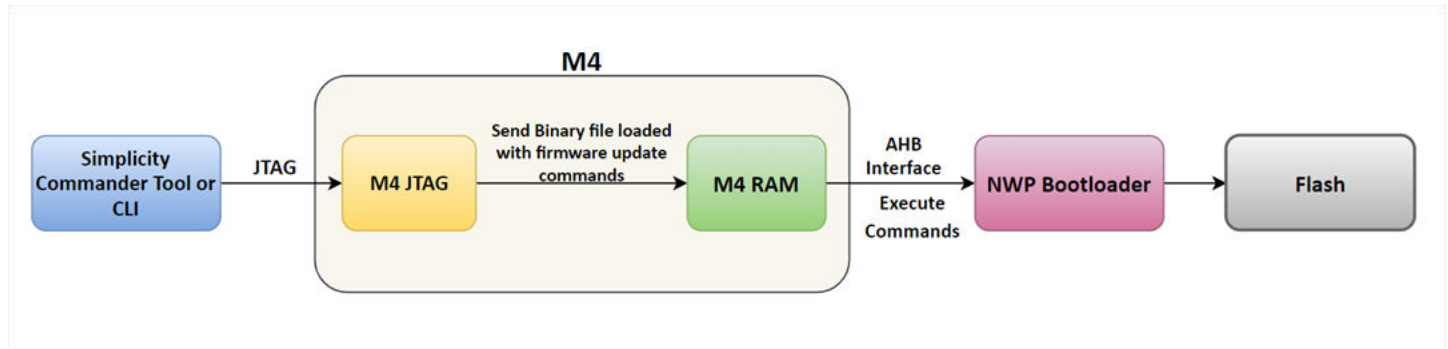


Figure 4.12. Firmware Update via Simplicity Commander

The general command line structure looks like this: **commander [command] [options] [arguments]** where:

- **commander** is the name of the tool
- **command** is one of the commands supported by Simplicity Commander, such as, flash, write, convert, and so on.
- **argument** is an item of information provided to Simplicity Commander when it is started. An argument is commonly used when the command takes one or more input files.
 - square brackets indicate optional parameters as in this **example**: commander write [filename(s)] [options]
 - angle brackets indicate required parameters as in this **example**: commander read --output <filename>

Firmware Update

- **Command**: commander rps load <filename.rps> -d si917
- **Example**: commander rps load ta_fw.rps -d si917

M4 Firmware Update

- **Command**: commander rps load [<filename.rps> -d si917]
- **Example**: commander rps load Secured_sl_si91x_calendar.rps -d si917

For more information about the Simplicity Commander Line Interface, refer to the [UG574: SiWx917 SoC Manufacturing Utility User Guide](#).

Note: If the security is enabled in your device, you have to load the secured image; otherwise, the loading fails.

4.2.3 Firmware Update via External Host Interfaces - SPI/SDIO (ISP Mode)

The SiWG917 should be kept in ISP mode for the firmware update vis external host interfaces (SPI/SDIO).

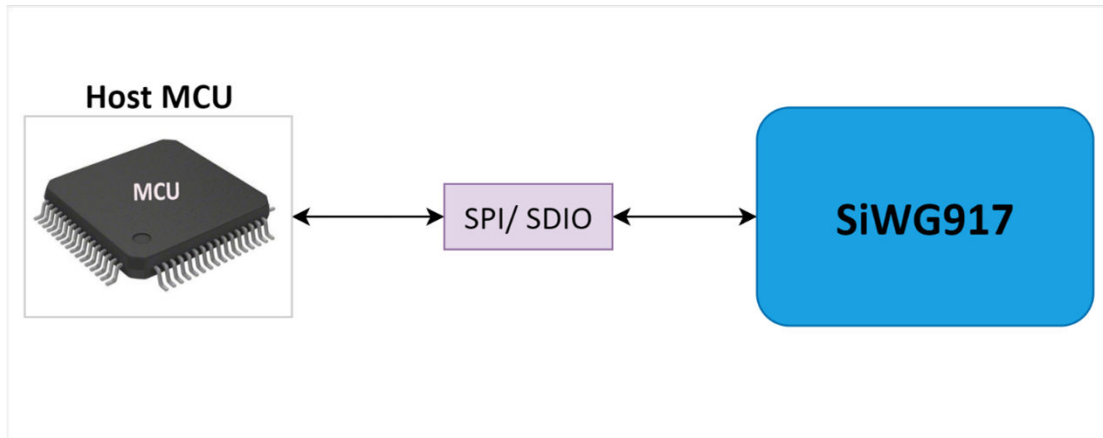


Figure 4.13. Firmware Update via External Host Interfaces - SPI/SDIO (ISP Mode)

The host uses two registers at address locations - 0x41050034 (**HOST_INTERACT_REG_IN**) and 0x4105003C (**HOST_INTERACT_REG_OUT**). These registers are used by the host and the device for giving commands and acknowledgments.

SPI or SDIO Interface – Steps for Firmware Update

The SiWG917 should be kept in ISP mode. To keep the device in ISP mode, press the ISP-button, and while holding down the ISP-button, press and release the Reset-button (on the main board) and release the ISP-button, this will enable ISP mode. You can exit the ISP mode by pressing the reset button on the main board.

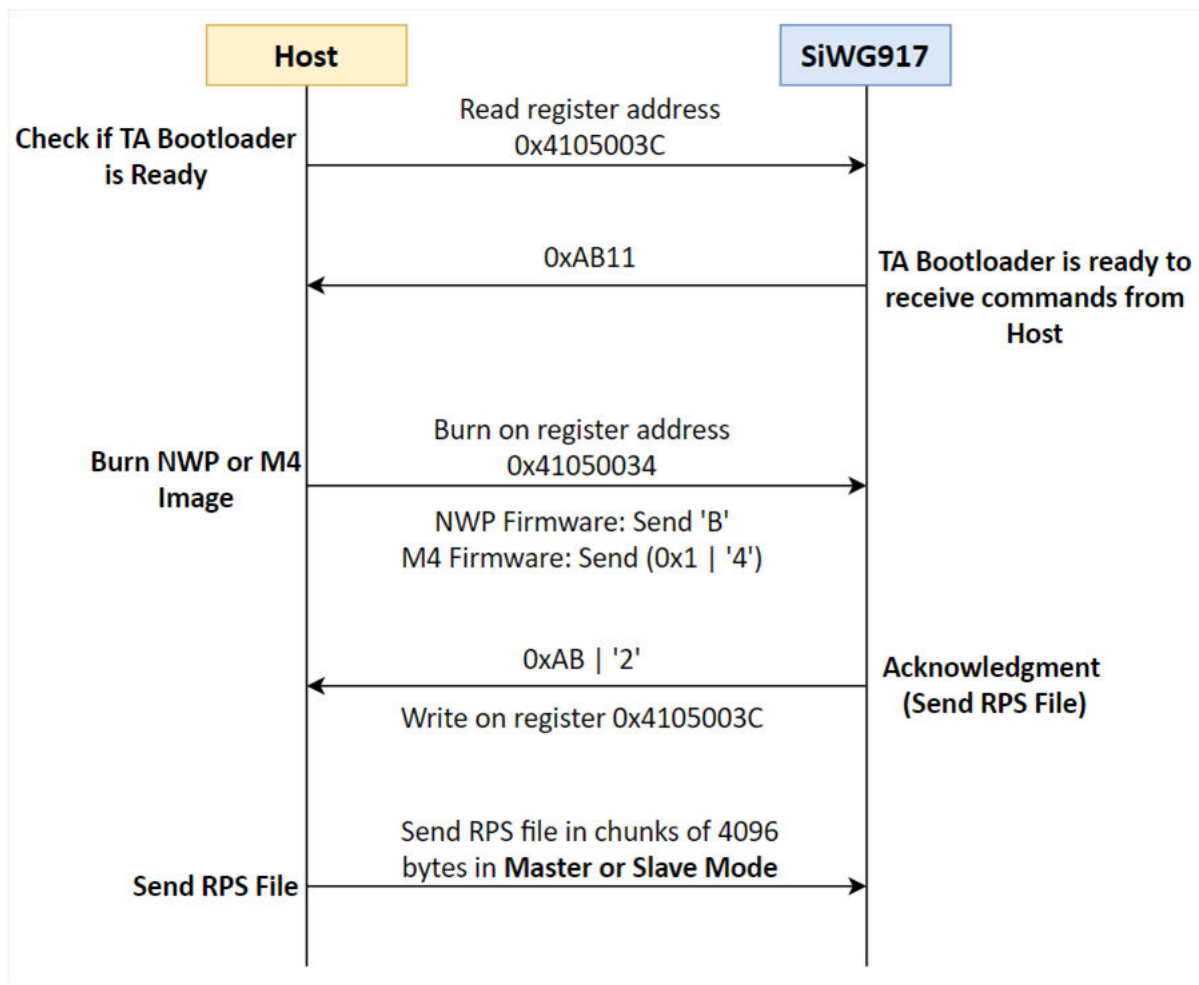


Figure 4.14. Steps for Firmware Update via External Host Interfaces - SPI/SDIO (ISP Mode)

1. Using SPI interface on the host and read the register 0x4105003C, which should read out 0xAB11. This means the NWP bootloader is ready to receive the commands from host.
2. Send "Burn" command to device. For NWP firmware, the command is 'B' and for M4 firmware, the command is 0x1|'4' ('4' OR'ed with 0x1 to indicate the image number).
 - This command should be written on register address 0x41050034.
3. Once the NWP bootloader receives the above command, the device will respond with acknowledgement which will be (0xAB | '2') written on register 0x4105003C. This response means the SiWG917 is requesting the host to send RPS file
4. Upon receiving above response, the host will send the RPS file in chunks of 4096 bytes.

Note:

- Refer to the [firmware update from the host MCU using SDIO](#) for reference application using EFM32 as host.
- We used an external MCU to flash the NWP and M4 firmware images through SPI interface to the SiWG917. The time taken for firmware update are as follows:
 - NWP firmware (1.6 MB): ~ 40 seconds
 - M4 firmware (104 KB): ~ 5 seconds

5. Secure Firmware Update

In case of Bootloader and OTA based firmware update mechanisms, a secure firmware update can be done by enabling the security in the firmware image using the Manufacturing Utility (Refer to the [UG574: SiWx917 SoC Manufacturing Utility User Guide](#)). The process of firmware update remains the same for Non-Secure and Secure Firmware updates, except that in the case of secure firmware image, the integrity checks are done.

The following flowchart shows the secure firmware update process.

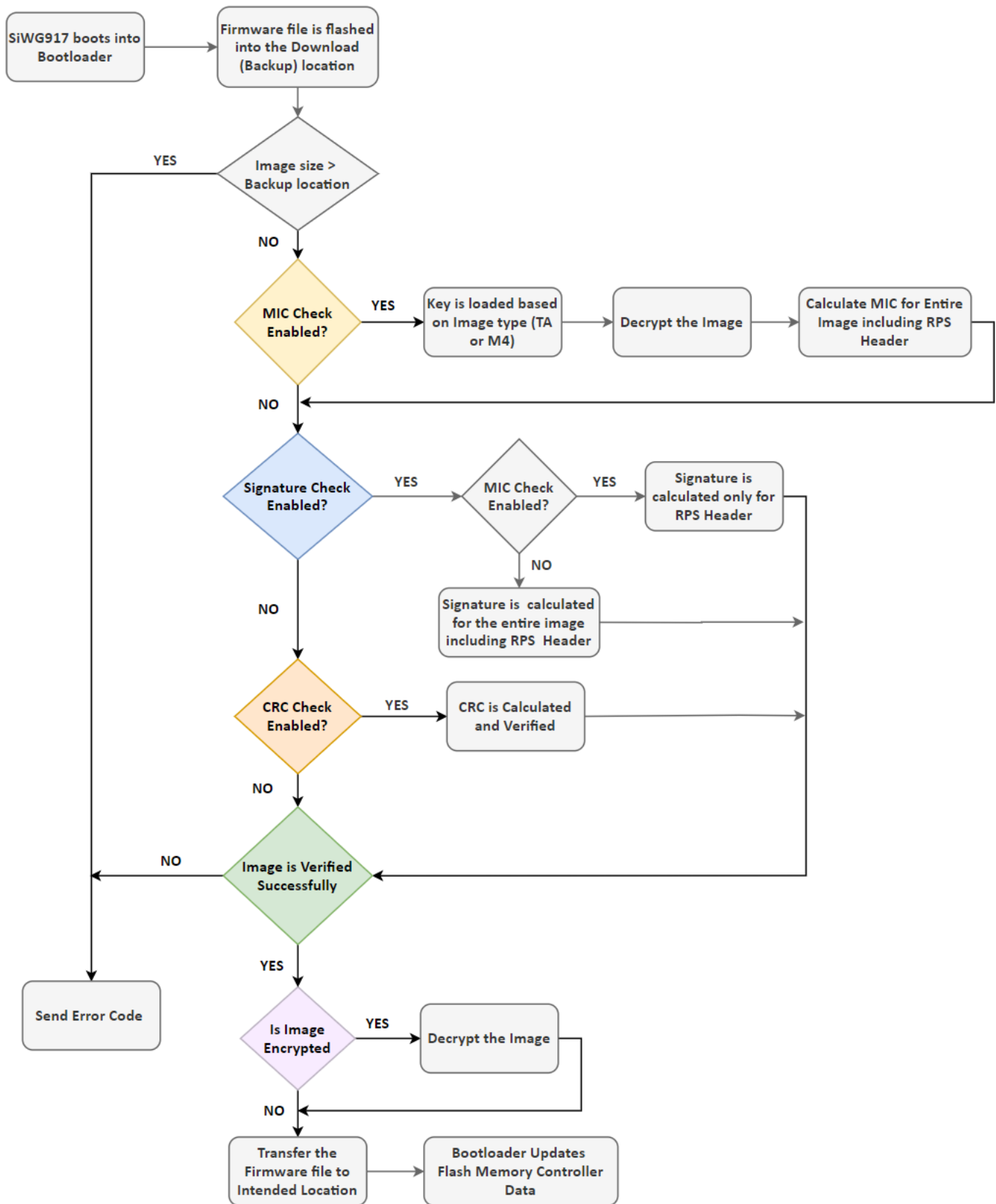


Figure 5.1. Secure Firmware Update Flowchart

The firmware image for both NWP and M4 supports the following:

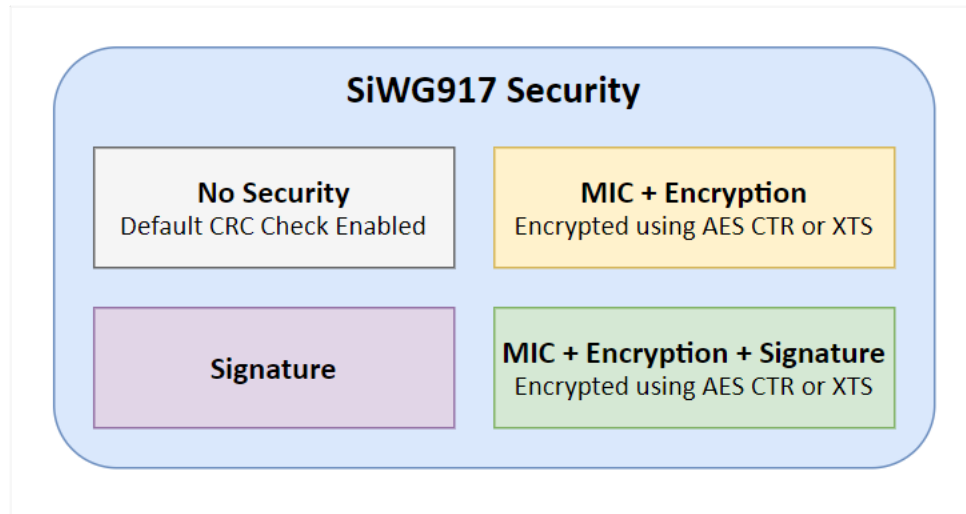


Figure 5.2. Supported Firmware Image Security

- **[No Security:]** Default CRC of the image within the RPS header.
- **[MIC + Encryption:]** The MIC is calculated for the whole plain image and saved within the RPS header. The image is encrypted and appended to the RPS header. The image can be encrypted using AES CTR or XTS mode.
- **[Signature:]** The size of the image in the RPS header is incremented by the size of the signature. Finally, the whole RPS file's (RPS Header+ image) signature is calculated and appended to the end of the file.
- **[MIC + Encryption + Signature:]** The MIC is calculated for the whole plain image and saved within the RPS header. The image is encrypted and appended to the RPS header. The image can be encrypted using AES CTR or XTS mode. The size of the image in the RPS header is incremented by the size of the signature. Finally, the whole RPS file's (RPS Header + image) signature is calculated and appended to the end of the file.

In each mode, the Control Flag of the RPS Header is updated with the correct configuration.

Flash Memory Controller

The Bootloader has a reserved space in flash where it saves the details of the current firmware. These details are saved when a new valid firmware is saved in the flash. This data includes the start address in the flash where the valid firmware is available, as well as the firmware's size, CRC and MIC value of the firmware, security flags, and the image's version number.

We have 2 FMCs for NWP and M4 separately. The FMCs are located in the flash and have flash protection, so they don't get corrupted. The FMC data itself is protected with CRC and MIC validation. Each time the FMC is modified after a new image is updated, the FMC is saved in a backup location and then the original location is updated to avoid any issues arising from the FMC getting corrupted while modifying it or a power down happening during the FMC modification process.

If there is a power outage during the copying of firmware from the backup (downloaded) location to the primary (target) location, the SiWG917 will restart the backup process from the beginning when you power up the device next time. It will not resume from where it left off. This behavior is the same for both NWP and M4. In the event of a power outage, the NWP FMC will retain the existing firmware start address. The NWP FMC is updated with the new firmware details only after the firmware update is fully completed.

6. Combined Image

The combined image is a single image which is obtained by combining the NWP and M4 images. The process of creating the combined image involves encrypting both the NWP and M4 images, and then adding a RPS header and signature. In the case of non-secure firmware, we will not be adding the signature.

- The Combined Image RPS header format will be the same as the M4 RPS header format with few reserved bytes changes.
- The signature for complete combined image will be calculated and appends at the end of image.
- MIC computation and signature can be used to maintain integrity and confidentiality of the combined image.
- Encryption of combined image is discarded as it will add overhead for firmware to decrypt and store into flash location. The NWP and M4 images are individually encrypted.

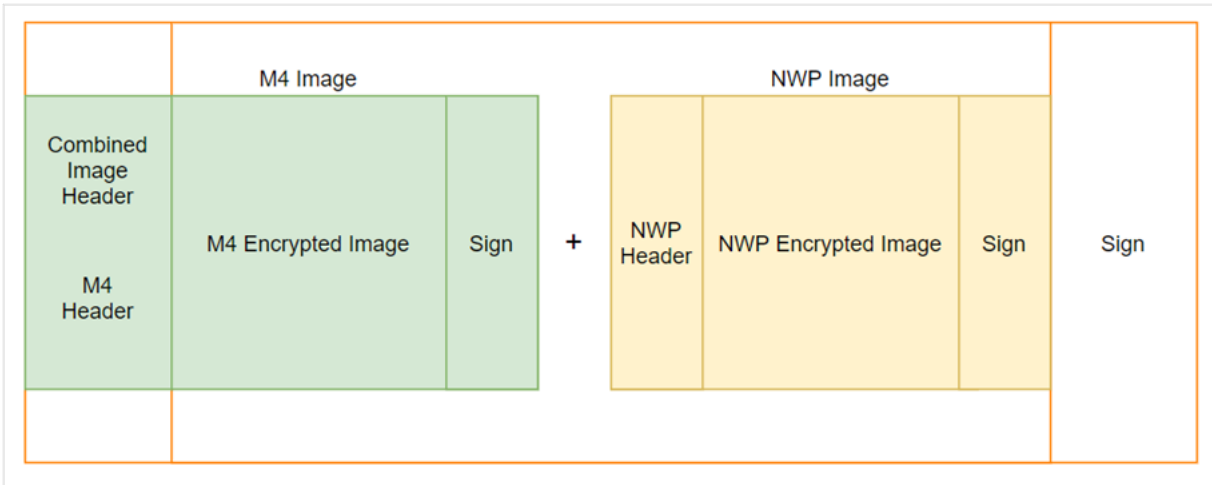


Figure 6.1. Combined Image with Signature

Note: For more information on combined image generation, refer to [UG574: SiWx917 SoC Manufacturing Utility User Guide](#).

6.1 Importance of Secure Over-the-Air Updates with Combined Images

- Over The Air (OTA) updates have become a popular way to update device firmware, ensuring that devices have the latest features and security enhancements. However, the security of OTA updates is critical, as attackers can exploit vulnerabilities in the update process to gain unauthorized access to devices and data.
- To address this, combined firmware images are now used for OTA updates. These images contain multiple firmware components, simplifying the update process, and reducing the risk of errors or incompatibilities.
- To ensure security, digital code signing is used to verify the authenticity of the combined image, while encryption protects the image during transmission to prevent unauthorized access and tampering.
- These measures provide robust protection, ensuring that OTA updates are performed safely and securely, without compromising the confidentiality or integrity of the device.
- Overall, OTA updates using combined firmware images are important for device manufacturers and end-users, enabling convenient and secure device firmware updates.

7. Guidelines and Recommendations

1. It is strongly recommended to use the latest version of firmware, especially for all major releases, as this has many changes compared to minor releases.
2. The SiWG917 should be kept powered on during the complete firmware update process, you need to ensure there are no power fluctuations.
3. While firmware update is happening, it is advisable to avoid doing multiple tasks.
4. The SoC should not be placed in power save mode during any portion of the firmware update process.
5. We recommend using the simplicity commander tool for firmware update via Bootloader.
6. Anti-rollback check is enabled as part of efuse configs and is used to block downgrading the firmware. It is disabled by default, but the users would enable it in the end of their development.

8. References

The below are few firmware update example references. Click on the link of your preferred example.

- Firmware update using a TCP server on a local PC or a cloud service such as Amazon AWS or Microsoft Azure: [GitHub Link](#)
- Firmware update using remote HTTP/s server or cloud storage server: [GitHub Link](#)
- M4 Firmware Update using a TCP server on a local PC or a cloud service such as Amazon AWS or Microsoft Azure: [GitHub Link](#)

9. Troubleshooting

- On a fresh SiWG917 chip with no NWP firmware present, if you try to run the application – You will see the error code: 0x56, this means Valid Firmware Not Present. Load the SiWG917 with the NWP firmware and run the application to resolve this issue.
- Loading non-secure firmware on secure board and vice-versa, results in 108 error (CRC check failure)
 1. If board is having secure MBR, we'll add security to the firmware image with the same keys and load the firmware in the device.
- Loading 1.6 MB M4 application in device with 1.8 MB MBR, results in 102 error (ISP mode ON)
 1. Check the MBR present in the device using SiWG917 manufacturing utility.
 2. If 1B → Load M4 application having 1.6MB flash addresses
 3. If 1F → Load M4 application having 1.8MB flash addresses
 4. Anything else (CC, 00) → program MBR again

10. Revision History

Revision 1.1

November 2024

- Added: [4.2.3 Firmware Update via External Host Interfaces - SPI/SDIO \(ISP Mode\)](#).
- Editorial changes throughout the document.

Revision 1.0

December 2023

- Initial release.

Smart. Connected. Energy-Friendly.



IoT Portfolio
www.silabs.com/products



Quality
www.silabs.com/quality



Support & Community
www.silabs.com/community

Disclaimer

Silicon Labs intends to provide customers with the latest, accurate, and in-depth documentation of all peripherals and modules available for system and software implementers using or intending to use the Silicon Labs products. Characterization data, available modules and peripherals, memory sizes and memory addresses refer to each specific device, and "Typical" parameters provided can and do vary in different applications. Application examples described herein are for illustrative purposes only. Silicon Labs reserves the right to make changes without further notice to the product information, specifications, and descriptions herein, and does not give warranties as to the accuracy or completeness of the included information. Without prior notification, Silicon Labs may update product firmware during the manufacturing process for security or reliability reasons. Such changes will not alter the specifications or the performance of the product. Silicon Labs shall have no liability for the consequences of use of the information supplied in this document. This document does not imply or expressly grant any license to design or fabricate any integrated circuits. The products are not designed or authorized to be used within any FDA Class III devices, applications for which FDA premarket approval is required or Life Support Systems without the specific written consent of Silicon Labs. A "Life Support System" is any product or system intended to support or sustain life and/or health, which, if it fails, can be reasonably expected to result in significant personal injury or death. Silicon Labs products are not designed or authorized for military applications. Silicon Labs products shall under no circumstances be used in weapons of mass destruction including (but not limited to) nuclear, biological or chemical weapons, or missiles capable of delivering such weapons. Silicon Labs disclaims all express and implied warranties and shall not be responsible or liable for any injuries or damages related to use of a Silicon Labs product in such unauthorized applications.

Trademark Information

Silicon Laboratories Inc.[®], Silicon Laboratories[®], Silicon Labs[®], SiLabs[®] and the Silicon Labs logo[®], Bluegiga[®], Bluegiga Logo[®], EFM[®], EFM32[®], EFR, Ember[®], Energy Micro, Energy Micro logo and combinations thereof, "the world's most energy friendly microcontrollers", Redpine Signals[®], WiSeConnect[®], n-Link, EZLink[®], EZRadio[®], EZRadioPRO[®], Gecko[®], Gecko OS, Gecko OS Studio, Precision32[®], Simplicity Studio[®], Telegesis, the Telegesis Logo[®], USBXpress[®], Zentri, the Zentri logo and Zentri DMS, Z-Wave[®], and others are trademarks or registered trademarks of Silicon Labs. ARM, CORTEX, Cortex-M3 and THUMB are trademarks or registered trademarks of ARM Holdings. Keil is a registered trademark of ARM Limited. Wi-Fi is a registered trademark of the Wi-Fi Alliance. All other products or brand names mentioned herein are trademarks of their respective holders.



Silicon Laboratories Inc.
400 West Cesar Chavez
Austin, TX 78701
USA

www.silabs.com