UG574: SiWx917 SoC Manufacturing Utility User Guide

Version 1.3

November 2024

Table of Contents

1 Introduction

This document describes the manufacturing procedure to be followed when procuring the SiWG917 (SiWx917 SoC) IC, based on the Ordering Part Number (OPN), flash mode, and security specifications.

The SiWG917 comprises two processors: Silicon Labs' Network Wireless Processor (NWP) and an ARM® Cortex® M4 Processor. The NWP subsystem is responsible for executing all networking and wireless stacks on independent threads and functions as the secure processing domain, overseeing secure boot, secure firmware updates, and debug lock. The Cortex-M4 processor is dedicated to peripheral and application-related processing

1.1 Manufacturing Procedure

Based on the OPN, the SiWG917 can be equipped with either 4 or 8 MB of "In-package" Quad SPI (QSPI) flash. Additionally, the SiWG917 supports external flash options of up to 16 MB[1]. The QSPI serves as the interface for accessing the flash memory.

The SiWG917 operates in the following flash modes:

- Common Flash - A single flash memory is shared between the NWP and M4 processors.
- Dual Flash - Independent flash memories are allocated for the NWP and M4 processors.

In case of no in-package flash OPNs, it is necessary to enable the flash mode of your choice, refer to the section Flash Mode Selection – No In-Package Flash OPN for the steps to select the appropriate flash mode.

For OPNs with 4 or 8 MB in-package flash, the default configuration is common flash mode. If you wish to configure the device in dual flash mode, refer to the section Dual Flash Mode.

The following flow chart illustrates the SiWG917 manufacturing procedure.

[1] External 16MB Flash is not supported by latest available SiWG917 MBR. For information regarding software roadmap features, lists of available features, and profiles, contact Silicon Labs or refer to document related to SiWG917 MBR (Release Notes and Reference Manuals).

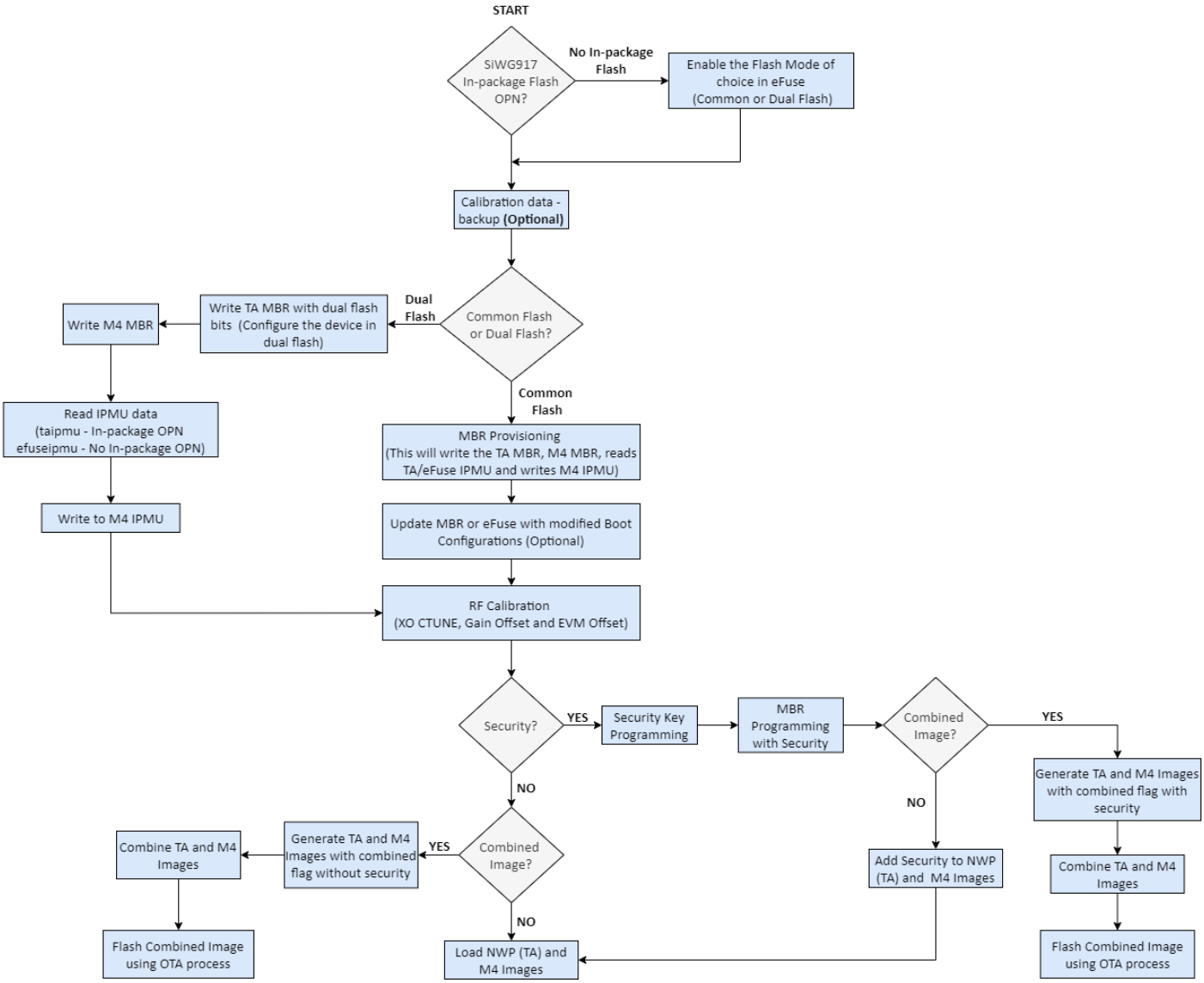


Figure 1.1: Manufacturing Procedure Flow of SiWx917

- **Verify the Flash Mode**
  - Check whether the SiWG917 is in common Flash mode or Dual Flash mode.
- **Master Boot Record (MBR) configuration based on OPN**
  - MBR is stored in flash and contains information such as clock frequencies, offsets of structures like device specific configurations, SPI configurations, external flash details, and so on.
- **Enable security**
  - The security fields in the MBR, PUF activation code, key descriptors, and the keys must be programmed in the device while enabling the security features.
- **RF Calibration** (Frequency and Gain Offset)
  - The Crystal Oscillator Capacitor tune (XO CTUNE) is used to adjust frequency, gain offset adjustments can be done in burst, continuous and continuous wave mode in the SiWG917 using the manufacturing utility.

1.2 Manufacturing Utility

The **Manufacturing Utility** operates in conjunction with Simplicity Commander via Command Line Interface (CLI). The CLI facilitates interaction with the commander to execute the manufacturing procedure. Multiple instances of the commander can support multiple devices, enabling simultaneous programming of production information into the SiWG917.

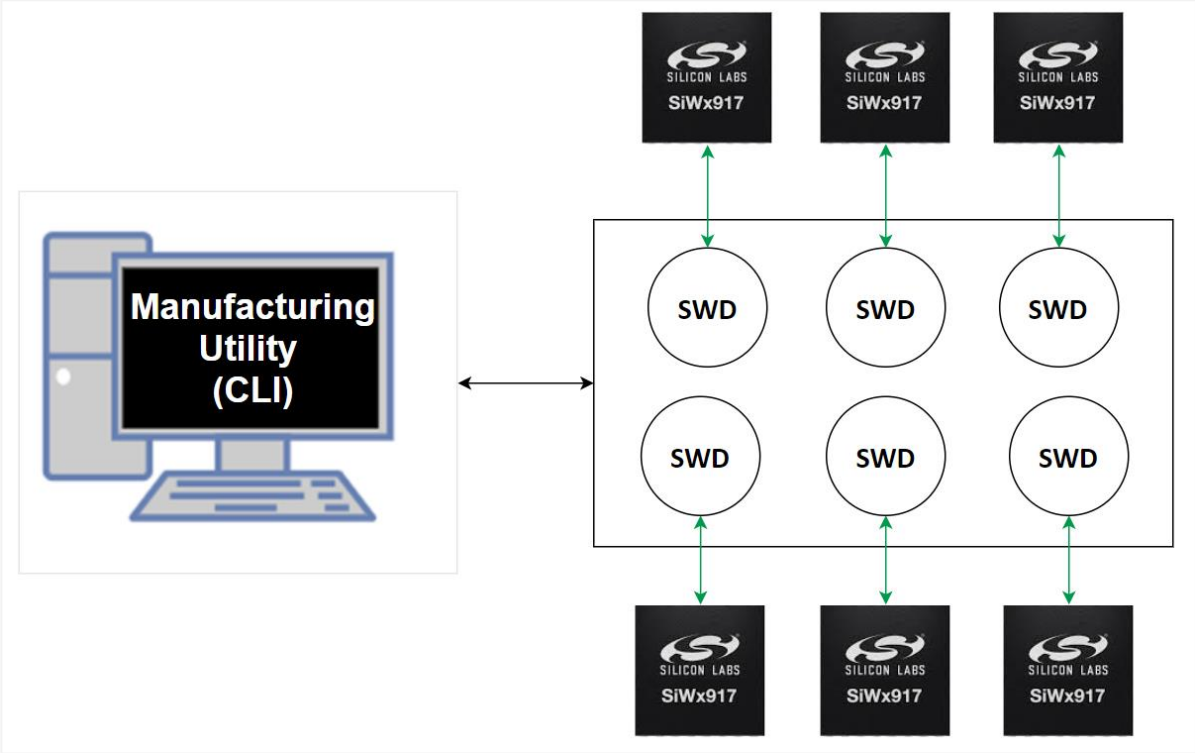


Figure 1.2: Manufacturing Utility

1.2.1 Simplicity Commander CLI

There are two methods to access the Simplicity Commander CLI.

- Simplicity Commander (as part of tools in Simplicity Studio)
- Simplicity Commander (individually installed)

Note: When configuring the SiWG917, you must use the latest version of **simplicity commander**. In contrast not doing so might result the device to act in an unexpected manner.

Enter “cmd” in the Simplicity Commander Installed path to open the commander CLI.

- Example: For Simplicity Commander (As part of tools in Simplicity Studio) – Go to the path: $:\...\SimplicityStudio\v5\developer\adapter_packs\commander and type ‘**cmd**’ in the address bar and press **Enter** key.

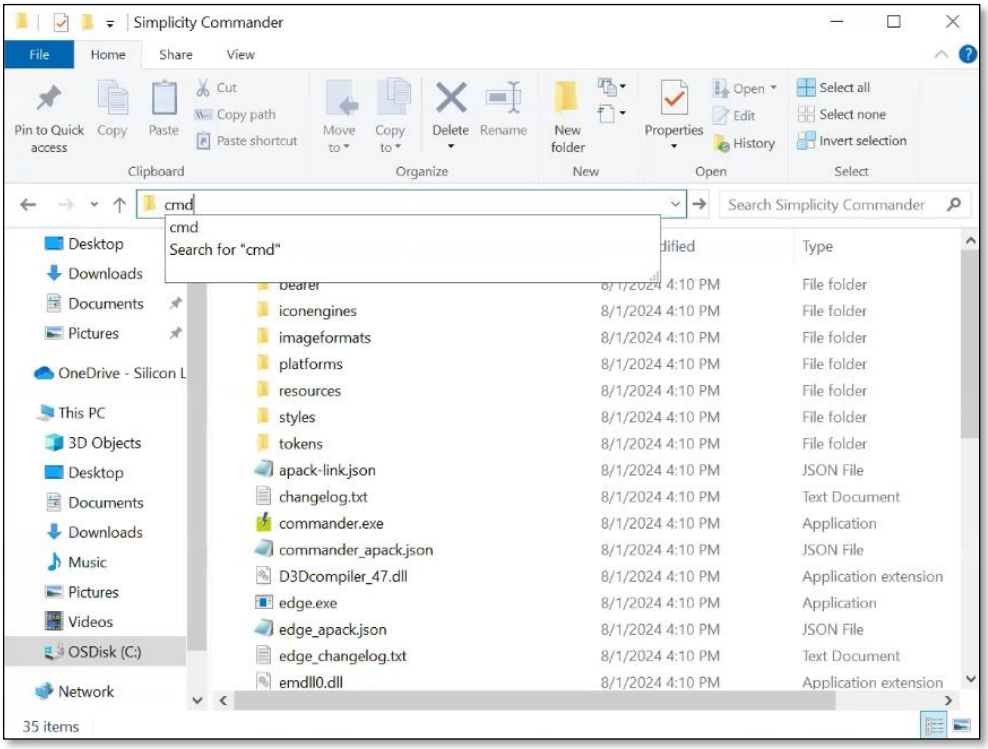


Figure 1.3: Simplicity Commander CLI

1.2.2 Manufacturing Utility Commands – Parameters

The following table lists the various parameters that are used by the commands mentioned in the subsequent sections.

Field	Description
provision	Writes TA MBR, M4 MBR, Reads, and Writes M4 IPMU in Common flash mode
write	Compares the provided MBR file in the command and the MBR file already present inside the device and writes the new changes into the MBR
read	Read the contents
init	Generate the activation code
--mbr	Master Boot Record
--keys	Security keys
m4mbrcf	Common flash M4 MBR
m4mbrdf	Dual flash M4 MBR
m4ipmucf	M4 IPMU data for common flash configuration
m4ipmudf	M4 IPMU data for dual flash configuration
efusecopy	Updating eFuse data to flash for selected region
<full opn>	Provide the **OPN** number. Example: SiWG917M111MGTBA
<updated-mbr-fields.json>	File in which security level is programmed
-d	Device
--skipload	Skip loading manufacturing firmware (loaded by the commander CLI in the device RAM, when the first command is given from the CLI) when the command is given. **Note:** Make sure the first command given to the device using commander CLI does not have the --**skipload** each time the device is powered ON.
--pinset [n]	Should be given for no in-package flash and in-package flash OPN (dual flash mode) **Pinset no [n]** | **GPIO set** 0 | GPIO 0 to 5

	2 GPIO 46 to 51 3 GPIO 52 to 57
--start	Turn on the radio transmission
--stop	Turn off the radio transmission
--noburst	Transmission happens continuously instead of being in bursts
--internalant	Select virtual internal RF switch. By default, the external antenna RF switch configuration is selected.
--ctuneoverride	It writes the XO CTUNE value into the flash instead of adding the frequency offset to already present value.
--store	Computed values are stored in flash
--storeinefuse	Compute values are stored in eFuse

Table 1.1: Manufacturing Utility Commands - Parameters

2 Flash Mode Selection – No In-Package Flash OPN

The following table lists the steps involved in enabling either common flash or dual flash mode in the eFuse (One Time Programmable) in a no in-package flash SiWG917 OPN.

Note: Once the Common flash or Dual flash mode is selected in the eFuse by following the processes outlined here, it is irreversible.

Steps	Description		Command (Syntax)
1	Read the present contents of the eFuse to an **efuse_value.json** file		`commander manufacturing read efuse --out efuse_value.json -d SiWG917M1xxXGTBA`
2	Enable Flash mode (Common or Dual Flash)	**Common Flash**	Open the efuse_value.json file (created in step 1) "`common_flash_enabled`" and set to 1
		Dual Flash	Open the efuse_value.json file (created in step 1) Following are listed: • "`m4_flash_address_width`" and set to 2 • "`m4_flash_address_width_valid`" and set to 1
3	Dry run and check the correct bit is set in the eFuse **(Optional, but recommended)**		`commander manufacturing write efuse --data efuse_value.json -d SiWG917M1xxXGTBA –dryrun`
4	Burn the bits in the eFuse		`commander manufacturing write efuse --data efuse_value.json -d SiWG917M1xxXGTBA`
5	Read back the contents of the eFuse		`commander manufacturing read efuse --property m4_and_security_config -d SiWG917M1xxXGTBA`

Table 2.1: Flash Mode Selection (No In-Package Flash OPN)

The preceding steps are for selecting the flash mode only. You must complete the following sections Manufacturing Procedure – Without Security, and Manufacturing Procedure – With Security as per your requirement.

3 Manufacturing Procedure – Without Security

This section outlines the sequence for programming devices without security.

Before proceeding with the following steps for Common flash or Dual flash mode, ensure that the flash mode is selected (Flash Mode Selection – No In-Package Flash OPN) for no in-package flash OPN.

Note: This section contains two manufacturing sequence of steps for both Common Flash Mode and Dual Flash Mode. Depending on the user's configuration, only one of these sections needs to be followed

3.1 Common Flash Mode

The following sequence needs to be followed for manufacturing procedure without security in common flash mode.

Steps	Description	Command (Syntax)
1	Backup – IPMU Calibration data	`commander manufacturing read <efuseipmu|taipmu> --out <filename.bin> -d <OPN Number> [--pinset n]` • efuseipmu – No In-package flash OPN • taipmu – In-package flash OPN
2	MBR Provisioning	`commander manufacturing provision –mbr <filename.bin|default> -d <OPN Number> [--skipload] [--pinset n]`
3	Boot Configurations Update in eFuse **(Optional)**	Refer to the section Boot Configurations Update – eFuse
4	RF (Frequency and Gain Offset) Calibration	Refer to section RF Calibration for calibration steps.
5	Load NWP and M4 Firmware Images	`commander rps load <filename.rps> -d <OPN Number>` **Note:** If you want to load a combined image (NWP +M4) instead of individual NWP and M4 images, refer to the section Combined Image (NWP + M4) for combined image related information

Table 3.1: Common Flash Mode (Without Security)

3.2 Dual Flash Mode

The following sequence must be followed for the manufacturing procedure without security in dual flash mode.

To configure devices with an in-package flash to dual flash mode, user must set the dual flash bits in the TA MBR (see dual_flash.json file).

Note:

- For dual flash mode and no in-package flash devices – you must mention the **[--pinset n]** in the manufacturing commands.
- Flash pinset is as below for SiWG917

Flash Pinset no	GPIO set
0	GPIO 0 to 5
2	GPIO 46 to 51
3	GPIO 52 to 57

Step	Description	Command (Syntax)
1	Backup – IPMU Calibration data	`commander manufacturing read <efuseipmu|taipmu> --out <filename.bin> -d <OPN Number> [--pinset n]` • efuseipmu – No In-package flash OPN • taipmu – In-package flash OPN
2	Write NWP MBR with dual flash bits enabled (using dual_flash.json file)	1. `commander manufacturing write tambr –data <filename.bin> -d <OPN Number> [--skipload] [--pinset n]` 2. `commander manufacturing write tambr –data dual_flash.json -d <OPN Number> [--skipload] [--pinset n]`
3	Write M4 MBR	commander manufacturing write m4mbrdf –data <filename.bin|filename.json> -d <OPN Number> [--skipload] [--pinset n]
4	Write IPMU calibration data to M4	Read IPMU (taipmu – In-package flash OPN, efuseipmu – No In-package flash OPN) 1. `commander manufacturing read <taipmu|efuseipmu> --out <filename.bin>` Write M4 IPMU 2. `commander manufacturing write <m4ipmucf|m4ipmudf> data <filename.bin> [--skipload] [--pinset n]`
5	Boot Configurations Update in eFuse **(Optional)**	Refer to the section Boot Configurations Update – eFuse
6	RF (Frequency and Gain Offset) Calibration	Refer to section RF Calibration for calibration steps.
7	Load NWP and M4 Firmware Images	`commander rps load <filename.rps> -d <OPN Number>` **Note:** If you want to load a combined image (NWP +M4) instead of individual NWP and M4 images, refer to the section Combined Image (NWP + M4) for combined image related information

Table 3.2: Dual Flash Mode (Without Security)

3.3 Example: SiWG917M111MGTBA (Common Flash Mode)

The manufacturing procedure without security for SiWG917M111MGTBA OPN in common flash mode is as follows.

1. **Backup – IPMU Calibration**
   - `commander manufacturing read taipmu --out filename.bin -d SiWG917M111MGTBA`
2. **MBR Provisioning**
   - `commander manufacturing provision –mbr default -d SiWG917M111MGTBA`
3. **RF Calibration – refer to section RF Calibration**
4. **Load NWP and M4 Images**
   - **NWP Image**
     `commander rps load SiWG917-B.2.12.2.1.0.9.rps -d SiWG917M111MGTBA`
   - **M4 Image**
     `commander rps load si91x_hello_world.rps -d SiWG917M111MGTBA`

Note: If no MBR is present in the device, you will see 0xa0ac error.

4 Security Features

The following sections provide information on enabling SiWG917 security features using the manufacturing utility, (Simplicity Commander CLI).

4.1 Secure Boot

Secure Boot is a feature designed to ensure that only authenticated code can run on the chip. If a device fails its security check, it is not permitted to run, and program control will typically stall in the validating module. SiWG917 possess two bootloaders: the Security Bootloader, and the Application Bootloader.

The Security Bootloader runs on the NWP, while the Application Bootloader runs on the Cortex M4 processor. **Secure Boot is implemented within the Security Bootloader.**

The Secure Boot process is as follows:

1. The Security Bootloader authenticates the MBR in the flash.
2. The Security Bootloader validates the integrity and authenticity of the firmware (NWP and M4) in the flash and subsequently invokes the Application Bootloader.

For enabling only Secure Boot or Secure Boot along with other security features, refer to the Security Levels section.

4.2 Secure Key Management and Protection

Inject custom public and private keys and other custom secret keys on the chips during manufacturing – safeguard your keys right from the beginning of their lifecycle.

Physically Unclonable Function (PUF) derived intrinsic keys are shown in the following table. These keys are unique per device and are generated randomly when the PUF is initialized.

Key	Description
Master Key	This key is used for MIC calculation and wrapping/ unwrapping of TA keys. Keys encrypted using Master Key is done using AES ECB Mode. MIC is calculated using AES CBC mode.
Unwrap Key	This key is used for MIC calculation and wrapping/ unwrapping of M4 keys. Keys encrypted using Unwrap Key is done using AES ECB Mode. MIC is calculated using AES CBC mode
TA FW key 1, TA FW key 2	These two keys are being used for encryption and inline decryption of TA firmware in CTR or XTS mode based on configuration from security configs of EFUSE
M4 FW key 1, M4 FW key 2	These two keys are being used for encryption and inline decryption of M4 firmware in CTR or XTS mode based on configuration from security configs of EFUSE

Table 4.1: PUF Derived Intrinsic Keys

Keys generated by Simplicity Commander and stored on the device are as follows.

Key	Description
OTP Symmetric Key	Used for flash content's (MIC) verification - AES CBC is used for calculation of the MIC value.
OTP Public Key	Used for flash content's (digital signature) verification (ECDSA-P256). This key is used to verify the signature of MBR
TA Public Key*	Used for TA firmware signature verification (ECDSA-P256) – by the security bootloader. This key will be wrapped with PUF derived intrinsic keys

TA OTA Key*	Used for TA firmware image encryption and MIC calculation of TA Firmware. This key will be wrapped with PUF derived intrinsic keys
M4 OTA Key*	Used for M4 firmware image encryption and MIC calculation of M4 firmware. This key will be wrapped with PUF derived intrinsic keys
M4 Public Key*	Used for M4 firmware signature verification (ECDSA-P256) by the security bootloader. This key will be wrapped with PUF derived intrinsic keys
Attestation Private Key	Used for signing secure attestation tokens(ECDSA-P256). This key will be wrapped with PUF derived intrinsic keys

Table 4.2: Keys Generated by Simplicity Commander

*** Note**: These keys are referred to as “static keys” in this document.

For information on how to generate the keys at commander, refer to Generate Static Keys from Simplicity Commander. For PUF activation, refer to Activation Code Generation for PUF and to generate intrinsic keys and load the keys, refer to Provisioning Static Keys.

The following table lists the Key type, Key storage requirements, and Storage type.

S.No	Key	Key Storage Requirements	Algorithm	Operation	Modes	Storage	
						NWP eFuse	NWP Flash
1	OTP Symmetric Key	16 bytes	AES	MIC Calculation	AES-CMAC	☑	–
2	OTP Public Key	91 bytes - NIST Curve P-256	ECDSA	Sign verification	–	☑	–
3	TA Public Key	96 bytes* - NIST Curve P-256	ECDSA	Sign verification	–	–	☑
4	TA OTA Key	32 bytes	AES	Encryption, MIC Calculation	Encryption: AES-ECB MIC Calculation: AES-CMAC	–	☑
5	M4 OTA Key	32 bytes	AES	Encryption, MIC Calculation	Encryption: AES-ECB MIC Calculation: AES-CMAC	–	☑
6	M4 Public key	96 bytes* - NIST Curve P-256	ECDSA	Sign verification	–	–	☑
7	Attestation Private Key	240 Bytes* - NIST Curve P-256	ECDSA	Sign	–	–	☑
8	Master Key	52 bytes key code used to initialize key in Key Holder	AES	Encryption, MIC Calculation	Encryption: AES-ECB MIC Calculation: AES-CMAC	–	☑
9	Unwrap Key	52 bytes key code used to initialize key in Key Holder	AES	Encryption	Encryption: AES-ECB MIC Calculation: AES-CMAC	–	☑
10	TA FW Key (2 keys)	52 bytes key code used to initialize key in Key Holder	AES	Encryption	AES CTR, AES XTS	–	☑
11	M4 FW Key (2 keys)	52 bytes key code used to initialize key in Key Holder	AES	Encryption	AES CTR, AES XTS	–	☑

Table 4.3: Keys Type, Storage Requirements and Storage Type

Note: In the preceding table ***** signifies:

- For TA and M4 Public key actual key size is 91 bytes, remaining 5 padding bytes added to make key size multiple of 16 for AES encryption.
- For Attestation key also, actual key size is 227 bytes.
- All keys stored in flash will be appended with MIC and other metadata that is not reflected in the table.

4.3 Secure Firmware Upgrade

The secure firmware update feature of the bootloader checks the authenticity of the new firmware image along with its integrity. The bootloader updates the image only after successfully validating the authenticity and integrity of the image. It prevents downgrade to a lower version of firmware using the anti-rollback feature if it is enabled.

The bootloader also supports transparent migration to a wirelessly updated image and protection against failures by providing recovery mechanisms.

The security bootloader uses a proprietary format for its upgrade images, called RPS. These files have extension ". rps".

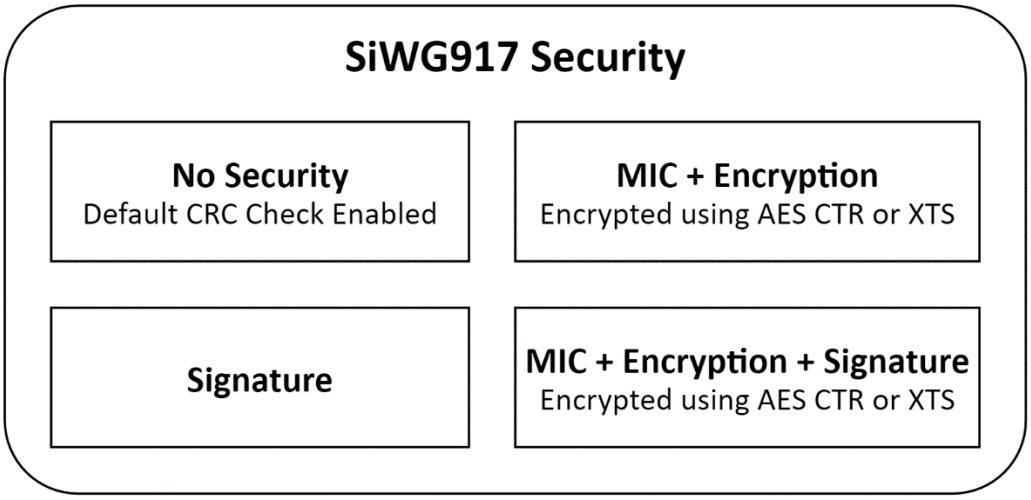


Table 4.1: SiWG917 Firmware Image Security

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

- *No Security:* Default CRC of the image within the RPS header.
- *Message Integrity Check (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 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. Finally, the whole RPS file's (RPS Header+ image) signature is calculated and appended to the end of the file.

To understand how to select the security level of your choice refer to Security Levels section.

To enable security for NWP and M4 images refer to Enable Security Configurations in NWP and M4 Firmware Images.

4.4 Debug Lock

Configure the debug port securely before the chips leave the factory with the Debug Lock Feature (which can be unlocked with a token)

The SiWG917 has a hardware debug solution. This provides high system visibility of the processor and memory through either a traditional JTAG port or a 2-pin Serial Wire Debug (SWD) port that is ideal for microcontrollers and other small package devices. The Secure Lock Feature locks the Debug Port. Any unauthorized access will be restricted.

This will be a one-time update of the MBR, however. This update should happen when the user has completed the development phase and entered the manufacturing phase for their products. Locking the NWP and M4 debug ports is a necessary step in production to prevent unauthorized access to sensitive information. Refer to the AN1428: SiWx917 Debug Lock for more information on enabling debug lock.

5 Manufacturing Procedure – With Security

The following steps show the manufacturing procedure to enable security configurations.

For devices with no in-package flash, ensure the flash mode is selected (Flash Mode Selection – No In-Package Flash OPN) before proceeding with the following steps for common flash or dual flash mode.

Note: This section contains two manufacturing sequence of steps for both Common Flash Mode and Dual Flash Mode. Depending on the user's configuration, only one of these sections needs to be followed

5.1 Common Flash Mode

The following sequence needs to be followed to enable security configurations in common flash mode during manufacturing.

Step	Description	Command (Syntax)
1	Backup – IPMU Calibration data	`commander manufacturing read <efuseipmu|taipmu> --out <filename.bin> -d <OPN Number> [--pinset n]` • efuseipmu – No In-package flash OPN • taipmu – In-package flash OPN
2	MBR Provisioning	`commander manufacturing provision --mbr <filename.bin|default> -d <OPN Number> [--skipload] [--pinset n]`
3	Security Key Programming	Refer to section Security Key Programming.
4	MBR Programming to Enable Security Configurations	Refer to section MBR Programming to Enable Security Configurations.
5	Boot Configurations Update in eFuse **(Optional)**	Refer to the section Boot Configurations Update – eFuse.
6	RF (Frequency and Gain Offset) Calibration	Refer to section RF Calibration for calibration steps.
7	Enable Security Configurations in NWP and M4 Firmware Images	Refer to section Enable Security Configurations in NWP and M4 Firmware Images.
8	Load NWP and M4 Firmware Images	`commander rps load <filename.rps> -d <OPN Number>` **Note:** If you want to load a combined image (NWP +M4) instead of individual NWP and M4 images, refer to the section Combined Image (NWP + M4) for combined image related information.
9	Debug Lock **(Optional, but Recommended)**	Refer to the section Debug Lock.

Table 5.1: Common Flash Mode (With Security)

Note: The security configurations of firmware images must match the security configurations on the device, that is MIC, signature and encrypted XiP. A mismatch can lead to firmware execution failure.

5.2 Dual Flash Mode

To configure devices with an in-package flash to dual flash mode, user must set the dual flash bits in the TA MBR (refer to dual_flash.json file).

Note:

- For dual flash mode and no in-package flash devices - you must mention the **[--pinset n]** in the manufacturing commands.
- Following are the Flash pinset for the SiWG917.

Flash Pinset no	GPIO set
0	GPIO 0 to 5
2	GPIO 46 to 51
3	GPIO 52 to 57

The following sequence needs to be followed to enable security configurations in dual flash mode during manufacturing.

Step	Description	Command (Syntax)
1	Backup – IPMU Calibration data	`commander manufacturing read <efuseipmu|taipmu> --out <filename.bin> -d <OPN Number> [--pinset n]` • efuseipmu – No In-package flash OPN • taipmu – In-package flash OPN
2	Write NWP MBR with dual flash bits enabled	1. `commander manufacturing write tambr –data <filename.bin> -d <OPN Number> [--skipload] [--pinset n]` 2. `commander manufacturing write tambr --data dual_flash.json -d <OPN Number> [--skipload] [--pinset n]`
3	Write M4 MBR	`commander manufacturing write m4mbrdf --data <filename.bin|filename.json> -d <OPN Number> [--skipload] [--pinset n]`
4	Write IPMU calibration data to M4	Read IPMU (taipmu – In-package flash OPN, efuseipmu - no In-package flash OPN) 1. `commander manufacturing read <taipmu|efuseipmu> --out <filename.bin>` Write M4 IPMU 2. `commander manufacturing write <m4ipmucf|m4ipmudf> data <filename.bin> [--skipload] [--pinset n]`
5	Security Key Programming	Refer to section Security Key Programming.
6	MBR Programming to Enable Security Configurations	Refer to section MBR Programming to Enable Security Configurations.
7	Boot Configurations Update in eFuse **(Optional)**	Refer to the section Boot Configurations Update – eFuse.
8	RF (Frequency and Gain Offset) Calibration	Refer to section RF Calibration for calibration steps.
9	Enable Security Configurations in NWP and M4 Firmware Images	Refer to section Enable Security Configurations in NWP and M4 Firmware Images.
10	Load NWP and M4 Firmware Images	`commander rps load <filename.rps> -d <OPN Number>` **Note:** If you want to load a combined image (NWP + M4) instead of individual NWP and M4 images, refer to the section Combined Image (NWP + M4) for combined image related information.

11	Debug Lock **(**Optional, but Recommended**)**	Refer to the section Debug Lock.

Table 5.2: Dual Flash Mode (With Security)

Note: The security configurations of firmware images should align with the security configurations on the device, such as MIC, signature, and encrypted XiP. A discrepancy may result in a failure of firmware execution.

5.3 Security Key Programming

This part of manufacturing includes generation of static keys, and Physically Unclonable Function (PUF) derived intrinsic keys. This section includes instructions to provision the keys into flash. While programming the static keys, the keys are saved in wrapped format in the flash. There are multiple keys that are generated inside and outside of device. Refer to Secure Key Management and Protection section for information on keys generated by simplicity commander and PUF derived intrinsic keys.

Following is the procedure for provision keys.

1. Activation code generation only for PUF
2. Power cycle the device to boot again after step 1
3. Generate static keys using simplicity commander
4. Provisioning static keys

5.3.1 Activation Code Generation for PUF

This command is used to generate an activation code from the PUF module present in SiWG917 device. Once generated, the firmware writes this activation code into NWP flash. The activation code is used as a seed to initialize the PUF.

Syntax:

```
commander manufacturing init --mbr <filename.bin|default> -d <full opn> [--skipload] [--
pinset n]
```

Return Code

- Success – 0xa05a
- In case of Failure – Refer to Possible Error Codes and try again.

5.3.2 Power Cycle the device

You must power cycle the device after the activation code generation and flash completion to successfully initialize the PUF and store the PUF derived intrinsic keys. This is a mandatory step.

5.3.3 Generate Static Keys from Simplicity Commander

These keys are generated by simplicity commander. The key generation will produce a json file containing the Static Keys. For asymmetric keys, both public and private keys are written to the JSON file.

Syntax:

```
commander util genkeyconfig --outfile <keys.json> -d <full opn>
```

Return Code

- Success – 0xa05a
- In case of Failure – Refer to Possible Error Codes and try again

5.3.4 Provisioning Static Keys

This step performs the actual provisioning of the static keys. This command instructs the initialized PUF to derive intrinsic keys and store them in NWP flash as a key code.

Syntax:

```
commander manufacturing provision --keys <keys.json> -d <full opn> [--skipload]
[--pinset n]
```

Note: keys.json file contains the static keys i.e. TA OTA key, M4 OTA key, TA public key, M4 public key and attestation key.

5.4 MBR Programming to Enable Security Configurations

By default, SiWG917 devices does not have security features enabled. The MBR and eFuse configurations of the device have all security features set to disable state. This means the PUF is not enabled, the activation code for the PUF is not programmed, and neither keys nor key descriptors are programmed.

5.4.1 Programmable Fields in MBR

The following table lists the relevant features in the MBR which can be programmed. The MBR fields are present in two structures efuse_data and mbr within the MBR.

Feature	Structure	MBR Field
To enable TA MIC and encryption	efuse_data	ta_secure_boot_enable = 1;
To enable M4 MIC and encryption	efuse_data	m4_secure_boot_enable = 1;
To enable TA Signature	efuse_data	ta_digital_signature_validation = 1;
To enable M4 Signature	efuse_data	m4_digital_signature_validation = 1;
To enable TA inline encryption	efuse_data	ta_encrypt_firmware = 1; (1 for CTR, 2 for XTS mode)
To enable M4 inline encryption	efuse_data	m4_encrypt_firmware = 1;
To select M4 firmware encryption mode	efuse_data	m4_fw_encryption_mode = 1; (1 for CTR, 2 for XTS mode)
Start address of the sector where the PUF Activation code must be saved	mbr	puf_activation_code_addr = 0x2000;

Table 5.3: MBR Fields - Features

- The security fields in MBR, PUF Activation code, Key descriptors, and the Keys are to be programmed in the device in order to enable the security configurations in the device.
- The preceding features mentioned can also be configured in eFuse, but it will be irreversible. For other boot configurations that can be configured in MBR, refer to the section Possible Boot Configurations.

5.4.2 Security Levels

There are three different security levels available. The levels are defined to make the security configurations easier for the user. It is important to note that these levels are not security configurations in themselves – they are a grouping of available security features to help users decide based on their security requirements.

1. Security Level 1 or Low Security Level
2. Security Level 2 or Partial Security Level
3. Security Level 3 or Full Security Level

5.4.2.1 Security Level 1 (Low Security)

If you want to configure only Message Integrity Check (MIC) while enabling secure boot, the configurations in this level should be considered. This security level enables the bootloader to carry out the MIC of the firmware image

during firmware loading and firmware update. In this configuration, the firmware update process is faster compared to other security levels, the bootup time is faster, and the firmware execution is also faster.

This configuration is selected using the following fields:

```
"efuse_data": {
        "ta_secure_boot_enable": 1,
        "m4_secure_boot_enable": 1
}
```

5.4.2.2 Security Level 2 (Partial Security)

If you want to add signature check of firmware image along with MIC while enabling secure boot, the configurations in this level should be considered. This security level enables the bootloader to carry out the MIC and signature validation of the firmware image during firmware loading and firmware update.

In this level, the firmware update process will be slower than Security Level 1; the bootup time will be relatively slower than the Security Level 1, and the firmware execution speed will be same as Security Level 1.

This configuration is selected using the below fields:

```
"efuse_data": {
        "ta_secure_boot_enable": 1,
        "ta_digital_signature_validation": 1,
        "m4_secure_boot_enable": 1,
        "m4_digital_signature_validation": 1
}
```

5.4.2.3 Security Level 3 (Full Security)

If you want to enable secure boot with MIC, signature check, and inline encryption, the configurations in this level should be considered. The configurations in this security level enables the bootloader to perform the MIC, signature validation and decryption of the firmware image while loading and updating firmware. Additionally, it enables encrypted execute-in-place(XiP) while saving the firmware in the flash.

In this configuration, the firmware update process is slow, the bootup time is high, and the firmware execution is slow compared to security levels 1 and 2.

This configuration is selected using the following fields:

```
"efuse_data": {
        "ta_secure_boot_enable": 1,
        "ta_digital_signature_validation": 1,
        "ta_encrypt_firmware": 1,      // 1 for CTR, 2 for XTS mode
        "m4_encrypt_firmware": 1,
        "m4_fw_encryption_mode": 1, // 1 for CTR, 2 for XTS mode
        "m4_secure_boot_enable": 1,
        "m4_digital_signature_validation": 1
}
```

5.4.3 Example JSON file with security parameters of MBR

The following example shows the structure and the available fields:

```
{
  "puf_activation_code_addr": 8192,
    "efuse_data": {
        "m4_digital_signature_validation": 1,
        "m4_encrypt_firmware": 1,
        "m4_fw_encryption_mode": 1,
        "m4_secure_boot_enable": 1,
        "ta_digital_signature_validation": 1,
        "ta_encrypt_firmware": 1,
        "ta_secure_boot_enable": 1
    },
}
```

You should create a JSON file using the preceding fields shown and configure it based on their security requirements and by referring to Security Levels section. The following section refers to this JSON file as **updated-mbr-fields.json** file

5.4.4 Programming MBR with security parameters

The security configurations enabled based on the security level chosen should be placed in a **.json** file. Refer to the example JSON file in the section Example JSON file with security parameters of MBR.

Use the following command to update the MBR with the desired security configurations. Once this command is issued, it updates the existing MBR with updated-mbr-fields.json file parameters

Syntax:

```
commander manufacturing provision --mbr <filename.bin|default> --data <updated-mbr-
fields.json> -d <full opn> [--skipload] [--pinset n]
```

Example

```
commander manufacturing provision --mbr  ta_mbr_SiWG917M111MGTBA.bin --data updated-mbr-
fields.json -d SiWG917M111MGTBA
```

- The updated-mbr-fields.json file provided should be updated by the user with desired security levels as mentioned in Security Levels section

5.5 Enable Security Configurations in NWP and M4 Firmware Images

When the security configurations are enabled, the device expects the NWP and the M4 firmware image files also to have the same configurations enabled. If you try to flash the mismatched security configurations, the bootloader returns an error.

The keys generated in the section Generate Static Keys from Simplicity Commander are used here to secure NWP and M4 images

5.5.1 Secure NWP Image

The following command is to secure the NWP firmware image.

Syntax:

```
commander rps convert <filename.rps> --taapp <original non-encrypted TA rps> --mic
<keys.json> --encrypt <keys.json> --sign <keys.json>
```

5.5.2 Secure M4 Image

The following command is to secure the M4 firmware image.

Syntax:

```
commander rps convert <filename.rps> --app  <original non-encrypted M4 rps> --mic
<keys.json> --encrypt <keys.json> --sign <keys.json>
```

5.6 Example: SiWG917M111MGTBA (Common Flash Mode)

The manufacturing procedure with security configurations for SiWG917M111MGTBA in Common flash mode is as follows:

1. **Backup – IPMU Calibration**
   - `commander manufacturing read taipmu --out filename.bin -d SiWG917M111MGTBA`
2. **MBR Provisioning**
   - `commander manufacturing provision --mbr default -d SiWG917M111MGTBA`
3. **Security Key Programming**
   - **Activation Code Generation for PUF (PUF Initialization)**
     `commander manufacturing init --mbr ta_mbr_SiWG917M111MGTBA.bin -d SiWG917M111MGTBA`
   - **Power cycle the device**
   - **Generate Keys with Simplicity Commander**
     `commander util genkeyconfig --outfile commanderkeys.json -d SiWG917M111MGTBA`
   - **Provision Static Keys**
     `commander manufacturing provision --keys commanderkeys.json -d SiWG917M111MGTBA`
4. **MBR Programming to Enable Security Configurations**
   - **Security Level 3 enabled in the updated_mbr_fields.json file**
     `commander manufacturing provision --mbr  ta_mbr_SiWG917M111MGTBA.bin --data updated_mbr_fields.json -d SiWG917M111MGTBA`
5. **RF Calibration – refer to section RF Calibration.**
6. **Convert NWP and M4 images to match Security Level 3**
   - **Security - NWP Image:**
     `commander rps convert secured_SiWG917-B.2.12.2.1.0.9.rps --taapp SiWG917-B.2.12.2.1.0.9.rps  --mic commanderkeys.json --encrypt commanderkeys.json --sign commanderkeys.json`
   - **Security - M4 Image:**
     `commander rps convert secured_si91x_hello_world.rps --app  si91x_hello_world.rps --mic commanderkeys.json --encrypt commanderkeys.json --sign commanderkeys.json`
7. **Flash NWP and M4 Images**
   - **TA Image:**
     `commander rps load secured_SiWG917-B.2.12.2.1.0.9.rps -d SiWG917M111MGTBA`
   - **M4 Image:**
     `commander rps load secured_si91x_hello_world.rps -d SiWG917M111MGTBA`

Note: If you want to update the boot configurations in eFuse refer to the section Boot Configurations Update – eFuse and for debug lock refer to the section Debug Lock.

5.7 Disable Security

The disabling of security does not erase the PUF activation code or erase keys, only re-writes the MBR. If the security configurations are programmed in eFuse, you cannot disable the security.

The following command is to disable the security.

Syntax:

```
commander manufacturing provision --mbr default -d <full opn> [--skipload] [--pinset n]
```

Note: If you want to enable security again, you need to give only one command, refer to the section Programming MBR with security parameters.

6 Combined Image (NWP + M4)

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

- Combined Image RPS header format is same as the M4 RPS header format with few reserved bytes changed.
- Signature for complete combined image is calculated and appended at the end of image.
- MIC computation and signature maintains the integrity and confidentiality of the combined image.
- Encryption of combined image is discarded as it would add overhead for firmware to decrypt and store into flash location. The TA and M4 images are encrypted individually.

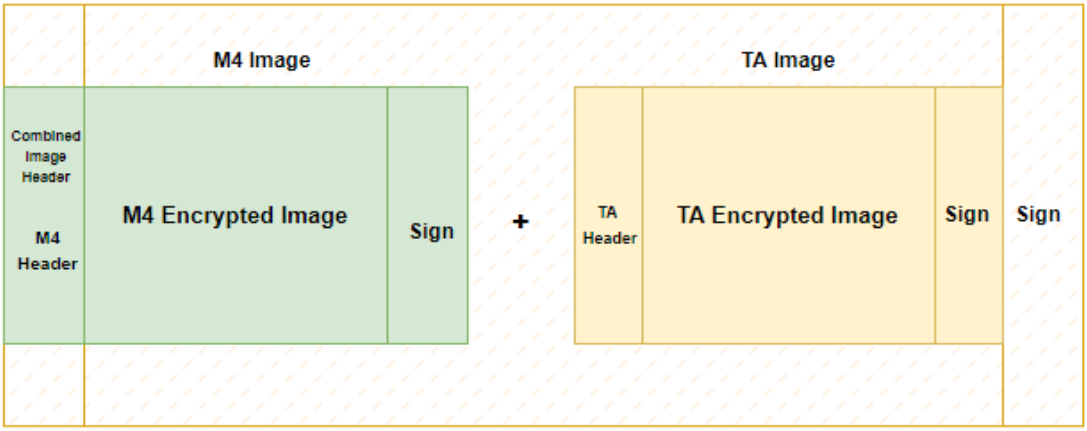


Figure 6.1: Combined Image (NWP + M4)

The following table lists the process to create and flash a combined image for Common Flash and Dual Flash mode devices.

Steps	Description	Security	Command (Syntax)
1	TA Image with combined flag	Disabled	`commander rps convert <filename.rps> --taapp <original non-encrypted TA rps> --combinedimage`
		Enabled	`commander rps convert <filename.rps> --taapp <original non-encrypted TA rps> --mic <keys.json> --encrypt <keys.json> --sign <keys.json> --combinedimage`
2	M4 Image with combined flag	Disabled	`commander rps convert <filename.rps> --app <original non-encrypted M4 rps> --combinedimage`
		Enabled	`commander rps convert <filename.rps> --app <original non-encrypted M4 rps> --mic <keys.json> --encrypt <keys.json> --sign <keys.json> --combinedimage`
3	Combine TA and M4 Images	Disabled	`commander rps convert <combined_image.rps> --app <m4_image_combinedflag.rps> --taapp <ta_image_combinedflag.rps>`
		Enabled	`commander rps convert <combined_image.rps> --app <m4_image_combinedflag.rps> --taapp <ta_image_combinedflag.rps> --sign <keys.json>`
4	Flash the Combined Image		You can flash the combined image using the OTA process only. Refer to the HTTP OTAF example.

Table 6.1: Combined Image Steps

7 Boot Configurations Update – eFuse

This section provides the commands to read and write eFuse data along with the list of the available configurations in eFuse that can be programmed by the user.

Once all the development and testing work is complete, the next step would be to program the eFuses. It is important to note that this is a permanent and one-time operation.

SiWG917 has eFuse memory of 1024 bytes. This memory is divided into four sections with control to enable write locks for each of these sections.

eFuse section	Byte Range
R1	0-127
R2	128-255
R3	256-767
R4	768-1024

Note: You cannot write to eFuse through the M4 application image or through NWP via APIs. You must have an active JTAG connection to write the eFuse bits.

7.1 Read eFuse Data

You can read the eFuse data/contents to an **.json** (for example, efuse_value.json) file.

The command to read efuse data is as follows.

Syntax:

```
commander manufacturing read efuse --out efuse_value.json -d <full opn>
```

Example

- `commander manufacturing read efuse –out efuse_value.json -d SiWG917M111MGTBA`

7.2 Write eFuse Data

In the case of eFuse (one time programmable) data, simplicity commander CLI first checks whether the requested update is possible (since bits can only be set to 1 and never cleared). Then it will ask for your confirmation.

The possible eFuse configurations are mentioned in the section Possible Boot Configurations.

You can modify the eFuse bits in the efuse_value.json file which is read in the section Read eFuse Data and use the following command which are used to write efuse data into flash.

Syntax:

- **`commander manufacturing write efuse --data efuse_value.json -d <OPN Number> [--skipload] [--pinset n] [-s jlinkserialno] [--noprompt] [--dryrun]`**
- **--noprompt: It is possible to skip the confirmation, although users need to note that this is a one-time operation.**
- **--dryrun: It is possible to check the results of the operation before physically going ahead with the one-time programming.**

Example

- `commander manufacturing write efuse --data efuse_value.json -d SiWG917M111MGTBA –skipload --dryrun`

7.3 Possible Boot Configurations

The following table shows the available user configurable eFuse bits.

Note: The following fields can also be configured in MBR, you can modify them during development in the MBR and program in the eFuse once all the development and testing is completed.

#	Field	Description	Number of bits	Default Setting	Default value in NWP eFuse
1	safe_upgrade_frm_host	When set to 1- upgrade the • NWP and M4 images from backup, instead of overwriting directly • M4 OTA and Public keys from backup, instead of overwriting directly When set to 0, the images or the keys will be directly overwritten to the target locations based on the target address which comes from RPS header.	1	Enabled	1
2	ta_secure_boot_enable	Enable NWP Secure Boot 1: Secure boot is enabled for NWP 0: Secure boot is disabled in NWP	1	Disabled	0
3	ta_anti_roll_back	1: Anti roll back check is enabled for NWP firmware (will not allow to update old versions) 0: Anti roll back check is not enabled for NWP firmware	1	Anti roll back check is not enabled for NWP firmware	0
4	ta_digital_signature_validation	1: Digital signature validation is enabled for NWP firmware 0: Digital signature validation is disabled for NWP firmware	1	Digital signature validation is disabled for NWP firmware	0
5	m4_anti_roll_back	1: Anti roll back check is enabled for M4 firmware (will not allow to update old versions) 0: Anti roll back check is not enabled for M4 firmware	1	Disabled	0
6	m4_digital_signature_validation	1: Digital signature validation is enabled for M4 firmware 0: Digital signature validation is disabled for M4 firmware	1	Disabled	0
7	enable_autobaud_detection	1: Auto baud rate detection for UART is enabled 0: Auto baud disabled - Default Config - 115200 bps	1	Disabled	0
8	ta_encrypt_firmware	00: NWP firmware stored in unencrypted form in flash 01: NWP firmware stored in encrypted form in flash with CTR mode encryption 10: NWP firmware stored in encrypted form in flash with XTS mode encryption	2	Disabled : NWP firmware stored in unencrypted form in flash	0

		11: Reserved			
9	m4_flash_present	This field is used to indicate that the M4 has a separate flash(the device is dual flash) 1- M4 has flash 0- M4 doesn't have flash	1	Disabled : M4 doesn't have flash - it's a common flash device	0
10	m4_flash_pinset	M4 flash pin set 1: GPIO_46_TO_51 2: GPIO_52_TO_57	4	Not applicable as the m4_flash_present bit is not set by default, device will not use the values from this field	0
11	m4_secure_boot_enable	1: Secure boot is enabled for M4 0: Secure boot is not enabled for M4	1	Secure boot is not enabled for M4	0
12	m4_encrypt_firmware	0: M4 Firmware stored in unencrypted form in flash 1: M4 Firmware stored in encrypted form in flash	1	M4 Firmware is stored in unencrypted form in the flash	0
13	common_flash_enabled	Enable Common flash configuration. Note: If enabled from eFuse, cannot be overridden from MBR 1: Common flash mode is enabled 0: Common flash mode not selected The value written for this bit during manufacturing changes between device types. • If the OPN is no in-package OPN, the eFuse configs which are written in eFuse will have this value set as 0 so that customer can update it when a flash is connected. The customer can decide to use Common flash or dual flash mode. This bit is set to 0 so that customer can decide which mode can be selected later. • If the OPN is with flash, the eFuse configs are written as part of MBR configs in flash. So, in that case the value is set as 1.	1	If writing in eFuse, the value is set to 0 If writing to MBR, the value is set to 1	0
14	ta_otp_lock	NWP eFuse programming lock for R1 address region (0-127) 1: NWP eFuse programming is locked 0: NWP eFuse programming is not locked When eFuse is locked, won't be able to modify secure boot related parameters present in R1 address region	1	NWP eFuse programming is not locked for 0-127 range	0

15	disable_ta_jtag	Disable NWP JTAG interface. Write 0 for enabling NWP JTAG. 0 : Enable NWP JTAG interface 1 : Disable NWP JTAG Interface	1	NWP JTAG interface enabled	0
16	otp_lock_1	NWP eFuse programming lock for address region (128-255) of eFuse. This is enabled by programming the eFuse offset 255 with value 1 1: NWP eFuse programming is locked 0: NWP eFuse programming is not locked When eFuse is locked, won't be able to modify secure boot related parameters present in R2 address region	1	NWP eFuse programming is not locked in range 128-255	0
17	otp_lock_2	NWP eFuse programming lock for address region (256-767). This is enabled by programming the eFuse offset 766 with value 1 1: NWP eFuse programming is locked 0: NWP eFuse programming is not locked When eFuse is locked, won't be able to modify secure boot related parameters present in R3 address region	1	NWP eFuse programming is not locked in range 256-767	0
18	otp_lock_3	NWP eFuse programming lock for R4 address region (768-1024). This is enabled by programming the eFuse offset 1023 with value 1 1: NWP eFuse programming is locked 0: NWP eFuse programming is not locked When eFuse is locked, won't be able to modify secure boot related parameters present in R4 address region	1	NWP eFuse programming is not locked in range 768-1024	0
19	m4_otp_programming_lock	1: M4 eFuse programming is locked 0: M4 eFuse programming is not locked When eFuse is locked, won't be able to modify secure boot related parameters	1	M4 eFuse programming is not locked	0
20	disable_m4_jtag	Disable M4 JTAG interface. 0 : Enable M4 JTAG interface, which is default 1 : Disable M4 JTAG Interface	1	JTAG Enabled	0
21	disable_m4_access_frm_tass_sec	When set, M4 can't access TASS memory or registers except for host communication registers. **Note:** Setting this e-Fuse must be the last step in	1	This is secure zone which is disabled by default	0

		manufacturing. Once set, any further operations with manufacturing tools will not be possible.			
22	mbr_mic_sign_enable	MIC/Signature for combined region of MBR, Boot descriptor and key descriptor table. 00 : CRC 01: MIC - Applicable only when OTP Symmetric key is written 10: reserved 11 : Sign - Applicable only when OTP public key is written	2	CRC check enabled	00
23	MIC protected Content length	This field depicts how much space is MIC protected using OTP key stored in eFuse at offset 659. The MIC value is stored in eFuse at offset 643	4	Not applicable	0000
24	m4_firmware_encryption_mode	Indicates M4 firmware encryption mode. Valid only if m4_encrypt_firmware is set to 1 else ignored 00 : NA 01: Firmware stored in encrypted form in M4 flash-CTR Mode 10: Firmware stored in encrypted form in M4 flash-XTS Mode 11: Reserved.	2	Not applicable as the m4_encrypt_firmware is disabled by default. This field will be neglected	00
25	disable PSRAM encryption	Contents in PSRAM will be encrypted if this bit is enabled. This field explicitly disables PSRAM encryption irrespective of M4 firmware encryption is enabled or not while firmware is in flash 0: PSRAM encryption using same key config as M4 flash. PSRAM encryption mode will be CTR independent of M4 flash encryption mode. 1: PSRAM encryption disabled.	1	PSRAM encryption enabled	0

Table 7.1: Possible Boot Configurations

8 RF Calibration

This section describes the RF calibration steps to adjust the XO CTUNE, gain offset, and EVM offset in the SiWG917 device.

Note:

- Refer to the section Manufacturing Utility Commands – Parameters for the description of the parameters used in the following sections of commands.
- The --skipload parameter is used to skip loading the manufacturing firmware. This will save manufacturing time during mass production.
- The first command given to the device when powered ON, should not have --skipload, since the manufacturing firmware needs to be loaded.

8.1 Steps for CTUNE Adjustments

The following table shows the steps for adjusting the CTUNE and storing it in flash or eFuse.

Step	Description	Command (Syntax)
1	Setup the radio and start transmission	commander manufacturing radio --channel <1-14> --power <1-31/127> --phy <xMBPS,MCSn,CW> --noburst --internalant --start –serialinterface -d <OPN Number>
2	Measure the frequency	
3	Adjust the CTUNE value *Offset = measured frequency (in kHz) – channel frequency*	commander manufacturing xocal --offset <frequency offset in kHz> --serialinterface --skipload -d <OPN Number>
4	Check the channel frequency is within expectations. If not, repeat from step 2, while adjusting the CTUNE value.	
5	Store the CTUNE value (The radio transmission will stop)	commander manufacturing xocal --store --storeinefuse --skipload --serialinterface -d <OPN Number>
6	Store the XO CTUNE value with CTUNE override, if you have a CTUNE value (Optional)	commander manufacturing xocal --ctuneoverride <ctune value> --store --storeinefuse --skipload --serialinterface -d <OPN Number>

Table 8.1: CTUNE Adjustments steps

8.1.1 Example: SiWG917M111MGTBA – CTUNE adjustment steps

1. **Setup the radio and start transmission**
   commander manufacturing radio --channel 1 --power 10 --phy 1MBPS –start -d SiWG917M111MGTBA
2. **Measure the frequency**
3. **Adjust the CTUNE value (*Offset = measured frequency (in kHz) – 2412000)***
   commander manufacturing xocal --offset <Offset> --skipload -d SiWG917M111MGTBA
4. **Store the CTUNE value**
   - Store in Flash
     commander manufacturing xocal --store --skipload -d SiWG917M111MGTBA
   - Store in eFuse
     commander manufacturing xocal --store --storeinefuse --skipload -d SiWG917M111MGTBA

8.2 Steps for Gain Offset Adjustments

The following table lists the steps for adjusting the gain offset and storing it in flash or eFuse.

Step	Description	Command (Syntax)
1	Setup the radio and start transmission	commander manufacturing radio --channel <1-14> --power <1-31/127> --phy <xMBPS,MCSn,CW> --noburst --internalant --start –serialinterface -d <OPN Number>
2	Measure the output power	
3	Calculate the offset to meet the expected output power for channel 1,6,11 and 14. Offset = ceil ((output power measure (dBm) + cable loss -- 16) * 2)	
4	Store gain offset for channel 1, 6, 11, 14	commander manufacturing gain --ch1 <gain ch.1> --ch6 <gain ch.6> --ch11 <gain ch.11> --ch14 <gain ch.14> --store --storeinefuse --skipload --serialinterface -d <OPN Number>
5	Stop the radio transmission	commander manufacturing radio --stop --skipload -d <OPN Number>

Table 8.1: Gain Offset Adjustments steps

8.2.1 Example: SiWG917M111MGTBA – Gain Offset adjustment steps

1. **Setup the radio and start transmission**
   commander manufacturing radio --channel 1 --power 16 --phy 1MBPS –start -d SiWG917M111MGTBA
2. **Measure the output power**
3. **Calculate the offset to meet the expected output power**
   Offset = ceil ((output power measure (dBm) + cable loss -- 16) * 2)
4. **Store the gain offset value**
   - Store in Flash
     commander manufacturing gain --ch1 <gain ch.1> --ch6 <gain ch.6> --ch11 <gain ch.11> --ch14 <gain ch.14> --store –skipload -d SiWG917M111MGTBA
   - Store in eFuse
     commander manufacturing gain --ch1 <gain ch.1> --ch6 <gain ch.6> --ch11 <gain ch.11> --ch14 <gain ch.14> --store --storeinefuse --skipload -d SiWG917M111MGTBA
5. Stop radio transmission
   commander manufacturing radio --stop --skipload -d SiWG917M111MGTBA

Note:

- The gain offset updated in flash with the preceding command in step 4. To force the usage of this gain offset in flash even though the whole calibration data is in eFuse, You need to give the EVM offset command mentioned section EVM Offset.
- This serves as an enable for gain offset to pick from flash when the remaining calibration data is in eFuse.
- There can be a variation of up to +/- 2dB in power across parts at Typical/Room temperature.
- It is recommended that the user makes the "Customer Gain-offset" correction on customer-end products to correct for IC part-to-part variation and insertion-loss variations in the RF front-end on customer boards. This calibration process would ensure accurate Transmit power control for Regulatory compliance @ volume production.

8.3 EVM Offset

Following is the syntax of the command to set the EVM offset. You can apply the offset for the rates where EVM is passed. Any of the offsets mentioned further for the respective data rates will have the offset applied.

Syntax:

commander manufacturing evmoffset --off0 <offset 11B> --off1 <offset 11G,24Mbps 11N,MCS0-2> --off2 <offset 11G,54Mbps 11N,MCS3-7> --off3 <offset 11N,MCS0> --off4 <offset 11N,MCS7> --store –storeinefuse --serialinterface -d <OPN Number>

Example

- **commander manufacturing evmoffset --off0 0 --off1 0 --off2 0 --off3 0 --off4 0 –store -d SiWG917M111MGTBA**
- This command updates the flash with all the evm offsets with 0 along with a magic byte.

Note: This magic byte when used in flash acts as a enabler for picking gain offset from flash even the whole calibration data is in the eFuse.

9 Manufacturing Utility – Commands

In this section some of the useful and commonly used manufacturing utility commands are mentioned.

9.1 SiWG917 Info

You can get the manufacturing and device information of the SiWG917 by using the following commands.

9.1.1 Manufacturing Info

The command provides the following information of the SiWG917 OPN.

- OPN
- Wi-Fi MAC address
- Wi-Fi MAC address (customer)
- Flash size
- Flash variant
- Flash type
- NWP firmware version
- MBR variant
- Manufacturing SW version
- Application region start address
- Application code start address
- Application region end address
- Flash configuration
- PSRAM option
- Mode
- NWP roll-back prevention
- NWP digital signature validation
- NWP firmware encryption
- NWP secure boot
- Application roll-back prevention
- Application digital signature validation
- Application code encryption
- Application secure boot

Syntax:

commander manufacturing info -d <OPN Number>

Note: The Manufacturing SW version provided by the above command is embedded into the SiWG917 during chip manufacturing at PTE. This serves as the factory default setting in the production testing software.

9.1.2 Device Info

The command provides the following information of the SiWG917 OPN.

- Part Number
- Product Revision
- Flash Size
- SRAM Size
- Unique ID

Syntax:

commander device info -d <OPN Number>

9.2 PSRAM Pinset Update

In any of the SiWG917 OPN except SiWG917M121XGTBA, and SiWG917M141XGTBA, by default the external PSRAM is configured on pins (52 to 57), that is pinset 3. If you would like to configure the PSRAM on pins (46 to 51), that is pinset 2, refer to the following steps. The MBR is updated with the psrampinsetupdate.json file.

9.2.1 Write TA MBR

The following command is used to write TA MBR.

Syntax:

commander manufacturing write tambr --data psrampinsetupdate.json [--skipload] -d <OPN Number>

Example

- commander manufacturing write tambr --data psrampinsetupdate.json -d SiWG917M111MGTBA

9.2.2 Write M4 MBR

The following command is used to write M4 MBR.

Syntax:

commander manufacturing write <m4mbrcf|m4mbrdf> --data psrampinsetupdate.json [--skipload] [--pinset n] -d <OPN Number>

Example

- commander manufacturing write m4mbrcf --data psrampinsetupdate.json -d SiWG917M111MGTBA

9.3 User Data – Update

The user data space in the M4 starts at 0x047f7000, and ends at 0x047fbfff. You can use the following commands to write, read, write to a location and erase user data.

9.3.1 Write User Data

The following command is used to read user data.

Syntax:

commander manufacturing write userdata --data <filename.bin>

Example

- commander manufacturing write userdata --data updateuserdata.bin

9.3.2 Read User Data

The following command is used to write user data.

Syntax:

commander manufacturing read userdata

9.3.3 Write User Data to a Location

The following command is used to write user data.

Syntax:

commander manufacturing write userdata --data <filename.bin> --address <location_address>

Example

- commander manufacturing write userdata --data updateuserdata.bin --address 0x047fbfe0

9.3.4 Erase User Data

The following command is used to write user data.

Syntax:

commander manufacturing erase userdata

9.4 MAC Address – Update

When you write the MAC address to the flash or eFuse, it is stored as the Wi-Fi MAC address (customer). The SiWG917 uses this MAC address instead of the default one. This process allows you to customize the MAC address for your specific use case.

You can give the command “commander manufacturing info” in the commander CLI and view both the default Wi-Fi MAC address, and Wi-fi MAC address (customer).

The Wi-Fi MAC address is updated using the manufacturing utility using the following structure and available fields which is stored as a **.json** file. For example,file is provided here: **MAC_Address_update_fields.json**.

You can modify MAC address in the the **customer_wlan_mac_address** field as per your requirement.

```
{
    "customer_wlan_info_magic_byte": 90,
    "customer_wlan_mac_address": "112233445566"
}
```

9.4.1 Write to Flash

The following command can be used to update the Wi-Fi MAC address in the flash

Syntax:

commander manufacturing write efusecopy --data MAC_Address_update_fields.json -d <full opn>

Example:

commander manufacturing write efusecopy --data MAC_Address_update_fields.json -d SiWG917M111MGTBA

9.4.2 Write to eFuse

The following command can be used to update the Wi-Fi MAC address in the eFuse.

Syntax:

commander manufacturing write efuse --data MAC_Address_update_fields.json -d <full opn>

Example:

commander manufacturing write efuse --data MAC_Address_update_fields.json -d SiWG917M111MGTBA

9.4.3 Read MAC Address

The following command can be used to read the Wi-Fi MAC address from the flash or eFuse.

Syntax:

commander manufacturing read <efusecopy|efuse> --property customer_wlan_mac_address -d <full opn>

Note:

- **efusecopy – For flash**
- **efuse – For eFuse**

Example:

- Read from flash

  commander manufacturing read efusecopy --property customer_wlan_mac_address -d SiWG917M111MGTBA

- Read from eFuse

  commander manufacturing read efusecopy --property customer_wlan_mac_address -d SiWG917M111MGTBA

9.5 Flash Type – Update

All the OPN MBR files except for SiWG917M110LGTBA contains configurations of GIGA flash by default. SiWG917M110LGTBA OPN will have XMC flash as default.

The flash configurations can be changed by the following steps.

9.5.1 For Macronix Flash

While loading MBR, add the "macronix.json" file in the following command using "--data" option.

Syntax:

commander manufacturing provision --mbr <filename.bin|default> --data macronix.json -d <full opn>

Example

commander manufacturing provision --mbr ta_mbr_SiWG917M111XGTBA.bin --data macronix.json

9.5.2 For XMC Flash

While loading MBR, add the "xmc.json" file in the following command using "--data" option.

Syntax:

commander manufacturing provision --mbr <filename.bin|default> --data xmc.json -d <full opn>

Example

commander manufacturing provision --mbr ta_mbr_SiWG917M111XGTBA.bin --data xmc.json

9.5.3 Change from XMC to GIGA

While loading MBR, add the "giga.json" file in the following command using "--data" option.

Syntax:

commander manufacturing provision --mbr <filename.bin|default> --data giga.json -d <full opn>

Example

commander manufacturing provision --mbr ta_mbr_SiWG917M110GTBA.bin --data giga.json

Note: By adding the "flash_pinset" field to the "qflash_config" section in the **json** files above, the flash pinset can be updated.

10 Possible Error Codes

Command	Error Code	Description	Cause of failure
MBR	0xa0ac	Verification failed from flash	If you entered wrong inputs to select pinset/damage MBR file.
Activation code	0xa0b0	Activation code generated failed	If the device is locked to use PUF, the device might throw an error indicating PUF enroll failure.
Intrinsic keys	0xa0b1	Intrinsic keys generation failed	It occurs if this command triggered before activation code command.
Static keys	0xa0b2	Static keys stored failed	It occurs if there are no intrinsic keys.
Update Firmware	108	Failed to load firmware	It occurs if header part of rps file is not proper.
Bootloader Timeout	102	Failed to load firmware	It occurs if the SiWG917 is in sleep or JTAG is not accessible. Keep the device in ISP mode to resolve this issue.
Start Radio	Cannot execute the radio command	Not able to start radio	This error is seen if the first command given to the device on power ON has --skipload in the command. Do not give the --skipload parameter in the first command given to the device each time when powered ON.

11 Revision History

Revision 1.3

November 2024

- Updated: 10. Possible Error Codes.
- Added a note for --skipload parameter in 8. RF Calibration.

Revision 1.2

November 2024

- Updated the complete document.
- Major changes
  - Added the following sections:
    - 2. Flash Mode Selection – No In-Package Flash OPN,
    - 7. Boot Configurations Update – eFuse
    - 9. Manufacturing Utility – Commands.
  - Manufacturing procedure explained for common flash and dual flash mode separately.

Revision 1.1

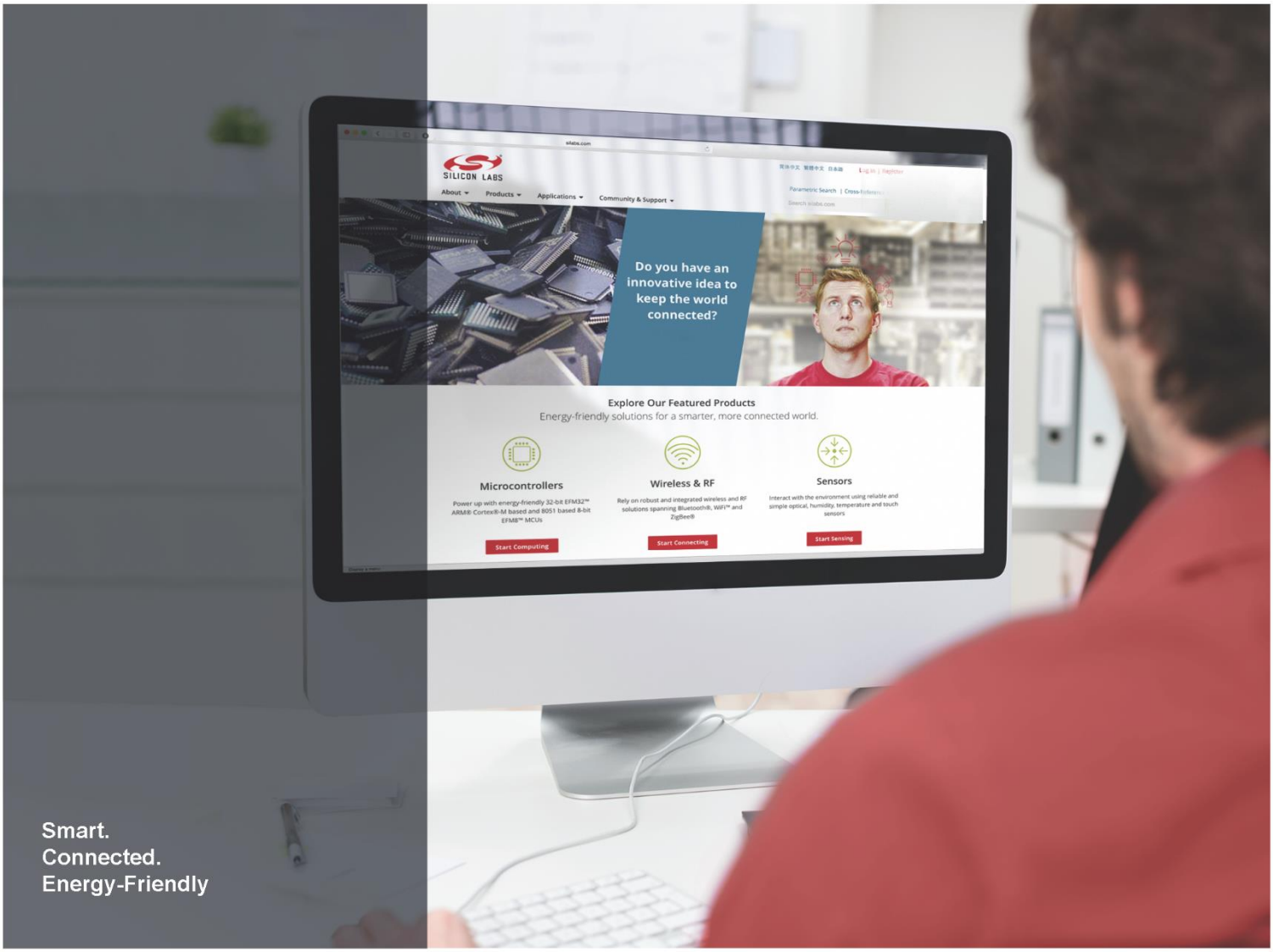
January 2024

- Updated: 8. RF Calibration.

Revision 1.0

December 2023

- Initial release



Products
www.silabs.com/products

Quality
www.silabs.com/quality

Support and Community
community.silabs.com

SILICON LABS

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

http://www.silabs.com