UG575: SiWx917 NCP Manufacturing Utility User Guide

Version 1.0

February 2024

Note: This content may contain offensive terminology that is now obsolete. Silicon Labs is replacing these terms with inclusive language wherever possible. For more information, visit www.silabs.com/about-us/inclusive-lexicon-project.

Contents

1 Introduction

SiWN917 (SiWx917 NCP) Network Co-Processor mode: In this configuration the application runs on the external microcontroller unit (MCU) host, and the connectivity stack runs on the SiWx917 chipset. In addition, in adherence to the Trusted Execution Environment architecture, this product allows customers to program the production information into the device under test (DUT), for example, like updating/loading the required Master Boot Record (MBR), Security Keys, Secure TA Firmware Update along with reading data from primary calibration location and write calibration data to flash.

1.1 Manufacturing Utility

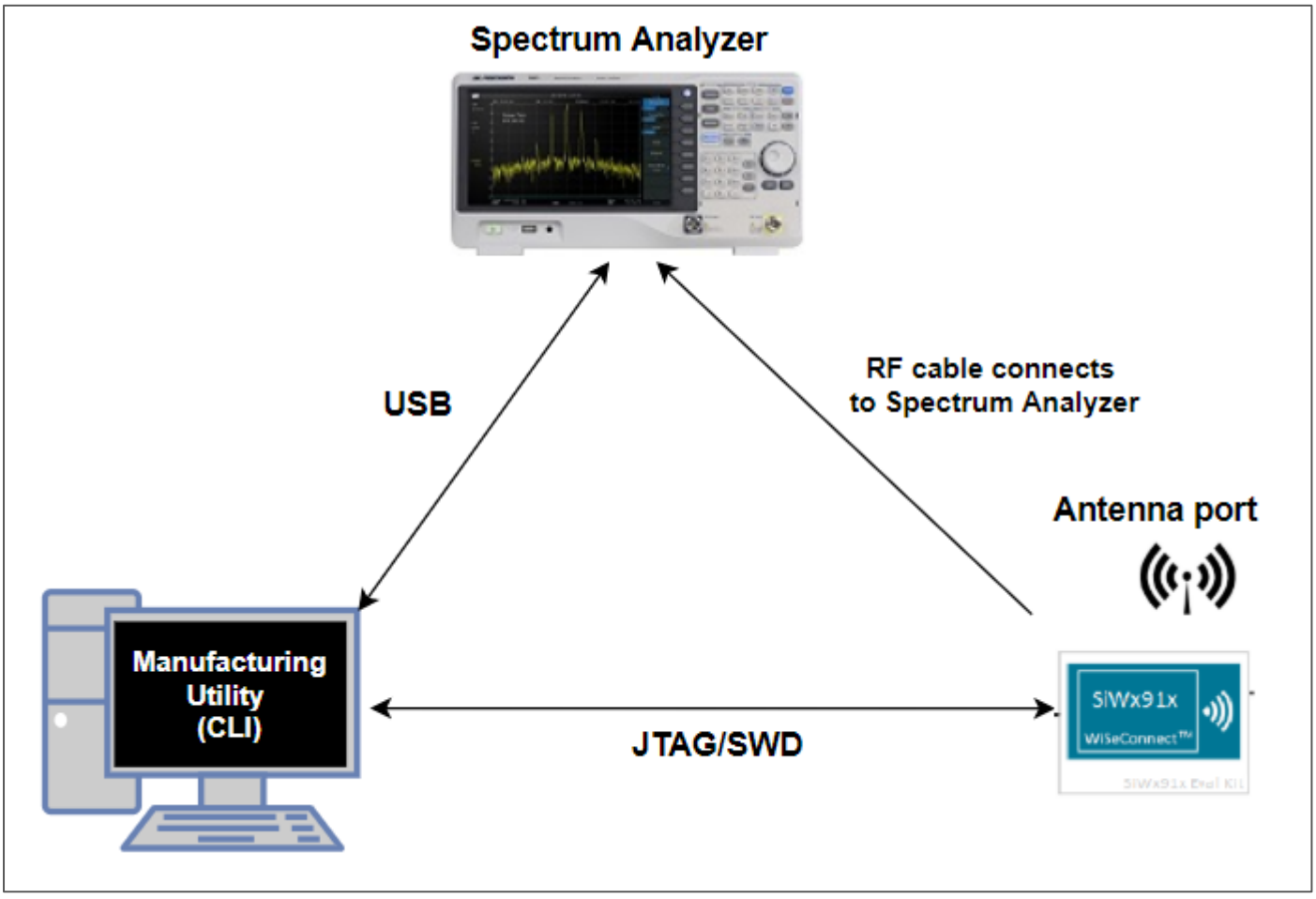
Manufacturing Utility or Simplicity Commander CLI (Command Line Interface) is used to update the required ***Master Boot Record (MBR)***, ***Enable the Security*** and ***Calibration*** in SiWN917.

This utility is enabled to work with Simplicity Commander through its Command Line Interface (CLI). Simplicity Commander CLI supports multiple commands for the user to program the production information into the device.

Master Boot Record (MBR): Stored in flash and contains information like clock frequencies, offsets of structures like FMC, eFuse copy, SPI configurations, External Flash details, etc. Any SiWx917 NCP that is shipped out of the factory will have a default MBR. Based on the OPN, the user can update it.

Enable the Security: The security fields in MBR, PUF Activation code, Key descriptors, and the Keys are to be programmed in the device while enabling the security features in the device.

Calibration: The CTUNE and gain offset adjustments can be done in burst and continuous mode in the SiWN917 using the manufacturing utility.



1.1.1 Simplicity Commander

Simplicity Commander is a single, all-purpose tool to be used in a production environment. It can be invoked using a simple Command Line Interface (CLI) that is also scriptable.

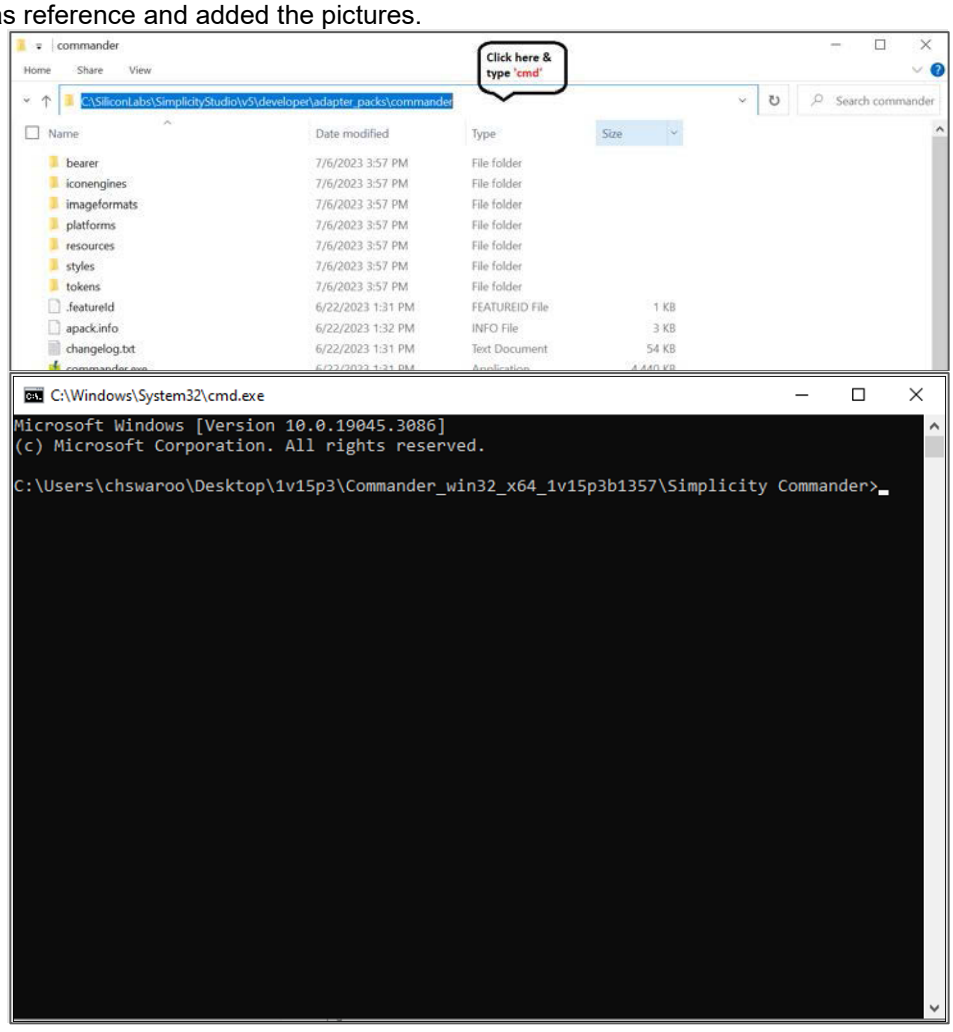
Simplicity Commander enables users to complete these essential tasks:

- Flash the application
- Configure the application
- Create binaries for production

1.1.2 Simplicity Commander CLI

To execute Simplicity Commander commands, start a Windows command window, and change to the Simplicity Commander directory, or you can also open Commander as described below.

- Navigate to C:\SiliconLabs\SimplicityStudio\v5\developer\adapter_packs\commander folder.
- Type ‘cmd’ in the address bar and press Enter key. The Simplicity Commander CLI will be seen. We have taken Windows PC as reference and added the pictures.



2 Baud Rate Configuration in EFR32

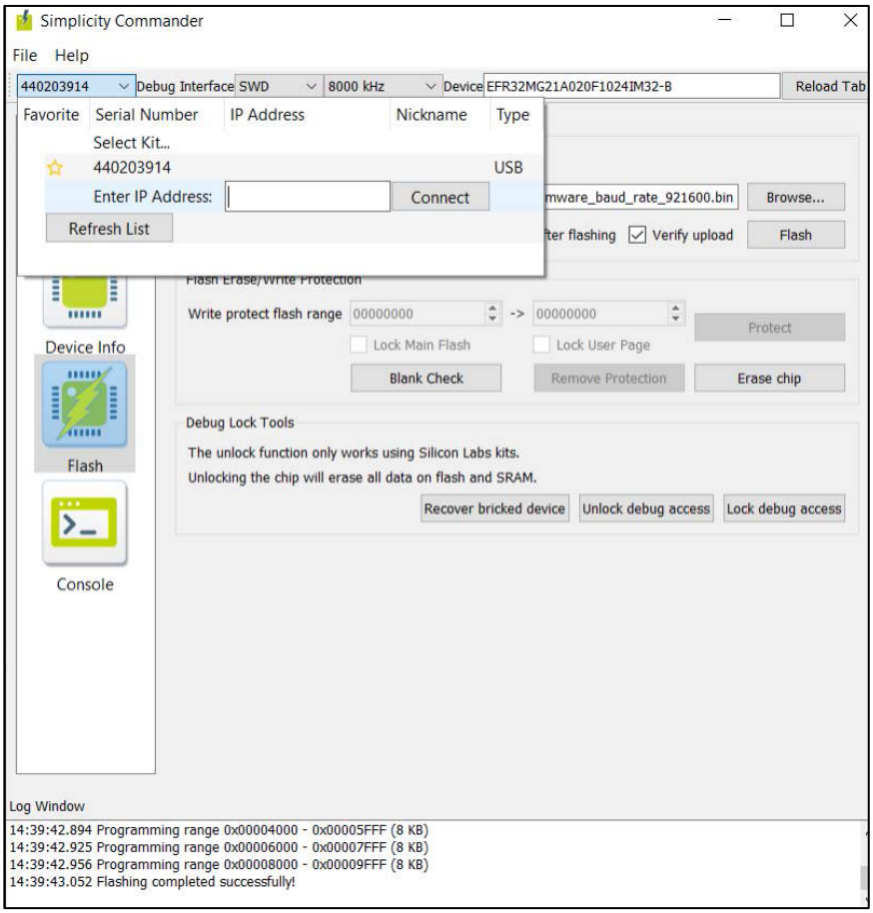
The below steps are required to increase the baud rate of the host MCU (EFR32MG21/EFR32MG24)

1. Plug in the EFR32 and use the command below to check the IP address.

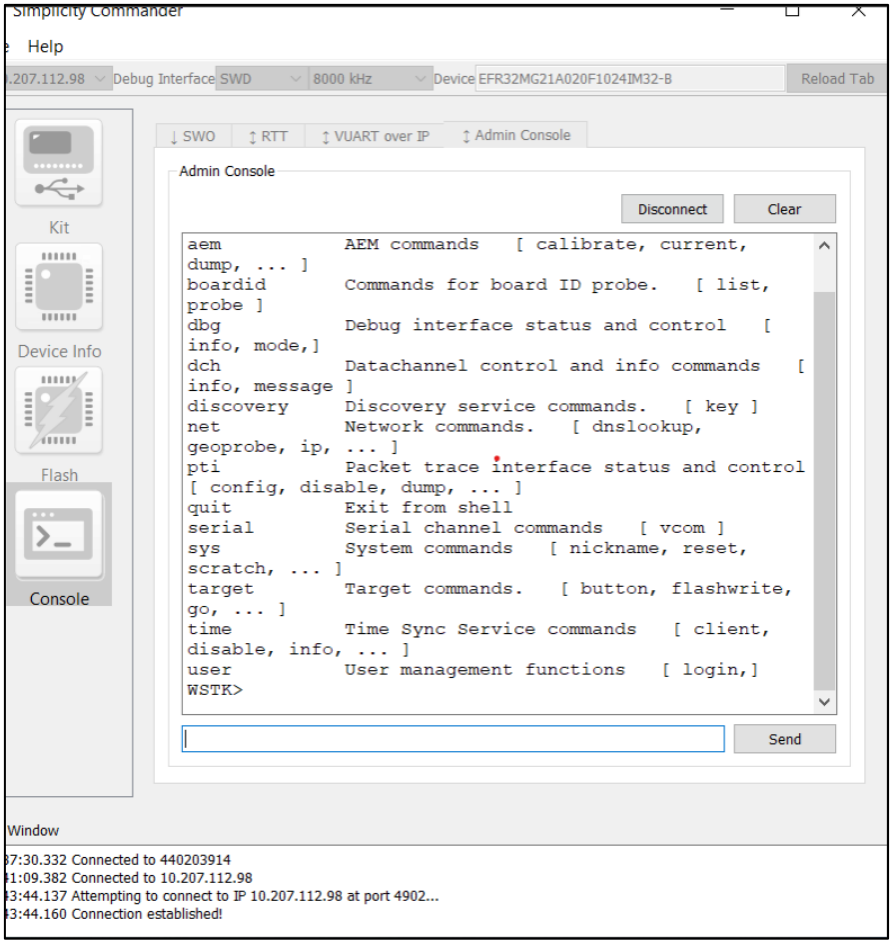
Command: **commander adapter ip --serialno <serial number>**

```
C:\Users\srnayudu\Downloads\Commander_win32_x64_1v16p0b14\Simplicity Commander>commander adapter ip
WARNING: No serial number or IP address given, cannot lock access to adapter.
IP Address: 10.207.112.146/16
Gateway    : 10.207.112.1
DNS Server: 10.207.110.11
DONE
```

2. After getting the IP address, open "commander GUI", and enter the IP address in the field provided for “Enter IP Address:” and then go to the console tab in the left menu and click on "admin console."



3. Next click on "connect” and send the command "help."



4. Next send the command "**serial vcom config speed 921600**"

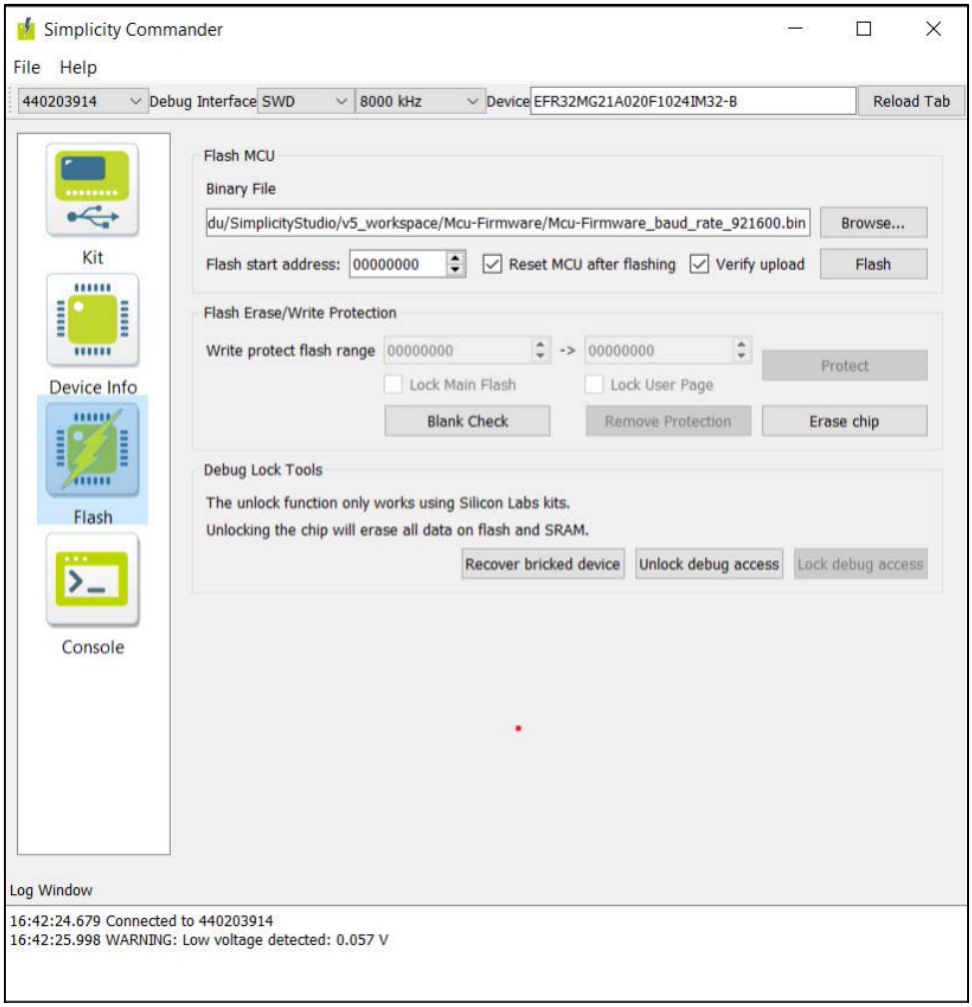
2.1 Flash EFR MCU Firmware to EFR Board

Below is the Firmware for EFR MCU Firmware.

MG21 Board – Mcu-Firmware.bin

MG24 Board - wifi_CMFG_Tool_App.bin

Download the MCU firmware based on the requirements and flash it in the EFR32 host MCU.
1. Open "simplicity commander" by navigating to "Tools"
2. click on "Select Kit" and then select "Flash" and Browse the .bin file to flash.
3. Then click on "Flash"



2.2 Error Recovery Procedure

If the board goes to reset mode in between sending commands, it means that commander is taking some files backup i.e., MBR, bootdesc, keydesctable, efusecopy, taipmu files.

These backup files can be found at location,
"**C:/Users/xxxxx/AppData/Local/SiliconLabs/commander/BackupStore/**" and the filename format is **Macaddress_field_YearMonthDay-HourMinuteSecond.bin**

```
WARNING: No serial number or IP address given, cannot lock access to adapter.
Reading 1024 bytes from 0x040003e0
Reading 496 bytes from 0x04000000
Reading 112 bytes from 0x040001f0
Reading 88 bytes from 0x04000300
Reading 1024 bytes from 0x040003e0
Reading 58 bytes from 0x04000561
Loading TA firmware. Writing 321776 bytes to 0x00400000...
Setting TA PC to 0...
Writing 0xA0AC to 0x41050034...
Command: 0xa022, Address: 0x00000000, Length: 0x00000000, Flags: 0x00000010
Writing 12 bytes to 0x0042d000...
Interrupting TA...
Timeout waiting for firmware to complete command (0x4105003c = 0x0000, PC0 = 0x79cd2a, PC2 = 0xb22c)
ERROR: Update flash info failed.
ERROR: Could not load TA manufacturing firmware.
ERROR: Could not execute provisioning sequence!

================================================================================
The failed write attempt MAY have corrupted the data on your device (6C:5C:B1:C4:36:5C).
The device's data in the affected regions was safely backed up before the write,
and is located here:
  C:/Users/srnayudu/AppData/Local/SiliconLabs/commander/BackupStore/6C5CB1C4365C_tambr_20231116-184458.bin
  C:/Users/srnayudu/AppData/Local/SiliconLabs/commander/BackupStore/6C5CB1C4365C_bootdesc_20231116-184458.bin
  C:/Users/srnayudu/AppData/Local/SiliconLabs/commander/BackupStore/6C5CB1C4365C_keydesctable_20231116-184458.bin
  C:/Users/srnayudu/AppData/Local/SiliconLabs/commander/BackupStore/6C5CB1C4365C_efusecopy_20231116-184458.bin
  C:/Users/srnayudu/AppData/Local/SiliconLabs/commander/BackupStore/6C5CB1C4365C_taipmu_20231116-184458.bin
================================================================================
DONE
```

3 Manufacturing Procedure – Non-Secure Device

MBR loading can be done based on the OPN number: a user can program the MBR by specifying the default option with the OPN number. This will load the factory default MBR. If the users receive a flash-less device or OPNs which have an external flash, MBR loading should not be done based on the OPN number.

MBR is to be loaded from a file given as input in **.bin** format.

Here is the sequence to follow for programming the devices in which security is disabled.

- Write TA MBR
- Write the Calibration data to the TA flash.

3.1 Write TA MBR

Below is the Syntax for writing the TA MBR.

Syntax:

commander manufacturing provision --mbr <filename.bin|default> -d <full opn> [--skipload] [--pinset n] --serialinterface --skipinit

Note:

default → To load MBR based on the OPN number given as input,

filename.bin → To load MBR from a bin file. In this case OPN number input not required.

Example:

1. Load the default MBR.
   Command: commander manufacturing provision --mbr **default** -d **SiWN917M100LGTBA --serialinterface** --skipinit
2. Load a specific MBR
   Command: commander manufacturing provision --mbr **ta_mbr_SiWN917M100LGTBA.bin** -d **SiWN917M100LGTBA --serialinterface** --skipinit

4 Manufacturing Procedure – Secure Device

To do the production programming of security enabled parts, we need to update the security key programming in the SiWN917 and then program the MBR by selecting security levels.

The following steps are to be followed for securing your SiWN917 device.

1. Back-up the original TA and eFuse data/content (Optional)
2. Security Key Programming
3. Add Security to TA firmware image files.
4. Flash the Secured images into flash.

4.1 Back-up the Original TA and eFuse data/content (Optional)

Recommendation: Back-up the original TA and the eFuse data/content before doing any updates. Backing up these is optional.

Field	Syntax	Example
TA MBR back-up	**commander manufacturing read <tambr> --out <filename.bin> serialinterface --skipinit**	commander manufacturing read tambr --out tambr.bin
eFuse data copy (back-up) - optional	**commander manufacturing read <efusecopy> --out <filename.bin> serialinterface --skipinit**	commander manufacturing read efusecopy --out efusecopy.bin

Note: If no MBR is present in the device, you will see an error. Refer to the Possible Error Codes section for more information on error codes.

4.2 Security Key Programming

This part of manufacturing includes generation of static keys and Physically Unclonable Function (PUF) generated keys and loading them 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 the device. Refer to Secure Key Management and Protection section for information on keys generated at commander and PUF Intrinsic keys.

The sequence to do security key programming is as follows:

1. Activation code generation for PUF Block.
2. Power cycle the device to boot again after step 1.
3. Generate keys from commander (generates keys at commander).
4. Intrinsic key generation (PUF generated keys) and loading keys (loads both device dependent keys and generated keys at commander)

4.2.1 Activation Code Generation for PUF Block

This command is used to generate activation code from the PUF module present in SiWN917 device. Once generated, the firmware will burn this activation code into flash.

Syntax:

commander manufacturing init --mbr <filename.bin|default> --data <updated-mbr-fields> -d <full opn> [--skipload] --serialinterface

MBR (filename.bin) files are present at the folder "Commander_xxx_xxx\Simplicity Commander\resources\jlink\Si917"s

Example:

commander manufacturing init –mbr **ta_mbr_SiWN917M100LGTBA.bin** --data **updated_mbr_file.json** -d **si917 --serialinterface**

Return Code

- Success – 0xa05a

- Failure – Try again.

4.2.2 Power Cycle the device

The users need to power cycle the device after the Activation Code Generation and flash completion. This is a mandatory step.

4.2.3 Generate Keys from Commander

These keys are generated at commander. The key generation will produce a json file containing the OTA keys, public keys, and Attestation key. For each of the public or private keys, the respective pair counterparts will be produced.

Syntax:

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

Example:

commander util genkeyconfig --outfile **commanderkeys.json** -d **si917**

Return Code

- Success – 0xa05a
- Failure – Try again.

4.2.4 Intrinsic Key Generation, Update MBR, and Loading Keys

This step performs the actual provisioning of the intrinsic keys (PUF generated keys), static keys, and final TA MBR and key descriptor table.

Along with keys loading and burning, the MBR will be updated with filename.bin or the "default" option (selected based on the OPN).

Syntax:

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

Note:

- **keys.json file contains the static keys i.e., TA OTA key, TA public key, and attestation key.**

Example:

commander manufacturing provision --mbr **ta_mbr_SiWN917M100LGTBA.bin** --keys **commanderkeys.json** --data **updatedmbrfields.json** -d **si917 --serialinterface**

Note: The updated-mbr-fields.json file provided is to be updated with different security levels discussed in MBR Programming with Security Section. Once we give this command, it updates default OPN based MBR with .json file parameters and loads to the flash after keys loading.

4.3 Add Security to TA Firmware Image Files

When the security is enabled in the device, it expects the TA file also to have the same level of security enabled. If the user tries to flash the non-secured image, the system will give an error.

4.3.1 Extract Keys

We need some of the keys generated in the section Generate Keys from Commander to secure the TA firmware image. We will have to extract these keys to a different folder.

Syntax:

commander util extractkeys <keys.json> --dir <destination foldername>

Example:

commander util extractkeys **commanderkeys.json** --dir **extracted-keys**

- This command extracts all the Keys to individual files in a specified folder that is already created (Make sure to create the folder whose name is given in this command before giving this command). In the above example, commanderkeys.json is the key file that was generated from the Generate Keys from Commander section, and extracted-keys is the folder name created.

4.3.2 Secure TA Firmware Images

TA Image

Syntax:

commander rps convert <filename.rps> --taapp <original non-encrypted TA rps> --mic "<OTA_KEY>" --encrypt "<OTA_KEY>" --sign <ta_private_key.pem>

Example:

commander rps convert **ta_fw.rps** --taapp **SiWG917-B.2.9.0.0.15.rps --**mic "**01a46070c0a6def656beef50555f6c5a771b6775003bf16a7cd5aa6c8ef9cc0a**" --encrypt "**01a46070c0a6def656beef50555f6c5a771b6775003bf16a7cd5aa6c8ef9cc0a**" --sign **ta_private_key.pem**

4.4 Flash the Secured Images into Flash

The user can flash the secured TA firmware images using the Simplicity Commander CLI. You need to load the .rps file format for the TA firmware images.

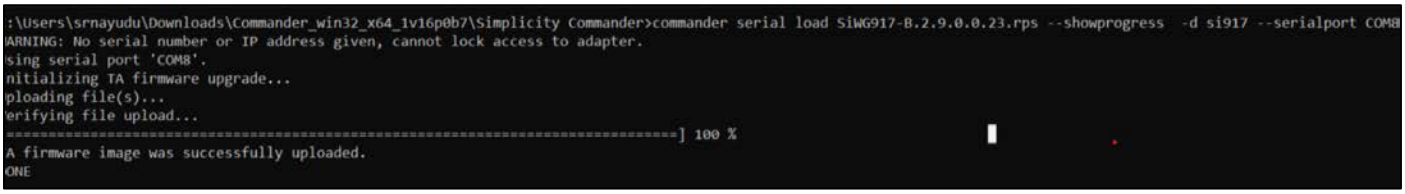
Flash TA Image

Syntax:

commander serial load <filename.rps> --showprogress --serialport < COM PortNo> -d full OPN

Example:

commander serial load ta.rps –showprogress –serialport COM7 -d **si917**



Note: If security is enabled in your device (according to section 4.2 - Security Key Programming), you will have to load the secured image. Or else the loading fails.

4.5 MBR Programming with Security

In the catalogue parts, the SiWN917 devices do not have security features enabled. The devices will have a default common flash configuration if the device's OPN has a flash.

In all the default cases, the device's MBR and EFUSE configurations have all the security features programmed to disabled state - the PUF is not enabled, the activation code for the PUF is not programmed, keys or key descriptors are not programmed.

- The security fields in MBR, PUF Activation code, Key descriptors, and the Keys are to be programmed in the device while enabling the security features in the device.
- The MBR includes the address of the sector where the PUF Activation code must be programmed as well as the address where the Key descriptors must be saved.
- This way the user can change the location of the PUF Activation code, Key descriptor table, and the address of the Keys.

- The MBR contains a field – **valids,** which has bits to set the PUF_ACTIVATION_CODE_ADDRESS_VALID and KEY_DESCRIPTOR_ADDRESS_VALID
- The device uses these bits to determine the address of the PUF Activation code, Key Descriptor, and the Keys.

4.5.1 Programmable Fields in MBR

The table below shows a few of the Requirements and the MBR fields which can be programmed.

Feature	Structure	MBR Field
To enable TA MIC and encryption	efuse_data	ta_secure_boot_enable = 1;
To enable TA Signature	efuse_data	ta_digital_signature_validation = 1;
To enable TA inline encryption	efuse_data	ta_encrypt_firmware = 1; (2 for XTS mode)
Start address of the sector where the PUF Activation code must be saved, 0x2000 is the default	mbr	puf_activation_code_addr = 0x2000;
Start address where the key descriptor table must be saved. (0x300 is the default)	mbr	key_desc_table_addr = 0x300;
If PUF Activation code's start address is being changed	mbr	valids |= PUF_ACTIVATION_CODE_ADDR_VALID;
If Key descriptor's start address is being changed	mbr	valids |= KEY_DESCRIPTOR_ADDRESS_VALID;

4.5.2 Security Levels

There are three different security levels available as described below. The levels are defined to make the security configurations easier for the user.

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

Security Level 1 (Low Security)

If the user wants to enable only secure boot, this level should be selected. This security level enables the bootloader to carry out the Message Integrity Check (MIC) of the firmware image during firmware loading and firmware update.

In this configuration, the firmware update process will be faster, the bootup time will be less, and the execution also will be faster.

This configuration is selected using the following fields:

```
"efuse_data": {
“ta_secure_boot_enable”: 1
“m4_secure_boot_enable”: 1
“disable_m4ss_kh_access”:0
}
```

Security Level 2 (Partial Security)

This level is for the user who wants to enable Secure boot along with Signature check of firmware image. This security level enables the bootloader to carry out the Message Integrity Check (MIC) and Signature validation of the firmware image during firmware loading and firmware update.

In this configuration, the Firmware update process will be slower but faster than Security Level 3, the bootup time will be relatively less than Security Level 3, and the execution will be same as Security Level 3.

This configuration is selected using the following fields:

```
"efuse_data": {
        “ta_secure_boot_enable”: 1
        “m4_secure_boot_enable”: 1
        “ta_digital_signature_validation”:1
        “m4_digital_signature_validation”: 1
        “disable_m4ss_kh_access”: 0
}
```

Security Level 3 (Full Security)

This level is for the user who wants to enable the Secure Boot, the Signature check, and the Inline Encryption. This security level enables the bootloader to carry out the Message Integrity Check (MIC), Encryption, Signature check of the firmware image during firmware loading and firmware update and Inline Encryption while saving the firmware in the flash.

In this configuration, the Firmware update process will be slower, the bootup time will be higher, and the execution also will be slower compared to security levels 1 and 2.

This configuration is selected using the following fields:

```
"efuse_data": {
        “ta_secure_boot_enable”:1
        “m4_secure_boot_enable”:1
        “ta_encrypt_firmware”: 1   // (1 for CTR, 2 for XTS mode)
        “m4_encrypt_firmware”: 1
        “m4_fw_encryption_mode “: 1   // (1 for CTR mode, 2 for XTS mode)
        “disable_m4ss_kh_access”: 0
        “ta_digital_signature_validation”: 1
        “m4_digital_signature_validation”: 1
}
```

4.6 Example JSON file with security parameters of MBR

The following example shows the structure and the available fields:

Download the .json file from the link

```
{
  "puf_activation_code_addr": 8192,
   "efuse_data": {
      "disable_m4ss_kh_access": 0,
      "m4_digital_signature_validation": 0,
      "m4_encrypt_firmware": 0,
      "m4_fw_encryption_mode": 0,
      "m4_secure_boot_enable": 0,
      "ta_digital_signature_validation": 1,
      "ta_encrypt_firmware": 1,
      "ta_secure_boot_enable": 1
   },
   "key_desc_table_addr": 768
}
```

5 Manufacturing Procedure – eFuse Data

In the case of eFuse/OTP data, Commander will first check that the requested update is possible (since bits can only be set to 1 and never cleared). Then it will ask for confirmation from the user. It is possible to skip the confirmation via the --noprompt option although note that this is a one-time operation.

It is possible to check the results of the operation before physically going ahead with the one-time programming by using the --dryrun option.

This command is used to write eFuse data into flash.

Syntax:

commander manufacturing write efuse --data file.json -d <full opn > [--skipload] [--pinset n] [-s jlinkserialno] [--noprompt] [–dryrun] --serialinterface --skipinit

6 Security Features – Manufacturing Utility

The following sections provide information on how to enable security using a manufacturing utility (Simplicity Commander CLI) to enable different SiWN917 Security features.

6.1 Secure Boot

Configure or enable/disable a pre-flashed secure bootloader on the chips to encrypt your software intellectual property (IP). Safeguard your competitive edge in the market.

Secure Boot is a feature for ensuring only an authenticated code can run on the chip. If any device fails its security check, it is not allowed to run, and program control will typically stall in the validating module.

The SiWx917 includes a Security Bootloader or Secure Bootloader. The Security Bootloader runs on the ThreadArch processor. On any reset, the Security Bootloader configures the module hardware based on the configuration present in the eFuse. It also validates the integrity and authenticity of the firmware in Flash and detects and prevents the execution of unauthorized software during the boot sequence.

Secure Boot Process:

1. Upon reset, the Security Bootloader configures the module hardware based on the configuration present in the ROM and Flash.
2. Device configurations in Flash are authenticated by Security Bootloader
3. Security Bootloader validates the integrity and authenticity of the firmware in the Flash and invokes the Application Bootloader.

To enable only secure boot or secure boot along with other security features refer to Security Levels section.

6.2 Secure Key Management and Protection

Inject custom public and private keys and other custom secret keys on the chips during manufacturing – safeguard your products right from the beginning of their lifecycle. The Bootloader uses public and private key-based digital signatures to recognize authentic software. The Security Bootloader provides provision for inline execution (XIP) of encrypted firmware from Flash.

If this feature is enabled, PUF engine will be invoked. PUF Keys will be generated internally via PUF and stored in flash. Images are encrypted and decrypted using PUF intrinsic keys.

The PUF Intrinsic Keys are device-dependent keys. They are unique for each device.

PUF Intrinsic Keys are as below. These are generated randomly.

- Master Key – Used to encrypt the TA OTA Key and TA Public Key for storing in the Flash.
- Unwrap Key – Used to encrypt the M4 OTA Key and M4 Public Key for storing in the Flash.
- TA FW Keys (2) – Used for Inline decryption.

Keys generated at the commander are as follows.

- OTP Symmetric Key – Used for flash content (Message Integrity Check) verification.
- OTP Public Key – Used for flash content (digital signature) verification.
- TA Public Key – Used for TA firmware signature verification.
- TA OTA key - Used for TA Firmware upgrade.
- Attestation Private Key – Used for secure attestation.
- M4 OTA Key – Used for M4 firmware upgrade.
- M4 Public Key – Used for M4 firmware signature verification.

For information on how to generate the keys at commander, refer to Generate Keys from Commander. For PUF activation, refer to Activation Code Generation, and to generate intrinsic keys and load the keys (both the intrinsic keys and keys generated at commander), refer to Intrinsic Key Generation and Loading Keys.

The table below shows the Key type, Key length, and Storage type.

S.No	Key Type	Key Length	Storage	
			OTP	Flash
1	OTP Symmetric key	16 bytes	☑	☐
2	OTP Public key	91 bytes	☑	☐
3	TA public key	96 bytes*	☐	☑
4	TA OTA key	32 bytes	☐	☑
5	M4 OTA key	32 bytes	☐	☑
6	M4 public key	96 bytes*	☐	☑
7	Attestation Private key	240 Bytes*	☐	☑
8	Master Key	52 bytes key code	☐	☑
9	Unwrap Key	52 bytes key code	☐	☑
10	TA FW Key(2 keys)	52 bytes key code	☐	☑
11	M4 FW Key(2 keys)	52 bytes key code	☐	☑

Note: For TA 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.

7 Calibration Using Manufacturing Utility

The following two flows are possible with the manufacturing utility (Simplicity Commander CLI) to calibrate the SiWN917 device.

7.1 Transmission in Burst Mode

7.1.1 Steps for Frequency Offset Correction.

1. Set up the radio and start transmission.
   commander manufacturing radio --power 16 --phy 6MBPS --channel 1 --start -d SiWG917M111MGTBA --serialinterface
2. Measure the frequency on the instrument which has the capability of measuring frequency error in modulated packets. For example, Keysight instruments with modulation analysis measurement setting capability reports the frequency error in burst packets.
3. Adjust the CTUNE values by providing the frequency error as input. This is a frequency offset correction, with the value ranging from –255 to 255

   Offset = measured frequency (in kHz) - 2412000

   commander manufacturing xocal --offset <Offset> --skipload -d SiWG917M111MGTBA --serialinterface --skipinit
4. Check the instrument and verify that the channel frequency is within expectations. If not, repeat from step 3.
5. Store the CTUNE values. The radio transmission will stop.
   commander manufacturing xocal --store --skipload -d SiWG917M111MGTBA --serialinterface --skipinit

7.1.2 Steps for Customer Gain-Offset

1. Setup the radio and start transmission on channel 1.
   commander manufacturing radio --power 16 --phy 6MBPS --channel 1 --start -d SiWG917M111MGTBA --serialinterface
2. Measure the output power with the instrument that has the capability of measuring burst power. For example, Keysight instruments with modulation analysis measurement setting capability reports the burst packet power.
3. Calculate the offset to meet the expected output power (16 dBm).
   *Offset = ceil ((output power measure (dBm) + cable loss -- 16) * 2)*
4. Store the gain values for channel 1
   commander manufacturing gain --ch1 <Offset> --skipload -d SiWG917M111MGTBA --serialinterface --skipinit
5. Repeat from step 1 for channel 6 and 11. In step 4m replace –ch1 with --ch6 and --ch11 when measuring for channel 6 and 11 respectively.
6. Stop the radio.
   commander manufacturing radio --stop --skipload -d SiWG917M111MGTBA --serialinterface --skipinit

7.2 Transmission in Continuous Mode

7.2.1 Steps for CTUNE Adjustments

1. Set up the radio and start transmission.
   commander manufacturing radio --power 16 --phy CW --channel 1 --start -d SiWG917M111MGTBA --serialinterface
2. Measure the frequency on the instrument using peak search (use the instrument which supports spectrum mode/modulation).
3. Adjust the CTUNE values by providing the frequency error as input. This is a frequency offset correction, with the value ranging from –255 to 255.
   Offset = measured frequency (in kHz) -- 2412000

commander manufacturing xocal --offset <Offset> --skipload -d SiWG917M111MGTBA --serialinterface --skipinit

4. Check the instrument and verify that the channel frequency is within expectations. If not, repeat from step 3.
5. Store the CTUNE values. The radio transmission will stop.
   commander manufacturing xocal --store --skipload -d SiWG917M111MGTBA --serialinterface --skipinit

7.2.2 Steps for Gain Offset Adjustments

1. Setup the radio and start transmission on channel 1.
   commander manufacturing radio --power 16 --phy 6MBPS --channel 1 --noburst --start -d SiWG917M111MGTBA --serialinterface
2. Measure the output power (use the instrument which supports spectrum mode/modulation). Measure the integrated power in 20MHz bandwidth.
3. Calculate the offset to meet the expected output power (16 dBm).
   *Offset = ceil ((output power measure (dBm) + cable loss -- 16) * 2)*
4. Store the gain values for channel 1.
   commander manufacturing gain --ch1 <Offset> --skipload -d SiWG917M111MGTBA --serialinterface --skipinit
5. Repeat from step 1 for channel 6 and 11. In step 4 the replace –ch1 with --ch6 and --ch11 when measuring for channel 6 and 11 respectively.
6. Stop the radio.
   commander manufacturing radio --stop --skipload -d SiWG917M111MGTBA --serialinterface –skipinit

Note:

- 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 Read Manufacturing Parameters

This command is used to read the manufacturing parameters from DUT and to save the data to a file.

Command: commander manufacturing read <tambr|taipmu|efuse|efusecopy> --out <filename.bin|filename.json> --serialinterface –skipinit

Read	Region	Region to read the data from
Read <region>	tambr	Read the TA flash MBR data available for memory region
	--out <filename>	Filename to store the read data in. "*.bin"
	taipmu	Read the TA flash ipmu data available for memory region
	efuse	Read the OTP data
	efusecopy	Read the efusecopy data in flash

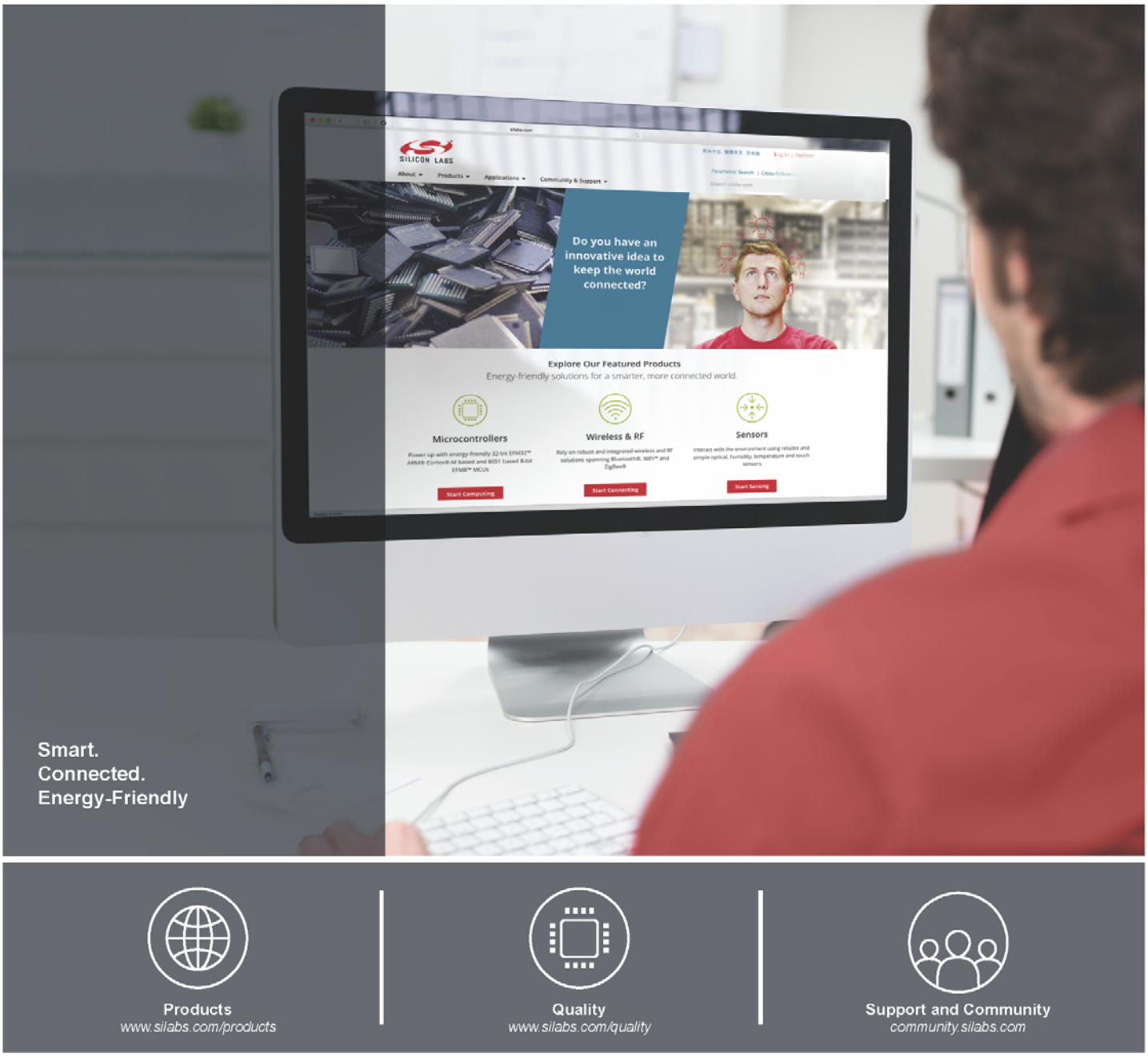
9 Possible Error Codes

Command	Error Code	Description	Cause of failure
MBR	0xa0ac	Verification failed from flash	There may be a wrong input from the user to select pinset/damaged MBR file
Activation code	0xa0b0	Activation code generated failed	security module might throw an error if PUF enroll fails, if the device is locked to use PUF
Intrinsic keys	0xa0b1	Intrinsic keys generation failed	This may occur if this command triggered before activation code command.
Static keys	0xa0b2	Static keys stored failed	This may occur if there are no intrinsic keys.
Update Firmware	108	Failed to load firmware	This may occur if the header part of rps file is not proper.

10 Appendix

10.1 Parameters - Manufacturing Utility

Field	Description
write	command for writing the MBR
read	read the contents
init	generate the activation code
--mbr	master boot record
--keys	security keys
tambr	Read the TA flash MBR data available for the memory region
--out <filename>	Filename to store the read data in. "*.bin", or "*.json" if a JSON parser is available for this memory region data.
taipmu	Read the TA flash IPMU data available for the memory region
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
--skipinit	to skip the initialization part
--skipload	Skip loading the TA provisioning firmware
--pinset	pin set (Self-explanatory)
n	pin set number Internal Flash pin set: **0** External flash pin set: [**2/3**]



Products
www.silabs.com/products

Quality
www.silabs.com/quality

Support and Community
community.silabs.com

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

http://www.silabs.com