Instruction

Manufacture Z-Wave product in volume

Document No.:	INS14285
Version:	11
Description:	-
Written By:	JFR;JSMILJANIC;CAOWENS;GAFARKAS
Date:	2022-09-12
Reviewed By:	OPP;PSH;CAOWENS;JCC
Restrictions:	Public

Approved by:

Date	CET	Initials	Name	Justification
2022-09-12	09:05:06	NTJ	Niels Johansen	

SILICON LABS

REVISION RECORD				
Doc. Rev	**Date**	**By**	**Pages Affected**	**Brief Description of Changes**
1	20190130	JFR	All	Initial draft
2	20190207	JFR	2	Clarified Lock Bit Page setting to protect IP and security material.
3	20190711	SGANGULY JFR	2	Split manufacturing flow for end devices and gateways. Also added details about bootloader and signing keys etc.
4	20190712	SCBROWNI	All	Minor typos
5	20190903	JFR	2.1 & 2.2	RAILTest replacing ApplicationTestPoll used in 500 Series
6	20190911	SCBROWNI	2.1 & 2.2	Edited and reviewed the changed sections.
7	20200327	JESMILJA	2.2	Updated instruction what keys to use for EFR32ZG14
8	20200427	JFR	2.1 & 2.2	Clarified that ZGM130S is based on EFR32FG13 and EFR32ZG14 is based on EFR32FG14.
9	20210319	JFR	2.2	Updated wrt. crystal frequency, keys and QR labelling.
10	20220203	JFR	Front page	Removed 700.
11	20220720	FG	3 & References	Add 800 series instructions

Table of Contents

1 INTRODUCTION

This document describes the manufacturing test flow for Z-Wave 700/800 SoC-based products.

2 MANUFACTURING FLOW FOR 700 SERIES

The following section describe the manufacturing flow for end devices (ZGM130S, etc.) and gateways using EFR32ZG14. It is not recommended to use the Silicon Labs public signing key and encryption key used in the apps.

2.1 End Devices

The manufacturing production test flow for end devices ZGM130S (based on EFR32FG13) must incorporate the following steps:

- Perform product-specific testing such as I/O, etc. Refer to *UG409: RAILtest User's Guide* under the SDK documentation section in the Simplicity Studio distribution.
- Perform RF testing, etc. Use RAILtest. Refer to [7] regarding RF testing. The 500 Series ApplicationTestPoll function is not available in the 700.
- Set the manufacturing codes:
    1. Download an OTA bootloader to the SoC target via the Serial Wire Debug (SWD) interface.
    2. Write your own public signing key and encryption key to the SoC target via the SWD interface. A readme.txt file in the Z-Wave SDK release describes how to generate your own keys and write them to the device Lock Bits Page. The path to the readme file in the Z-Wave SDK release is:

       ```
       <Your ZWAVE Installation Directory>\BootLoader\sample-keys\
       ```
    3. Download the application firmware to the SoC target via the SWD interface. Do not set the Lock Bit in this step.

4. The application in the SoC signals when the security materials, etc., are in place in the Lock Bit Page via the manufacturing token TOKEN_MFG_ZW_INITIALIZED. The following steps are performed in the SoC at the application startup:
    a. If public/private keypair and QR code are already present in the Lock Bit page (check manufacturing token TOKEN_MFG_ZW_INITIALIZED), jump to the last step continuing normal operation. Refer to [1] for details about manufacturing tokens.
    b. Calculate the public/private key based on Curve25519.
    c. Construct the QR code using the public key, product type, and product ID (latter two from the application) as described in [2].
    d. Calculate the SHA-1 checksum as per [2] and incorporate it in the QR code.
    e. Write the QR code to the Lock Bit Page as manufacturing token TOKEN_MFG_ZW_QR_CODE.
    f. Write the private/public keypair to the Lock Bit Page as manufacturing tokens TOKEN_MFG_ZW_PRK and TOKEN_MFG_ZW_PUK.
    g. Write completion of the Lock Bit Page initialization as manufacturing token TOKEN_MFG_ZW_INITIALIZED. This token can be used to sync completion of data to Lock Bits Page in a production system.
    h. Continue normal startup.
5. Read the QR code from the SoC.
6. Set the Lock Bit Page [6] to protect IP and security material against untrusted entities.
7. Label the product with the QR code. Refer to [5] for details.

    The QR code format enables customization of the QR code with extra TLVs (e.g., MaxInclusionRequestInterval, proprietary serial number, etc.) instead of using the internally generated one. The manufacturing line programmer must then read out the public key, etc., compose the wanted QR code, and print it to a label. The new QR code can also be stored in the User Data Page, for example.

    Set the following registers in the Lock Bit Page [6] as a minimum to protect IP and security material:

    DLW = Disable the debug port by clearing the four LSBs
    ULW = Ignore
    MLW = Optional (disable mass erase through MSC)
    ALW = Optional (disallow a mass erase operation)
    CLW1 = Ignore
    CLW2 = Ignore
    PLW[0…121] = Ignore

2.2 Gateways

The manufacturing production test flow for gateways using EFR32ZG14 (based on EFR32FG14) must incorporate the following steps:

1. Product-specific testing such as I/O, etc. Refer to 'Using RAIL Test' under the SDK documentation section in the Simplicity Studio distribution.
2. Calibrate the 39MHz crystal used on each EFR32ZG14-based product to ensure the RF frequency is correct, see [4]. The crystal calibration can be done by using a RailTest firmware, see KB - Z-Wave 700: EFR32ZG14 CTUNE Calibration.
3. The RF performance testing for each product can also be done by using the same RailTest firmware. Refer to [7] regarding RF performance testing. The 500 Series ApplicationTestPoll function is not available in 700.
4. Download Z-Wave OTW bootloader to the SoC target via Serial Wire Debug (SWD) interface.
5. Generate your own public signing key and encryption key and write them to the SoC target via the Serial Wire Debug (SWD) interface. These keys are necessary for upgrading the firmware in the field. Following simplicity commander commands will be used for writing keys into the device's Lock Bits Page.
    1. commander flash --tokengroup znet --tokenfile zg14_encrypt.key --tokenfile zg14_sign.key-tokens.txt -d EFR32ZG14
    2. The key files (do not use the Silicon Labs keys) are locked in the Z-Wave release in the following path on your SDK installation

       ```
       <Your ZWAVE Installation Directory>\BootLoader\ZG14-keys\
       ```
6. Download the application firmware to the SoC target via the Serial Wire Debug (SWD) interface. Do not set the Lock Bit in this step.
7. The application in the SoC signals when security materials, etc., are in place in the Lock Bit Page via manufacturing token TOKEN_MFG_ZW_INITIALIZED. The following steps are performed in the SoC at the application startup:
    1. If the public/private keypair and QR code are already present in the Lock Bit page (Check manufacturing token TOKEN_MFG_ZW_INITIALIZED), jump to the last step continuing normal operation. Refer to [1] for details about manufacturing tokens.
    2. Calculate the public/private key based on Curve25519.
    3. Construct the QR code using public key, product type, and product ID (latter two from application) as described in [2].
    4. Calculate SHA-1 checksum as per [2] and incorporate it in the QR code.
    5. Write the QR code to Lock Bit Page as manufacturing token TOKEN_MFG_ZW_QR_CODE.
    6. Write private/public keypair to the Lock Bit Page as manufacturing tokens TOKEN_MFG_ZW_PRK and TOKEN_MFG_ZW_PUK.

    7. Write completion of Lock Bit Page initialization as manufacturing token TOKEN_MFG_ZW_INITIALIZED. This token can be used to sync completion of data to the Lock Bits Page in a production system.

    8. Continue normal startup.

8. Read the QR code from the SoC.

9. Set the Lock Bit Page [6] to protect IP and security material against untrusted entities.

10. Label the product with the QR code. It is optional to label a gateway in case the QR code is accessible via the UI. Refer to [5] for details.

    The QR code format enables customization of the QR code with extra TLVs (e.g., MaxInclusionRequestInterval, proprietary serial number, etc.) instead of using the internally generated one. The manufacturing line programmer must then read out the public key, etc., and compose the wanted QR code and print it to a label. The new QR code can also be stored in, e.g., the User Data Page.

    Set the following registers in the Lock Bit Page [6] as a minimum to protect IP and security material:

    DLW = Disable the debug port by clearing the four LSBs
    ULW = Ignore
    MLW = Optional (disable mass erase through MSC)
    ALW = Optional (disallow a mass erase operation)
    CLW1 = Ignore
    CLW2 = Ignore
    PLW[0…121] = Ignore

3 MANUFACTURING FLOW FOR 800 SERIES

The following section describe the manufacturing flow for end devices and gateways using 800 series.

It is not recommended to use the Silicon Labs public signing key and encryption key used in the apps.

Further reading about the available security features and their usage can be found in the UG103.05: IoT Endpoint Security Fundamentals.

3.1 End Devices

The manufacturing production test flow for end devices based on 800 series must incorporate the following steps:

- Update the SE firmware, further readings AN1222: Production Programming of Series 2 Devices
- Update the Bootloader, further readings AN1222: Production Programming of Series 2 Devices
- Perform product-specific testing such as I/O, etc. Refer to *UG409: RAILtest User's Guide* under the SDK documentation section in the Simplicity Studio distribution.
- Perform RF testing, etc. Use RAILtest. Refer to [7] regarding RF testing. The 500 Series ApplicationTestPoll function is not available in the 800.
- Set the manufacturing codes:
    1. Write your own public signing key and encryption key to the SoC target via the SWD interface. A readme.txt file in the Z-Wave SDK release describes how to generate your own keys and write them to the device Lock Bits Page. The path to the readme file in the Z-Wave SDK release is:

       ```
       <Your ZWAVE Installation Directory>\BootLoader\sample-keys\
       ```

    2. Download the application firmware to the SoC target via the SWD interface.

3. The application in the SoC signals when the security materials, etc., are in place in the Lock Bit Page via the manufacturing token TOKEN_MFG_ZW_INITIALIZED. The following steps are performed in the SoC at the application startup:

    i. If public/private keypair and QR code are already present in the Lock Bit page (check manufacturing token TOKEN_MFG_ZW_INITIALIZED), jump to the last step continuing normal operation. Refer to [1] for details about manufacturing tokens.

    j. Calculate the public/private key based on Curve25519.

    k. Construct the QR code using the public key, product type, and product ID (latter two from the application) as described in [2].

    l. Calculate the SHA-1 checksum as per [2] and incorporate it in the QR code.

    m. Write the QR code to the Lock Bit Page as manufacturing token TOKEN_MFG_ZW_QR_CODE.

    n. Write the private/public keypair to the Lock Bit Page as manufacturing tokens TOKEN_MFG_ZW_PRK and TOKEN_MFG_ZW_PUK.

    o. Write completion of the Lock Bit Page initialization as manufacturing token TOKEN_MFG_ZW_INITIALIZED. This token can be used to sync completion of data to Lock Bits Page in a production system.

    p. Continue normal startup.

4. Read the QR code from the SoC.

5. Label the product with the QR code. Refer to [5] for details.

    The QR code format enables customization of the QR code with extra TLVs (e.g., MaxInclusionRequestInterval, proprietary serial number, etc.) instead of using the internally generated one. The manufacturing line programmer must then read out the public key, etc., compose the wanted QR code, and print it to a label. The new QR code can also be stored in the User Data Page, for example.

- Perform Key Provisioning, further readings AN1222: Production Programming of Series 2 Devices
- Set the debug access, further reading AN1190: Series 2 Secure Debug
- Set the Anti-Tamper protection, further reading AN1247: Anti-Tamper Protection Configuration and Use

3.2 Gateways

The manufacturing production test flow for gateways based on 800 series must incorporate the following steps.

Further reading about the available security features and their usage can be found in the UG103.05: IoT Endpoint Security Fundamentals.

- Update the SE firmware, further readings AN1222: Production Programming of Series 2 Devices
- Update the Bootloader, further readings AN1222: Production Programming of Series 2 Devices
- Product-specific testing such as I/O, etc. Refer to 'Using RAIL Test' under the SDK documentation section in the Simplicity Studio distribution.
- Calibrate the 39MHz crystal used on each EFR32ZG14-based product to ensure the RF frequency is correct, see [4]. The crystal calibration can be done by using a RailTest firmware, see KB - Z-Wave 700: EFR32ZG14 CTUNE Calibration.
- The RF performance testing for each product can also be done by using the same RailTest firmware. Refer to [7] regarding RF performance testing. The 500 Series ApplicationTestPoll function is not available in 800 series.
- Generate your own public signing key and encryption key and write them to the SoC target via the Serial Wire Debug (SWD) interface. These keys are necessary for upgrading the firmware in the field. Following simplicity commander commands will be used for writing keys into the device's Lock Bits Page.
  1. commander flash --tokengroup znet --tokenfile zg14_encrypt.key --tokenfile zg14_sign.key-tokens.txt -d EFR32ZG14
  2. The key files (do not use the Silicon Labs keys) are locked in the Z-Wave release in the following path on your SDK installation

     ```
     <Your ZWAVE Installation Directory>\BootLoader\ZG14-keys\
     ```
- Download the application firmware to the SoC target via the Serial Wire Debug (SWD) interface.
- The application in the SoC signals when security materials, etc., are in place in the Lock Bit Page via manufacturing token TOKEN_MFG_ZW_INITIALIZED. The following steps are performed in the SoC at the application startup:
  1. If the public/private keypair and QR code are already present in the Lock Bit page (Check manufacturing token TOKEN_MFG_ZW_INITIALIZED), jump to the last step continuing normal operation. Refer to [1] for details about manufacturing tokens.
  2. Calculate the public/private key based on Curve25519.
  3. Construct the QR code using public key, product type, and product ID (latter two from application) as described in [2].
  4. Calculate SHA-1 checksum as per [2] and incorporate it in the QR code.
  5. Write the QR code to Lock Bit Page as manufacturing token TOKEN_MFG_ZW_QR_CODE.
  6. Write private/public keypair to the Lock Bit Page as manufacturing tokens TOKEN_MFG_ZW_PRK and TOKEN_MFG_ZW_PUK.
  7. Write completion of Lock Bit Page initialization as manufacturing token TOKEN_MFG_ZW_INITIALIZED. This token can be used to sync completion of data to the Lock Bits Page in a production system.

8. Continue normal startup.

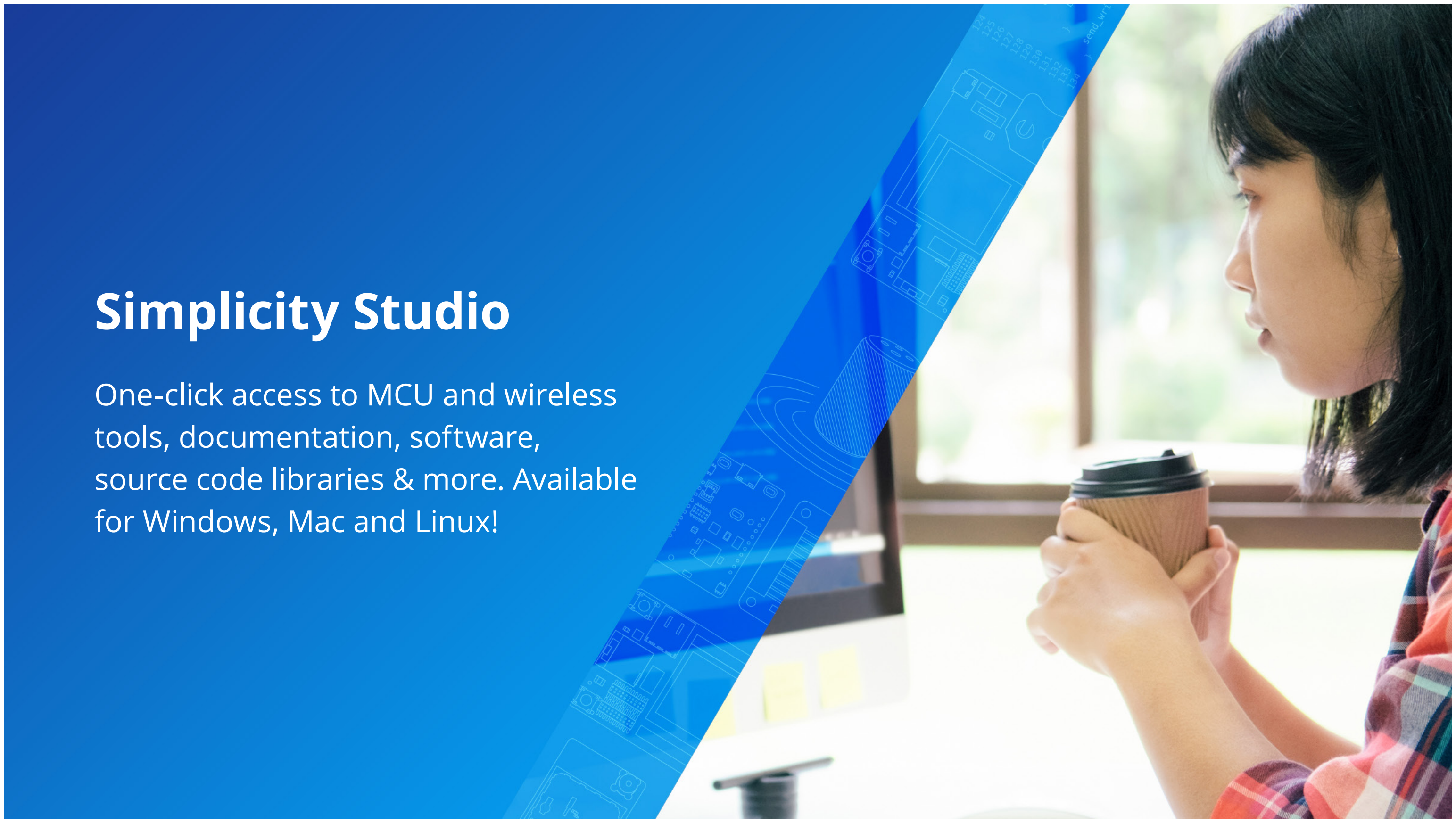
- Read the QR code from the SoC.
- Label the product with the QR code. It is optional to label a gateway in case the QR code is accessible via the UI. Refer to [5] for details.

  The QR code format enables customization of the QR code with extra TLVs (e.g., MaxInclusionRequestInterval, proprietary serial number, etc.) instead of using the internally generated one. The manufacturing line programmer must then read out the public key, etc., and compose the wanted QR code and print it to a label. The new QR code can also be stored in, e.g., the User Data Page.

- Perform Key Provisioning, further readings AN1222: Production Programming of Series 2 Devices
- Set the debug access, further reading AN1190: Series 2 Secure Debug
- Set the Anti-Tamper protection, further reading AN1247: Anti-Tamper Protection Configuration and Use

REFERENCES

[1] Silicon Labs, SDS14306, Software Design Specification, Z-Wave 700 Lock Bits and User Data Page Contents.
[2] Silicon Labs, INS13975, Instruction, SmartStart Production Control.
[3] Silicon Labs, SDS13937, Software Design Specification, Node Provisioning QR Code Format.
[4] Silicon Labs, UG522, Instruction, Mandatory crystal adjustment.
[5] Z-Wave Alliance, Z-Wave Security 2 (S2) and SmartStart Product Labeling Requirements.
[6] Silicon Labs, EFR32xG14 Wireless Gecko Reference Manual. Rev. 1.3.
[7] Silicon Labs, UG523, Instruction, Bring-up/test HW development.
[8] Silicon Labs, UG103.05, IoT Endpoint Security Fundamentals.
[9] Silicon Labs, AN1222, Production Programming of Series 2 Devices.
[10] Silicon Labs, AN1190, Series 2 Secure Debug.


Simplicity Studio
One-click access to MCU and wireless tools, documentation, software, source code libraries & more. Available for Windows, Mac and Linux!

IoT Portfolio	SW/HW	Quality	Support & Community
www.silabs.com/IoT	www.silabs.com/simplicity	www.silabs.com/quality	www.silabs.com/community

SILICON LABS

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

www.silabs.com