AN1507: SiWx917 RCP Manufacturing Utility Application Note

Manufacturing means programming production-specific information into the device. This application note provides information about the manufacturing procedure and the utility in SiWx917 RCP (referred to as SiWT917).

KEY POINTS

- SiWx917 Manufacturing utility procedure
- Manufacturing information and device calibration
- Supported commands and its usage

 Copyright © 2025 by Silicon Laboratories

1. Introduction

Customer manufacturing software tool for SiWx917 RCP (SiWT917) can be used to update manufacturing information and perform device calibration on the fly. This document will provide information about the features and functionalities of the application.

2. Requirements

Hardware:

- BRD4346A - SiWx917 Wi-Fi 6 and Bluetooth LE Co-Processor Radio Board (hereafter referred to as SiWx917 radio board)
- BRD8045B - EXP Adapter Board for SiWx917 Co-Processors (hereafter referred to as adapter board)
- Raspberry Pi 4 with Raspberry Pi 4 OS for BRD8045B
- Windows /Linux/ Raspberry PI
- LAN cable

Software:

- SiWx91x RCP Driver
- Commander CLI

3. Setup Image

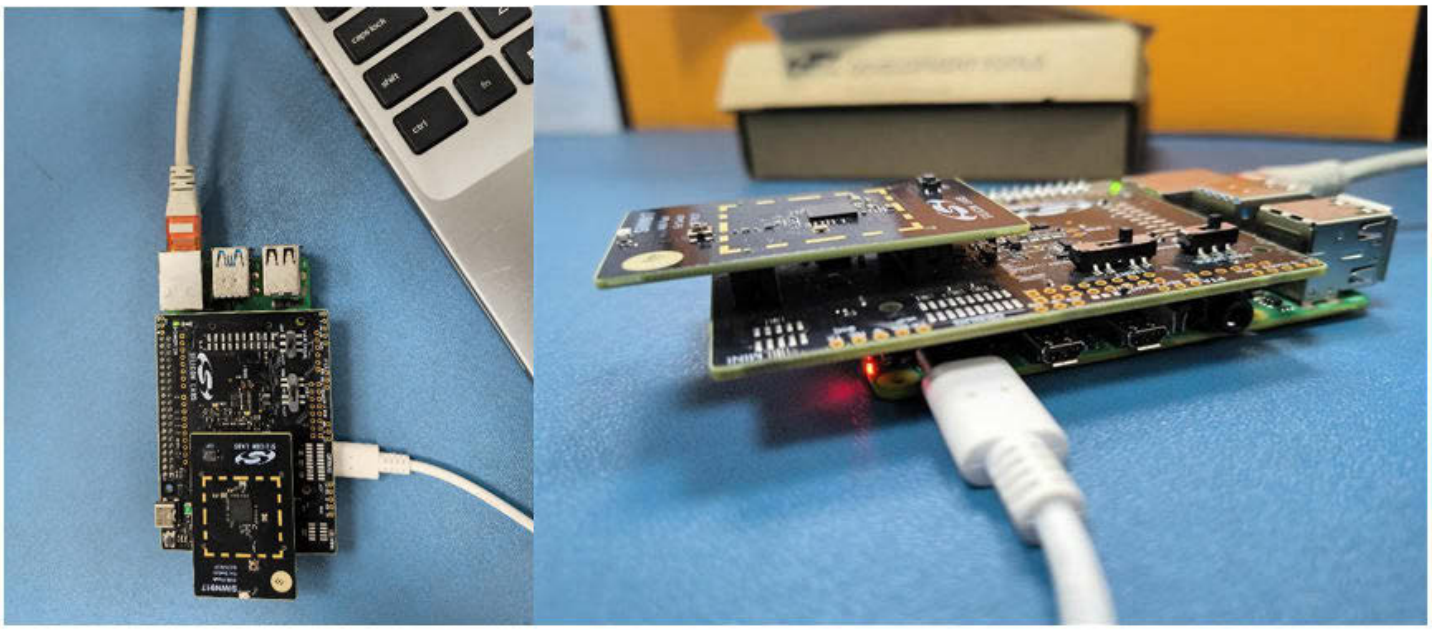

Figure 3.1. Setup Image

4. Installation

1. Download and unzip the release package.
2. Install the driver without loading the firmware using the following steps:
   a. Search for the following line in Makefile and uncomment it:

   EXTRA_CFLAGS += -DNO_FIRMWARE_LOAD_SUPPORT

```
#Uncomment below line to skip loading firmware
EXTRA_CFLAGS += -DNO_FIRMWARE_LOAD_SUPPORT
```

   b. Save the changes made in Makefile and compile the driver using the make command:

```
#make clean;make
```

   c. After successful compilation, install all the dependencies.
   d. Insert the modules using the following arguments:
      i. modprobe mac80211
      ii. modprobe cfg80211
      iii. modprobe Bluetooth
   e. Insert the kernel modules rsi_91x.ko and rsi_sdio.ko using the following commands:
      i. `insmod rsi_91x.ko dev_oper_mode=1 rsi_zone_enabled=0x601 skip_fw_load=1`
      ii. `insmod rsi_sdio.ko`
3. Ensure that the driver is inserted correctly using dmesg. Use `dmesg` command to observe the prints.
4. The print should show "FIRMWARE LOADING SKIPPED", indicating that the driver did not load any firmware.

```
[ 2571.612327] Bluetooth: HCI device and connection manager initialized
[ 2571.612341] Bluetooth: HCI socket layer initialized
[ 2571.612348] Bluetooth: L2CAP socket layer initialized
[ 2571.612357] Bluetooth: SCO socket layer initialized
[ 2604.781975] rsi_91x: loading out-of-tree module taints kernel.
[ 2795.444367] rsi_91x: ===============================================
[ 2795.444387] rsi_91x: ========== FIRMWARE LOADING SKIPPED ==========
[ 2795.444398] rsi_91x: ===============================================
[ 2795.444407] rsi_91x: Driver Version  : SiWT917.2.13.0.1
[ 2795.444418] rsi_91x: Operating mode  : 0 [Unknown]
[ 2795.444613] rsi_91x: obm_configure_region: Default Country Code 0 selected
[ 2795.445064] rsi_91x: rsi_probe: ***** 9117_B0 Module *****
root@raspberrypi:/home/pi#
```

5. Once the driver is installed, download the Simplicity Commander onto your host.
6. Unzip the commander to the directory of your choice.
7. Navigate to the release directory (driver path) where the mfg application is present after using the make command (for example, `SiWT917.x.x.x.x/release/`).
8. Open a terminal and run the application using `./mfg`. The output should be as shown below:

```
root@raspberrypi:/home/pi/SiWT917.2.13.0.1/release# ./mfg
********** Server is up and listening on TCP PORT NUMBER : 51917 **********
```

9. Open another terminal and navigate to the path where the commander-cli was previously extracted as per step 6.

```
pi@raspberrypi:~ $ sudo su
root@raspberrypi:/home/pi# cd commander-cli/
```

10. Type the commands described in section 5 to perform read or write operations for the different memory regions from the commander-cli path.

5. Working Commands

Note:

1. Before proceeding, make the connections as shown in 3. Setup Image.
2. Get the IP address of the DUT using ipconfig or ifconfig.
3. For any command that is used, the port number always remains 51917.

5.1 Init

This command is used to generate an activation code. Once generated, the firmware will burn this activation code into flash.

Command syntax: `commander manufacturing init --mbr <filename.bin|default> --data <updated-mbr-fields> -d < OPN Number > [--skipload] [--pinset n]`

Example: `commander manufacturing init –host 192.168.10.100:51917 –serialinterface –device SiWG917M100MGTBA`

Fields	Description
`init`	Generate the activation code
`--mbr <filename.bin|default>`	Binary file to read the MBR data from. If "default", the MBR data will be populated from a template for the connected device.
`--data <filename>`	JSON containing MBR fields to update the MBR with.
`-d <full opn>`	Provide OPN number example: SiWG917M100MGTBA
`--pinset <index>`	By default, pinset value is 0
`--skipload`	Skip loading the TA provisioning firmware

```
root@raspberrypi:/home/pi/SiWT917.2.13.0.1/release/commander-cli# ./commander-cli  mfg917 init --host 192.168.30.10:51917 --serialinte
rface --device SiWG917M100MGTBA
Using host 192.168.30.10, port 51917.
Initializing target...
No MBR data provided, using the default PUF activation key address...
Loading RAM code (321776 bytes)...
Starting RAM code...
Activation code generated successfully
DONE
```

Figure 5.1. Init Command

5.2 Read

This command is used to read the specified memory region from the DUT and save it to a file.

Command syntax: `commander manufacturing read <tambr|taipmu|m4mbrcf|m4mbrdf|m4ipmucf|m4ipmudf|efuse|efusecopy|efuseipmu> --out <filename.bin|filename.json> -d <OPN Number>`

Example: `commander manufacturing read tambr -output -host 192.168.10.100:51917 --serialinterface --device SiWG917M100MGTBA`

<region>	Information
tambr	Read the TA flash MBR data available for memory region
m4mbrcf	Read the common flash M4 MBR data
m4mbrdf	Read the dual flash M4 MBR data
m4ipmucf	Read the common flash M4 ipmu data
m4ipmudf	Read the dual flash M4 ipmu data
--out <filename>	Store the read data in. "*.bin", or "*.json" if a JSON parser is available for this memory
taipmu	Read the TA flash ipmu data available for memory region
efuse	Read the OTP data
efusecopy	Read the efusecopy data in flash
efuseipmu	Read the ipmu data from OTP/Efuse
-d <full opn>	Provide OPN number example: SiWG917M100MGTBA

```
root@raspberrypi:/home/pi/SiWT917.2.13.0.1/release/commander-cli# ./commander-cli manufacturing read tambr --host 192.168.30.10:51917
--serialinterface --device SiWG917M100MGTBA
Using host 192.168.30.10, port 51917.
Initializing target...
Reading data from region: tambr
Reading 496 bytes from 0x04000000
{address:  0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F}
04000000: 5A 5A 00 00 F0 01 00 00 A8 72 EB 28 31 7A 03 00
04000010: 00 00 02 40 00 00 28 00 D8 00 78 18 C7 00 20 00
04000020: 01 00 05 00 80 07 00 20 00 00 00 00 00 00 00 00
04000030: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
04000040: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
04000050: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
04000060: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
04000070: 00 00 00 00 F0 1F E6 6F 00 00 00 00 00 00 00 00
04000080: 00 00 00 00 00 00 00 00 20 00 00 00 00 00 00 00
04000090: 00 00 00 00 00 00 00 00 0B 00 00 03 00 00 00 00
040000a0: 19 09 0E 00 00 48 E8 01 00 5A 62 02 88 90 00 00
040000b0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
040000c0: 00 00 00 00 00 00 00 00 61 62 63 64 31 32 33 34
040000d0: 65 66 35 36 37 38 61 62 00 00 00 00 00 00 00 00
040000e0: 00 00 00 00 00 00 00 00 61 61 62 62 63 63 64 64
040000f0: 65 65 66 66 00 00 00 00 00 00 00 00 00 00 00 00
04000100: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
04000110: 00 FF FF 02 00 00 00 00 00 00 00 00 00 00 00 00
04000120: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
04000130: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
04000140: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
04000150: 0B 11 00 40 80 10 00 00 00 00 06 00 02 00 48 00
04000160: 19 00 28 00 01 80 00 00 70 91 00 00 00 F0 3C 00
04000170: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
04000180: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
04000190: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
040001a0: 00 00 00 0C AA 62 EB 3F 31 7A 03 30 00 00 38 6A
040001b0: 00 00 20 00 00 00 78 00 00 00 00 00 00 00 00 00
040001c0: 0D 5D 00 00 03 00 00 00 07 02 01 00 00 00 00 00
040001d0: 00 00 00 00 00 00 00 00 00 00 00 00 FF FF 07 00
040001e0: 00 00 00 00 00 00 00 00 00 00 00 00 78 8C 13 80

DONE
root@raspberrypi:/home/pi/SiWT917.2.13.0.1/release/commander-cli# ^C
```

Figure 5.2. Read Command

5.3 Write

This command is used to write to the specified memory region in DUT. If the file is provided in binary form, it will be flashed verbatim. If it's provided as json, it will be considered an update to the flash content.

Command syntax: `commander manufacturing write <region> --data <filename.bin|filename.json> -d < OPN Number > [--skipload] [--pinset n]`

Example: `commander manufacturing write tambr -data resources/jlink/Si917/ta_mbr_SiWG917M100MGTBA.bin --pinset 0 --host 192.168.10.100:51917 --serialinterface --device SiWG917M100MGTBA`

Field	Description
`--data <filename.bin|filename.json>`	Provide MBR file
`-d <full opn>`	Provide OPN number example: SiWG917M100MGTBA
`--pinset <index>`	By default, pinset value is 0. Select pinset for external flash configurations
`--skipload`	Skip loading the TA provisioning firmware

```
root@raspberrypi:/home/pi/vikas/commander-cli# ./commander-cli manufacturing write tambr --data resources/jlink/Si917/ta_mbr_SiWG917M100MGTBA.bin --pinset
 0 --host 192.168.250.230:51917 --serialinterface --device SiWG917M100MGTBA
WARNING: No serial number or IP address given, cannot lock access to adapter.
Using host 192.168.250.230, port 51917.
Initializing target...
Parsing file resources/jlink/Si917/ta_mbr_SiWG917M100MGTBA.bin...
WARNING: The provided data (528 B) is larger than the target region (496 B); the data will be truncated before it is written to the device.
Writing data to region: tambr
Loading RAM code (321776 bytes)...
Starting RAM code...
Attempting reset before trying again...
Loading RAM code (321776 bytes)...
Starting RAM code...
Data loaded successfully
Region 'tambr' was successfully written to device.
DONE
root@raspberrypi:/home/pi/vikas/commander-cli#
```

Figure 5.3. Write Command

5.4 Provision

This command is used to flash the MBR content for a selected region or to recover the board. MBR (filename.bin) files are present at the following path.

Path: "Commander_win32_x64_1v15p3b1357\Simplicity Commander\resources\jlink\Si917"

Command syntax: `commandermanufacturing provision --mbr <filename.bin|default> -d <OPN Number>`

Example: `commander manufacturing provision -mbr resources/jlink/Si917/ta_mbr_SiWG917M100MGTBA.bin --pinset 0 --host 192.168.10.100:51917 --serialinterface --device SiWG917M100MGTBA`

```
root@raspberrypi:/home/pi/vikas/commander-cli# ./commander-cli manufacturing provision --mbr resources/jlink/Si917/ta_mbr_SiWG917M100MGTBA.bin --serialint
erface --device SiWG917M100MGTBA --pinset 0 --host 192.168.250.230:51917
WARNING: No serial number or IP address given, cannot lock access to adapter.
Using host 192.168.250.230, port 51917.
Initializing target...
Loading RAM code (321776 bytes)...
Starting RAM code...
Attempting reset before trying again...
Loading RAM code (321776 bytes)...
Starting RAM code...
WARNING: Changing 'boot_descriptor_offset' field in MBR structure to new default value (0x00000238).
Programming TA MBR...
The on-device TA MBR is already up to date.
Programming ROM patch...
ROM patch is already present on the device.
Programming M4 MBR...
Data loaded successfully
DONE
root@raspberrypi:/home/pi/vikas/commander-cli#
```

Figure 5.4. Provision Command

5.5 Channel and Power (Radio)

This command is used to set the channel and power to the device.

Command syntax: `commander manufacturing radio [--channel <1-14>] [--power <1-31/127>] [--phy <xMBPS,MCSn,CW>] [--vmcu18] [--noburst] [--internalant] [-stop|-start] [-serialinerface ] [-skipinit] -d <OPN Number>`

Example: `commander manufacturing radio -channel 1 -power 10 -phy 1MBPS -start -host 192.168.10.100:51917 -serialinterface -d SiWG917M100MGTBA`

Note:

start - Turn ON the radio transmission

stop - Turn OFF the radio transmission

noburst - Transmission will happen continuously instead of being in bursts

internalant - select virtual internal switch. By default, the external antenna switch configuration

vmcu18 - selects parameters for operation at 1.8 V

```
root@raspberrypi:/home/pi/vikas/commander-cli# ./commander-cli manufacturing radio --channel 1 --power 10 --phy 1MBPS --start --serialinterface -d SiWN917
M100LGTBA --host 192.168.250.230:51917
WARNING: No serial number or IP address given, cannot lock access to adapter.
Using host 192.168.250.230, port 51917.
Initializing target...
Loading RAM code (355328 bytes)...
Starting RAM code...
Attempting reset before trying again...
Loading RAM code (355328 bytes)...
Starting RAM code...
Setting radio power:00000010, channel:00000001.
DONE
root@raspberrypi:/home/pi/vikas/commander-cli# ./commander-cli manufacturing radio --stop --serialinterface -d SiWN917M100LGTBA --host 192.168.250.230:519
17
WARNING: No serial number or IP address given, cannot lock access to adapter.
Using host 192.168.250.230, port 51917.
Initializing target...
Loading RAM code (355328 bytes)...
Starting RAM code...
Attempting reset before trying again...
Loading RAM code (355328 bytes)...
Starting RAM code...
Setting radio power:00000001, channel:00000001.
Radio stopped.
DONE
root@raspberrypi:/home/pi/vikas/commander-cli#
```

Figure 5.5. Channel and Power Command

5.6 Xocal

This command adjusts the tuning to compensate for the specified frequency offset and initiates the radio based on the current configuration, which must be set before this command.

Command syntax: `commander manufacturing xocal [--ctuneoverride <ctune value>] [--offset <frequency offset in kHz>] [--store] [-storeinefuse] [-serialinterface ] [-skipinit] -d <OPN Number>`

Example: `commander manufacturing xocal -host 192.168.10.100:51917 -store --serialinterface -d SiWG917M100MGTBA`

Note:

store - Compute values are stored in flash

Storeinefuse - Compute values are stored in efuse

```
root@raspberrypi:/home/pi/vikas/commander-cli# ./commander-cli manufacturing xocal --host 192.168.250.230:51917 --store --serialinterface -d SiWG917M100MG
TBA
WARNING: No serial number or IP address given, cannot lock access to adapter.
Using host 192.168.250.230, port 51917.
Initializing target...
Storing calibration data in Flash
Loading RAM code (355328 bytes)...
Starting RAM code...
Stopping the radio...
Setting radio power:00000001, channel:00000001.
Radio stopped.
DONE
root@raspberrypi:/home/pi/vikas/commander-cli#
```

Figure 5.6. Xocal Command

```
root@raspberrypi:/home/pi/vikas/commander-cli# ./commander-cli manufacturing xocal --host 192.168.250.230:51917 --offset 12 --serialinterface -d SiWG917M1
00MGTBA
WARNING: No serial number or IP address given, cannot lock access to adapter.
Using host 192.168.250.230, port 51917.
Initializing target...
Loading RAM code (355328 bytes)...
Starting RAM code...
Attempting reset before trying again...
Loading RAM code (355328 bytes)...
Starting RAM code...
Calculated CTUNE: 0
DONE
root@raspberrypi:/home/pi/vikas/commander-cli#
```

6. Troubleshooting

- If any timeout occurs, remove the modules and re-insert them before trying to re-run the application.
- Verify if the right commander-cli is present according to the system architecture.
- Ensure that the right IP address has been specified while running the commands.
- If write is being performed after a previous write is already done, the --skipload flag must be added at the end to prevent firmware loading.
- The flag used to skip device init is –skipinit.
- XOCAL and RADIO commands should be run first after installing the driver or else it will result in a timeout or error.
- During the initial write command, --skipload flag must not be used.
- Ensure that you are using the right bin for the specified purpose. Bin files are usually present in <path-to-commander>/resources/jlink/Si917/.

7. Revision History

Revision 1.0

March, 2025

Initial release.

Simplicity Studio

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

IoT Portfolio
www.silabs.com/IoT

SW/HW
www.silabs.com/simplicity

Quality
www.silabs.com/quality

Support & Community
www.silabs.com/community

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

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

SILICON LABS

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

www.silabs.com