One-time programmable (OTP) memory programming - CYW5557x

About this document

Scope and purpose

This application note describes the method for programming the one-time programmable (OTP) non-volatile memory in the Infineon AIROC™ CYW5557x device using the PCIe or SDIO host interface for WLAN.

Intended audience

This document is intended for design and applications engineers. It contains information on:

- PCIe/SDIO HW header programming
- Basic NVRAM parameters and CIS tuple tags
- WLAN MAC address programming
- BD_ADDR programming

Acronyms and Abbreviations

In most cases, acronyms and abbreviations are defined on first use.

For a comprehensive list of acronyms and other terms used in Infineon document, click **here**.

Table of contents

1 Introduction

Infineon AIROC™ CYW5557x is a complete dual-band (2.4 GHz and 5 GHz) 802.11ax 2×2 MIMO + Bluetooth® 5.2 MAC/PHY/radio system-on-a-chip. One-time programmable (OTP) nonvolatile memory is included in the WLAN section of the device to store board-specific information such as PCIe header, product ID, manufacturer ID, and MAC address. Excluding the internal header information, up to 348 bytes of user-accessible OTP memory is available on CYW5557x for WLAN information. The application note provides the OTP programming information for both PCIe and SDIO host interfaces.

The OTP memory content, along with an editable NVRAM file (*nvram.txt* file), provides all configuration information used by the WLAN device driver to initialize and configure CYW5557x.

1.1 IoT Resources

The wealth of data available **here** will help you to select the right IoT device for your design, and quickly and effectively integrate the device into your design. You can access a wide range of information, including technical documentation, schematic diagrams, product bill of materials, PCB layout information, and software updates. You can acquire technical documentation and software from the **support community website**.

2 NVRAM content development and memory programming flow

Figure 1 shows the *nvram.txt* file content development and the OTP memory programming flow. Parameters in the *nvram.txt* file can be divided into basic and advanced categories. This document introduces basic parameters for OTP programming, advanced parameters will be introduced in the future release.

~~*Note:*~~ *Conduct the NVRAM development and OTP programming flow shown in **Figure 1** on fewer boards/modules during the product development stage. Once this process is complete and the production version of the nvram.txt file and OTP memory file is approved for production use, programming can begin for high-volume mass production as defined by each manufacturer.*

NVRAM content development and memory programming flow

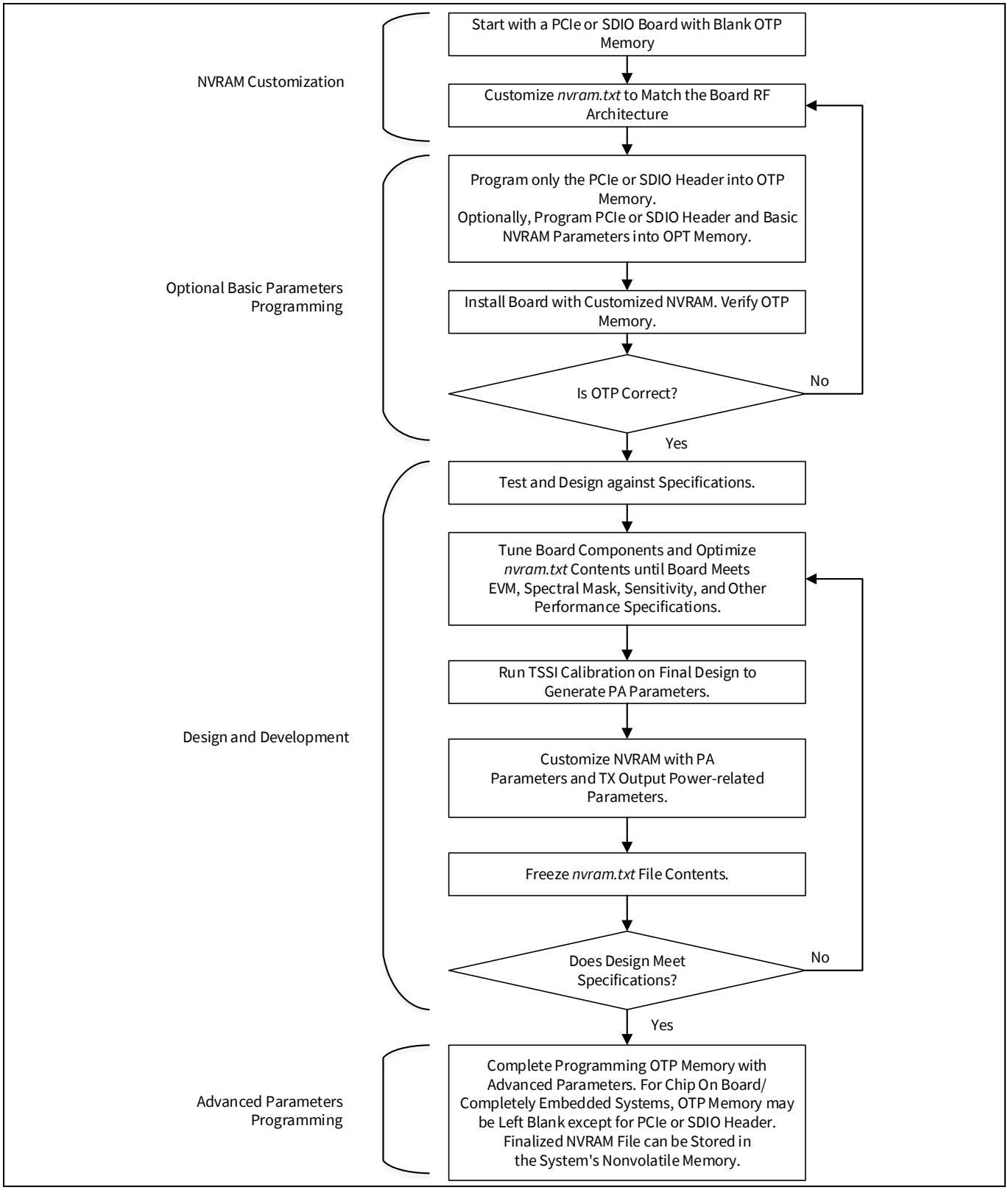


Figure 1 NVRAM development and programming flow of OTP memory

2.1 Programming the OTP memory

One item that is required in the OTP memory is the PCIe or SDIO header. When using the PCIe or SDIO interface with CYW5557x, there are certain PCIe or SDIO function settings (such as L1 sub-state for low power) that are read before the firmware and NVRAM are downloaded. To properly set these settings, the PCIe or SDIO header must be programmed into their OTP (one-time programmable, nonvolatile memory).

Note that the PCIe or SDIO header should be created as a collaboration between Infineon and the customer. A majority of the PCIe or SDIO header fields are either generic (and do not need to be changed) or are Infineon-specific, while a few are customer-specific. Coordinate with the Infineon Hardware Applications team supporting the design to confirm the appropriate PCIe or SDIO header. Note that the PCIe or SDIO header is a set block of data with a predetermined order. It does not use tuples.

2.2 Programming basic parameters into the OTP memory

Parameters in the *nvram.txt* file that are to be programmed into the OTP memory must be entered in the OTP binary map after the PCIe or SDIO header. A CIS tuple is required for each parameter in the CIS structure. Most parameters in the *nvram.txt* file have a unique identifier called the “CIS tuple tag”. The driver recognizes and parses each CIS tuple by its tag number.

Note: The PCIe or SDIO header does not use tuples, but is a set block of data with a specific ordering.

Table 1 lists the basic NVRAM parameters, the associated tag number, and the number of bytes each parameter occupies in the OTP memory. Basic parameters typically have fixed values specific to a particular device or board. The value of these parameters is often retained throughout the life of the device/board. Therefore, it is generally acceptable to program these basic parameters into the OTP memory early in development, before the design is finalized.

Table 1 Basic NVRAM parameters and CIS tuple tags

NVRAM parameter	CIS tuple tag	Length of value (in bytes)
Sromrev	0x00	1
Boardrev	0x02	2
Broadtype	0x1b	2
Macaddr	0x19	6
ccode[1]	0x0a	2
pa2ccka0[2]	0x86	6
pa5gbw4080a0, pa5gbw4080a1	0x89	48
pa2ccka1	0xA1	6
subband5gver,maxp2ga0, pa2ga0, maxp5ga0, pa5ga0	0x59	38
maxp2ga1, pa2ga1, maxp5ga1, pa5ga1	0x5A	36

In the OTP binary map, each tuple is formed by the four fragments described in **Table 2**.

[1] The value for ccode in the *nvram.txt* file is in ASCII format. It must be converted to hexadecimal format before entering it into the OTP binary map (for example, “US” = “0x55 0x53”).

[2] Test footnote (References -> Insert Footnote)

NVRAM content development and memory programming flow

Table 2 CIS tuple format

Fragment	Description
80	Indicates the beginning of a new tuple. 0x80 is specific to Infineon tuple subtags.
Length	Defines the total size (in bytes) of the tag plus the value of the tuple that occupies the OTP memory space.
Tag	Identifies a parameter in the *nvram.txt* file. A tag usually takes one byte in memory.
Value	Specifies the value of the parameter in little-endian format (first byte is the least significant byte (LSB)).

For example, the tuple is defined by the fragments that follow:

```
80          03          02          00          11
```

- 80 – Beginning of a new tuple.
- 03 – The tag (1 byte) and the value (2 bytes) occupy 3 bytes (total) in the OTP memory.
- 02 – Tag of 0x02 is the identifier for broadrev in the *nvram.txt* file.
- 00 11 – The value of boardrev in reverse hexadecimal byte or 0x1100.

Table 3 and **Table 4** provide an example OTP binary map for a CYW5557x device that contains the PCIe or SDIO header and some of the *nvram.txt* file parameters listed in **Table 1**.

Note:

1. *CIS tuples do not have to be listed in any order because each tuple begins with a unique identifier.*
2. *OTP bytes can be written only once. Only blank and zero-programmed bytes can be programmed during subsequent write cycles.*
3. *The PCIe or SDIO header is a set block of data with a predetermined order. In PCIe or SDIO header order, do not use tuples. The tuples must be programmed into the OTP memory for all PCIe or SDIO functions (such as L1SS) to operate properly.*

NVRAM content development and memory programming flow

Table 3 CYW5557x OTP map for PCIe (required in OTP)

Offset	0x0	0x1	0x2	0x3	0x4	0x5	0x6	0x7	0x8	0x9	0xa	0xb	0xc	0xd	0xe	0xf
00000040	33	00	50	00 (1)	00	00 (2)	BE	12 (3)	AF	0C	7E	3B	08	EB	C4	2B
00000050	64	2A	64	29	64	2C	E7	3C	00	46	3C	01	32	01	03	02
00000060	12	18	05	9E	14	8A	20	00	20	00	16	00	00	00	00	00
00000070	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00000080	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00000090	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
000000a0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
000000b0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
000000c0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
000000d0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
000000e0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
000000f0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00000100	31	BD	00	80	02	00	00	00	F7	3F	00	18	00	00	00	00
00000110	37	BD	00	80	02	00	00	00	F7	8F	32	19	00	00	00	00

PCIe Header
(Required in OTP)

CYW5557x OTP header

50	00 (1)
00	00 (2)
BE	12 (3)
31	BD(4)

XTAL frequency. 0x3800 for 37.4 MHz

PCI Subsystem ID, vendor specific. Use 0x0000 if unknown

PCI Subsystem vendor ID. 0x12BE is Infineon' vendor ID

Device ID. 0xBD31 is device ID for CYW5557x

WLAN PCIE HW Region = 1792 bits = 224 bytes

NVRAM content development and memory programming flow

Table 4 CYW5557x OTP map for SDIO3.0

Offset	0x0	0x1	0x2	0x3	0x4	0x5	0x6	0x7	0x8	0x9	0xa	0xb
00000120	4B	00	FF	FF	C0	00	20	04	B4	04	31	BD

CYW5557x OTP map for SDIO2.0

Offset	0x0	0x1	0x2	0x3	0x4	0x5	0x6	0x7	0x8	0x9	0xa	0xb
00000120	4A	00	FF	FF	C0	00	20	04	B4	04	31	BD

SDIO HW header

WLAN SDIO HW region = 96 bits = 12 bytes

2.3 Creating and editing the OTP binary map

Use a hexadecimal text editor to create and edit an OTP binary map. A hexadecimal text editor preserves the formatting of the *nvram.txt* file. Writing to the OTP memory requires a .bin file that fits in the OTP memory space.

For CYW5557x, the maximum size of the OTP memory is 348 bytes.

Note: *Do not use Notepad to edit the nvram.txt file. Edit the nvram.txt file using a properly formatted text editor such as Notepad++ or WordPad++ to preserve the original format of the file. Using a non-formatted text editor such as Notepad could corrupt the format of the NVRAM map, causing the driver to incorrectly read the nvram.txt file.*

1. Add or edit each byte in the OTP binary map to populate the PCIe hardware header and the CIS tuple, as described in the OTP binary map instructions provided in Programming basic parameters into the OTP .

Note: *Ensure that the OTP binary map file is been edited to match the example CYW5557x OTP binary map described in **Table 3** and **Table 4**.*

2. Save the OTP binary map as a binary image file (*.bin* extension) to the directory containing the *wl file*.

Note: *Save the filename with a .bin file extension so that the data it contains can be programmed into the OTP memory. In this application note, this file is referred as* 5557x_OTP.bin*.*

2.4 Programming WLAN parameters into the OTP memory

Use the following commands to write the MAC address into the OTP memory.

Command:

```
./wl otpraw 12160 72 <Bn>.......<B2><B1><B0><Tuple-Tag(19)><Length(07)><Tag(80)>
# This programs the MAC address <B0> is MSB and <Bn> is LSB
```

Command example:

```
./wl otpraw 12160 72 0x00904C2D800C190780
```

After OTPed:

```
[root@55572]#./wl cur_etheraddr
cur_etheraddr 0C:80:2D:4C:90:00
```

3 Programming the CYW5557x OTP memory using Jetson Xavier NX board

This section outlines the procedure to program the PCIe header to the OTP of a CYW5557x device using a Jetson Xavier Linux platform from NVIDIA.

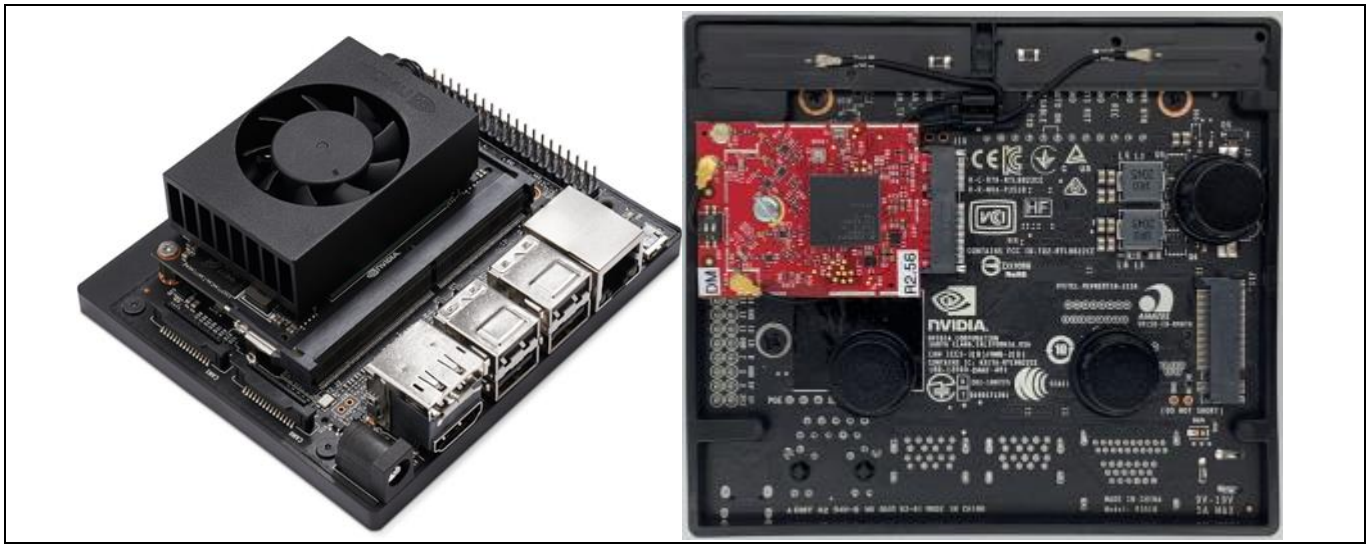


Figure 2 Jetson Xavier NX system example

The required hardware includes:

- 1x CYW5557x PCIe board with M.2 interface– this is the “DUT”
- 1x Jetson Xavier NX board from NVIDIA with Infineon image
- 1x USB mouse
- 1x USB keyboard
- 1x external monitor with HDMI or DVI connection
- 1x HDMI-to-HDMI cable or HDMI-to-DVI cable
- 1x Ethernet cable
- 1x CYW9TX2H2EVK-01 interposer card (inserted into the M.2 slot on the Jetson Xavier NX board)

The required software includes:

- Infineon PCIe MFG driver package containing driver files for CYW5557x in Linux FC19 (4.12) 64-bit platform (typically provided by Infineon).

Note: *The OTP.bin file contains the CYW5557x PCIe or SDIO header information. Follow the procedure in We can write SDIO/PCIe header information in respective HW regions. This means, same part with same OTP can be used for either SDIO or PCIe host interfaces based on GPIO strap configuration and no separate inventory required from the OEM end as well for each host interfaces.*

Programming the OTP to program the OTP memory using the *OTP.bin* file.

Note: *We can write SDIO/PCIe header information in respective HW regions. This means, same part with same OTP can be used for either SDIO or PCIe host interfaces based on GPIO strap configuration and no separate inventory required from the OEM end as well for each host interfaces.*

3.1 Programming the OTP memory

Use the MFG firmware and follow these steps to program the OTP memory:

1. While powered OFF, connect the Jetson Xavier NX board to Ethernet, USB mouse and keyboard, and monitor (an HDMI-to-DVI cable is required to connect to a DVI monitor).
2. Connect the DUT to the 60-pin connector located in the Jetson Xavier NX board.
3. Plug in the power to the Jetson Xavier NX board; the Jetson Xavier NX system should turn ON automatically. On the monitor, you should see the screen booting up to Linux.
4. At prompt, log in as "root". When logged in, type the following to go to the Linux GUI:
   ```
   > startx
   ```
5. Go to **Activities** > **Terminal** to open a command prompt. At the terminal:

   a) Type the following command:
   ```
   > ifconfig -a.
   ```
   b) Copy the MAC address for eth0 (for example, 74:d4:35:47:84:d9).

   c) Open another terminal using a text editor of your choice.

   d) If using vi, type the following command:
   ```
   > vi /etc/sysconfig/network-scripts/ifcfg-eth0p
   ```
   e) In this file, modify the MAC address to match it with the copied MAC address (for example, HWADDR=74:d4:35:47:84:d9). Then, save the file.
6. On the terminal, type the following command to reboot the Jetson Xavier NX board.
   ```
   > reboot
   ```
   The Ethernet connectivity should work after reboot, and the Jetson Xavier NX board should be able to connect to the network.
7. Once in Linux, copy the CYW5557x driver files and the *OTP.bin* file to the desired directory.

Note: *Check that the results returned by `lspci` includes the slot number of the DUT (03:00.0). The command can be used to check the revision ID after programming and a power cycle.*

8. Go to the directory where you copied the CYW5557x driver files to. Issue the driver load command as you would normally do on a Linux system:
   ```
   > rmmod bcmdhd
   > modprobe rfkill
   > insmod cfg80211.ko
   > insmod dhd.ko firmware_path=rtecdc.trxse nvram_path=nvram.txt
   > ifconfig eth1 192.168.1.101 up
   > ./wl ver
   ```

Note: *If the driver loads successfully, the command `wl ver` will return the WL version and the driver version.*

9. Once the driver is loaded successfully, you are ready to program the OTP memory.

   a) Run the following command to check the OTP dump in the OTP memory:
   ```
   > wl otpdump
   ```

Programming the CYW5557x OTP memory using Jetson Xavier NX board

b) Confirm that the CYW5557x device has never been programmed. If so, your device is ready to be programmed. Go to the directory where you copied the *OTP.bin* file to.

For PCIe, run the following command:

```
> wl otphwregn --pcie -b OTP.bin
```

For SDIO, run the following command:

```
> wl otphwregn --sdio -b OTP.bin
```

c) After programming is completed, confirm the OTP memory by dumping the OTP memory contents again:

```
> wl otpdump
```

Note: *Depending on the contents of your .bin file, the OTP dump might vary.*

If the OTP dump matches your *OTP.bin* file, the OTP programming is successful. This means that the PCIe or SDIO header is correctly programmed into your CYW5557x device.

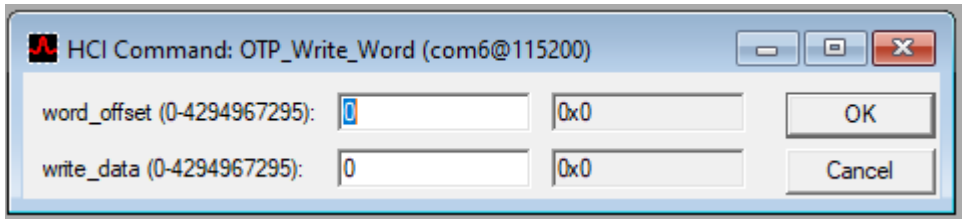
Note: *Make sure that you remove the device from the PCIe or SDIO slot before power cycling.*

4 Programming the CYW5557x OTP BD address

OTP Write Word HCI vendor-specific command:

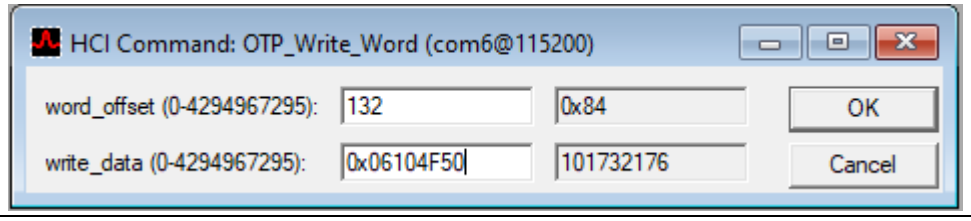
1. A new vendor-specific command has been added for H2-A1 that allows writing to the OTP memory. This command requires the following parameters:

- Word offset (32-bit-aligned offset (from OTP_START) in the OTP memory to which the OTP bits need to be configured)
- Write data (32-bit data)

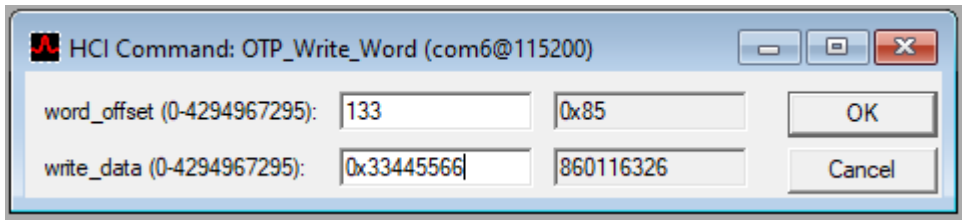


Configuring the BD address in the OTP memory:

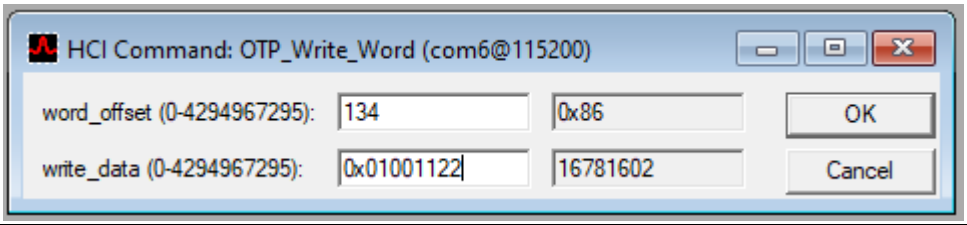
1. Issue the `OTP_Write_Word` HCI vs command with the word offset set to 132, and write data set to 0x06104F50. This configures the OTP signature and config header.



2. Issue the `OTP_Write_Word` HCI vs command with the word offset set to 133, and the write data set to 0x33445566. This configures the lower four bytes of the BD address.



3. Issue the `OTP_Write_Word` HCI vs command with the word offset set to 133, and the write data set to 0x01001122. This configures the higher four bytes of the BD address and blank section (add padding to make it word-aligned).



4. Toggle WL_REG_ON and BT_REG_ON.
5. Issue the `Read_BD_Address` HCI command. This shows the latest BD address configured in the OTP memory.

Programming the CYW5557x OTP BD address

Command example:

```
06:52.667  com31  c>  OTP_Write_Word word_offset: 0x84
HCI Command
com31@115200
[EC FD 08]: 84 00 00 00 50 4F 10 06
opcode = 0xFDEC (65004, "OTP_Write_Word")
word_offset = 0x84 (132)
write_data = 0x6104F50 (101732176)

06:52.683  com31  <c  OTP_Write_Word
HCI Command Complete Event
com31@115200
[0E 04]: 01 EC FD 00
event = 0xE (14, "Command Complete")
Num_HCI_Command_Packets = 0x1 (1)
Command_Opcode = 0xFDEC (65004, "OTP_Write_Word")
Status = 0x0 (0, "Success")

15:07.088  com31  c>  OTP_Write_Word
word_offset: 0x85
HCI Command
com31@115200
[EC FD 08]: 85 00 00 00 88 77 66 55
opcode = 0xFDEC (65004, "OTP_Write_Word")
word_offset = 0x85 (133)
write_data = 0x33445566 (860116326)

15:07.102  com31  <c  OTP_Write_Word
HCI Command Complete Event
com31@115200
[0E 04]: 01 EC FD 00
event = 0xE (14, "Command Complete")
Num_HCI_Command_Packets = 0x1 (1)
Command_Opcode = 0xFDEC (65004, "OTP_Write_Word")
Status = 0x0 (0, "Success")

15:46.294  com31  c>  OTP_Write_Word
word_offset: 0x86
HCI Command
com31@115200
[EC FD 08]: 86 00 00 00 33 22 00 01
opcode = 0xFDEC (65004, "OTP_Write_Word")
word_offset = 0x86 (134)
```

```
write_data = 0x1001122 (16781602)

15:46.311  com31  <c  OTP_Write_Word
HCI Command Complete Event
com31@115200
[0E 04]: 01 EC FD 00
event = 0xE (14, "Command Complete")
Num_HCI_Command_Packets = 0x1 (1)
Command_Opcode = 0xFDEC (65004, "OTP_Write_Word")
Status = 0x0 (0, "Success")

19:18.988  com31  <c  Read_Local_Name Name: CYW55560A1 UART FCBGA iPA dLNA
HCI Command Complete Event
com31@115200
[0E FC]:
01 14 0C 00 43 59 57 35 35 35 36 30 41 31 20 55 41 52 54 20 46 43 42 47 41 20 69 50
41 20 64 4C
4E 41 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
event = 0xE (14, "Command Complete")
Num_HCI_Command_Packets = 0x1 (1)
Command_Opcode = 0xC14 (3092, "Read_Local_Name")
Status = 0x0 (0, "Success")
Name = "CYW55560A1 UART FCBGA iPA dLNA"

19:29.320  com31  c>  Read_BD_ADDR
HCI Command
com31@115200
[09 10 00]
opcode = 0x1009 (4105, "Read_BD_ADDR")

19:29.345  com31  <c  Read_BD_ADDR  BD_ADDR: 112233445566
HCI Command Complete Event
com31@115200
[0E 0A]: 01 09 10 00 66 55 44 33 22 11
```

Programming the CYW5557x OTP BD address

```
event = 0xE (14, "Command Complete")
Num_HCI_Command_Packets = 0x1 (1)
Command_Opcode = 0x1009 (4105, "Read_BD_ADDR")
Status = 0x0 (0, "Success")
BD_ADDR = "112233445566"
```

Revision history

Document version	Date of release	Description of changes
**	2022-02-21	Initial release.

Trademarks
All referenced product or service names and trademarks are the property of their respective owners.

Edition 2022-02-21

Published by

Infineon Technologies AG

81726 Munich, Germany

Do you have a question about this document?

Go to www.Infineon.com/support

Document reference

002-34838 Rev.**

IMPORTANT NOTICE

The information contained in this application note is given as a hint for the implementation of the product only and shall in no event be regarded as a description or warranty of a certain functionality, condition or quality of the product. Before implementation of the product, the recipient of this application note must verify any function and other technical information given herein in the real application. Infineon Technologies hereby disclaims any and all warranties and liabilities of any kind (including without limitation warranties of non-infringement of intellectual property rights of any third party) with respect to any and all information given in this application note.

The data contained in this document is exclusively intended for technically trained staff. It is the responsibility of customer's technical departments to evaluate the suitability of the product for the intended application and the completeness of the product information given in this document with respect to such application.

For further information on the product, technology, delivery terms and conditions and prices please contact your nearest Infineon Technologies office (**www.infineon.com**).

WARNINGS

Due to technical requirements products may contain dangerous substances. For information on the types in question please contact your nearest Infineon Technologies office.

Except as otherwise explicitly approved by Infineon Technologies in a written document signed by authorized representatives of Infineon Technologies, Infineon Technologies' products may not be used in any applications where a failure of the product or any consequences of the use thereof can reasonably be expected to result in personal injury.