AIROC™ CYW8x570: Programming the one-time programmable (OTP) nonvolatile memory

About this document

Scope and purpose

This application note describes the method for programming the one-time programmable (OTP) nonvolatile memory in the Infineon AIROC™ CYW8x570 Automotive Wi-Fi & Bluetooth® combo chip device using the PCIe or SDIO host interface for WLAN.

Intended audience

This document is intended for design and application 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 the Infineon document, click here.

Table of contents

Table of contents

About this document **1**
Table of contents **2**
1 Introduction **3**
1.1 Internet of Things (IoT) resources 3
2 OTP memory programming flow **4**
3 Introduction of PCIe/SDIO header **6**
3.1 Creating and editing the OTP binary map 6
3.2 Programming PCIe/SDIO header into the OTP memory 9
4 Basic parameters and CIS tuple tags **11**
4.1 Programming PA parameters into the OTP memory 12
4.1.1 Programming PA parameters for 2.4 GHz/5 GHz bands core 0 12
4.1.2 Programming PA parameters for 2.4 GHz/5 GHz bands core 1 13
4.1.3 Programming PA parameters for 2.4 GHz band 11b rate 14
4.1.4 Programming PA parameters for 6 GHz band both cores 14
4.1.5 Programming maximum power setting of 6 GHz band both cores 15
4.2 Programming WLAN MAC address into the OTP memory 15
4.3 Programming customvar 1/2/3 into the OTP memory 16
5 Programming CYW8x570 BD address to OTP memory **17**
Revision history **21**
Disclaimer **22**

1 Introduction

Infineon AIROC™ CYW8x570 Automotive Wi-Fi & Bluetooth® combo chip is a tri-band (2.4/5/6 GHz) 2×2 MIMO IEEE 802.11ax WLAN MAC/baseband/radio + 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 CYW8x570 for WLAN information. The application note provides the OTP programming information for PCIe and SDIO host interfaces.

The OTP memory content with an editable NVRAM file (*nvram.txt* file) provides all configuration information used by the WLAN device driver to initialize and configure CYW8x570.

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 read the nvram.txt file incorrectly.*

1.1 Internet of Things (IoT) resources

The wealth of data available here helps you to select the suitable 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 bills of materials, PCB layout information, and software updates. You can acquire technical documentation and software from the Infineon Developer Community.

2 OTP memory programming flow

File a technical support case to request the release of NVRAM (*nvram.txt*) and PCIe/SDIO header image (OTP.bin) files which match the board RF architecture.

Figure 1 shows the OTP memory programming flow. This document introduces PCIe and SDIO hardware header and basic parameters for OTP programming.

Note: *Conduct the OTP programming flow shown in Figure 1 on fewer boards/modules during the product development stage. After 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.*

OTP memory programming flow

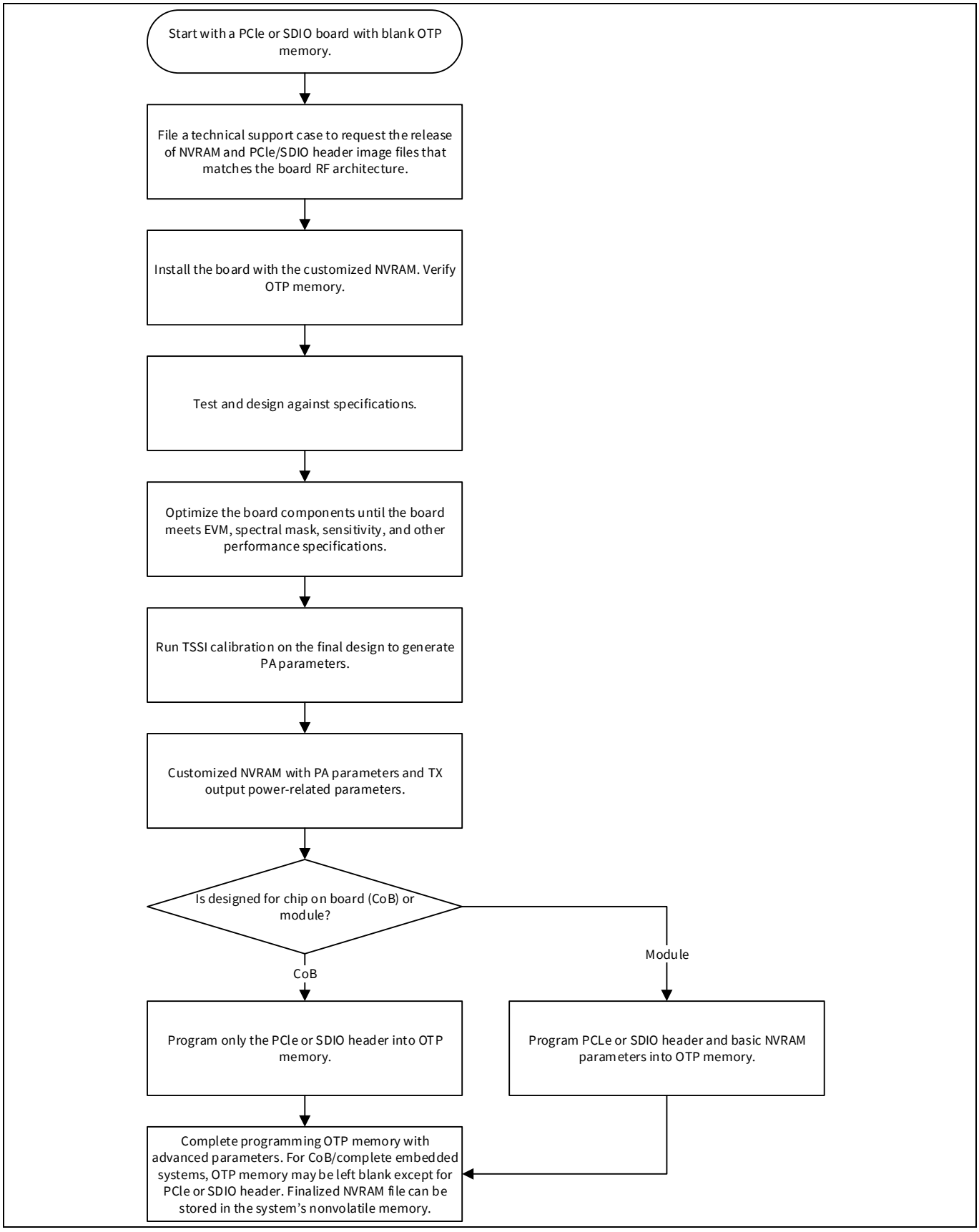


Figure 1 Programming flow of OTP memory

3 Introduction of PCIe/SDIO header

One item required in the OTP memory is the PCIe or SDIO header. When using the PCIe or SDIO interface with CYW8x570, certain PCIe or SDIO function settings (such as the L1 sub-state for low power) are read before the firmware and NVRAM downloads. To properly set these settings, program the PCIe or SDIO header into their OTP (one-time programmable, nonvolatile memory) region.

Note:

1. *Create the PCIe or SDIO header as a collaboration between Infineon and the customer. Most 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.*
2. *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.*

3.1 Creating and editing the OTP binary map

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

For CYW8x570, the maximum size of the OTP memory is 348 bytes.

1. Add or edit each byte in the OTP binary map to populate the PCIe hardware header

Note: *Ensure to edit the OTP binary map file to match the example CYW8x570 OTP binary map described in Table 1 and Table 2.*

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

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 to as 8x570_OTP.bin.*

Table 1 CYW8x570 OTP map for PCIe HW header (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	3F	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)

CYW8x570 OTP header

50	00 (1)
00	00 (2)
BE	12 (3)
3F	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. 0xBD3F is device ID for CYW89570

WLAN PCIE HW Region = 1792 bits = 224 bytes

Table 2 CYW8x570 OTP map for SDIO3.0 HW header

Offset	0x0	0x1	0x2	0x3	0x4	0x5	0x6	0x7	0x8	0x9	0xa	0xb
00000120	5B	00	FF	FF	C0	00	20	04	B4	04	31	BD

CYW8x570 OTP map for SDIO2.0 HW header

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

3.2 Programming PCIe/SDIO header into the OTP memory

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

1. Bring up the Linux host platform with DUT. You should see the screen booting up to Linux.
2. At the prompt, log in as "root". When logged in, type the following to go to the Linux GUI:
   ```
   > startx
   ```
3. Copy the CYW8x570 driver files and the *OTP.bin* file to the desired directory.
4. Go to **Activities** > **Terminal** to open a command prompt. At the terminal:

Note: *Check that the results returned by the `lspci` command include 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.*

5. Go to the directory where you copied the CYW8x570 driver files. 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 and driver versions.*

6. After the driver is loaded successfully, dump the return message with the `dmesg` command to a log file before programming it for future debugging.
7. Use the following steps to program the OTP memory.

   a) Run the following command to check the OTP dump in the OTP memory:
   ```
   > ./wl otpdump
   ```
   0x0040: 0x0000 0x0000 0x0000 0x0000
   0x0048: 0x0000 0x0000 0x0000 0x0000
   0x0050: 0x0000 0x0000 0x0000 0x0000
   0x0058: 0x0000 0x0000 0x0000 0x0000
   0x0060: 0x0000 0x0000 0x0000 0x0000
   0x0068: 0x0000 0x0000 0x0000 0x0000

   b) Confirm that the CYW8x570 device has never been programmed whose value is 0x0 from offset 0x0040 to 0x011f. If so, your device is ready to be programmed. Go to the directory where you copied the *OTP.bin* file.

   For PCIe, run the following command:
   ```
   > ./wl otphwregn --pcie -b OTP.bin
   > ./wl otpraw 507 1
   > ./wl otpraw 508 1
   ```

Introduction of PCIe/SDIO header

For SDIO, run the following command:

```
> ./wl otphwregn –-sdio -b OTP.bin
> ./wl otpraw 507 1
> ./wl otpraw 509 1
```

c) After programming is complete, 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 CYW8x570 device.

4 Basic parameters and CIS tuple tags

Parameters in the *nvram.txt* file that need to be programmed into the OTP memory must be entered in the OTP binary map after the PCIe or SDIO header. Require a CIS tuple 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 3 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 finalizing the design.

Table 3 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
subband5gver,maxp2ga0, pa2ga0, maxp5ga0, pa5ga0	0x59	38
maxp2ga1, pa2ga1, maxp5ga1, pa5ga1	0x5a	36
pa2ga2, pa2ga3	0xbc	12
pa6ga0, pa6ga1	0xbd	48
maxp6ga0, maxp6ga1	0xbe	4

In the structure of CIS tuple format, each tuple is formed by the four fragments described in Table 4.

Table 4 CIS tuple format

Fragment	Description
0x80	Indicates the beginning of a new tuple. 0x80 is specific to Infineon tuple sub tags.
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 (the first byte is the least significant byte (LSB)).

[1] The value for code 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”).

Basic parameters and CIS tuple tags

For example, the tuple is defined by the fragments as follows:

```
80              03              02              00              11
```

Where,

- 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 1 and Table 2 provide an example OTP binary map for a CYW8x570 device that contains the basic NVRAM parameters.

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.*

4.1 Programming PA parameters into the OTP memory

Use the following commands to write basic parameters into the OTP memory.

Note:

1. *The wl tool must be '18.15 RC1.49' or later.*
2. *The driver must be: ASTER_REL_100_10_77 or later.*
3. *The FW must be: LEM1RC143_REL_18_53_169 (mfgtest) or later.*

4.1.1 Programming PA parameters for 2.4 GHz/5 GHz bands core 0

Following are 2.4 GHz and 5 GHz bands core 0 PA parameters that need to be programmed with tuple tag (0x59)

```
subband5gver=0x<B0><B1> #must have 2 bytes in the OTP memory
maxp2ga0=0x<B0><B1>
pa2ga0=0x<B0><B1>, 0x<B2><B3>, 0x<B4><B5>
maxp5ga0=0x<B0>, 0x<B1>, 0x<B2>, 0x<B3>
pa5ga0=0x<B0><B1>,0x<B2><B3>, 0x<B4><B5>,0x<B6><B7>,.......,
0x<B20><B21>,0x<B22><B23>
```

Command to write 2G/5G core 0 PA parameters:

```
./wl otptlv 0x80 0x59 38
0x<<subband5gver<B1><B0>><<maxp2ga0<B1><B0>><<pa2ga0<B1><B0><B3><B2><B5><B4
>><<maxp5ga0<B0><B1><B2><B3>><<pa5ga0<B1><B0><B3><B2><B5><B4>.......<B21><B
20><B23><B22>>
# This programs the PA params <B0> is MSB and <Bn> is LSB
```

Basic parameters and CIS tuple tags

Command example:

```
[root@8x570]#./wl otptlv 0x80 0x59 38
0x0400500096ff181941fd484a4c4e48ffab18fdfc4fff181812fd54ff9b180afd4ffff9171
8fd
```

After OTPed:

```
subband5gver=0x0004
maxp2ga0=80 #hex = 0x0050
pa2ga0=-106,6424,-703 #hex = 0xff96,0x1918,0xfd41
maxp5ga0=72,74,76,78 #hex = 0x48,0x4a,0x4c,0x4e
pa5ga0=-184,6315,-771,-177,6168,-750,-172,6299,-758,-177,6137,-744
#hex =
0xff48,0x18ab,0xfcfd,0xff4f,0x1818,0xfd12,0xff54,0x189b,0xfd0a,0xff4f,0x17f
9,0xfd18
```

4.1.2 Programming PA parameters for 2.4 GHz/5 GHz bands core 1

Following are 2.4 GHz and 5 GHz bands core 1 PA parameters that need to be programmed with the same tuple tag (0x5a).

```
maxp2ga1=0x<B0><B1>
pa2ga1=0x<B0><B1>, 0x<B2><B3>, 0x<B4><B5>
maxp5ga1=0x<B0>, 0x<B1>, 0x<B2>, 0x<B3>
pa5ga1= 0x<B0><B1>,0x<B2><B3>, 0x<B4><B5>,0x<B6><B7>,.......,
0x<B20><B21>,0x<B22><B23>
```

Command to write 2G/5G core 1 PA parameters:

```
./wl otptlv 0x80 0x5a 36
0x<<maxp2ga1<B1><B0>><<pa2ga1<B1><B0><B3><B2><B5><B4>><<maxp5ga1<B0><B1><B2
><B3>><<pa5ga1<B1><B0><B3><B2><B5><B4>.......<B21><B20><B23><B22>>
# This programs the PA params <B0> is MSB and <Bn> is LSB
```

Command example:

```
[root@8x570]#./wl otptlv 0x80 0x5a 36 0x500088ff9d1842fd484a4c4e
51ff36180ffd53ff791809fd4bff401723fd4bff2b1806fd
```

After OTPed:

```
maxp2ga1=80 #hex = 0x0050
pa2ga1=-120,6301,-702 #hex = 0xff88,0x189d,0xfd42
maxp5ga1=72,74,76,78 #hex = 0x48,0x4a,0x4c,0x4e
pa5ga1= -175,6198,-753,-173,6265,-759,-181,5952,-733,-181,6187,-762
```

Basic parameters and CIS tuple tags

```
#hex =
0xff51,0x1836,0xfd0f,0xff53,0x1879,0xfd09,0xff4b,0x1740,0xfd23,0xff4b,0x182
b,0xfd06
```

4.1.3 Programming PA parameters for 2.4 GHz band 11b rate

Following are PA parameters for 2G 11b rate that need to be programmed with the same tuple tag (0xbc).

```
pa2ga2=0x<B0><B1>, 0x<B2><B3>, 0x<B4><B5>
pa2ga3=0x<B0><B1>, 0x<B2><B3>, 0x<B4><B5>
```

Command to write 2G 11b rate PA parameters for both cores:

```
./wl otptlv 0x80 0xbc 12
0x<<pa2ga2<B1><B0><B3><B2><B5><B4>><<pa2ga3<B1><B0><B3><B2><B5><B4>>
# This programs the PA params <B0> is MSB and <Bn> is LSB
```

Command example:

```
[root@8x570]#./wl otptlv 0x80 0xbc 12 0x8cff101a17fda2ff461b15fd
```

After OTPed:

```
pa2ga2=-116,6672,-745 #hex = 0xff8c,0x1a10,0xfd17
pa2ga3=-94,6982,-747 #hex = 0xffa2,0x1b46,0xfd15
```

4.1.4 Programming PA parameters for 6 GHz band both cores

Following are PA parameters for 6 GHz band on both cores that need to be programmed with the same tuple tag (0xbd).

```
pa6ga0=0x<B0><B1>,0x<B2><B3>, 0x<B4><B5>,0x<B6><B7>,.......,
0x<B20><B21>,0x<B22><B23>
pa6ga1=0x<B0><B1>,0x<B2><B3>, 0x<B4><B5>,0x<B6><B7>,.......,
0x<B20><B21>,0x<B22><B23>
```

Command to write 6 GHz band both cores PA parameters:

```
./wl otptlv 0x80 0xbd 48
0x<<pa6ga0<B1><B0><B3><B2><B5><B4>.......<B21><B20><B23><B22>><<pa6ga1<B1><
B0><B3><B2><B5><B4>.......<B21><B20><B23><B22>>
# This programs the PA params <B0> is MSB and <Bn> is LSB
```

Command example:

```
[root@8x570]#./wl otptlv 0x80 0xbd 48
0x6bffc11817fd8aff4d1afcfc5bff451802fd67ff65171bfd57ffff18effc75ffbc19fbfc4
0ffb8161bfd87ffd7180efd
```

Basic parameters and CIS tuple tags

After OTPed:

```
Pa6ga0= -149,6337,-745,-118,6733,-772,-165,6213,-766,-153,5989,-741
#hex =
0xff6b,0x18c1,0xfd17,0xff8a,0x1a4d,0xfcfc,0xff5b,0x1845,0xfd02,0xff67,0x176
5,0xfd1b
Pa6ga1= -169,6398,-786,-140,6587,-774,-193,5815,-742,-122,6358,-755
#hex =
0xff57,0x18ff,0xfcef,0xff75,0x19bc,0xfcfb,0xff40,0x16b8,0xfd1b,0xff87,0x18d
7,0xfd0e
```

4.1.5 Programming maximum power setting of 6 GHz band both cores

Following are parameters for the maximum power setting of 6 GHz band on both cores that need to be programmed with the same tuple tag (0xbe).

`maxp6ga0=0x<B0>,0x<B1>`

`maxp6ga1=0x<B0>,0x<B1>`

Command to write maximum power setting for both cores on 6 GHz band:

```
./wl otptlv 0x80 0xbe 4 0x<<maxp6ga0<B0><B1>><<maxp6ga1<B0><B1>>
```

Command example:

```
[root@8x570]#./wl otptlv 0x80 0xbe 4 0x433f4f40
```

After OTPed:

```
Maxp6ga0=67,63 #hex = 0x43,0x3f
Maxp6ga1=68,64 #hex = 0x4f,0x40
```

4.2 Programming WLAN MAC address into the OTP memory

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

Note: MAC address is reprogrammable in OTP memory with the wl wrvar command

Command example:

```
[root@8x570]#./wl wrvar macaddr=00:01:02:03:04:05
```

After OTPed:

```
[root@8x570]#./wl cur_etheraddr
cur_etheraddr 00:01:02:03:04:05
```

4.3 Programming customvar 1/2/3 into the OTP memory

Use the following commands to write the customvar1, customvar2, and customvar3 into the OTP memory.

Note: *Use the FW revision LEM1RC143_REL_18_53_121 (or later revision) and wl app revision CYPRESSEXE_REL_18_15_1_49 (or later revision) for customvar 1/2/3 programming.*

Command example:

```
[root@8x570]#./wl wrvar customvar1=0x12345678
[root@8x570]#./wl wrvar customvar2=0x9abcdef0
[root@8x570]#./wl wrvar customvar3=0x87654321
```

After OTPed:

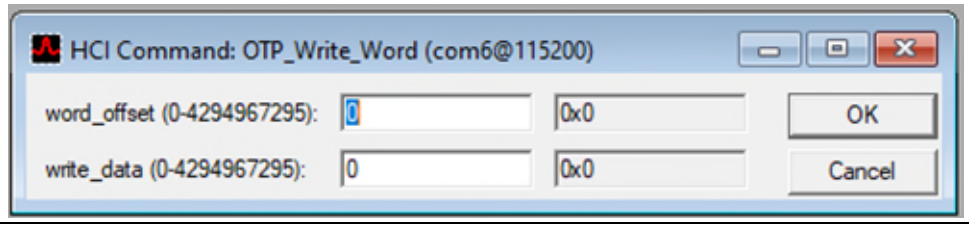
```
[root@8x570]#./wl rdvar customvar1
customvar1=0x12345678
[root@8x570]#./wl rdvar customvar2
customvar1=0x9abcdef0
[root@8x570]#./wl rdvar customvar3
customvar1=0x87654321
```

5 Programming CYW8x570 BD address to OTP memory

OTP Write Word HCI vendor-specific command:

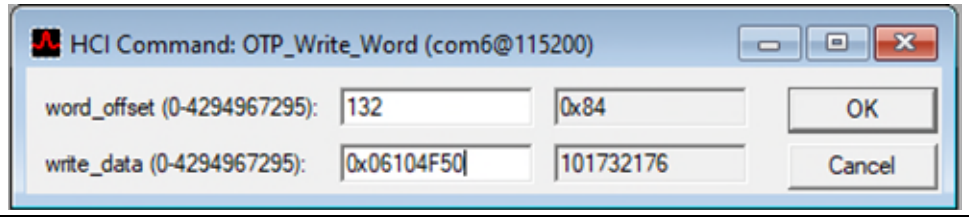
1. A new vendor-specific command has been added for H2-A1 that allows writing to the OTP memory. This command requires the following parameters:

- 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)

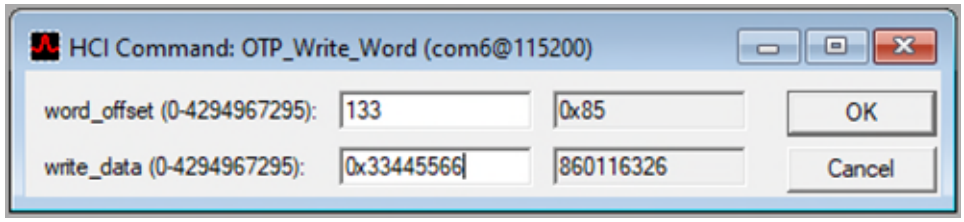


Configuring the BD address in the OTP memory:

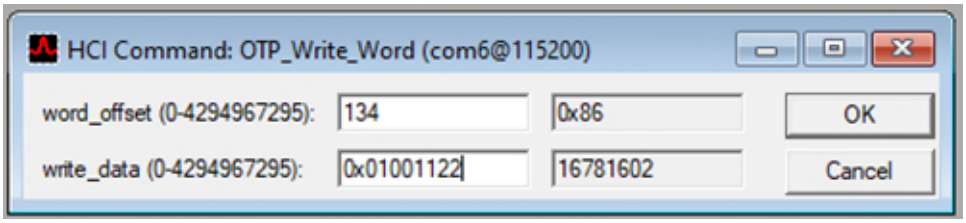
1. Issue the `OTP_Write_Word` HCI vs command with the word offset set to 132, and write the data set to 0x06104F50. This configures the OTP signature and config header.



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`.

Programming CYW8x570 BD address to OTP memory

5. Issue the `Read_BD_Address` HCI command. This shows the latest BD address configured in the OTP memory.

Code Listing 1 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")
```

Programming CYW8x570 BD address to OTP memory

Code Listing 1 Command example

```
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
```

Programming CYW8x570 BD address to OTP memory

Code Listing 1 Command example

```
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
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 revision	Date	Description of changes
**	2022-09-16	Initial release.
*A	2023-04-27	Modified programming flow of OTP memory. Added basic parameters for 6E band.

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

Edition 2023-04-27
Published by

Infineon Technologies AG
81726 Munich, Germany

© 2023 Infineon Technologies AG. All Rights Reserved.

Do you have a question about this document?
Email: erratum@infineon.com

Document reference
002-36181 Rev. *A

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.

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.