AN214841

CYW43903/CYW43907/CYW43909 OTP Programming and NVRAM Development

Associated Part Family: CYW4390X

This application note describes the method for creating and programming an nvram.txt file. This file is used to test a new board design, optimize NVRAM values, and program the one-time programmable (OTP) nonvolatile memory in CYW4390X devices, including the CYW43903, CYW43907, and CYW43909.

Contents

1 Introduction 1
1.1 Purpose and Audience 1
1.2 Cypress Part Numbering Scheme 2
1.3 Before You Begin 2
1.4 Acronyms and Abbreviations 2
2 IoT Resources 2
3 OTP Programming Considerations 2
4 NVRAM Content Development and OTP Programming Flow 4
5 Customizing the nvram.txt File 5
5.1 Using the nvram.txt File Template 5
5.2 Editing the nvram.txt File 6
5.3 Finalizing the nvram.txt File 6
6 Programming Preparation 7
6.1 OTP 7
7 OTP Programming Procedure 11
Document History Page 17
Worldwide Sales and Design Support 18

1 Introduction

The Cypress CYW4390X devices are single-chip IEEE802.11 a/b/g/n devices with integrated ARM Cortex-based processors intended for embedded Internet-of-Things (IoT) applications. For the WLAN section of the device, a one-time programmable (OTP) nonvolatile memory is available for storing board-specific information such as product ID, manufacturer ID, MAC address, and more. Up to 356 bytes of OTP memory is available for WLAN information on the CYW4390X.

The OTP memory content, together with an editable NVRAM file (referred to throughout this document as the nvram.txt file), combines to create a complete card information structure (CIS) that the device driver uses to initialize and configure the CYW4390X.

1.1 Purpose and Audience

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

- NVRAM content development and OTP programming flow
- Customizing the nvram.txt file
- OTP programming procedure

1.2 Cypress Part Numbering Scheme

Cypress is converting the acquired IoT part numbers from Broadcom to the Cypress part numbering scheme. Due to this conversion, there is no change in form, fit, or function as a result of offering the device with Cypress part number marking. The table provides Cypress ordering part number that matches an existing IoT part number.

Table 1. Mapping Table for Part Number between Broadcom and Cypress

Broadcom Part Number	Cypress Part Number
BCM4390X	CYW4390X
BCM43903	CYW43903
BCM43907	CYW43907
BCM43909	CYW43909

1.3 Before You Begin

It is recommended that the users of this application note request the following items from Broadcom's Customer Support Portal (CSP):

- A CYW4390X board reference design package that contains:
  - The reference board schematic, bill of materials, and layout.
  - The reference board `nvram.txt` template file.
- The correct WICED-SDK package.

See IoT Resources for details on accessing the Broadcom® CSP. If necessary, contact your Sales or Engineering support representative.

1.4 Acronyms and Abbreviations

In most cases, acronyms and abbreviations are defined upon first use. For a more complete list of acronyms and other terms used in Cypress documents, go to: http://www.cypress.com/glossary.

2 IoT Resources

Cypress provides a wealth of data at http://www.cypress.com/internet-things-iot to help you to select the right IoT device for your design, and quickly and effectively integrate the device into your design. Cypress provides customer access to a wide range of information, including technical documentation, schematic diagrams, product bill of materials, PCB layout information, and software updates. Customers can acquire technical documentation and software from the Cypress Support Community website (https://community.cypress.com/)

3 OTP Programming Considerations

For designs where the host and device are permanently connected together, which is typically done with a hard-wired SDIO interface, programming the OTP memory in production is optional. It is equally acceptable to store all NVRAM parameters in host firmware and keep the OTP blank in production. For devices that may be installed on different hosts, the OTP can be programmed to protect the unique MAC address and to prevent end-users from altering power control parameters (such as maximum output power and other power amplifier parameters).

For host platforms running the Linux® or Windows® XP operating system, it is not necessary to program the OTP memory during board bring-up and hardware tuning. Instead, store all required board variables in the nvram.txt file. Although OTP programming is not required for devices used on these host operating systems, nvram.txt file development is still required.

In applications where a CYW4390X chip is used as a stand-alone system, OTP can be left blank and the NVRAM parameters can be stored in external flash memory. The integrated processor in the CYW4390X can read/write the NVRAM file to/from the external flash as needed.

The initial state of all OTP bits in an unprogrammed device is 0. Individual bits can be set to 1, but once set, they can never be reset back to 0. The entire OTP array can be programmed in a single-write cycle using "wl" commands provided with the SDIO driver. Alternatively, multiple-write cycles can be used to selectively program specific fields, but only the bits that are still in the 0 state can be set to the 1 state during each programming cycle.

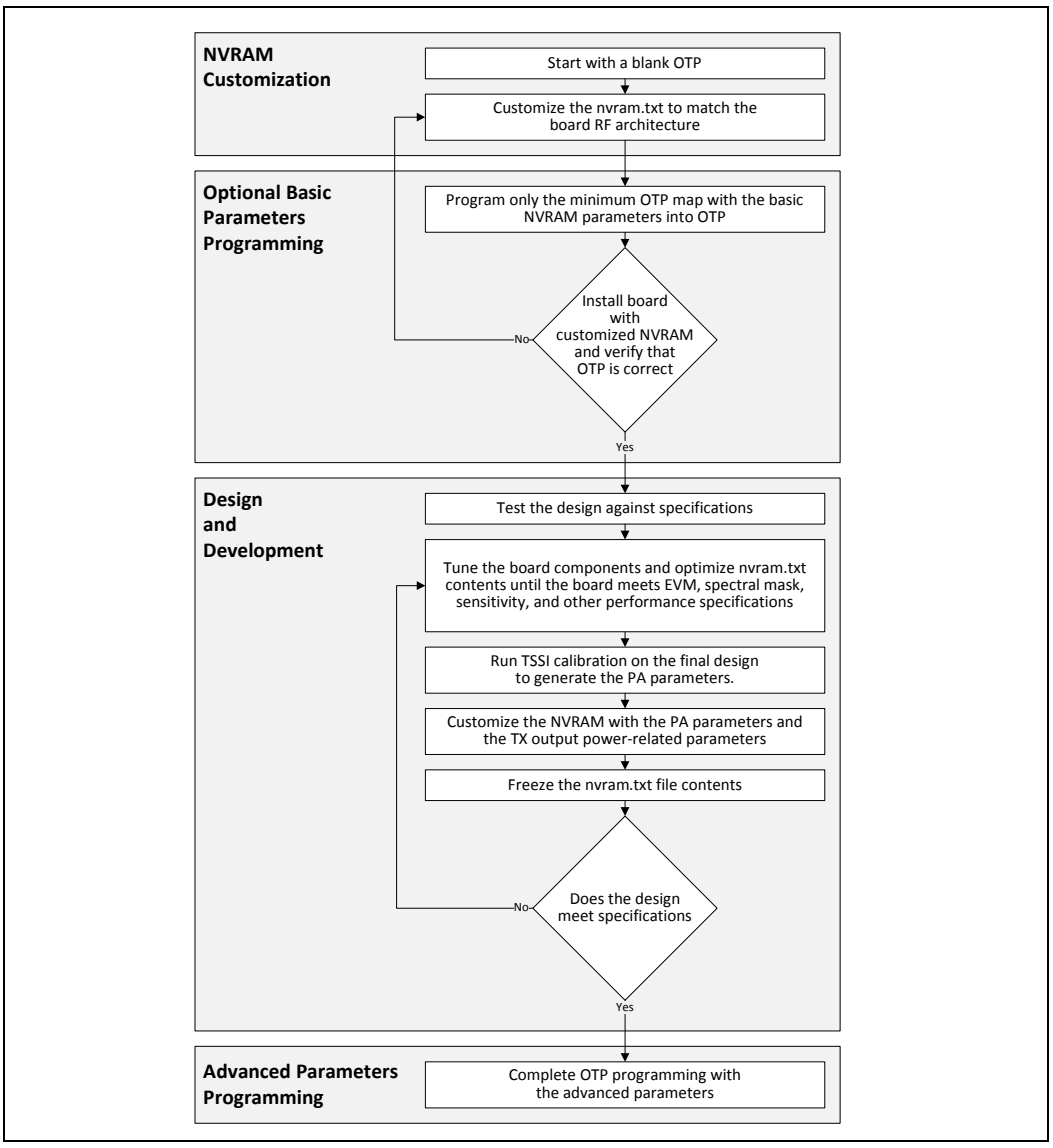
Because the OTP programming process is irreversible, Cypress recommends that board designers finalize all parameters before programming the OTP memory. Boards and modules should be tested using only the editable nvram.txt file. The nvram.txt parameters are loaded by the driver into on-chip RAM, allowing the chip to be tested even if the OTP memory has not yet been programmed. This method lets board designers tune RF components and alter critical parameters while testing boards using different versions of the nvram.txt file. As an option, a few basic parameters (such as the board type and MAC address) can be programmed into the OTP prior to board testing during development. If a parameter is present in both the on-chip OTP and the nvram.txt file, the value from the OTP overrides the value from the nvram.txt file; the WLAN driver ignores the corresponding value in the nvram.txt file.

Caution: Due to the irreversible OTP programming process, board development should be done on boards with blank OTP memory using the parameters in the editable nvram.txt file. Do not program the OTP memory until the contents of the nvram.txt file have been verified and frozen.

4 NVRAM Content Development and OTP Programming Flow

Figure 1 shows the nvram.txt file content development and the OTP programming flow. Parameters in the nvram.txt can be divided into two groups: basic parameters and advanced parameters. Pertinent OTP programming details for each phase can be found in OTP Programming Procedure.

Figure 1. NVRAM Development and OTP Programming Flow



Note: The OTP programming flow shown in Figure 1 is used only during the development stages of the project on small quantities of boards or modules. Once this process is complete and a "golden" nvram.txt file or OTP file is established, the development phase can be bypassed, and the programming can be done in high volume for mass production, following the correct manufacturing procedure defined by each manufacturer.

5 Customizing the nvram.txt File

This section describes customizing, editing, and finalizing the nvram.txt file for OTP programming.

5.1 Using the nvram.txt File Template

For each Cypress reference board design an nvram.txt file is provided, which is exactly matched to that specific-board design. Typically, the file is named after the board it supports (for example, bcm943909wcd.txt). It may be provided with the reference board design package or with the driver release. The latest version of the file can be obtained by submitting a request on the Cypress CSP. Use this file as a sample or template to begin the customization to match your own board design.

A sample nvram.txt file, with parameters that are common to Cypress CYW4390X reference design boards, is shown in Table 2. No specific order is required for the parameters in the nvram.txt file.

Parameters listed in Table 2 are design variables which must be reviewed prior to starting board or module testing. Specifically, the boardflags, the swctrlmap variables, and the number of antennas must be customized to match the board RF architecture. During the board development phase, start with the default power amplifier (PA) parameters provided in the nvram.txt template. The PA parameters are eventually optimized using Cypress's transmit signal strength indicator (TSSI) calibration tools.

Note: The parameters in Table 2 typically require tuning to each specific-board or module design. This is not an exhaustive list. Additional parameters may be added by Cypress at any time to control RF performance-related attributes of the driver. Always check with Cypress for the latest version of the nvram.txt file for the reference design before starting customization for your board design.

Table 2. NVRAM Parameters That Require Customizing for Each Board Design

NVRAM Parameter	Example Data	Description
vendid	0x14e4	Vendor ID–identifies the IC vendor: always 0x14e4, which is the Cypress PCI vendor ID.
devid	0x43d00	Device ID–identifies the device. Does not necessarily correspond to the chip number.
boardtype	0x0755	Board type: a critical parameter that should be copied form a similar reference design. The main CYW43909 (43907) reference board uses 0x0755 (0x7bb).
boardrev	0x1101	Board revision tracked by the internal test tool (optional). Example: •0x1101 converts to P101 •0x1208 converts to P208
boardflags	0xa00	Board configuration flags that define power topology, external components (ePA, eLNA), etc.
boardflags2		
boardflags3		
macaddr	00:90:4c:c5:12:38	Sets the device MAC address.
sromrev	11	Revision number of this file.
xtalfreq	37400	Onboard XTAL or oscillator frequency, in kHz.
nocrc	1	On SDIO firmware images, the image has a 32-bit CRC appended to it, and when it starts up after download it does an internal CRC check and fails if the CRC fails. nocrc=1 causes the image to skip this check.
aa2g	1	Number of antennas available for the 2.4 GHz, in bit-mapped binary format: 1 = 01b for one antenna
ccode	All	Country code.
pa2ga0	–145, 6074, –668	PA parameter values for 2.4 GHz band obtained from TSSI calibrations.
pa5ga0	blank (no text)	PA parameter values for four 5 GHz sub-bands.
cckpwroffset0	5	CCK power offset adjustment
maxp2ga0	74	The maximum possible output power in 2.4 GHz in quarter dBs called the *board maximum*. Various modulation formats (for example, data rates) have powers backed off from this limit. Generally, higher rates such as mcs7 have larger offsets compared to those for lower data rates (such as rate 1, or 2). The units are quarter dBs. So maxp2ga0 = 74 translates to 0.25 × 74 dBm = 18.5 dBm
maxp5ga0	70	

Table 2. NVRAM Parameters That Require Customizing for Each Board Design (Continued.)

NVRAM Parameter	Example Data	Description
il0macaddr[a]	00:90:4c:c5:12:38	The il0 MAC address format enables the firmware to use a default (dummy) MAC address if the OTP is blank (that is, if no valid MAC address has previously been programmed into the OTP).

a. When devices with blank OTP are used, the firmware may fail to load unless a MAC address is provided. Cypress recommends using an il0macaddr value during board development, which provides the driver with a dummy MAC address. With il0macaddr in nvram.txt, the driver loads even when the OTP is blank. As an alternative a unique MAC address can be programmed into OTP using the basic parameters OTP map shown in Figure 2 . In production, each board or module should have a valid and unique MAC address programmed into its OTP. The il0macaddr is ignored by the driver when a MAC address is programmed in the OTP.

5.2 Editing the nvram.txt File

The nvram.txt file content should be edited in a properly formatted text editor, such as Notepad++ or WordPad++, so that the original format of the file is preserved. Using a non-formatted text editor (such as Notepad) may corrupt the format of the NVRAM map, thus causing the driver to fail to correctly read the nvram.txt file.

5.3 Finalizing the nvram.txt File

After the final PA parameters for the design have been generated, edit the nvram.txt file to update the PA parameters derived from using the TSSI tool, and then adjust the Tx output power-related parameters in the nvram.txt file. Run output power tests (using the updated nvram.txt file) to verify that these parameters are providing the correct output power. Verify that the RF performance (such as EVM, spectral mask, and rxper) meets design specifications.

Cypress recommends running a regulatory prescan to verify that the required output power can be delivered without violating the band-edge limits. If the band-edge limits cannot be met, it may be necessary to reduce the output power at the band-edge channels.

After all prototype tests have passed and all nvram.txt file parameters have been optimized and frozen, users can select the needed parameters to program the OTP for production.

The CYW4390X chips have up to 356 bytes of space in the OTP memory available for user data. Given the limited space in the OTP, it is impossible to program the entire nvram.txt file to the OTP. The programmer must be very careful to select only the necessary parameters that go into the OTP. Parameters that typically go into the OTP are those that are unique to the board (such as MAC address) and those that are required to satisfy local regulatory requirements, which are usually output power-related parameters (such as maximum output power, power offset per-rate, PA parameters, country code, etc.).

6 Programming Preparation

6.1 OTP

Prior to OTP programming, an OTP binary map file must be prepared and edited with correct values. The SDIO OTP data format is based on the CIS as defined by the PCMCIA/SD Card Association. The CIS data contains the hardware header followed by one or more data blocks, where each data block (or tuple) contains the type, length, and value of the tuple. Refer to Appendix A: “CIS Map" for details.

At the start of the OTP map, a string called the SDIO hardware header must be present preceding any NVRAM variables. When a driver detects content in the OTP, the SDIO hardware header is required to boot up the CYW4390X devices via the SDIO interface. Therefore, the SDIO hardware header is the minimum set of parameters when programming an OTP. The hardware header is shown in Figure 2 . Any other parameters needed to be programmed to the OTP are appended after the SDIO hardware header (refer to Creating and Editing the OTP Binary Map).

6.1.1 Basic Parameters

Parameters in the nvram.txt file that are to be programmed to the OTP must follow the SDIO hardware header in the OTP binary map. Each parameter requires a CIS tuple 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. For a list of the CIS tuples and their tag numbers, see Appendix A: “CIS Map".

Table 3 lists the common basic nvram.txt file parameters with their tag numbers and the byte size they occupy in the OTP memory space. Basic parameters are typically values fixed to a specific device or board and tend to retain their values across the life of the device/board. For this reason, it is generally acceptable to program these basic parameters to the OTP early in the development, before the design is frozen.

Table 3. Basic NVRAM Parameters CIS Tuple Tags

NVRAM Parameter	CIS Tuple Tag	Length of Value (in Bytes)
sromrev	0x00	1
boardtype	0x1b	2
macaddr	0x19	6

In the OTP binary map, each tuple is formed by the four fragments described in Table 4.

Table 4. CIS Tuple Format

Fragment	Description
80	This number indicates the beginning of a new tuple. 0x80 is specific to Cypress tuple subtags.
Length	The length defines the total size (in bytes) of the tag plus the value of the tuple that occupies the OTP memory space.
Tag	The tag identifies a parameter in the nvram.txt file. A tag usually takes one byte in memory.
Value	The value of the parameter is in little-endian format (that is, the first byte is the least-significant byte).

For example, a tuple that looks like the following is defined by the fragments listed in Table 5:

8 0	0 3	1 b	5 5	0 7

Table 5. An Example of Tuple Definition

Fragment	Description
80	Beginning of a new tuple.
03	The tag (1 byte) and the value (2 bytes) will occupy 3 bytes total in the OTP memory.
00	Tag of 0x00 is the identifier for boardtype in the nvram.txt file.
5507	The value of boardtype in reverse binary byte. It is 0x0755 as seen in the nvram file (boardtype = 0x0755)

Figure 2 shows an example of the OTP binary map for the CYW4390X devices that contains the SDIO header and the nvram.txt file parameters listed in Table 3. The memory address is relative to the fixed constant 0x0040. So, the memory at absolute byte-address 0x0040 (64, in decimal) contains the value of 0x4b as shown in Figure 2.

Figure 2. Example OTP Binary Map Containing Basic Parameters

Relative Offset	*0x0*	*0x1*	*0x2*	*0x3*	*0x4*	*0x5*	*0x6*	*0x7*	*0x8*	*0x9*	*0xa*	*0xb*	*0xc*	*0xd*	*0xe*	*0xf*
0x0000	4b	00	ff	ff	06	00	20	04	d0	02	85	ab	80	02	00	0b
0x0008	80	03	1b	55	07	80	07	19	66	55	44	33	22	11	00	00
0x0010	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0x0020	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0x0030	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
...	" "	" "	" "	" "	" "	" "	" "	" "	" "	" "	" "	" "	" "	" "	" "	" "
0x0150	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0x0160	00	00	ff	ff												

Legend:
SDIO Header
sromrev
boardtype: 0775 (change)
MAC Address Change: 66:55:44:22:11
End of CIS map

In this example, the values for each parameter are as follows:

- The SDIO header is:
  - CYW43903: 4b 00 ff ff 06 00 20 04 d0 02 7f ab
  - CYW43907: 4b 00 ff ff 06 00 20 04 d0 02 83 ab
  - CYW43909: 4b 00 ff ff 06 00 20 04 d0 02 85 ab
- sromrev = 0x0b (or 11) is fixed (the tuple for sromrev is 80 02 00 0b).
- boardtype = 0x755 (should be copied from the Cypress reference board on which the customer design is based.) In general, the value of 0x0755 (0x07bb) should work for CYW43909 (CYW43907). May use 0x0755 also for the CYW43903.
- Macaddr = 66:55:44:33:22:11. A unique MAC address needs to be programmed for each board.

Note:

- CIS tuples do not have to be in a particular order because each tuple begins with a unique identifier (80).
- OTP bytes can be written to only once, so only blank or zero-programmed bytes can be programmed on subsequent write cycles.
- The end marker sets the end of the CIS region (user partition) so that the so that the "ff ff" pattern falls on the 0x0162 and 0x0163 byte locations. This gives maximum available user space of 356 bytes.

6.1.2 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 formatting of the `nvram.txt` file. *Do not use Notepad* as it modifies formatting and corrupts the `nvram.txt` file. Writing to the OTP requires a `.bin` file that fits within the OTP size. For the CYW4390X devices, the user partition of the OTP has a maximum size of 356 bytes. Figure 3 shows an example of this partition for the CYW43909. For the CYW43903 and CYW43907, change the SDIO header and board type, as discussed earlier (see Figure 4 and Figure 5).

Figure 3. Basic CIS Map for the CYW43909

Legend:
- SDIO Header
- sromrev
- boardtype
- MAC address
- End of CIS map

Rel. Address	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00000000	4B	00	FF	FF	06	00	20	04	D0	02	85	AB	80	02	00	0B
00000010	80	03	1B	55	07	80	07	19	66	55	44	33	22	11	00	00
00000020	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00000030	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00000040	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00000050	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00000060	00	00	00	00	00	00	00	00	00	00	00	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	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00000110	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00000120	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00000130	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00000140	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00000150	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00000160	00	00	FF	FF												

Add or edit each byte in the map to fill in the SDIO hardware header and the CIS tuple according to the OTP binary map instructions described earlier in this section. The map shown in Figure 3 has been edited to match the CYW4390X OTP binary map example in Figure 2 .

When editing is complete, save the file and manually change the `.txt` file extension to `.bin`. The file name must have `.bin` extension so that it can be programmed to the OTP. Store this `.bin` file in the working directory that contains the remote WL utility, which is discussed in the next section.

For example purposes, this file is referred to as `4390Xcis_map.bin` in the following instructions.

6.1.3 Setting Up the Programming Environment

The CYW4390X OTP memory is programed using remote WL over a UART connection. Remote WL is a Windows application that sends WL commands (for example, `ciswrite`, `cisdump`, etc.) over the serial connection to the `mfg_test` WICED™ application that is running on the CYW4390X ACPU. The ACPU forwards the commands to the WLAN firmware, which then programs the OTP.

The procedure details are specified below:

1. Connect the CYW4390X WICED board to a Windows PC with an FTDI USB cable then verify that UART communication is successful.
2. Compile and download the mfg_test WICED application (default is Wiced-SDK\apps\test\mfg_test) to the CYW4390X WICED board using the following target build string:
   `$Wiced-SDK> make test.mfg_test-ThreadX-NetX_Duo-BCM943909WCD1_3 download run`
3. Power cycle the CYW4390X WICED board.
4. From the Windows PC console, change the directory to `$Wiced-SDK\libraries\test\wl_tool`, where the remote WL Windows application (`wl43909B0.exe`) is present.
5. Execute the WL command to print the WLAN firmware version (`ver`) using the remote WL Windows application (`wl43909B0.exe --serial <serial_port>`).

Note: The <serial_port> number will differ from one computer to another.

Example:

```
$Wiced-SDK\libraries\test\wl_tool> wl43909B0.exe --serial 6 ver
7.16 RC99.19
wl0: Jun 18 2015 10:58:40 version 7.15.168 (TOB) (r521252 WLTEST) FWID 01-3da9c1dd
```

If successful, the WLAN firmware version will be printed as shown in the example above. This confirms that the emote WL and `mfg_test` communication infrastructure is working as expected.

Note: If another serial terminal application is open on the same COM port, WL will fail to open.

The remote WL application setup can be used to execute any WL command supported by the WLAN firmware.

Note: In the remainder of this document `wl` is replaced with `wl43909B0.exe --serial <serial_port>` in all programming commands.

7 OTP Programming Procedure

To program the above OTP binary map and some other important parameters to the CYW4390X devices:

1. Set up the programming environment as detailed earlier.
2. Program the OTP binary map: `wl ciswrite 4390Xcis_map.bin`.

Note: The `*cis_map.bin` binary file is different for each device.
See Programming the CIS Map and then return here.

3. Confirm that the OTP binary map is programmed successfully by running the command `> wl cisdump`.
4. Optionally, the following command may also be used to output OTP contents to the console:

```
> wl otpdump
```

The wl otpdump command outputs the entire set of OTP contents with absolute addresses. The user partition is a subset that spans address spaces from 0x40 to 0x01A3 inclusive.

5. Power cycle the CYW4390X WICED board.

7.1 Programming the CIS Map

Figure 3 shows the basic CIS map for the CYW43909 after the MAC address is updated.

Figure 4 shows the basic CIS map for the CYW43907 after the MAC address is updated.

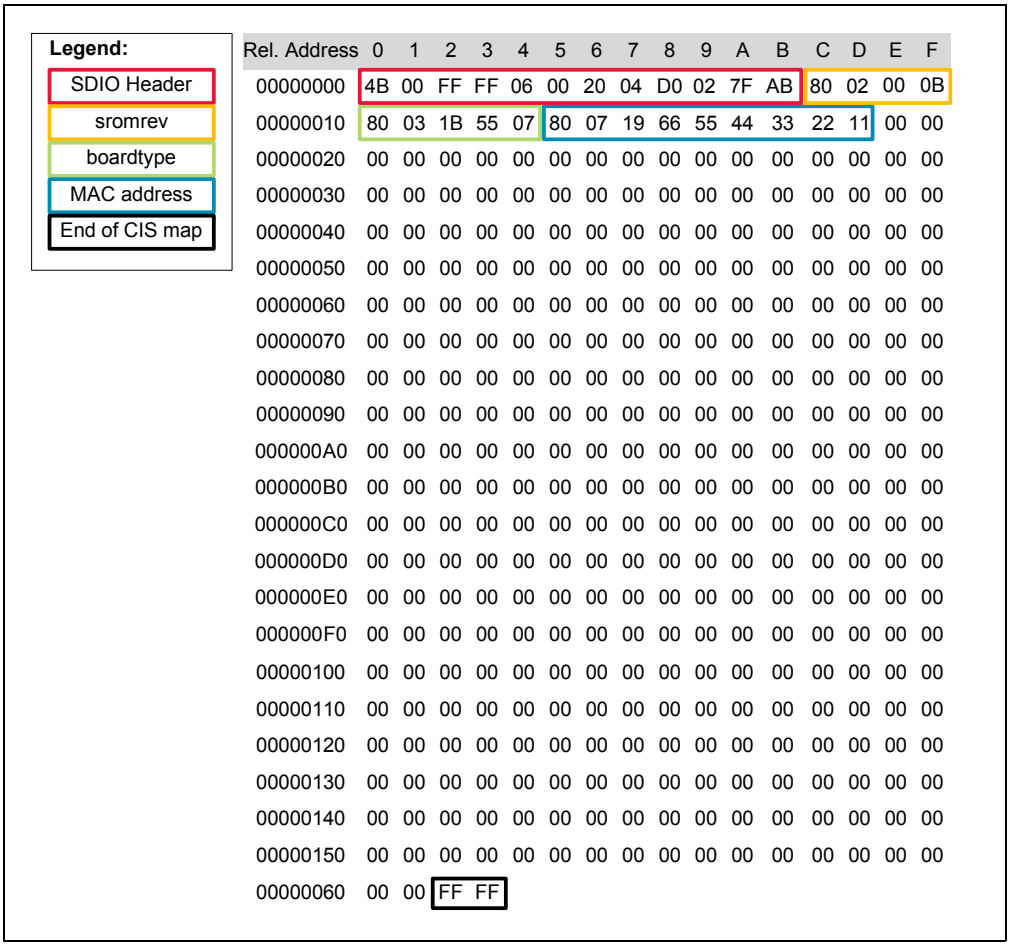
Figure 5 shows the basic CIS map for the CYW43903 after the MAC address is updated.

Figure 4. Basic CIS Map for the CYW43907

Legend:
- SDIO Header
- sromrev
- boardtype
- MAC address
- End of CIS map

Rel. Address	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00000000	4B	00	FF	FF	06	00	20	04	D0	02	83	AB	80	02	00	0B
00000010	80	03	1B	BB	07	80	07	19	66	55	44	33	22	11	00	00
00000020	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00000030	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00000040	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00000050	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00000060	00	00	00	00	00	00	00	00	00	00	00	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	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00000110	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00000120	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00000130	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00000140	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00000150	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00000160	00	00	FF	FF												

Figure 5. Basic CIS Map for the CYW43903



Rel. Address	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00000000	4B	00	FF	FF	06	00	20	04	D0	02	7F	AB	80	02	00	0B
00000010	80	03	1B	55	07	80	07	19	66	55	44	33	22	11	00	00
00000020	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00000030	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00000040	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00000050	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00000060	00	00	00	00	00	00	00	00	00	00	00	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	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00000110	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00000120	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00000130	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00000140	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00000150	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00000060	00	00	FF	FF												

Appendix A: CIS Map

Table 6 and Table 7 list the CIS map (standard tuple tags and Cypress subtags) for SDIO devices.

Table 6. Standard Tuple Tags

Name	Tag	Length	Format	Variables	Description
CISTPL_VERS_1	0x15			manf	CIS version, manufacturer, device, and version strings.
				productname	
CISTPL_MANFID	0x20	4		manfid	Manufacturer and device ID.
				prodid	
CISTPL_FUNCID	0x21				Function identification
CISTPL_FUNCE	0x22				Function extensions
CISTPL_FUNCE	0x22	8			Subtype = FUNCE_mac(0x4), value: 6 bytes MAC address.
CISTPL_CFTABLE	0x1b	2		regwindowsz	Configuration table entry
CISTPL_FID_SDIO	0x0c				Extensions defined by SDIO specification
CISTPL_BRCM_HNBU	0x80				Cypress specific tuple subtag identifier
CISTPL_END	0xff				End of the CIS tuple chain

Table 7. Cypress Tuple Subtags

Name	Tag	Length	Format	Variables	Description
HNBU_SROMREV	0x00	1		sromrev	SROM revision
HNBU_CHIPID	0x01	4/6/8/10		vendid	Vendor and device ID
				devid	
				chiprev	
				subvendid	
				subdevid	
				boardtype	
HNBU_BOARDREV	0x02	1/2		boardrev	Board revision
HNBU_PAPARMS	0x03	2/8/9		pa0b0	PA parameters: 8 (sromrev = 1) or 9 (sromrev > 1) bytes
				pa0b1	
				pa0b2	
				pa0itssit	
				pa0maxpwr	
				opo	
HNBU_AA	0x06	1/2		aa2g	Antennas available
				aa5g	
HNBU_AG	0x07	1/2/3/4		ag0	Antenna gain
				ag1	
				ag2	
				ag3	

Table 7. Cypress Tuple Subtags (Continued.)

Name	Tag	Length	Format	Variables	Description
HNBU_BOARDFLAGS				boardflags	Board flags depend on the front-end module used. Please confirm this value with Cypress.
HNBU_CCODE	0x0a	3		ccode	Country code (2 bytes ASCII + 1 byte CCTL) The CCTL means indoor/outdoor, but it is never used.
				cctl	
HNBU_CCKPO	0x0b	2		cckpo	CCK power offsets
HNBU_OFDMPO	0x0c	4		ofdmpo	11g OFDM power offsets
HNBU_PAPARMS5G	0x0e	22		pa1b0	5G PA parameters for low/mid/high band.
				pa1b1	
				pa1b2	
				pa1lob0	
				pa1lob1	
				pa1lob2	
				pa1hib0	
				pa1hib1	
				pa1hib2	
				pa1itssit	
				pa1maxpwr	
				pa1lomaxpwr	
				pa1himaxpwr	
HNBU_ANT5G	0x0f	2		aa5g	5G antennas available/gain
				ag1	
HNBU_XTALFREQ	0x13	4		xtalfreq	Crystal frequency in kilohertz.
HNBU_TRI2G	0x14	1		triso2g	2G TR isolation
HNBU_TRI5G	0x15	3		triso5gl	5G TR isolation
				triso5g	
				triso5gh	
HNBU_RXPO2G	0x16	1		rxpo2g	2G RX power offset
HNBU_RXPO5G	0x17	1		rxpo5g	5G RX power offset
HNBU_BOARDNUM	0x18	2		boardnum	Board serial number, independent of MAC address.
HNBU_MACADDR	0x19	6		macaddr	MAC address override for the standard CIS LAN_NID.
HNBU_BOARDTYPE	0x1b	2		boardtype	Board type

Table 7. Cypress Tuple Subtags (Continued.)

Name	Tag	Length	Format	Variables	Description
HNBU_FEM	0x23	2/4		antswctl2g(15-11)	Front-end module variables.
				triso2g(10-8)	
				pdetrange2g(7-3)	
				extpagain2g(2-1)	
				tssipos2g(0)	
				antswctl5g(15-11)	
				triso5g(10-8)	
				pdetrange5g(7-3)	
				extpagain5g(2-1)	
				tssipos5g(0)	
HNBU_PO_CCKOFDM	0x28	6/18		cck2gpo	Power offset for 2G in CCK and OFDM.
				ofdm2gpo	
				ofdm5gpo	
				ofdm5glpo	
				ofdm5ghpo	
HNBU_OFDMPO5G	0x37	12		ofdm5gpo	Power offset for 5G in OFDM.
				ofdm5glpo	
				ofdm5ghpo	
HNBU_PO_MCS2G	0x29	16		mcs2gpo0	Power offset for 2G in modulation coding scheme (MCS) rate.
				mcs2gpo1	
				mcs2gpo2	
				mcs2gpo3	
				mcs2gpo4	
				mcs2gpo6	
				mcs2gpo7	
HNBU_PO_MCS5GM	0x2a	16		mcs5gpo0	Power offset for 5G mid-band in MCS rate.
				mcs5gpo1	
				mcs5gpo2	
				mcs5gpo3	
				mcs5gpo4	
				mcs5gpo6	
				mcs5gpo7	

Table 7. Cypress Tuple Subtags (Continued.)

Name	Tag	Length	Format	Variables	Description
HNBU_PO_MCS5GLH	0x2b	32		mcs5glpo0	Power offset for 5G low/high band in MCS rate.
				mcs5glpo1	
				mcs5glpo2	
				mcs5glpo3	
				mcs5glpo4	
				mcs5glpo6	
				mcs5glpo7	
				mcs5ghpo0	
				mcs5ghpo1	
				mcs5ghpo2	
				mcs5ghpo3	
				mcs5ghpo4	
				mcs5ghpo6	
				mcs5ghpo7	
HNBU_PO_40M	0x2e	2		bw40po	2g: bits 0–3
					5g: bits 4–7
					5gl: bits 8–11
					5gh: bits 12–15
HNBU_PO_40MDUP	0x2f	2		bwduppo	2g: bits 0–3
					5g: bits 4–7
					5gl: bits 8–11
					5gh: bits 12–15
HNBU_CCKFILTTYPE	0x36	1		cckdigfilttype	CCK digital filter selection option.

Document History Page

Document Title: AN214841 - CYW43903/CYW43907/CYW43909 OTP Programming and NVRAM Development Document Number: 002-14841				
Rev.	**ECN No.**	**Orig. of Change**	**Submission Date**	**Description of Change**
**	-	-	06/23/2015	4390X-AN100-R Initial release
*A	-	UTSV	07/29/2015	4390X-AN101-R • Updated: • OTP Programming Considerations . • Using the nvram.txt File Template . • Figure 2: “Example OTP Binary Map Containing Basic Parameters". • Note on page 8. • Creating and Editing the OTP Binary Map . • Setting Up the Programming Environment . • OTP Programming Procedure . • Figure 4: “Basic CIS Map for the CYW43907". • Figure 5: “Basic CIS Map for the CYW43903". **Removed:** • “Programming Secure Boot and Associated Keys” • “Programming Secure Boot and Secure Flash” • “Enabling Secure SFlash”. • “OTP Maps and Programming Commands”.
*B	5445162	UTSV	09/28/2016	Added Cypress Part Numbering Scheme and Mapping table. Updated to Cypress format.
*C	5817681	AESATP12	07/14/2017	Updated logo and copyright.

Worldwide Sales and Design Support

Worldwide Sales and Design Support

Cypress maintains a worldwide network of offices, solution centers, manufacturers' representatives, and distributors. To find the office closest to you, visit us at Cypress Locations.

Products

ARM® Cortex® Microcontrollers	cypress.com/arm
Automotive	cypress.com/automotive
Clocks & Buffers	cypress.com/clocks
Interface	cypress.com/interface
Internet of Things	cypress.com/iot
Memory	cypress.com/memory
Microcontrollers	cypress.com/mcu
PSoC	cypress.com/psoc
Power Management ICs	cypress.com/pmic
Touch Sensing	cypress.com/touch
USB Controllers	cypress.com/usb
Wireless Connectivity	cypress.com/wireless

PSoC® Solutions

PSoC 1 | PSoC 3 | PSoC 4 | PSoC 5LP | PSoC 6

Cypress Developer Community

Forums | WICED IOT Forums | Projects | Video | Blogs | Training | Components

Technical Support

cypress.com/support

All other trademarks or registered trademarks referenced herein are the property of their respective owners.

Cypress Semiconductor
198 Champion Court
San Jose, CA 95134-1709

© Cypress Semiconductor Corporation, 2015-2017. This document is the property of Cypress Semiconductor Corporation and its subsidiaries, including Spansion LLC ("Cypress"). This document, including any software or firmware included or referenced in this document ("Software"), is owned by Cypress under the intellectual property laws and treaties of the United States and other countries worldwide. Cypress reserves all rights under such laws and treaties and does not, except as specifically stated in this paragraph, grant any license under its patents, copyrights, trademarks, or other intellectual property rights. If the Software is not accompanied by a license agreement and you do not otherwise have a written agreement with Cypress governing the use of the Software, then Cypress hereby grants you a personal, non-exclusive, nontransferable license (without the right to sublicense) (1s) under its copyright rights in the Software (a) for Software provided in source code form, to modify and reproduce the Software solely for use with Cypress hardware products, only internally within your organization, and (b) to distribute the Software in binary code form externally to end users (either directly or indirectly through resellers and distributors), solely for use on Cypress hardware product units, and (2) under those claims of Cypress's patents that are infringed by the Software (as provided by Cypress, unmodified) to make, use, distribute, and import the Software solely for use with Cypress hardware products. Any other use, reproduction, modification, translation, or compilation of the Software is prohibited.

TO THE EXTENT PERMITTED BY APPLICABLE LAW, CYPRESS MAKES NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS DOCUMENT OR ANY SOFTWARE OR ACCOMPANYING HARDWARE, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. To the extent permitted by applicable law, Cypress reserves the right to make changes to this document without further notice. Cypress does not assume any liability arising out of the application or use of any product or circuit described in this document. Any information provided in this document, including any sample design information or programming code, is provided only for reference purposes. It is the responsibility of the user of this document to properly design, program, and test the functionality and safety of any application made of this information and any resulting product. Cypress products are not designed, intended, or authorized for use as critical components in systems designed or intended for the operation of weapons, weapons systems, nuclear installations, life-support devices or systems, other medical devices or systems (including resuscitation equipment and surgical implants), pollution control or hazardous substances management, or other uses where the failure of the device or system could cause personal injury, death, or property damage ("Unintended Uses"). A critical component is any component of a device or system whose failure to perform can be reasonably expected to cause the failure of the device or system, or to affect its safety or effectiveness. Cypress is not liable, in whole or in part, and you shall and hereby do release Cypress from any claim, damage, or other liability arising from or related to all Unintended Uses of Cypress products. You shall indemnify and hold Cypress harmless from and against all claims, costs, damages, and other liabilities, including claims for personal injury or death, arising from or related to any Unintended Uses of Cypress products.

Cypress, the Cypress logo, Spansion, the Spansion logo, and combinations thereof, WICED, PSoC, CapSense, EZ-USB, F-RAM, and Traveo are trademarks or registered trademarks of Cypress in the United States and other countries. For a more complete list of Cypress trademarks, visit cypress.com. Other names and brands may be claimed as property of their respective owners.