AN214915

OTP Programming and NVRAM Development in SDIO Mode - CYW43241

Associated Part Family: CYW43241

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 the CYW43241 device.

Contents

1 About This Document

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
- SDIO Windows® XP driver installation
- 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
BCM43241	CYW43241

1.3 Acronyms and Abbreviations

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

1.4 Before You Begin

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

- A CYW43241 board reference design package that contains:
  - The reference board schematic, bill of materials, and layout. Be sure to specify either the FCBGA or WLCSP package, and either single-band (2.4 GHz only) or dual-band (2.4 GHz and 5 GHz).
  - An nvram.txt template file for the reference board.
- A Windows XP or Linux® device driver for the relevant SDIO device
- Cypress transmit signal strength indicator (TSSI) calibration tools

Refer to IoT Resources on page 2 for details on accessing Cypress CSP. If necessary, contact your Sales or Engineering support representative.

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 (http://community.cypress.com/).

3 Introduction

The Cypress CYW43241 is a single-chip IEEE802.11 a/b/g/n 2 × 2 MIMO + BT/FM device intended for embedded 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. Including header information, up to 232 bytes of OTP memory is available for WLAN information on the CYW43241. Although the CYW43241 WLAN section provides the option of using either SDIO or HSIC for the host interface, this application note addresses only SDIO applications.

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 CYW43241.

4 OTP Programming Considerations

For designs where the host and device are permanently connected, which is typically done with a hardwired 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.

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.

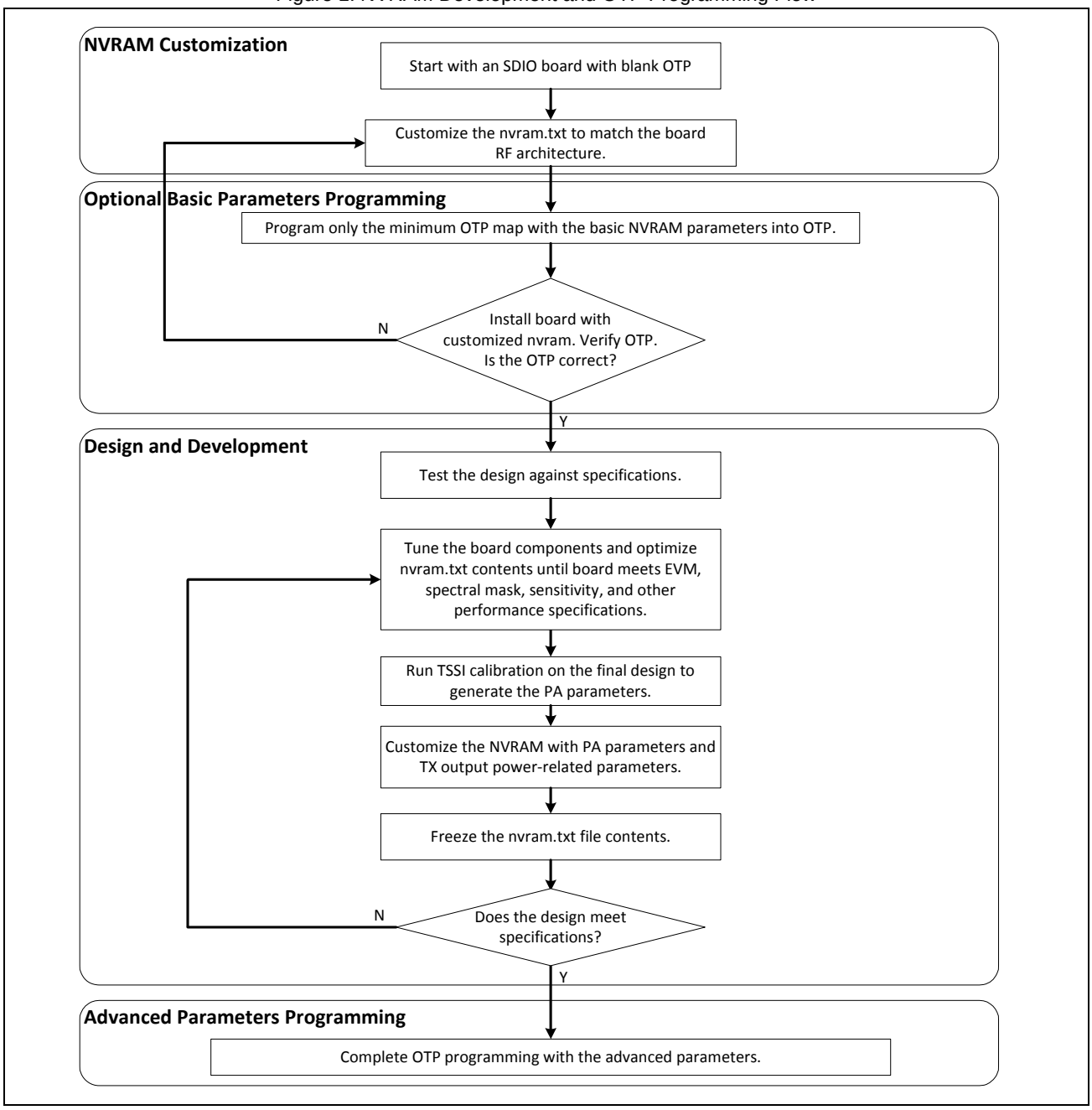
Because the OTP programming process is irreversible, Cypress recommends that board designers finalize all parameters before programming the OTP memory. Boards and modules should be tested using only the editable nvram.txt file. The nvram.txt parameters are loaded by the driver into on-chip RAM, allowing 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.

5 NVRAM Content Development and OTP Programming Flow

The nvram.txt file content development and the OTP programming flow are shown in Figure Note:. 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 on page 7.

Figure 1. NVRAM Development and OTP Programming Flow



Note: The OTP programming flow shown in Figure Note: 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.

6 Customizing the nvram.txt File

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

6.1 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, bcm943241ehlsdb.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.

Note: When a change is made in the nvram.txt file, the file must be saved, and the wireless device driver must be disabled and reenabled in the Windows Device Manager for the change to take effect. Save the file as “nvram.txt” and place it in the C:\Windows\system32\drivers\ directory. Delete or replace any previous file with the same name in this directory.

A sample nvram.txt file, with parameters that are common to Cypress’s dual-band SDIO reference design boards, is shown in Table 2 and Table 3. No specific order is required for the parameters in the nvram.txt file.

Note: Parameters listed in Table 2 are used and specified by Cypress only and should not be changed by customers

Table 2. Cypress-Specified NVRAM Parameters

NVRAM Parameter	Example Data	Description
boardtype	0x5f0	Board type. This is a critical parameter that should be copied from a similar Cypress reference board design. 0x5f0 applies only to a design with FCBGA, dual-band, external PAs, and LNAs. Different board type values apply to other board architectures.
sromrev	9	SROM revision.
txchain	0x3	Selects two transceiver chains for 2 x 2 MIMO.
rxchain	0x3	
pa0itssit	62	Defines power detector dynamic range.
pa1itssit	62	

Parameters listed in Table 3 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 3 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 Broadcom for the latest version of the nvram.txt file for the reference design before starting customization for your board design.

Table 3. NVRAM Parameters Requiring Customizing for Each Board Design

NVRAM Parameter	Example Data	Description
boardrev	0x1108	Board revision tracked by the Broadcom internal test tool (optional). Examples: 0x1108 converts to P108.
boardflags	0x200	Board configuration flags that define power topology, external components (ePA, eLNA), etc.
xtalfreq	37400	Onboard XTAL or oscillator frequency in kHz.
swctrlmap_2g	0x00400040, 0x00030001, 0x00010001, 0x40301, 0x1ff	Defines front-end RF switch or front-end module (FEM) control logic for 2.4 GHz band.

Table 3. NVRAM Parameters Requiring Customizing for Each Board Design (Cont.)

NVRAM Parameter	Example Data	Description
aa2g, aa5g	3	Number of antennas available for the 2.4 GHz and 5 GHz bands, respectively, in bit-mapped binary format: 1 = 01b for one antenna. 3 = 11b for two antennas.
ag0	0x2	Antenna gain (in dBi) defined by converting hexadecimal to 8-bit binary: lower 0–5 bits = signed 2s complement in units of dB. higher 6–7 bits = unsigned number in units of quarter dB. Examples: 0x82 = 2.5 dB (2 + 2 × 0.25) 0x7f = –0.75 dB (–1 + 1 × 0.25)
maxp2ga0, maxp2ga1	0x46	Maximum output power for each chain in the 2.4 GHz band in hexadecimal format. Units of 0.25 dB. This applies to all complementary code keying (CCK) rates as measured at antenna port. The nominal target power in dBm for CCK packets is (0.25 × maxp2ga0 in decimal) – 1.5 dB. The value can be entered in either hexadecimal or decimal format. In the example shown for 0 x 46, the maximum output power is (16 x 4 + 6)/4 = 17.5 dBm, and the nominal power is 17.5 – 1.5 = 16.0 dBm.
maxp5ga0, maxp5ga1, maxp5gla0, maxp5gla1, maxp5gha0, maxp5gha1	0x40	Maximum output power for each chain in the 5 GHz mid, low, and high bands in hexadecimal format. Units of 0.25 dB. This applies to all legacy orthogonal frequency division multiplexing (OFDM) rates as measured at antenna port. The nominal target power in dBm is (0.25 × maxp5ga0 in decimal) – 1.5 dB. The value can be entered in either hexadecimal or decimal format.
pa2gw0a0, pa2gw0a1	0xfe72, 0xfe84	Chain 0 and chain 1 PA parameters for the 2.4 GHz band based on TSSI calibration.
pa2gw1a0, pa2gw1a1	0x174d, 0x1761	
pa2gw2a0, pa2gw2a1	0xfa51, 0xfa5d	
pa5glw0a0, pa5glw0a1	5424	PA parameters for the 5 GHz low-band based on TSSI calibration.
pa5glw1a0, pa5glw1a1	–684	
pa5glw2a0, pa5glw2a1	–176	
pa5gw0a0, pa5gw0a1	5431	PA parameters for the 5 GHz mid-band based on TSSI calibration.
pa5gw1a0, pa5gw1a1	–695	
pa5gw2a0, pa5gw2a1	–193	
pa5ghw0a0, pa5ghw0a1	5563	PA parameters for the 5 GHz high-band based on TSSI calibration.
pa5ghw1a0, pa5ghw1a1	–704	
pa5ghw2a0, pa5ghw2a1	–193	
legofdmbw202gpo	0x66666666	The 2.4 GHz OFDM back-off from the maximum output power as defined by maxp2ga0. Resolution is 0.5 dB per step. Values are applied to the eight transmission rates: 54, 48, 36, 24, 18, 12, 9, and 6 Mbps. Rate 6 = LSB.
legofdmbw205gmpo, legofdmbw205glpo, legofdmbw205ghpo	0x66666666	The 5 GHz OFDM back-off from the maximum output power for the mid, low, and high bands, as defined by maxp5ga0/1. Resolution is 0.5 dB per step. Values are applied to the eight transmission rates: 54, 48, 36, 24, 18, 12, 9, and 6 Mbps. Rate 6 = LSB.
mcsbw202gpo	0x66666666	MCS0 to MCS7 per rate transmit power offset from maxp2ga0/1 for 2.4 GHz HT20. One nibble per rate. Step size is 0.5 dB. MCS0 = LSB.
mcsbw402gpo	0x66666666	MCS0 to MCS7 per rate transmit power offset from maxp2ga0/1 for 2.4 GHz HT40. One nibble per rate. Step size is 0.5 dB. MCS0 = LSB.
mcsbw205gmpo, mcsbw205glpo, mcsbw205ghpo	0x44444444	The 5GHz HT20 mid, low, and high band MCS rate backoffs from the maximum output power as defined by maxp5ga0/1. Resolution is 0.5 dB per step. Values are applied to the corresponding rates from MCS7–MCS0. MCS0 = LSB.
mcsbw405gmpo, mcsbw405glpo, mcsbw405ghpo	0x44444444	The 5GHz HT40 mid, low, and high band MCS rate backoffs from the maximum output power as defined by maxp5ga0/1. Resolution is 0.5 dB per step. Values are applied to the corresponding rates from MCS7–MCS0. MCS0 = LSB.
triso2g	2	Defines 2.4 GHz T/R switch isolation (0 means default).
triso5g	4	Defines 5 GHz T/R switch isolation (0 means default).
ccode	ALL	Country code.
macaddr[a]	00:90:4c:c5:12:38	This macaddr entry enables the firmware to use a 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. This entry provides the driver with a ‘dummy’ MAC so 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 . In production, each board or module should have a valid and unique MAC address programmed into its OTP. The macaddr is ignored by the driver when a MAC address is programmed in the OTP.

6.2 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.

6.3 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 CYW43241 has 232 bytes of space in the OTP memory available to the user. Given the limited space in the OTP, it is impossible to program the entire nvram.txt file to the OTP. The programmer must carefully 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.).

7 OTP Programming Procedure

Prior to OTP programming, an OTP binary map file must be prepared and edited with correct values. An OTP binary map completely defines the parameters that have to be programmed to the OTP memory. 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 1: “CIS Map,” on page 12 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 CYW43241 device 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 on page 9. 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 on page 10). When an OTP binary map contains only the SDIO hardware header, the binary map is called a minimum OTP map.

Table 4 shows the minimum OTP map for the CYW43241 device.

Table 4. CYW43241 Minimum OTP Map

232 Bytes OTP MAP																
Offset	0x0	0x1	0x2	0x3	0x4	0x5	0x6	0x7	0x8	0x9	0xa	0xb	0xc	0xd	0xe	0xf
0x0000	4a	f0	ff	ff	00	00	20	04	d0	02	24	43	00	00	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
0x0040	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0x0050	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0x0060	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0x0070	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0x0080	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0x0090	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0x00a0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0x00b0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0x00c0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0x00d0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0x00e0	00	00	00	00	00	00	ff	ff								

For the CYW43241 device, the OTP map is terminated with 0xff 0xff.

7.1 Programming Basic Parameters into OTP

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, refer to Appendix 1: “CIS Map,” on page 12.

Table 5 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 5. Basic NVRAM Parameters CIS Tuple Tags

NVRAM Parameter	CIS Tuple Tag	Length of Value (in Bytes)
sromrev	0x00	1
boardrev	0x02	2
boardtype	0x1b	2
macaddr	0x19	6
ccode[a]	0x0a	2

a. The value for `ccode` in the nvram.txt file is in ASCII format. It must be converted to hexadecimal format before it is entered into the OTP map (for example, “US” = “0x55 0x53”).

In the OTP binary map, each tuple is formed by the four fragments described in Table 6.

Table 6. 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.

Table 6. CIS Tuple Format

Fragment	Description
Tag	The tag identifies a parameter in the nvram.txt file. A tag usually takes 1 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 7:

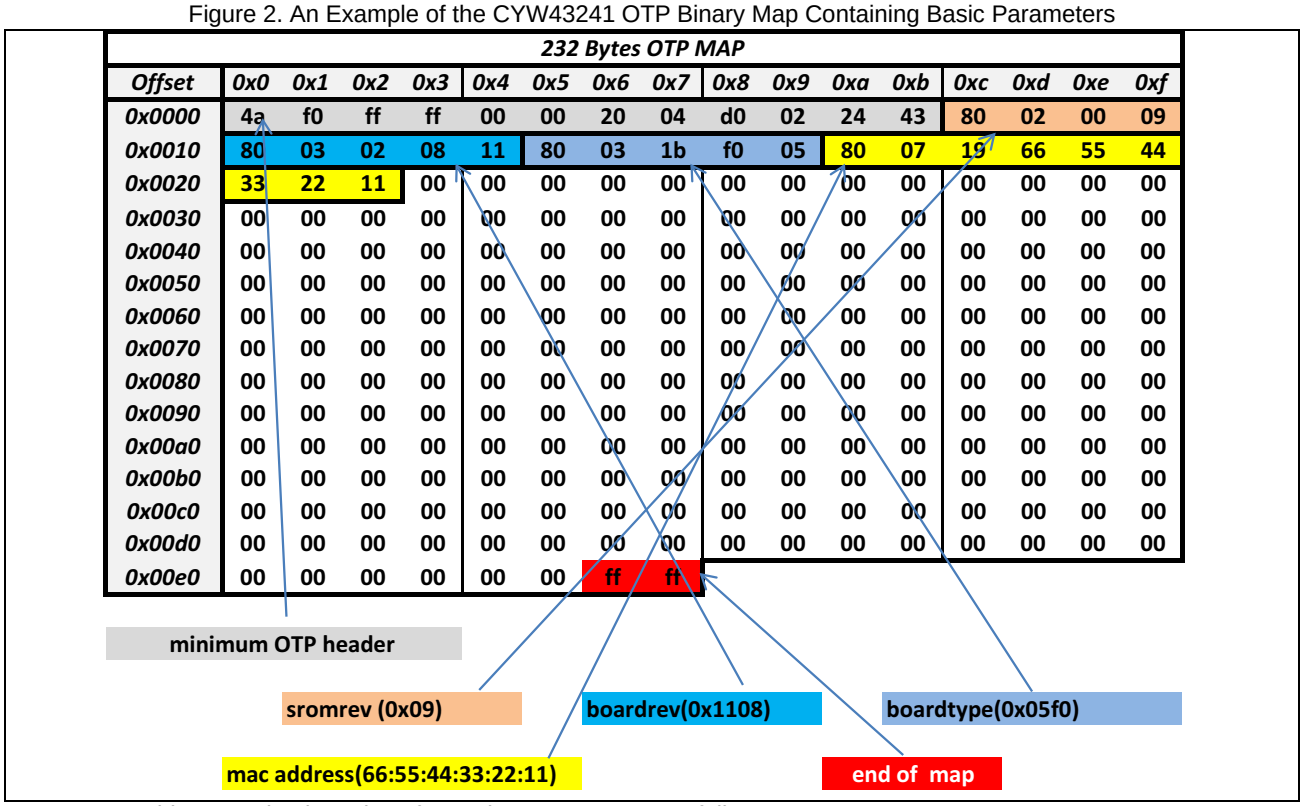
```
8       0       0       0       1
0       3       2       8       1
```

Table 7. 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.
02	Tag of 0x02 is the identifier for boardrev in the nvram.txt file.
08 11	The value of boardrev in reverse binary byte or 0x1108.

Figure shows an example of the OTP binary map for the CYW43241 device that contains some of the nvram.txt file parameters listed in Table 5 on page 8.

Figure 2. An Example of the CYW43241 OTP Binary Map Containing Basic Parameters



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

- sromrev = 0x09
- boardrev = 0x1108
- boardtype = 0x05f0
- macaddr = 66:55:44:33:22:11

Note:

- CIS tuples do not have to be in any particular order because each tuple begins with a unique identifier.
- OTP bytes can be written to only once, so only blank or zero-programmed bytes can be programmed on subsequent write cycles.

7.2 Creating and Editing the OTP Binary Map

To create and edit the OTP map:

1. 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 CYW43241 device, the OTP maximum size limit is 232 bytes including the hardware header and closing flags. The Hexadecimal OTP map template is shown below.

43241_otp_basic_parameters.bin

Address	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
00000000	4a	f0	ff	ff	00	00	20	04	d0	02	24	43	80	02	00	09
00000010	80	03	02	08	11	80	03	1b	f0	05	80	07	19	66	55	44
00000020	33	22	11	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	ff	ff								

2. 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 above has been edited to match the CYW43241 OTP binary map example in Figure on page 9.
3. 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.
4. Store this ".bin" file in the working directory that contains the wl.exe file.
   For example purposes, this file is referred to as "43241_OTP.bin" in the following instructions.

7.3 Programming Procedure Using wl Commands

To program the OTP binary map to the CYW43241 device:

1. Load the driver (refer to Appendix 2: “Driver Installation and Setup,” on page 12) to the CYW43241 device with the customized nvram.txt file. Verify the driver installation was successful by giving a few wl commands (for example, wl ver).
2. Type the following wl command to program the 43241_OTP.bin to the OTP:

```
> wl ciswrite 43241_OTP.bin
```

3. In the Windows Device Manager, disable the wireless device and then enable it.
4. Confirm OTP is programmed successfully by running the following command:

```
> wl cisdump
```

The following MAC address is used as an example: 11 22 33 44 55 66. The output should match exactly the OTP binary map created in Figure 2.

Figure 3. An Example of a wl cisdump Output

```
D:\WLAN_Drivers\PHOENIX2_REL_6_10_87>wl cisdump
Source: 2 (Internal OTP)
Maximum length: 232 bytes
Byte   0: 0x4a 0xf0 0xff 0xff 0x00 0x00 0x20 0x04
Byte   8: 0xd0 0x02 0x24 0x43 0x80 0x02 0x00 0x09
Byte  16: 0x80 0x03 0x02 0x08 0x11 0x80 0x03 0x1b
Byte  24: 0xf0 0x05 0x80 0x07 0x19 0x66 0x55 0x44
Byte  32: 0x33 0x22 0x11 0x00 0x00 0x00 0x00 0x00
Byte  40: 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
Byte  48: 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
Byte  56: 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
Byte  64: 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
Byte  72: 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
Byte  80: 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
Byte  88: 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
Byte  96: 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
Byte 104: 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
Byte 112: 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
Byte 120: 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
Byte 128: 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
Byte 136: 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
Byte 144: 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
Byte 152: 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
Byte 160: 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
Byte 168: 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
Byte 176: 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
Byte 184: 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
Byte 192: 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
Byte 200: 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
Byte 208: 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
Byte 216: 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
Byte 224: 0x00 0x00 0x00 0x00 0x00 0x00 0xff 0xff

D:\WLAN_Drivers\PHOENIX2_REL_6_10_87>_
```

If the cisdump is verified to match the OTP binary map, the OTP programming is complete. Once programmed, additional blank spaces (00) in the OTP can still be written by filling in those corresponding blank spaces in the OTP binary map. There is no restriction on how many times a device can be programmed, provided that each programming is writing to only blank, unwritten spaces. Follow the same procedure to program the additional blank spaces.

A. Appendix

A.1 CIS Map

Table 8 and Table 9 list the CIS map (standard tuple tags and Cypress subtags) for SDIO devices.

Table 8. 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 9. 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_CCODE	0x0a	3		ccode	Country code (2 bytes ASCII + 1 byte CCTL). The CCTL means indoor/outdoor, but it is never used.
				cctl	
HNBU_MACADDR	0x19	6		macaddr	MAC address override for the standard CIS LAN_NID.
HNBU_BOARDTYPE	0x1b	2		boardtype	Board type.

A.2 Driver Installation and Setup

This appendix provides steps:

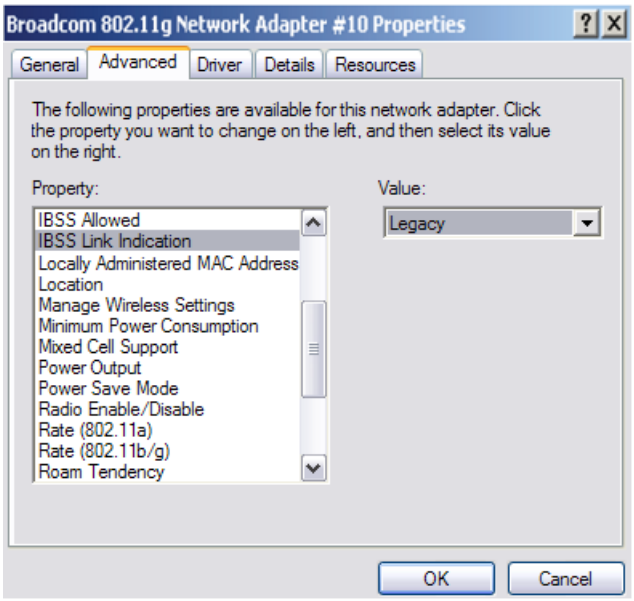
- To install a Windows XP driver for an SDIO device.
- To set the static IP address of the Cypress WLAN adapter.
- To install the Cypress WLAN tools for testing.

A.2.1 Installing an SDIO Driver for Windows XP

Note: In development environments where previous drivers have been installed, it may be necessary to uninstall a previously installed driver before proceeding with driver installation. If so, refer to Appendix 3: “Driver Removal,” on page 15.

To install an SDIO device driver:

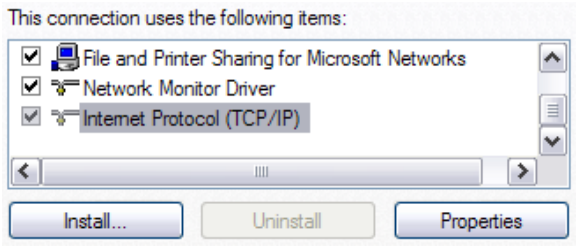
1. Rename the NVRAM file (named after the board it supports) to “nvram.txt” and copy it to the C:\Windows\system32\drivers\ directory.
2. Turn off the power to the Windows XP-based PC test system.
3. Install the Cypress adapter in the PC.
4. Power-on the PC and allow Windows XP to start.
5. In Windows Control Panel, double-click **Administrative Tools**.
6. Within the **Administrative Tools**, double-click **Computer Management**, and then click **Device Manager**.
7. In the right pane of **Computer Management**, right-click **Network adapters**, and then click **Scan for hardware changes**.
8. Follow the Windows on-screen instructions to install the SDIO device driver.
9. Double-click on the newly installed network adapter to view the adapter properties.
10. On the **Advanced** tab of the **Network Adapter Properties**, set the **IBSS Link Indication** property value to **Legacy**, scroll down and set the **IBSS 54g™ Mode** property value to **54g-Auto**, and then click **OK**.



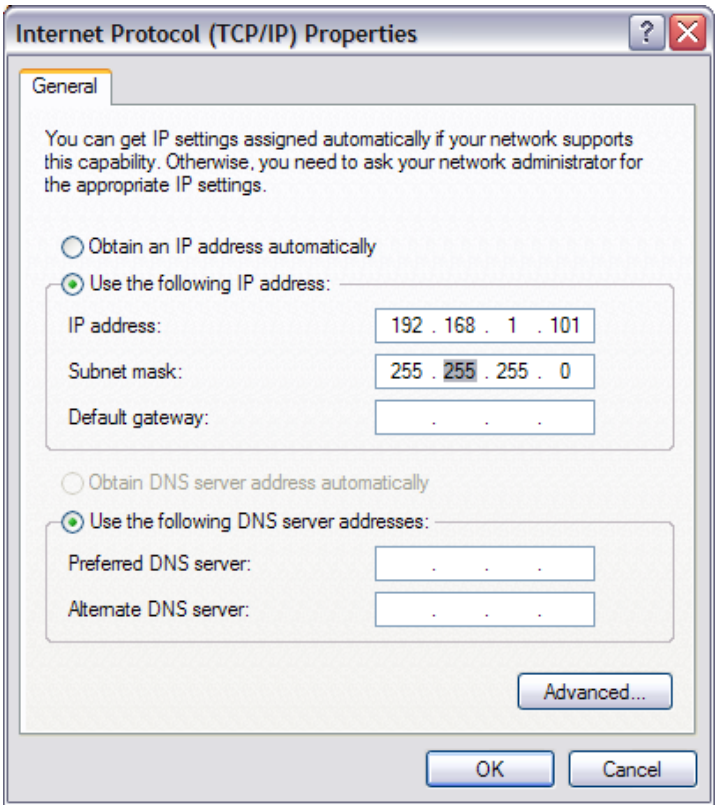
A. 2.2 Setting a Static IP Address for the WLAN Adapter

To set the static IP address of the Cypress IEEE 802.11g SDIO WLAN adapter:

1. In Windows Control Panel, double-click **Network Connections**.
2. Right-click **Wireless Network Connection**, and then select **Properties**.
3. In **Wireless Network Connection Properties**, click **Internet Protocol (TCI/IP),** and then click the **Properties** button.



4. Select the **Use the following IP address** option.
5. Set the IP address to **192.168.1.101** and the Subnet mask to **255.255.255.0**, then click **OK**.



A. 2.3 Installing the WLAN Test Tools

To install the WLAN test tools for enabling the driver-test commands, follow these instructions:

1. Copy the wl.exe file to the C:\Windows\system32\ directory.
2. Copy the brcm_wlu.dll file to the C:\Windows\system32\ directory.

A.3 Driver Removal

A. 3.1 Removing a Driver

In development environments where previous drivers have been installed, it may be necessary to remove a previously installed driver before proceeding with the new driver installation. Follow these instructions to remove a driver:

1. In Windows Control Panel, double-click **Administrative Tools**.
2. Within the **Administrative Tools**, double-click **Computer Management**, and then click **Device Manager**.
3. In the right pane of **Computer Management** under **Network adapters**, right-click **Broadcom 802.11g Network Adapter #10**, and then click **Uninstall**.
4. Click **OK** to **Confirm Device Removal**.

5. Delete the following files (refer to Using the nvram.txt File Template on page 5):

Filename	Location
bcmsddhd.sys	C:\Windows\system32\drivers\
nvram.txt	C:\Windows\system32\drivers\
oem#.inf	C:\Windows\inf\
oem#.pnf	C:\Windows\inf\
wl.exe	C:\Windows\system32\
brcm_wlu.dll	C:\Windows\system32\

The following files are typical SDIO driver files released with the Windows XP driver or design package:

nvram.txt
bcmsddhd.inf
bcmsddhd.sys
brcm_wlu.dll
wl.exe

A. 3.2 Deleting the oem#.inf and oem#.pnf Files

In some cases, multiple instances of Cypress network adapters might be installed. To locate the correct oem#.inf and oem#.pnf files for deletion, follow these steps:

1. Start the Windows **Search** option.
2. In the **Search Companion** pane of **Search Results**, type **C:\Windows\inf\oem*.inf**. In the **All or part of the file name** box, type **Broadcom Corporation**. In the **A word or phrase in the file** box, select **Local Hard Drives (C:)** from the **Look in** list, and then click **Search**.
3. If more than one INF file appears in the **Search Results**, open each in a text editor.

   Delete the Broadcom oem#.inf file and the associated Broadcom oem#.pnf file that looks similar to the oem#.inf file shown here.

```
;; bcmsddhd.inf
;;
;; Copyright 1998-2005, Broadcom Corporation.
;; All Rights Reserved.
;;
;; This is UNPUBLISHED PROPRIETARY SOURCE CODE of Broadcom Corporation;
;; the contents of this file may not be disclosed to third parties, copied or
;; duplicated in any form, in whole or in part, without the prior written
;; permission of Broadcom Corporation.
;;

[version]
    Signature   = "$Windows NT$"        ; Combined Win9x/Win2k inf
    Class=Net
    ClassGUID   = {4d36e972-e325-11ce-bfc1-08002be10318}
    Provider    = %V_BCM%
    Compatible  = 1
DriverVer=06/10/2009, 4.218.84.1
    CatalogFile=BCM43XX.CAT
    CatalogFile.NTamd64=BCM43XX64.CAT

[Manufacturer]
    %V_BCM% = BROADCOM, NTamd64
```

Document History Page

Document Title: AN214915 - OTP Programming and NVRAM Development in SDIO Mode - CYW43241 Document Number: 002-14915				
Rev.	**ECN No.**	**Orig. of Change**	**Submission Date**	**Description of Change**
**	–	–	05/14/2012	43241-AN100-R Initial release
*A	5456162	UTSV	09/30/2016	Updated in Cypress template Added Cypress part numbering scheme
*B	5834576	BENV	07/27/2017	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

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