Please note that Cypress is an Infineon Technologies Company.
The document following this cover page is marked as “Cypress” document as this is the company that originally developed the product. Please note that Infineon will continue to offer the product to new and existing customers as part of the Infineon product portfolio.

Continuity of document content
The fact that Infineon offers the following product as part of the Infineon product portfolio does not lead to any changes to this document. Future revisions will occur when appropriate, and any changes will be set out on the document history page.

Continuity of ordering part numbers
Infineon continues to support existing part numbers. Please continue to use the ordering part numbers listed in the datasheet for ordering.

CYPRESS
EMBEDDED IN TOMORROW™

AN230546

OTP Memory Programming and NVRAM Development - CYW43455

Author: Roy Shyu

This application note describes the method for creating an *nvram.txt* file, which is then used to test a new board design, optimize NVRAM values, and program the one-time programmable (OTP) nonvolatile memory in the CYW43455 device using the SDIO host interface for WLAN.

Contents

1 Introduction

The Cypress CYW43455 is a single-chip IEEE 802.11a/b/g/n/ac + BT 4.2 device for embedded and IoT applications. OTP nonvolatile memory is included in the WLAN section of the device to store board-specific information such as SDIO header, product ID, manufacturer ID, and MAC address. Excluding the internal header information, up to 498 bytes of user accessible OTP memory is available on CYW43455 for WLAN and BT information. The application note provides OTP programming information for only SDIO host interface.

The OTP memory content, along with an editable NVRAM file (*nvram.txt* file), provides all configuration information used by the WLAN device driver to initialize and configure CYW43455.

1.1 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/).

2 OTP Memory Programming Considerations

In embedded designs, the host and device are permanently connected, which is typically done using a hardwired SDIO interface. The only entry which is mandatory to be programmed into the OTP memory is the SDIO header. This is because there are certain SDIO function settings which are read before the firmware and NVRAM are downloaded. To properly set these settings, the SDIO header must be programmed into their OTP memory.

Other than the SDIO header, all other NVRAM parameters can be stored in the host's nonvolatile memory rather than in the OTP memory. For non-embedded devices that may be installed on different hosts, the OTP memory can be programmed to protect the unique MAC address and prevent end-users from altering the power control parameters, such as maximum output power.

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

The OTP programming process is irreversible, so it is recommended that you finalize all NVRAM parameters before programming any parameter into the OTP memory. Test the boards and modules using only the editable *nvram.txt* file.

The driver loads the parameters stored in the *nvram.txt* file onto an on-chip RAM, allowing the chip to be tested even if the OTP memory has only been programmed with the SDIO header. This method allows you to tune the RF components and alter critical parameters using different versions of the *nvram.txt file* while testing boards. Optionally, a few basic parameters, such as the board type and MAC address, can be programmed into the OTP memory prior to testing the board during development.

Note: If a parameter is present in both the on-chip OTP memory and the *nvram.txt* file, the value in the OTP memory takes priority over the value in the *nvram.txt* file.

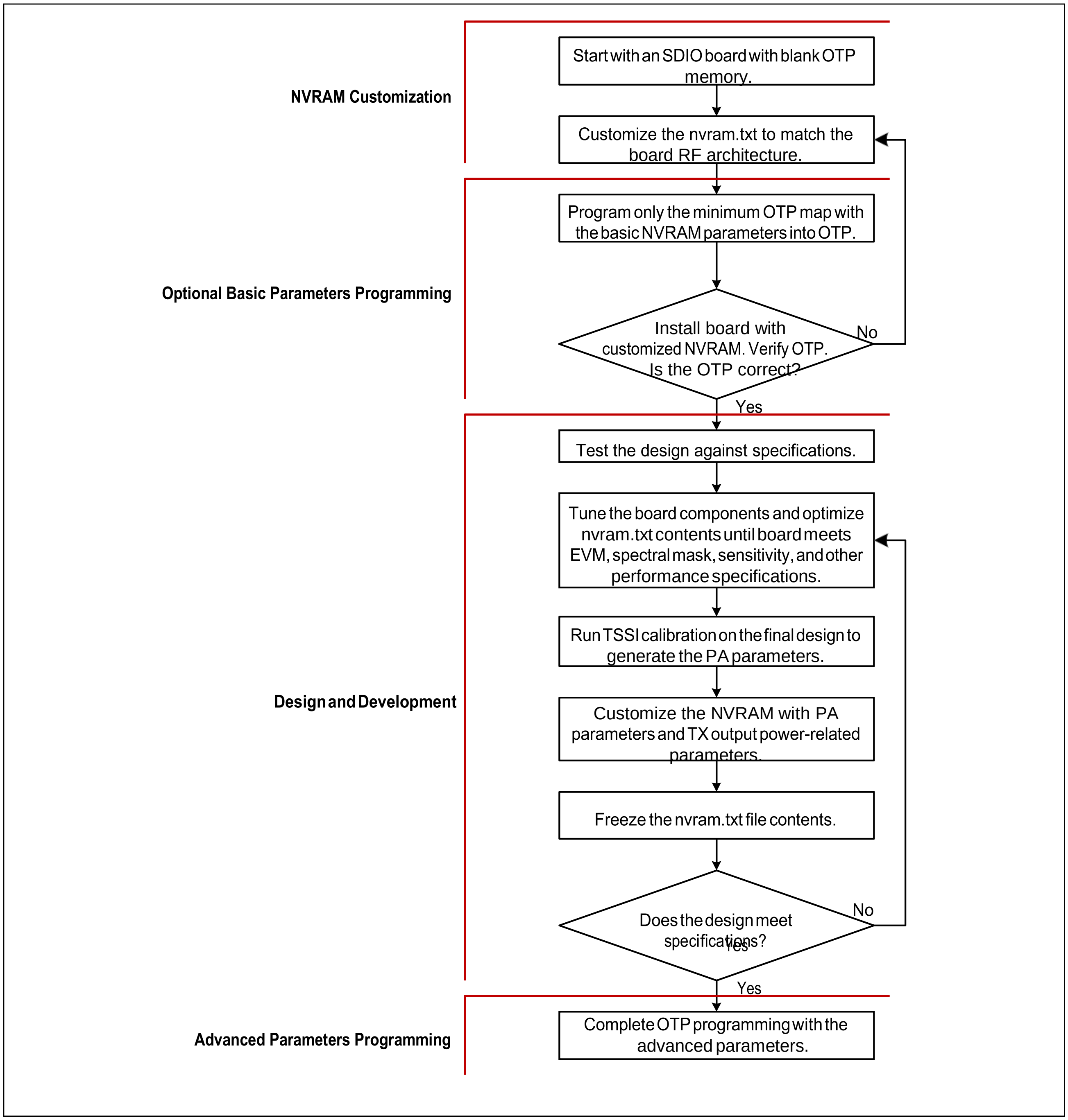
Note: The programming process of an OTP memory is irreversible. Cypress strongly recommends conducting development on boards using the parameters provided in the editable *nvram.txt* file. Do not program the OTP memory until the contents of the *nvram.txt* file have been verified and the file has been finalized for production use. The one exception to this is the SDIO header, which must be programmed into OTP memory for full SDIO functionality.

3 NVRAM Content Development and Memory Programming Flow

Figure 1 shows the *nvram.txt* file content development and the OTP memory programming flow. Parameters in the *nvram.txt* file can be divided into basic (see Table 3) and advanced (see Table 2) categories.

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

Figure 1. NVRAM Development and Programming Flow of OTP Memory



4 Customizing nvram.txt File

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

4.1 Using nvram.txt File Template

For each reference board design, Cypress provides an *nvram.txt* file for the specific board design. Typically, the file is named in accordance with the board it supports (for example, *cyw943455wlsdio.txt*).

The *nvram.txt* file might be included with the reference board design package or the driver release. You can download the latest version of the file from the Cypress Developer Community.

Table 1 and Table 2 provides a list of parameters in a typical *nvram.txt* file that are common to dual-band 802.11ac SDIO reference design boards.

Parameters in the *nvram.txt* file do not need to be entered in any specific order.

Note: The parameters listed in Table 1 are used and specified by Cypress and should only be changed by Cypress. It is important that Cypress reviews a customer's design early in the development process. Some of the parameters in Table 1 may need to be changed by Cypress to accommodate differences in the RF front end between the customer's design and the Cypress reference design from which it was derived.

Table 1. Cypress-specific NVRAM Parameters

NVRAM Parameter	Example Data	Description
sromrev	11	SROM revision for 802.11ac chips
boardtype	0x06e4	This is a critical parameter that should be copied from a similar Cypress reference board design.
tssipos2g	1	This represents if TSSI has positive slope for 2.4 GHz. For CYW43455, set the value to 1.
tssipos5g	1	This represents if TSSI has positive slope for 5 GHz. For CYW43455, set the value to 1.
rxchain	1	This specifies the number of rx paths (bit mask). For CYW43455, set the value to 1.
txchain	1	This specifies the number of tx paths (bit mask). For CYW43455, set the value to 1.
venid	0x14e4	PCI Vendor ID
devid	0x43ab	Chip ID, CYW43455
manfid	0x2d0	Manufacturer ID
nocrc	1	Check for CRC errors when loading firmware
boardflags boardflags2 boardflags3	0x00080201 0x40800000 0x48200100	Board configuration flag that defines the power topology, external components (iPA/iLNA or ePA/eLNA), and so on
tworangetssi2g tworangetssi5g	1 1	2.4 GHz and 5 GHz TSSI dual power range flag, which iPA chips support
xtalfreq	37400	Describes the reference oscillator frequency in kHz. ‘37400’ stands for 37.4 MHz
extpagain2g	2	Supports 2.4 GHz external PA. Use 2 for iPA boards, and use 0 for ePA boards.
extpagain5g	2	Supports 5 GHz external PA. Use 2 for iPA boards, and use 0 for ePA boards.
aa2g, aa5g	1	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
phycal_tempdelta	15	This parameter is for Cypress internal use only **Note**: Do not modify.
AvVmid_c0	1, 165, 2, 100, 2, 100, 2, 100, 2, 100	This parameter is for Cypress internal use only. **Note**: Do not modify.

NVRAM Parameter	Example Data	Description
swctrlmap_2g,	0x00000000, 0x00000000, 0x00000000, 0x00010000, 0x3ff	Describes how to control the external 2.4 GHz FEM (front-end module) or TR-SW.
swctrlmap_5g,	0x00100010, 0x00200020, 0x00200020, 0x00010000, 0x3fe	Describes how to control the external 5 GHz FEM (front-end module) or TR-SW.
swctrlmapext_2g/5g	0x00000000, 0x00000000, 0x00000000, 0x00000000,0x0	Describes how to control the external 2.4 GHz and 5 GHz FEM (front-end module) or TR-SW.

The design variables listed in Table 2 must be reviewed prior to beginning board or module testing. During the development phase, start with the default power amplifier (PA) parameters listed in the provided *nvram.txt* file. The PA parameters are eventually optimized using Cypress transmit signal strength indicator (TSSI) calibration tools.

The parameters in Table 2 typically require tuning for each board-specific or module design. This is not an exhaustive list. Cypress might add additional parameters at any time to control the 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 any board customization efforts.

Note: To avoid unexpected operating results, contact a technical support representative before attempting to add NVRAM parameters.

Table 2. NVRAM Parameters Requiring Customization

NVRAM Parameter	Example Data	Description
boardrev	0x1100	Board revision used by the WLAN driver. Examples: 0x1100 converts to P100
ccode	0	Country code for regulatory. Specifies which regulatory tables are to be loaded. **Note**: Together, the ccode and regrev parameters set the power and other limitations necessary to meet the country-specific regulatory requirements.
regrev	0	The regulatory revision code for regulatory use, and specifies which regulatory tables are to be loaded. **Note**: Together, the ccode and regrev parameters set the power and other limitations necessary to meet the country-specific regulatory requirements.
agbg0, aga0	0x7f	Antenna gain (in dBi) defined by converting hexadecimal to 8-bit binary: (agba0: 2.4 GHz antenna gain, aga0: 5 GHz antenna gain) • 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)
pa2ga0	-148, 5828, -679	PA parameters for the 2.4 GHz band based on TSSI calibration.

NVRAM Parameter	Example Data	Description
pa5ga0	83, 6045, -553, 57,5940, -566, 12,5919, -605, -17,5899, -640	PA parameters for the 5 GHz band based on TSSI calibration (Low / Mid / High / X1). Sub-band frequency range. Channel Range: • Low 5180 to 5240MHz 36-48 • Mid 5260 to 5320MHz 52-64 • High 5500 to 5700MHz 100-140 • X1 5745 to 5825MHz 149-165 **Note:** Each of four subbands has three parameters
pdoffset2g40ma0	0x10	2GHz, 40 MHz BW power detect (PD) offset (1/4 dB steps) in 2s complement format.
pdoffset40ma0	0x8888	5 GHz, 40 MHz BW power detect (PD) offset (1/4 dB steps) in 2s complement format. 4 bits for each sub-band. The most significant nibble is the X1 sub-band offset.
pdoffset80ma0	0x8888	5 GHz, 80 MHz BW PD offset (in 1/4 dB steps) in 2s complement format 4 bits for each sub-band. The most significant nibble is the X1 sub-band offset.
maxp2ga0	0x50	Maximum output power for 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 formats. 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.
cckbw202gpo	0x0000	CCK unsigned power offsets (in 1/2 dB steps) for the 20 MHz rates (11, 5.5, 2, 1 Mbps). The most significant nibble is the 11 Mbps offset.
cckbw20ul2gpo	0x0000	CCK unsigned power offsets in 1/2 dB steps for 20 U/L rates (11, 5.5, 2, 1 Mbps). The most significant nibble is the 11 Mbps offset
dot11agofdmhrbw202gpo	0x4442	OFDM unsigned power offsets in 1/2 dB steps for 54, 48, 36, and 24 Mbps. The most significant nibble is the 54 Mbps rate offset
ofdmlrbw202gpo	0x0022	OFDM 2.4 GHz, unsigned power offsets in 1/2 dB steps: • MCS1 and MCS2: 11n and 11ac 40 MHz BW • (most significant nibble) • MCS1 and MCS2: 11n and 11ac 20 MHz BW • 12 and 18 Mbps: 11g • 6 and 9 Mbps: 11g
mcsbw202gpo	0x98444422	11n/ac MCS0/1/2, 3-7, C8, C9 2.4 MHz unsigned power offsets in 1/2 dB steps – C9/C8/M7/M6/M5/M4/M3/M0-2. (If separate control of MCS1 and MCS2 is required, then use ofdmlrbw202gpo).
maxp5ga0	0x52, (low) 0x52, (mid) 0x52, (high) 0x52 (X1)	Maximum output power for the 5 GHz band 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.
mcslr5glpo	0x0000	5 GHz band low sub-band 12/18 Mbps and MCS1/2, unsigned power offsets in 1/2 dB steps: • (0) 20 MHz (least significant nibble) • (1) 40 MHz • (2) 80 MHz • (3) 160 MHz

NVRAM Parameter	Example Data	Description
mcsbw205glpo	0xb9555000	5 GHz, low sub-band, 11n/ac, 20 MHz, unsigned power offsets in 1/2 dB steps. The most significant nibble is the power offset for MCS9 and the least significant nibble is for MCS0–2. • C9/C8/M7/M6/M5/M4/M3/M0-2
mcsbw405glpo	0xb9555000	5 GHz, low sub-band, 11n/ac, 40 MHz, unsigned power offsets in 1/2 dB steps. The most significant nibble is the power offset for MCS9 and the least significant nibble is for MCS0–2. • C9/C8/M7/M6/M5/M4/M3/M0-2
mcsbw805glpo	0xb9555000	5 GHz, low sub-band, 11n/ac, 80 MHz, unsigned power offsets in 1/2 dB steps. The most significant nibble is the power offset for MCS9 and the least significant nibble is for MCS0–2. • C9/C8/M7/M6/M5/M4/M3/M0-2
mcslr5gmpo	0x0000	5 GHz, mid sub-band, 11ag/11n/11ac, QPSK, unsigned power offsets in 1/2 dB steps with respect to BPSK: MCS1/2 with respect to MCS0/1/2, and 12/18 Mbps with respect to 6/9 Mbps. • (0) 20 MHz (least significant nibble) • (1) 40 MHz • (2) 80 MHz • (3) 160 MHz
mcsbw205gmpo	0xb9555000	5 GHz, mid sub-band, 11n/ac, 20 MHz, unsigned power offsets in 1/2 dB steps. The most significant nibble is the power offset for MCS9 and he least significant nibble is for MCS0–2. • C9/C8/M7/M6/M5/M4/M3/M0-2
mcsbw405gmpo	0xb9555000	5 GHz, mid sub-band, 11n/ac, 40 MHz, unsigned power offsets in 1/2 dB steps. The most significant nibble is the power offset for MCS9 and the least significant nibble is for MCS0–2. • C9/C8/M7/M6/M5/M4/M3/M0-2
mcsbw805gmpo	0xb9555000	5 GHz, mid sub-band, 11n/ac, 80 MHz, unsigned power offsets in 1/2 dB steps. The most significant nibble is the power offset for MCS9 and the least significant nibble is for MCS0–2. • C9/C8/M7/M6/M5/M4/M3/M0-2
mcslr5ghpo	0x0000	5 GHz, high/X1 band 11ag/11n/11ac, QPSK, unsigned power offsets in 1/2 dB steps with respect to BPSK: MCS1/2 with respect to MCS0/1/2, and 12/18 Mbps with respect to 6/9 Mbps. • (0) 20 MHz (least significant nibble) • (1) 40 MHz • (2) 80 MHz • (3) 160 MHz
mcsbw205ghpo	0xb9555000	5 GHz, high/X1 sub-band, 11n/ac, 20 MHz, unsigned power offsets in 1/2 dB steps. The most significant nibble is the power offset for MCS9 and the least significant nibble is for MCS0–2. • – C9/C8/M7/M6/M5/M4/M3/M0-2
mcsbw405ghpo	0xb9555000	5 GHz, high/X1 sub-band, 11n/ac, 40 MHz, unsigned power offsets in 1/2 dB steps. The most significant nibble is the power offset for MCS9 and the least significant nibble is for MCS0–2. • C9/C8/M7/M6/M5/M4/M3/M0-2

NVRAM Parameter	Example Data	Description
mcsbw805ghpo	0xb9555000	5 GHz, high/X1 sub-band, 11n/ac, 80 MHz, unsigned power offsets in 1/2 dB steps. The most significant nibble is the power offset for MCS9 and the least significant nibble is for MCS0–2. • C9/C8/M7/M6/M5/M4/M3/M0-2
sb20in40hrpo	0	20in40 OFDM signed power offsets (in 1/2 dB steps) with respect to 20in20 for 64 QAM and above. LSB nibble to MSB nibble: • (0) 2.4 GHz band • (1) 5 GHz low sub-band • (2) 5 GHz mid sub-band • (3) 5 GHz high/X1 sub-band
sb20in80and160hr5glpo	0	20in40 OFDM signed power offsets (in 1/2 dB steps) with respect to 20in20 for 64 QAM and above. LSB nibble to MSB nibble: • (0) 2.4 GHz band • (1) 5 GHz low sub-band • (2) 5 GHz mid sub-band • (3) 5 GHz high/X1 sub-band
sb20in80and160hr5glpo	0	5 GHz low sub-band 20in80, 20in160 OFDM signed power offsets (in 1/2 dB steps) for 64 QAM and above. LSB nibble to MSB nibble: • (0) 20in80 with respect to 20in20 • (1) 20in160 with respect to 20in20 • (2) 20in80 - 20LL/UU with respect to 20LU/UL • (3) 20in160 - 20LLL/UUU with respect to other 20in160 sub-bands
sb40and80hr5glpo	0	5 GHz low sub-band 40in80, 40in160 OFDM signed power offsets (in 1/2 dB steps) for 64 QAM and above. LSB nibble to MSB nibble: • (0) 40in80 with respect to 40in40 • (1) 40in160 with respect to 40in40 • (2) 80in160 with respect to 80in80 • (3) 40in160 -40LL/UU with respect to 40LU/UL
sb20in80and160hr5gmpo	0	5 GHz mid sub-band 20in80, 20in160 OFDM signed power offsets (in 1/2 dB steps) for 64 QAM and above. LSB nibble to MSB nibble: • (0) 20in80 with respect to 20in20 • (1) 20in160 with respect to 20in20 • (2) 20in80 - 20LL/UU with respect to 20LU/UL • (3) 20in160 - 20LLL/UUU with respect to other 20in160 sub-bands
sb40and80hr5gmpo	0	5 GHz mid sub-band 40in80, 40in160 OFDM signed power offsets (in 1/2 dB steps) for 64 QAM and above. LSB nibble to MSB nibble: • (0) 40in80 with respect to 40in40 • (1) 40in160 with respect to 40in40 • (2) 80in160 with respect to 80in80 • (3) 40in160 -40LL/UU with respect to 40LU/UL
sb20in80and160hr5ghpo	0	5 GHz high/X1 sub-band 20in80, 20in160 OFDM signed power offsets (in 1/2 dB steps) for 64 QAM and above. LSB nibble to MSB nibble: • (0) 20in80 with respect to 20in20 • (1) 20in160 with respect to 20in20 • (2) 20in80 - 20LL/UU with respect to 20LU/UL • (3) 20in160 - 20LLL/UUU with respect to other 20in160 sub-bands

NVRAM Parameter	Example Data	Description
sb40and80hr5ghpo	0	5 GHz high/X1 sub-band 40in80, 40in160 OFDM signed power offsets (in 1/2 dB steps) for 64 QAM and above. LSB nibble to MSB nibble: • (0) 40in80 with respect to 40in40 • (1) 40in160 with respect to 40in40 • (2) 80in160 with respect to 80in80 • (3) 40in160 -40LL/UU with respect to 40LU/UL
sb20in40lrpo	0	20in40 OFDM signed power offsets (in 1/2 dB steps) with respect to 20in20 for 16 QAM and below. LSB nibble to MSB nibble: • (0) 2.4 GHz band • (1) 5 GHz low sub-band • (2) 5 GHz mid sub-band • (3) 5 GHz high/X1 sub-band
sb20in80and160lr5glpo	0	5 GHz low sub-band 20in80, 20in160 OFDM signed power offsets (in 1/2 dB steps) for 16 QAM and below. LSB nibble to MSB nibble: • (0) 20in80 with respect to 20in20 • (1) 20in160 with respect to 20in20 • (2) 20in80 - 20LL/UU with respect to 20LU/UL • (3) 20in160 - 20LLL/UUU with respect to other 20in160 sub-bands
sb40and80lr5glpo	0	5 GHz mid sub-band 20in80, 20in160 OFDM signed power offsets (in 1/2 dB steps) for 64 QAM and above. LSB nibble to MSB nibble: • (0) 20in80 with respect to 20in20 • (1) 20in160 with respect to 20in20 • (2) 20in80 - 20LL/UU with respect to 20LU/UL • (3) 20in160 - 20LLL/UUU with respect to other 20in160 sub-bands
sb40and80hr5gmpo	0	5 GHz mid sub-band 40in80, 40in160 OFDM signed power offsets (in 1/2 dB steps) for 64 QAM and above. LSB nibble to MSB nibble: • (0) 40in80 with respect to 40in40 • (1) 40in160 with respect to 40in40 • (2) 80in160 with respect to 80in80 • (3) 40in160 -40LL/UU with respect to 40LU/UL
sb20in80and160hr5ghpo	0	5 GHz high/X1 sub-band 20in80, 20in160 OFDM signed power offsets (in 1/2 dB steps) for 64 QAM and above. LSB nibble to MSB nibble: • (0) 20in80 with respect to 20in20 • (1) 20in160 with respect to 20in20 • (2) 20in80 - 20LL/UU with respect to 20LU/UL • (3) 20in160 - 20LLL/UUU with respect to other 20in160 sub-bands
sb40and80hr5ghpo	0	5 GHz high/X1 sub-band 40in80, 40in160 OFDM signed power offsets (in 1/2 dB steps) for 64 QAM and above. LSB nibble to MSB nibble: • (0) 40in80 with respect to 40in40 • (1) 40in160 with respect to 40in40 • (2) 80in160 with respect to 80in80 • (3) 40in160 -40LL/UU with respect to 40LU/UL
sb20in40lrpo	0	20in40 OFDM signed power offsets (in 1/2 dB steps) with respect to 20in20 for 16 QAM and below. LSB nibble to MSB nibble: • (0) 2.4 GHz band • (1) 5 GHz low sub-band • (2) 5 GHz mid sub-band • (3) 5 GHz high/X1 sub-band

NVRAM Parameter	Example Data	Description
sb20in80and160lr5glpo	0	5 GHz low sub-band 20in80, 20in160 OFDM signed power offsets (in 1/2 dB steps) for 16 QAM and below. LSB nibble to MSB nibble: • (0) 20in80 with respect to 20in20 • (1) 20in160 with respect to 20in20 • (2) 20in80 - 20LL/UU with respect to 20LU/UL • (3) 20in160 - 20LLL/UUU with respect to other 20in160 sub-bands
sb40and80lr5glpo	0	5 GHz low sub-band 40in80, 40in160 OFDM signed power offsets (in 1/2 dB steps) for 16 QAM and below. LSB nibble to MSB nibble: • (0) 40in80 with respect to 40in40 • (1) 40in160 with respect to 40in40 • (2) 80in160 with respect to 80in80 • (3) 40in160 -40LL/UU with respect to 40LU/UL
sb20in80and160lr5gmpo	0	5 GHz mid sub-band 20in80, 20in160 OFDM signed power offsets (in 1/2 dB steps) for 16 QAM and below. LSB nibble to MSB nibble: • (0) 20in80 with respect to 20in20 • (1) 20in160 with respect to 20in20 • (2) 20in80 - 20LL/UU with respect to 20LU/UL • (3) 20in160 - 20LLL/UUU with respect to other 20in160 sub-bands
sb40and80lr5gmpo	0	5 GHz mid sub-band 40in80, 40in160 OFDM signed power offsets (in 1/2 dB steps) for 16 QAM and below. LSB nibble to MSB nibble: • (0) 40in80 with respect to 40in40 • (1) 40in160 with respect to 40in40 • (2) 80in160 with respect to 80in80 • (3) 40in160 -40LL/UU with respect to 40LU/UL
sb20in80and160lr5ghpo	0	5 GHz high/X1 sub-band 20in80, 20in160 OFDM signed power offsets (in 1/2 dB steps) for 16 QAM and below. LSB nibble to MSB nibble: • (0) 20in80 with respect to 20in20 • (1) 20in160 with respect to 20in20 • (2) 20in80 - 20LL/UU with respect to 20LU/UL • (3) 20in160 - 20LLL/UUU with respect to other 20in160 sub-bands
sb40and80lr5ghpo	0	5 GHz high/X1 sub-band 40in80, 40in160 OFDM signed power offsets (in 1/2 dB steps) for 16 QAM and below. LSB nibble to MSB nibble: • (0) 40in80 with respect to 40in40 • (1) 40in160 with respect to 40in40 • (2) 80in160 with respect to 80in80 • (3) 40in160 - 40LL/UU with respect to 40LU/UL
dot11agduphrpo	0	11a/g duplicate mode signed power offsets (in 1/2 dB steps) for 64 QAM. Common power offset for Dup40, Dup40in80, and Dup40in160 with respect to 40in40 11n/11ac, Quad80 and Quad80in160 with respect to 11ac 80in80, Oct160 with respect to 11ac 160in160. LSB to MSB nibble: • (0) 2.4 GHz band • (1) 5 GHz low sub-band • (2) 5 GHz mid sub-band • (3) 5 GHz high/X1 sub-band
dot11agduplrpo	0	Bits 11a/g duplicate mode signed power offsets (in 1/2 dB steps) for 16 QAM and below. Common power offset for Dup40, Dup40in80, and Dup40in160 with respect to 40in40 11n/11ac, Quad80 and Quad80in160 with respect to 11ac 80in80, Oct160 with respect to 11ac 160in160. LSB to MSB nibble: • (0) 2.4 GHz band • (1) 5 GHz low sub-band • (2) 5 GHz mid sub-band • (3) 5 GHz high/X1 sub-band
mux_enab	0x11	Specifies GPIO pin for out-of-band (OOB) interrupts.

NVRAM Parameter	Example Data	Description
btc_mode	1	Specifies BT-COEX mode. Needed only for sLNA configuration.
ltecxmux	0	Specifies LTE Coex settings.
cckdigfilttype	2	Specifies filter type for 11b mode.

4.2 Editing nvram.txt File

Edit the *nvram.txt* file using a properly formatted text editor such as Notepad++ or WordPad++ to preserve the original format of the file. Using a non-formatted text editor such as Notepad could corrupt the format of the NVRAM map, causing the driver to incorrectly read the *nvram.txt* file.

4.3 Finalizing nvram.txt File

After the final PA parameters have been generated, edit the *nvram.txt* file to update the PA parameters derived using the Cypress TSSI tool, and then adjust the Tx output power-related parameters in the file. Using the updated *nvram.txt* file, run output power tests to verify that the parameters are providing the correct output power. Also, verify that RF performance (EVM, spectral mask, and PER) meets design specifications.

Cypress recommends running a regulatory pre-scan 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 finalized, the required parameters can be selected and the OTP memory programmed for production.

Note: The CYW43455 has 498 bytes of space in the OTP memory available for user data and this is for Wi-Fi only (CIS dump). Given the limited space in the OTP memory, it is impossible to program the entire *nvram.txt* file to the OTP memory. Make sure that you select only the necessary parameters that go into the OTP memory.

Parameters that typically go into the OTP memory 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, and country code. Alternately, with many embedded systems, various NVRAM variables are stored in the system's nonvolatile memory as opposed to OTP memory.

5 Programming OTP Memory

Prior to programming the OTP memory, an OTP binary map file must be prepared and populated with the correct values.

The OTP binary map file completely defines the parameters that have to be programmed into the OTP memory. OTP binary map file may only contain the SDIO hardware header or includes some NVRAM parameters.

One item that is required in the OTP memory is the SDIO hardware header. The SDIO header string must be present at the beginning of OTP binary map and must precede all NVRAM variables. After SDIO header string, the SDIO OTP data format for NVRAM parameters is based on the CIS as defined by the PCMCIA/SD Card Association. The CIS tuple data contains the type, length, the tag of the tuple, and one or more bytes of NVRAM value.

When using the SDIO interface with the CYW43455, there are certain SDIO function settings which are read before the firmware and NVRAM are downloaded. To properly set these settings, the SDIO header must be programmed into the OTP, nonvolatile memory.

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

Contact Cypress Support team to request the binary map file.

5.1 Programming Basic Parameters into OTP Memory

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

Note: The SDIO hardware header does not use tuple format, 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. For this reason, it is generally acceptable to program these basic parameters into the OTP memory early in the development, before the design is finalized.

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

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

For example, the tuple is defined by the fragments that follow(by hexadecimal format):

80 03 02 07 11

- 80 – Beginning of a new tuple.
- 03 – length, 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 boardrev in the *nvram.txt* file.
- 07 11 – The value of boardrev in reverse hexadecimal byte or 0x1107.

Table 5 provides an example OTP binary map for a CYW43455 that contains the SDIO hardware header and some CIS tuples of NVRAM parameters.

Note: CIS tuples do not have to be listed in any order because each tuple begins with a unique identifier.

Note: OTP bytes can be written only once. Only blank and zero-programmed bytes can be programmed during subsequent write cycles.

Note: The SDIO header is a set block of data with a predetermined order. Do not use tuples in the SDIO header order. The tuples must be programmed into OTP memory for all SDIO functions to operate properly.

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

Table 5. Example of CYW43455 OTP Binary Map for SDIO

Offset	0x0	0x1	0x2	0x3	0x4	0x5	0x6	0x7	0x8	0x9	0xa	0xb	0xc	0xd	0xe	0xf
0x000000	4B	00	FF	FF	00	00	20	04	D0	02	BF	A9	80	07	19	66
0x000010	55	44	33	22	11	80	03	02	07	11	80	02	00	0b	80	03
0x000020	1b	4a	08	00	00	00	00	00	00	00	00	00	00	00	00	00
0x000030	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0x000040	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0x000050	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0x000060	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0x000070	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0x000080	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0x000090	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0x0000a0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0x0000b0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0x0000c0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0x0000d0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0x0000e0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0x0000f0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0x000100	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0x000110	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0x000120	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0x000130	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0x000140	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0x000150	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0x000160	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0x000170	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0x000180	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0x000190	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0x0001a0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0x0001b0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0x0001c0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0x0001d0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0x0001e0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0x0001f0	FF	FF														

SDIO HW Header
macaddr=66:55:44:33:22:11
sromrev=11
boardrev=0x1107
boardtype=0x084a
OTP end
Max WLAN SW/HW Region size = 368 bytes

.

5.2 Creating and Editing 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. Writing to the OTP memory requires a bin file that fits in the OTP memory space.

For the CYW43455, the maximum size of the OTP memory is 498 bytes.

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

1. Add or edit each byte in the OTP binary map to populate the CIS tuple, as described in the OTP binary map instructions provided in Programming Basic Parameters into OTP Memory.

   Note: The OTP binary map file (see Table 6) has been edited to match the example CYW43455 OTP binary map described in Table 5.

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

   Note: Save the file with a *.bin* file extension so that the data it contains can be programmed into the OTP memory. In this application note, this file is referred as *43455_OTP.bin.*

Table 6 shows the hexadecimal OTP binary map template for the CYW43455 SDIO revision, respectively.

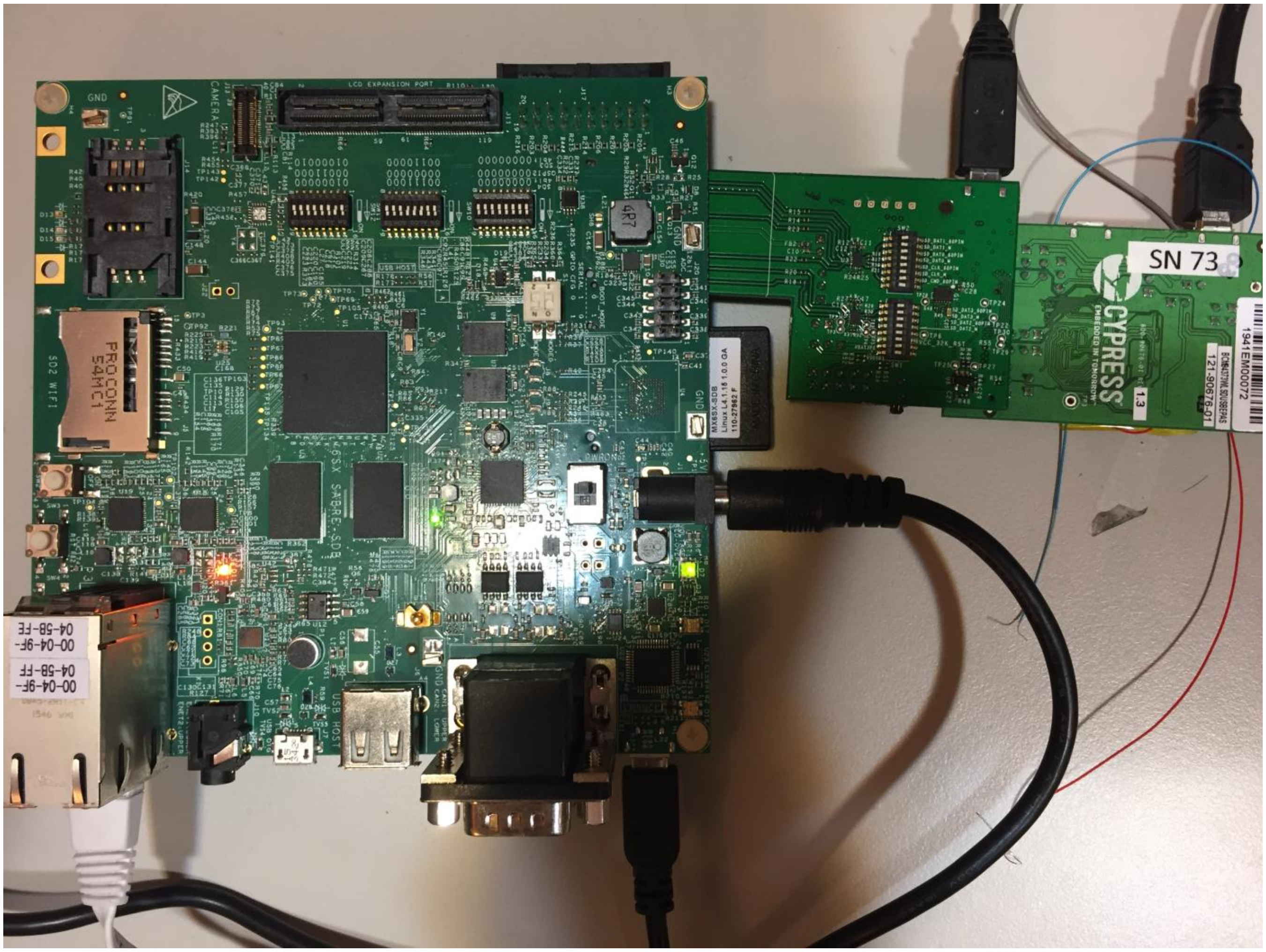
Table 6. CYW43455 SDIO Hexadecimal OTP Binary Map Template

Offset	0x0	0x1	0x2	0x3	0x4	0x5	0x6	0x7	0x8	0x9	0xa	0xb	0xc	0xd	0xe	0xf
00000000	4B	00	FF	FF	00	00	20	04	d0	02	BF	A9	80	07	19	66
00000010	55	44	33	22	11	80	03	02	07	11	80	02	00	0B	80	03
00000020	1B	4A	08	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	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00000170	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00000180	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00000190	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
000001A0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
000001B0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
000001C0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
000001D0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
000001E0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
000001F0	FF	FF														

6 Programming CYW43455 OTP Memory Using iMX6SX

This section outlines the procedure to program the SDIO binary file to the OTP of a CYW43455 device using an iMX6SX FMAC system.

Figure 2. iMX6SX FMAC System Example



The required hardware includes:

- 1x CYW43455 SDIO board – this is the “DUT”
- 1x iMX6SX system with Cypress image that has FMAC Kernel installed (4.14.0 or later) in SD card
- 1x Ethernet cable
- 1x CYW9SDIOAD_1 interposer card (inserted into the SD3 slot on iMX6SX)

The required software includes:

- Cypress SDIO MFG driver package containing driver files for CYW43455 in FMAC (4.14.0) platform (typically provided by Cypress).
- *OTP.bin* file containing the CYW43455 SDIO header string or some CIS tuples of NVRAM parameters. Follow the procedure in Programming OTP Memory to program OTP memory using the *OTP_bin* file.

6.1 Programming OTP Memory

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

1. While powered OFF, connect the iMX6SX to Ethernet.
2. Connect DUT to the 60-pin connector located in the iMX6SX.
3. Plug in the power to the iMX6SX and the iMX6SX system will be turned ON automatically.
4. At prompt, with a specific COM port for iMX6SX, log in as "root".
5. Copy the CYW43455 driver files and the *OTP.bin* file to a desired directory.
6. Go to the directory where you copied the CYW43455 driver files. Issue the driver load command as you would normally do on a FMAC system, or:

```
> insmod compat.ko
> insmod mmc_core.ko
> modprobe sdhci-pci
> modprobe rfkill
> insmod cfg80211.ko
> insmod brcmutil.ko
> insmod brcmfmac.ko debug=0x100004
> ifconfig wlan0 192.168.1.101 up
> ./wl ver
```

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

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

   a. Run the following command to check the CIS dump in the OTP:

```
> ./wl cisdump
```

b. If your CYW43455 device has never been programmed with the SDIO header in the OTP, check if the cisdump is similar to the following:

```
Source: 2 (Internal OTP)
Maximum length: 496 bytes
Byte   0: 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
Byte   8: 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
Byte  16: 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
Byte  24: 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
Byte  32: 0x00 0x00 0x00 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 0x00 0x00
Byte 232: 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
Byte 240: 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
Byte 248: 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
Byte 256: 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
Byte 264: 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
Byte 272: 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
```

```
Byte 280: 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
Byte 288: 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
Byte 296: 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
Byte 304: 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
Byte 312: 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
Byte 320: 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
Byte 328: 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
Byte 336: 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
Byte 344: 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
Byte 352: 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
Byte 360: 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
Byte 368: 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
Byte 376: 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
Byte 384: 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
Byte 392: 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
Byte 400: 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
Byte 408: 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
Byte 416: 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
Byte 424: 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
Byte 432: 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
Byte 440: 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
Byte 448: 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
Byte 456: 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
Byte 464: 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
Byte 472: 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
Byte 480: 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
Byte 488: 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
Byte 496: 0x00 0x00
```

c. If you can confirm that CYW43455 device has never been programmed, then your device has blank CIS and is ready to be programmed. Go to the directory where you copied the *OTP.bin* file.

   Run the following command:

   ```
   >./wl ciswrite OTP.bin
   ```

d. After programming is completed, confirm the OTP by dumping CIS again(power cycle is not necessary):

   ```
   >./wl cisdump
   ```

   If programming is successful, you should see the dump that looks similar to the following (for SDIO OTP):

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

```
Source: 2 (Internal OTP)
Maximum length: 368 bytes
Byte   0: 0x4b 0x00 0xff 0xff 0x00 0x00 0x20 0x04
```

```
Byte   8: 0xd0 0x02 0xbf 0xa9 0x80 0x07 0x19 0x66
Byte  16: 0x55 0x44 0x33 0x22 0x11 0x80 0x03 0x02
Byte  24: 0x07 0x11 0x80 0x02 0x00 0x0b 0x80 0x03
Byte  32: 0x1b 0x4a 0x08 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 0x00 0x00
Byte 232: 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
Byte 240: 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
Byte 248: 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
Byte 256: 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
Byte 264: 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
Byte 272: 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
Byte 280: 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
Byte 288: 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
Byte 296: 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
Byte 304: 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
Byte 312: 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
```

```
Byte 320: 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
Byte 328: 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
Byte 336: 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
Byte 344: 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
Byte 352: 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
Byte 360: 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
Byte 368: 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
Byte 376: 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
Byte 384: 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
Byte 392: 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
Byte 400: 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
Byte 408: 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
Byte 416: 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
Byte 424: 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
Byte 432: 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
Byte 440: 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
Byte 448: 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
Byte 456: 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
Byte 464: 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
Byte 472: 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
Byte 480: 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
Byte 488: 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
Byte 496: 0xff 0xff
```

If the CIS dump matches your *OTP.bin* file, the OTP programming is successful, and the SDIO header is correctly programmed to your CYW43455 device.

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

7 Programming CYW43455 OTP BD Address

Command:

```
#./wl otpraw <bitoffset> <length> <value>
```

Bit offset:

The offset of BD address is fix in the OTP memory.

Value:

50//Signature, OTP bit offset 4496 is the start of BT OTP signature.

4f//Signature, only 2 bytes of signature for this chip.

10 // Header. Use fixed value of 0x10.

06 // Size of OTP after this byte itself. If you need to program only the BD ADDR, use the size value of 0x06.

ff // BDADDR, 6 bytes; (Assuming BD address is 0xaabbccddeeff).

ee // BDADDR, 6 bytes.

dd // BDADDR, 6 bytes.

cc // BDADDR, 6 bytes.

bb // BDADDR, 6 bytes.

aa // BDADDR, 6 bytes.

Command example:

```
#./wl otpraw 4496 80 0xaabbccddeeff06104f50
```

Document History

Document Title: AN230546 - OTP Memory Programming and NVRAM Development - CYW43455

Document Number: 002-30546

Revision	ECN	Date	Description of Change
**	6892182	06/03/2020	New Application Note.

Worldwide Sales and Design Support

Cypress maintains a worldwide network of offices, solution centers, manufacturer's 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 MCU

Cypress Developer Community

Community | Code Examples | Projects | Videos | Blogs | Training | Components

Technical Support

cypress.com/support

Cypress Semiconductor
An Infineon Technologies Company
198 Champion Court
San Jose, CA 95134-1709
www.cypress.com
www.infineon.com