AR93xx/AR94xx/ AR95xx EEPROM Device Configuration Guide

March 2011

Notice

The information in this document has been carefully reviewed and is believed to be accurate. Nonetheless, this document is subject to change without notice, and Atheros Communications, Inc. (Atheros) assumes no responsibility for any inaccuracies that may be contained in this document, and makes no commitment to update or to keep current the contained information, or to notify a person or organization of any updates. Atheros reserves the right to make changes, at any time, in order to improve reliability, function or design and to attempt to supply the best product possible. Atheros does not represent that products described herein are free from patent infringement or from any other third party right.

No part of this document may be reproduced, adapted or transmitted in any form or by any means, electronic or mechanical, for any purpose, except as expressly set forth in a written agreement signed by Atheros. Atheros or its affiliates may have patents or pending patent applications, trademarks, copyrights, maskwork rights or other intellectual property rights that apply to the ideas, material and information expressed herein. No license to such rights is provided except as expressly set forth in a written agreement signed by Atheros.

ATHEROS MAKES NO WARRANTIES OF ANY KIND WITH REGARD TO THE CONTENT OF THIS DOCUMENT. IN NO EVENT SHALL ATHEROS BE LIABLE FOR DIRECT, INDIRECT, SPECIAL, INCIDENTAL SPECULATORY OR CONSEQUENTIAL DAMAGES ARISING FROM THE USE OR INABILITY TO USE THIS PRODUCT OR DOCUMENTATION, EVEN IF ADVISED OF THE POSSIBLITY OF SUCH DAMAGES. IN PARTICULAR, ATHEROS SHALL NOT HAVE LIABILITY FOR ANY HARDWARE, SOFTWARE, OR DATA TRANSMITTED OR OTHERWISE USED WITH THE PRODUCT, INCLUDING THE COSTS OF REPAIRING, REPLACING, INTEGRATING, INSTALLING OR RECOVERING SUCH HARDWARE, SOFTWARE OR DATA. ATHEROS SPECIFICALLY DISCLAIMS THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR A PARTICULAR PURPOSE AS THEY MIGHT OTHERWISE APPLY TO THIS DOCUMENT AND TO THE IDEAS, MATERIAL AND INFORMATION EXPRESSED HEREIN.

Document Number: MKG-15342 Ver. 2.0

Revision History

Revision	Description of Changes
March 2011	Added AR94xx and AR95xx support. This version applies to these chips: AR9331, AR9333, AR9341, AR9342, AR9344, AR9350, AR9380, AR9381, AR9382, AR9390, AR9392, AR9462, AR9463, AR9482, AR9485, AR9580, AR9582, AR9590, AR9592.
November 2010	Initial release for AR93xx

Contents

List of Tables

Preface

This document provides information on board calibration and board variation values. Storing board data and board calibration information of the Atheros AR93xx/AR94xx/AR95xx devices is required for these solutions. This data is typically stored in an EEPROM, on-chip one-time programmable (OTP) memory, or target Flash memory system. While not always existing in a physical EEPROM, this collection of data is often referred to throughout this document as EEPROM configuration data.

NOTE: This version applies to these chips:
AR9331, AR9333, AR9341, AR9342, AR9344, AR9350, AR9380, AR9381, AR9382, AR9390, AR9392, AR9462, AR9463, AR9482, AR9485, AR9580, AR9582, AR9590, AR9592.

About this Document

This document consists of the following chapters and appendix:

Chapter 1	**Device Configuration**—Describes the contents stored on the EEPROM.
Chapter 2	**Board Data**—Describes the EEPROM board data.
Appendix A	**EEPROM Board Data Structure File**—Describes the EEPROM board data contents structure file.
Appendix B	**Characterizing Temperature Slope**—Describes temperature slope measurement.

Audience

This document is intended for Atheros customers involved with the definition, design, and implementation of modules deploying the Atheros AR93xx/AR94xx/AR95xx chip sets.

Additional Resources

Atheros Reference Design hardware, software, and documentation contain proprietary information of Atheros Communications, Inc., and are provided under a license agreement containing restrictions on use and disclosure, and are also protected by copyright law. Reverse engineering of this hardware, software, or documentation is prohibited.

his guide assumes that the reader is familiar with the *AR93xx/AR94xx/AR95xx ART2 Reference Guide*.

1

Device Configuration

This chapter describes the details of the device configuration information stored on the AR93xx/AR94xx/AR95xx solution. The target drivers use configuration information to ensure optimum and regulatory certified performance of the wireless network interface.

The target driver loads three types of information from the board data information: solution-specific parameters to make the device function correctly for all external board components and regulatory requirements, individual card calibration data to account for part variance and achieve matching system results across a solution, and AR93xx/AR94xx/AR95xx-specific values that identify the version of the board.

NOTE: Although complexity has increased with up to three radios working in concert, Atheros has tried to maintain the known working techniques used by previous designs by extending them to cover three radios working as a single WLAN device.

Determining Concepts

This section discusses concepts used to determine what information is stored in board data and how to use that information. These concepts include:

- "Piecewise Linear Abstraction"
- "Frequency Piers"
- "Power Calibration"
- "Target Power"
- "Target Power Frequencies"
- "CTL Indexes"
- "Conformance Testing Limits"
- "Country or Domain Code"
- "Support of Multiple Regulatory Domains"
- "Operating Power Algorithm"
- "Switch Table Operation"

This section presents the techniques Atheros uses to calibrate the analog properties of each individual card. In most cases these techniques take a large amount of measured data across frequency and power levels and presents the data representation in the EEPROM's confined space.

Piecewise Linear Abstraction

Piecewise linear abstraction (PLA) technique captures general dependence accurately if it is sampled at appropriate turning points (TPs) and linearly interpolated between the TPs. Figure 1-1 demonstrates how the PLA scheme maintains general dependence accuracy if appropriate TPs are selected. This example shows the broader 5 GHz spectrum with various turning points.

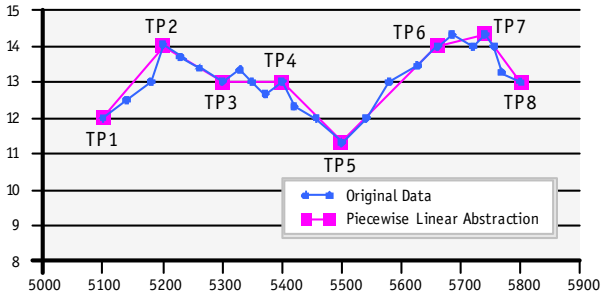

Figure 1-1. **PLA Scheme Applied to a General Dependence**

NOTE: Because these dependences arise from statistical variations of parameters and their interplay, the nature of dependences can vary from card to card. Therefore TPs may be at different locations for each card and fixing the locations of sampling points will, in general, not preserve the data with high accuracy for all cards.

A high degree of accuracy for each card can be preserved if the locations of the TPs are also stored with the sampled values for a sufficient number of TPs. This theme is central in the approach adopted by Atheros to store any NIC-specific or subsystem-specific calibration information on the EEPROM. This enables card manufacturers to deliver the highest level of performance accuracy tailored for each individual card.

Frequency Piers

The PLA concept applies to any kind of dependence. Figure 1-2 demonstrates how the PLA scheme can extend to a set of curves to accurately reproduce an original data set by sampling a few TPs if the sampling points are chosen well.

When the PLA scheme is applied to the dataset obtained by measuring the output power over a range of frequencies, the TPs for this family of curves are referred to as the frequency piers.

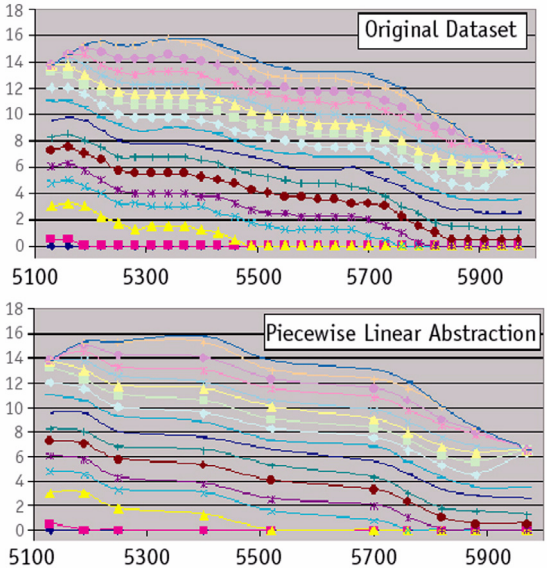

Figure 1-2. **Abstraction of a Set of Curves Through PLA Scheme**

For AR93xx/AR94xx/AR95xx, the output power level is controlled by the power calibration method (see "Power Calibration").

Manufacturing calibration can measure output power across channels, correlate these values for similarity, and then design a list of the best TP to capture the data. However, for calibration expediency, this process is performed once per solution and then a fixed list of frequency piers applies to all boards using that solution. These piers can consist of up to eight frequency piers for 5 GHz, and up to three for 2 GHz are determined from the pilot runs. The list of piers is specified within the calibration command arguments. Power measurements are performed only for these channels and stored on the reference design programmable memory as the calibration data.

Power Calibration

Each wireless device is required to comply with local emission regulations and 802.11 spectral limitations. Due to the sensitive nature of RF circuit design, for each manufactured device to provide an optimal level of throughput performance and output power, it is essential to calibrate each device and store the raw performance capability information in the reference design programmable memory.

Open Loop Power Control

To achieve a specific Tx power when transmitting 802.11 a/b/g/n (WiFi) packets, our software specifies a target power in a baseband (BB) register. BB picks the nearest Tx gain table entry to achieve the specified target power and send it to the analog transmitter. A gain table entry consists of information that directly controls operation of analog components such as amplifiers.

However, temperature changes and voltage swings can affect the behavior of analog components, causing the actual Tx power to deviate from the desired target power. In previous Atheros chips, an external power detector detects the feedback voltage. That information is then fed back to the BB, which measures the difference between the desired target power and measured Tx power, and takes this difference into account when picking a gain table entry from the gain table for subsequent packets. This scheme is called Closed Loop Power Control.

Starting with the Atheros AR93xx/AR94xx/AR95xx chips, the new Open Loop Power Control (OLPC) scheme means that the external power detector is no longer needed.

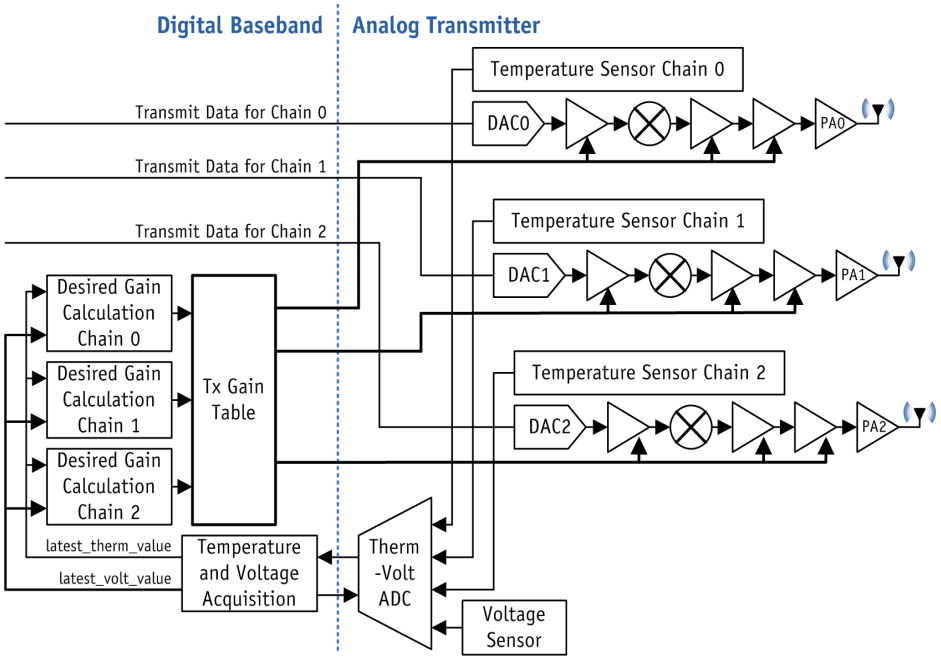

Figure 1-3. **Open Loop Power Control Block Diagram**

With OLPC, AR93xx/AR94xx/AR95xx chips can measure temperature and voltage continuously. Temperature and voltage information is fed into an algorithm that determines the nearest gain table entry to pick to achieve the target power specified by the software. In this scheme, any temperature changes and voltage swings are tracked continuously, resulting in quick response in terms of potential transmitter gain change required to adjust the Tx power to get as close as possible to the target power.

There are two main components in the OLPC scheme: "Temperature Voltage Acquisition" and "Desired Gain Calculation".

Temperature Voltage Acquisition

Temperature Sensor, Voltage Sensor and Therm-Volt ADC

In the AR93xx/AR94xx/AR95xx chips, each analog Tx chain contains a temperature sensor. Each chip also has a single voltage sensor, as shown in Figure 1-3. During measurement, only one of the four temperature sensors is turned on. The BB controls selection of temperature sensor and its turn on/off. Including one temperature sensor in each chain gains flexibility; thus software allows three-chain Tx configuration as well as the option to transmit packets using a two- or single-chain configuration. If a single chain is enabled for transmission, for example, that chain's temperature sensor is turned on.

A new ADC digitizes temperature and voltage information into a digital value. The new therm-volt ADC is shared by the temperature sensors and the voltage sensor. At any measurement time, either temperature or voltage sensor output is digitized by the ADC.

Temperature and Voltage Measurement and Application

In AR93xx/AR94xx/AR95xx chips, a scheme inside the BB determines:

- When temperature and voltage are measured
- How frequent temperature and voltage are measured
- How temperature and voltage are measured
- When measured values are applied to the calculation of desired gain
- When to measure

The measurement of temperature or voltage is carried out only during transmission of a packet because chip internal temperature may be different between Tx and Rx. In Tx, the power amplifier (PA) is turned on, which contributes to the rise of temperature. To make the Tx power control more accurate, temperature is measured only during transmit when the PA is on.

Applying Temperature and Voltage Computation

Just before each transmit, the BB calculates the desired gain variable, which is used as an input to lookup the Tx gain table. The lookup result is sent to the analog block to configure all analog components (mainly different amplifiers). Transmit starts once configuration is done. Calculation of the desired gain for the current packet uses the temperature and voltage measured in the previous packets when the last measurement took place. While the current packet is transmitting, new temperature or voltage is measured. The measurement results are saved into LATEST_THERM_VALUE or LATEST_VOLT_VALUE, which are later used to calculate the desired gain for the next Tx packet.

Desired gain for the first and second Tx packets after the system reset is calculated because software estimates the LATEST_THERM_VALUE and LATEST_VOLT_VALUE values and may be inaccurate, thus desired gain for the first Tx packet may not be accurate. During the first Tx packet after the system reset, temperature is measured at the end of the first Tx packet and LATEST_THERM_VALUE gets the accurately measured temperature result. Desired gain calculation for the second Tx packet uses this accurate LATEST_THERM_VALUE, and the initial estimated LATEST_VOLT_VALUE. Voltage is measured during the second Tx packet, thus at the end of the second Tx packet, LATEST_VOLT_VALUE gets the accurately measured voltage result. From then on, all subsequent Tx packet scans use accurately measured temperature and voltage results to calculate their desired gain.

Desired Gain Calculation

Calculations of desired gain for each Tx chain just before the start of a Tx packet uses the equation:

```
desired gain = target_power - olpc_gain_delta - therm_gain_corr -
volt_gain_corr
therm_gain_corr = alpha_therm x (latest_therm_value -
therm_cal_value)
volt_gain_corr = alpha_volt x (latest_volt_value - volt_cal_value)
```

Where:

alpha_therm	Describes the relationship between temperature change and gain change (assumes linear relationship and no absolute temperature reference). This value is obtained during lab characterization and it is done only once. It applies to all Tx chains and to all chips.
alpha_volt	Describes the relationship between voltage change and gain change (assumes linear relationship and assumes no absolute temperature reference). This value is obtained during lab characterization and it is done only once. It applies to all Tx chains and to all chips.
latest_therm_value	Latest temperature value; see "Applying Temperature and Voltage Computation" on page 1-5 for calculation information
latest_volt_value	Latest voltage value; see "Applying Temperature and Voltage Computation" on page 1-5 for calculation information
olpc_gain_delta	A correction factor calculated during TPC manufacturing calibration. Its value is channel frequency and Tx chain dependent, i.e., calibration must be carried out for each chain and channel frequency.
target_power	The expected target Tx power specified by software. Different MCS values could have different target powers, controlled by software. Desired gain calculations for all Tx chains use the same target_power.
therm_cal_value/ volt_cal_value	Temperature and voltage reading after manufacturing calibration. All Tx chains have the same therm_cal_value and volt_cal_value.

Target Power

The maximum power that satisfies all IEEE specification requirements (e.g., spectral mask) and performance criteria (that is, < 10% packet error rate (PER)) is determined for a particular subsystem design through a pilot run over a statistical ensemble of NICs. This power is referred to as target power.

This measurement is not performed individually for each card, and the target power does not take into account the regulatory domain's limited power. Target power is an indication of raw capability of a particular card type, regardless of the regulatory domain where the cards are used. Target powers can also be solution specific as required by a vendor to match previous product power level constraints.

Generally, a unique target power exists for each rate. However, for all Atheros reference designs, the rates of 6–24 Mbps and the rates of 1L – 5L have the same target power (spectral mask limited). The rates 5.5S, 11, 36, 48, and 54 Mbps have their own target power (PER-limited) over the entire frequency range.

AR93xx/AR94xx/AR95xx designs support target powers for the new 802.11n MCS rates. Target powers can be specified for these MCS rate groupings:

- MCS0, MCS8 and MCS16
- MCS1, MCS2, MCS3, MCS9, MCS10, MCS11, MCS17, MCS18, MCD19
- MCS4
- MCS5
- MCS6
- MCS7
- MCS12
- MCS13
- MCS14
- MCS15
- MCS20
- MCS21
- MCS22
- MCS23

Two sets of MCS rates (HT20 and HT40) are stored and retrieved across frequencies. The 802.11n draft document also contains a convention of using a control and an extension channel to allow HT40 transmissions while monitoring and using control traffic to manage two legacy channels. When running in dynamic HT40 mode, calibration target powers are computed separately for the frequencies of the control channel, the extension channel, and the HT40 channel that operates on top of the two legacy channels.

Target Power Frequencies

Target power for all rates or rate groups depends on the frequency. Board data has provisions for vendors to specify a number of TPs per target power rate group of this dependence under the PLA scheme. These TPs, used to convey target power information, are referred to as target power frequencies and are determined by the vendor after analyzing data gathered during the pilot run and conveyed to the calibration routine by the target power file. See the *AR93xx/AR94xx/AR95xx Atheros Radio Test 2 Reference Guide* for more information.

Conformance Testing Limits

CTL Indexes

In a regulatory domain, the frequency bands open for public infrastructure are typically interspersed with the restricted bands for military or government use. The extreme operating channels in the open frequency bands are referred to as band edges. For example, the band edge channel centers in a UNII-1 band of 5.15 GHz–5.25 GHz.

Special consideration is needed to determine the transmit power at the band edges to ensure compliance with the regulations in the adjacent restricted frequency band. Channels that fall within the open band (between the band edges) do not require this special consideration.

Because OFDM, CCK, HT20 and HT40 modes have different power signatures, they require different power levels. The 802.11b band edges provide CCK band edge information. Given 802.11g operation requires both CCK and OFDM, the 802.11g band edges specify the OFDM operation limits and an 802.11g CTL implies a search through the CTL indexes for a matching domain 802.11b CTL to get the CCK band edge limitations. AR93xx/AR94xx/AR95xx designs require specifying a number of regulatory domains as well as a number of modes, thus 21 indexes are provided in the board data structure.

The world mode operation (applies to client designs only) initializes without knowing the current country location (until an 802.11d beacon is heard). Thus, world mode must initialize its power levels with the lowest power across all CTL indexes that applies to the channel being used.

Conformance Testing Limits

Significant similarities exist in the boundaries of the open bands in several regulatory domains due to how the frequency bands have been opened for allocation in the 5 GHz and 2 GHz range worldwide. Thus several regulatory domains exist with identical sets of band edges. Conformance testing limits (CTLs) leverage this overlap, delivering a simplified mechanism supporting several regulatory domains in manufacturing. It is essential to convey band edge maximum power information to the driver using board configuration data for all regulatory domains where the NIC is targeted. This data is subsystem design-specific and gathered during the pilot run. If a regulatory domain-based approach is used to store this information on the board, considerable redundancy exists for domains with overlapping band edges.

AR93xx/AR94xx/AR95xx designs contain multiple radios; all or one of the radios can transmit at a time. To conserve space in the board configuration structure, only one CTL power value can be set regardless of how many radios are transmitting. To alleviate this redundancy and maximize the number of regulatory domains supported by a NIC, a CTL is defined to be a unique set of band edges and adjacent restricted band regulations, e.g.:

- If RD1 and RD2 have a permitted band from 5180–5240 MHz and the same set of restrictions for frequencies below 5180 and frequencies higher than 5240, then RD1 and RD2 can belong to the same CTL.
- If RD3 also has a permitted band from 5180–5240 MHz, but tolerates higher power for frequencies below 5180 (i.e., 2 dB higher tolerance at 5160), then RD3 can not belong to the same CTL as RD1 and RD2.
- If RD4 does not permit Tx in 5180–5240 MHz, but has a permitted band from 5400–5520 MHz, then RD4 can belong to the same CTL as RD1 and RD2 because the two bands do not overlap. The CTL now contains band edges 5180, 5240, 5400, and 5520, which is possible because software contains a list of legal channels in each regulatory domain (so for RD4, it does not even look at 5180 and 5240).

Regulatory domains of the band edges appearing in a CTL belong to that CTL. A CTL may contain additional band edge pairs, providing data for one CTL and enabling support for all regulatory domains belonging to it. Up to 21 CTLs are supported in the board configuration structure. CTLs also contain continuous application flags: one for each CTL frequency. In these 1-bit flags, 0 indicates that the CTL frequency is a band edge; 1 indicates that the CTL acts as a band edge and a continuing limit for frequencies greater than this band edge until the next band edge in the list. CTL flags apply to frequencies greater than or equal to the CTLs frequency up to but not including the frequency listed in the following CTL. In some cases, regulatory stipulations imposed outside of this band may restrict power output at not only the band edges, but also for some channels within the band. It then becomes necessary to specify limits on these in-band channels for that CTL. Non-edge flags are introduced to handle such cases. The design to use these flags is:

- All frequencies specified in a CTL must be arranged in ascending order.
- An in-band frequency marks the beginning of the channel range to apply the corresponding CTL limit to. This range goes up to and includes all following channels. It is permitted to specify only in-band frequency CTLs or even a single in-band CTL to cover an entire regulatory band.

These examples demonstrate how in-band frequency can be used.

Example 1

A band exists from 5420–5600 MHz. Out of band regulations require the power be limited to 12 dBm at the band edges, 13 dBm for 5420–5520, and 12.5 dBm for 5540–5580 MHz. Table 1-1 on page 1-10 demonstrates how to convey this information using the non-edge flags.

Table 1-1. **Non-Edge Flag Usage in CTLs (Example 1)**

	Band Edge	In-band Freq	In-band Freq	Band Edge	Next Band
CTL Freq	5400	5420	5540	5600	...
CTL Limit	12	13	12.5	12	...
Non-edge Flag	0	1	1	0	...

Example 2

A band exists from 5400–5500 MHz, out of band regulations require the power be limited to 13 dBm at starting band edge (5400), 15 dBm for 5420–5480 MHz, and 12 dBm at the ending band edge (5500). Table 1-2 demonstrates how to convey this information using the non-edge flags.

Table 1-2. **Non-Edge Flag Usage in CTLs (Example 2)**

	Previous Band	Band Edge	In-band Freq	Band Edge	Next Band
CTL Freq	...	5400	5420	5500	...
CTL Limit	...	13	15	12	...
Non-edge Flag	...	0	1	0	...

Country or Domain Code

A unique 14-bit code identifies the intended country or domain, of operation/sale. The target driver uses this code with the Country Code Selector (CCS) and worldwide roaming (WWR) flags to determine the current operating region and overlay the appropriate regulatory domain requirements on top of the target power and the band edge maximum power data. See the support bulletin *Worldwide Roaming Design Specification* for details on how this information is used.

Support of Multiple Regulatory Domains

The following information is coded in the driver to allow support of multiple regulatory domains:

- A mapping of each country code to a regulatory domain
- An association of all regulatory domains to the appropriate CTLs
- All allowed channels and the maximum legal power limits in all regulatory domains

It is important to program a comprehensive set of CTLs in the board configuration structure at manufacturing calibration. Thus supporting new frequency allocations in various countries (domains), or changes in regulations in existing regulatory domains, becomes possible through a software release of the NDIS driver or AP software update with the NICs already deployed in the field.

Operating Power Algorithm

The target driver uses information stored in the board data to determine the maximum transmit power for a given channel using the algorithm:

1. Read the country code from EEPROM.
2. Obtain a list of permitted channels for this country from the driver's regulatory domain table. If the current channel does not appear in the list of permitted channels, no transmission is initiated at this channel.
3. Reconstruct the calibration table for the current channel from the calibration data sampled at the frequency piers stored in the EEPROM interpolating as appropriate under the PLA scheme. Program the calibration table into MAC/BB processor chip.
4. Obtain the target power for each rate at the current channel from the data stored in board data. Target powers are set on a per-chain basis.
5. The driver determines the CTL for this country code and retrieves data for this CTL from board data. If the current channel is determined to be a band edge in this CTL, obtain band edge maximum power at this channel based on the number of radios transmitting. CTL values are measured and set on a per-chain basis.
6. The driver determines the current channel local regulatory power limit as well as any user configured or outside power limits. Software derived regulatory maximum and power limit values are often set in total power and are decreased when applying to multiple chains operating at once. Often –3 dB for two radios and –4.5 dB for three radios although this delta can be set by the EEPROM.
7. Compute the minimum of the target power, band edge max power, and local regulatory power limit at the current channel values for each rate and program the max power for all supported rates into the MAC/BB processor chip.

Switch Table Operation

This section describes the switch table used by the BB to control external and radio control signals. It also controls AGC receive attenuation with respect to the state the device has entered (see Table 1-3).

The switching table is broken into three tables for AR93xx/AR94xx/AR95xx. The first provides chain specific AGC control to attenuate large input signals. The second allows control of external signals for changing between transmit, idle, and BlueTooth coexistence, but is not chain specific. The third allows control of external signals for receive and is also not chain specific. All tables drive external pins and the analog internal LNA enable. The tables can be programmed in whatever manner the solution requires.

Table 1-3 shows valid states for the device that select an output set. Receive attenuation steps 4, 5, and 6 are only used by the chain-specific switch table.

Table 1-3. **Switch Table Operation**

Setting	Bits	State	Definition[1]
EEPROM word for the chain-specific table is defined as:	This parameter word is repeated once for each chain		
	31:12	RES	Reserved
	11:10	table_blueTooth	BlueTooth coexistence state table_blueTooth [11:10]
	9:8	table_atten1&2	Receive with both attenuation additions (never used)
	7:6	table_atten1	Receive with first attenuation addition
	5:4	table_receive	Receive with no attenuation addition
	3:2	table_transmit	Transmit state. table_transmit [3:2]
	1:0	table_idle	Idle state. table_idle [1:0]
The EEPROM word for the common table1 is defined as:	15:12	table_com_blueTooth	{sw_com[3, 2, 1, 0]} when BlueTooth
	11:8	table_com_transmit_t2	{sw_com[3, 2, 1, 0]} when tx ant2[2]
	7:4	table_com_transmit_t1	{sw_com[3, 2, 1, 0]} when tx ant1
	3:0	table_com_idle	{sw_com[3, 2, 1, 0]} when idle
The EEPROM word for the common table2 is defined as:	19:16	table_com2_ra12	{sw_com[3, 2, 1, 0]} when ant1&2 and lna1&2 combining[2]
	15:12	table_com2_ra212	{sw_com[3, 2, 1, 0]} when rx ant2, lna2[2]
	11:8	table_com2_ra112	{sw_com[3, 2, 1, 0]} when rx ant1, lna2[2]
	7:4	table_com2_ra211	{sw_com[3, 2, 1, 0]} when rx ant2, lna1
	3:0	table_com2_ra111	{sw_com[3, 2, 1, 0]} when rx ant1, lna1

[1] For this table, SW_*x*0 (where *x* is the chain) maps to the XLNABIAS pin in the AR938x/AR958x chips.
[2] AR93xx/AR94xx/AR95xx does not use ant2 or lna2. For these chips, these values are set to the ant1 and lna1 values, respectively.

These switch common lines are connected to any external components that need to be flipped between state transitions: LNAs, PAs, and Tx/Rx switches. As these table have both 5 GHz and 2 GHz copies, lines may also be used for 2 GHz to 5 GHz switching on dual-band products. Define the table based on the board layout needs of polarity and external component switching.

The large signal receive has two attenuation stages that require two additional sets of parameters. Both the atten 1 and atten 2 stage require setting EEPROM parameters to specify when the large signal attenuation should be activated, which acts as a hysteresis, and how much attenuation is gained by the attenuation stage (see Table 1-4).

Table 1-4. **Attenuation Steps**

Step	Definition
Atten1	First attenuation step. The DB attenuation AGC can expect when using atten 1 must be written to the BswAtten parameter. The margin/hysteresis for AGC to use for this stage must be written to BswMargin.
Atten2	Second attenuation step. The DB attenuation AGC can expect when using atten 2 must be written to the txrxatten field. The margin/hysteresis for AGC to use for this stage must be written to the rxTxmargin field.

2

Board Data

This chapter describes board data configuration for AR93xx/AR94xx/AR95xx solutions.

Reference Design Storage for Board Data

AR93xx/AR94xx/AR95xx reference designs support board data storage in three storage types.

Table 2-1. **Board Data Storage Types**

Type	Description
On Chip One Time Programmable Memory (OTP)	The AR93xx/AR94xx/AR95xx reference design has 8 Kbits of on-chip, one time programmable memory (OTP). Once a value of 1 is written to any memory bit, its value cannot be changed. To fit all board data in OTP and still be able to calibrate a board multiple times, board data is stored in OTP in a compressed format. For subsequent calibration, writes to OTP after calibration append to the end of the existing board data blocks. How many times a board can be recalibrated depends on what information changes between writes.
EEPROM	AR93xx/AR94xx/AR95xx reference designs support multiple EEPROM sizes. ART2 software can be configured to store data to EEPROM either using the same compression scheme as for OTP storage or the calibration structure can be stored uncompressed if the size of the EEPROM is at least 16k bits.
Flash (AP solutions only)	For access point based reference designs, board data can be stored on flash. When stored in flash, the board data is stored uncompressed.

Selection of where the data should be stored is made via ART2 during manufacturing and calibration.

Board Data High Level Description

Board data consists of two key components:

- Bus specific initialization values, which are automatically loaded to chip registers upon card insertion
- WLAN configuration and calibration information

Storage location for these components is different for OTP, EEPROM, and for Flash.

OTP and EEPROM High Level Layout

Figure 2-1 shows the high level layout of OTP and EEPROM.

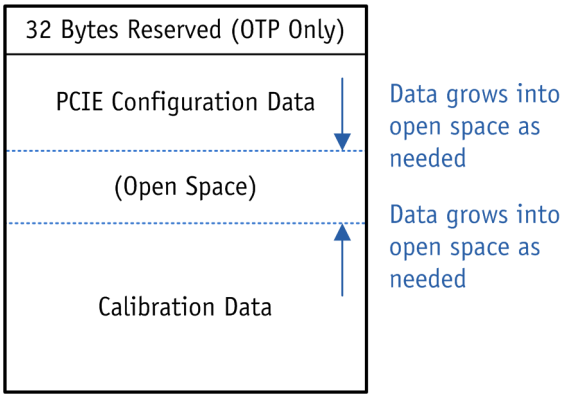

Figure 2-1. **OTP and EEPROM High Level Layout**

PCIE configuration data grows down into the open space as needed. WLAN-specific calibration data grows up into the open space as needed.

Flash High-Level Layout

The sector of flash allocated for the calibration structure is shown in Figure 2-2. Space has been allocated for up to 2 radios. MAC0 is assigned with ETH0 and MAC1 is assigned with ETH1. LAN/WAN can be programmed to MAC0 or MAC1. It is programmable from the Ethernet driver.

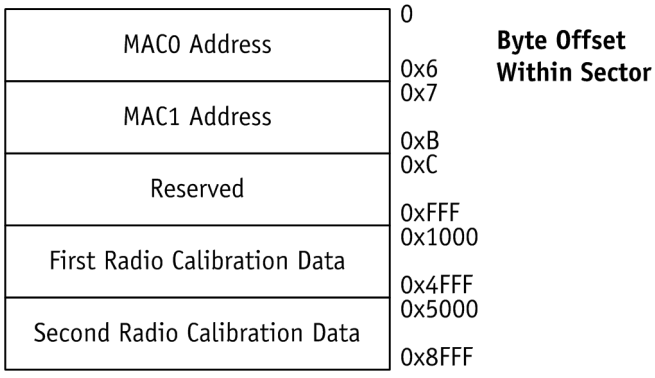

Figure 2-2. **Flash High-Level Layout**

Board Data Detailed Description

This section details the contents of the bus and WLAN-specific information.

Bus Specific Initialization Information

The bus specific initialization space (either starting at location 0 of a hardware EEPROM, 32 bytes from start of OTP, or relative location 0 of a flash sector allocated for board data storage) contains non-WLAN calibration initialization for the board. Table 2-2 shows the values contained if a physical EEPROM is present (such as on MiniPCI, CardBus, and PCIE devices).

Table 2-2. **EEPROM Values**

Value	Definition
Magic Half Word	A 16-bit magic half word, 0xA55A, that indicates that the EEPROM has been programmed at 16-bit offset 0.
EEPROM Read/ Write Mask	A 16-bit EEPROM read/write mask that can protect the EEPROM contents at 16-bit offset 1.
Location Link	A 16-bit location link to the beginning of board setup tuples at 16-bit offset 2. This value is always 0x0003 to point to 16-bit offset 3.
Offset 3	The 16-bit offset 3 begins groups of 48-bit board setup tuples containing a 16-bit register location followed by 32 bits of register data (least significant 2 bytes first). First address 0xFFFF ends auto initialization.

Upon cold reset, hardware parses through these board setup tuples and programs hardware registers to initialize register space including: PCIE configuration setup, LED initialization, and possibly sleep or timer registers requiring setup before the device is probed by the attached bus. The 16-bit offset specified in the triple tuple should contain the full register offset. Table 2-3 specifies offsets to use in accessing PCIE configuration registers.

Table 2-3. **PCIE Configuration Address Mapping**

Register Group	Offset
PCI Express Configuration	0x5000

If a physical EEPROM or OTP is not used (such as for an integrated access point), the device responds to bus probing with default hardware deviceID and subvendorDeviceID information. In this case, this area can store out-of-band information such as LAN MAC addresses, subVendorDeviceIDs, or software-required version/revision information. Figure 2-3 shows the EEPROM initialization board data details.

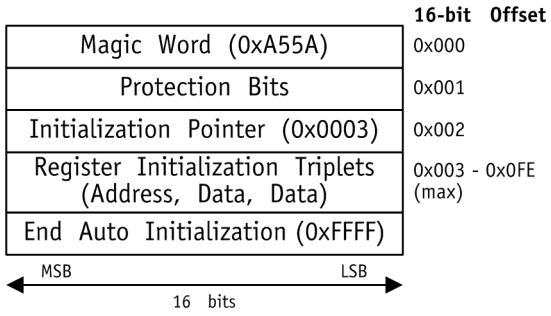

Figure 2-3. **EEPROM Initialization**

OTP supports a similar scheme for auto-initialization of hardware values, however the layout is slightly difference and some extra precautions have been built in case a write to OTP fails. The first 32 bytes of OTP are reserved for Atheros use, so the initialization area starts from byte address 32. At this location will be the first address (unlike in EEPROM). In OTP, 32 bits are used to specify the address and 32 bits for data. Figure 2-4 shows the format of the address. Note that the EEPROM structure ends with 0xFFFF and the OTP ends with 0x0000.

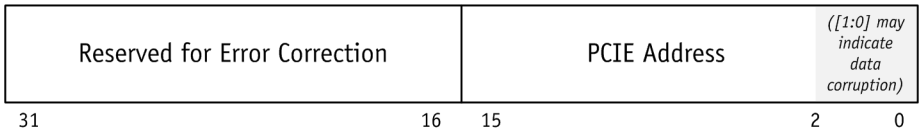

Figure 2-4. **OTP Address Format**

NOTE: Address is specified in the lower 16 bits. Address values increment by 4 bytes so bits [1:0] will not be set as part of a valid address. These bits are set to indicate data corruption.

Writes to OTP are not 100% reliable, so there is a small chance that an address may not write correctly. When writing these values to the OTP the ART2 calibration software will check for the address and data pairs to be written correctly. If there is an error in either address or data, the software will invalidate the address by writing 1 to bits [1:0] of the address. A 1 in either of these bit locations indicates to the hardware that it should skip that address/data pair. If software is unable to write a 1 to either of these bits, then the chip should be discarded or add an EEPROM.

Table 2-4 lists the location in EEPROM/OTP of the initialization triplets of the common PCIE configuration registers that typically get initialized.

Table 2-4. **Common Register Initialization Triplets**

EEPROM Byte Offset	OTP Byte Offset	Description	
0x0006	0x0020	Contains 0x5000	Address for PCIE vendor and DeviceID register
0x0008	0x0024	PCIE VendorID	Fixed at 0x168C for Atheros cards
0x000A	0x0026	PCIE DeviceID	Fixed at 0x0030 for AR93xx/AR94xx/AR95xx PCIE cards
0x000C	0x0028	Contains 0x5002C	Address for PCIE subvendor and subsystemID
0x000E	0x002C	PCIE SubVendorID	Configurable by customer
0x000f	0x002E	PCIE SubsystemID	Configurable by customer; varies by reference design
0x0010	0x0030	Contains 0x5008	The address for the PCIE class code
0x0012	0x0034	Contains 0x0001	Revision ID and the LSB of PCIE class code
0x0014	0x0036	Contains 0x0280	2 MSB of PCIE class code
0x0016	0x0038	Contains 0x407C	Atheros proprietary register for PCIE operation
0x0018	0x003A	Contains 0x0003	Atheros proprietary
0x001A	0x003E	Contains 0x0000	Atheros proprietary
0x001C	0x0040	Contains 0x4004	Atheros proprietary register for PCIE operation
0x001E	0x0044	Contains 0x021B	Atheros proprietary
0x0020	0x0046	Contains 0x0102	Atheros proprietary
0x0022	0x0048	Contains 0x4040	Atheros proprietary register for PCIE operation
0x0024	0x004C	Contains 0x2E5E	Atheros proprietary
0x0026	0x004E	Contains 0x0821	Atheros proprietary
0x0028	0x0050	Contains 0x4040	Atheros proprietary register for PCIE operation
0x002A	0x0054	Contains 0x003B	Atheros proprietary
0x002C	0x0056	Contains 0x0008	Atheros proprietary
0x002E	0x0058	Contains 0x4044	Atheros proprietary register for PCIE operation
0x0030	0x005C	Contains 0x0000	Atheros proprietary
0x0032	0x005E	Contains 0x0000	Atheros proprietary
0x0034	0x0060	Contains 0x570C	Atheros proprietary register for PCIE operation
0x0036	0x0064	Contains 0x3F01	Atheros proprietary
0x0038	0x0066	Contains 0x173F	Atheros proprietary

Board Data Solution-Specific Information

This information is typically obtained through a pilot run on a statistical ensemble of devices of this solution type. All manufactured devices of this design are expected to result in an optimum level of performance upon use of these settings by the software. These settings are not individually measured or calibrated for each device.

In this revision of the board data layout, space has been allocated to support operation in 802.11a, 802.11b, 802.11g, and 802.11n (5 GHz and 2 GHz) modes. Care has been taken to allocate sufficient space for all calibration parameters based on Atheros' history with a wide variety of customer solutions.

Board Data Device-Specific Information

The device-specific information accounts for all variances across boards due to component differences to produce solutions that maintain similar performance.

The principle device-specific information is the calibration points across frequency that describe power levels seen at the power detector. These levels take into account the gain variance across all the gain stages and provide reference level calibration of the transmit (Tx) power engine to reach the desired power levels.

Board Data Layout

This section describes the details of the WLAN specific section. Byte offsets for components within the structure are no longer provided since in OTP and sometimes in EEPROM, the data is stored in a compressed manner as described in "Compression Scheme" on page 2-47, it is no longer possible to predict the location of where these bytes will be within EEPROM and OTP.

Table 2-5 shows the board data categories.

NOTE: The get/set Command Parameter column in these tables lists the ART2 parameter that should be used with a `set` or `get` command to change that parameter's value within the calibration structure. The `set` commands have corresponding `get` commands (which are used without the arguments).

Use `help set` or `help get` to display a complete list of up-to-date parameters for the `set` and `get` commands.

Table 2-5. **Board Data Categories**

Bytes	Category	Figure	Table
8	Miscellaneous Data	Figure 2-5	Table 2-6
20	Customer Data	—	Table 2-7
25	Base EEPROM Header	Figure 2-6	Table 2-8
71	Modal EEPROM Header 2 GHz	Figure 2-7	Table 2-9
14	Base Extension_1	Figure 2-8	Table 2-10
3	Power Calibration Channels 2 GHz	—	Table 2-12
54	Power Calibration Data 2 GHz	Figure 2-10	Table 2-13
11	Target Power Channels 2 GHz	Figure 2-11	Table 2-14
8	Target Powers 802.11bg CCK	Figure 2-13	Table 2-16
12	Target Powers 802.11g OFDM	Figure 2-14	Table 2-17
42	Target Powers 2 GHz 802.11n HT20 OFDM	Figure 2-15	Table 2-18
42	Target Powers 2 GHz 802.11n HT40 OFDM	Figure 2-16	Table 2-19
12	CTL Indexes 2 GHz	—	Table 2-21
48	CTL Band Edges 2 GHz	Figure 2-18	Table 2-22
48	CTL Power Data 2 GHz	Figure 2-19	Table 2-22
71	Modal EEPROM Header 5 GHz	Figure 2-7	Table 2-9
14	Base Extension_2	Figure 2-9	Table 2-11
8	Power Calibration Channels 5 GHz	—	Table 2-12
144	Power Calibration Data 5 GHz	Figure 2-10	Table 2-13
24	Target Power Channels 5 GHz	Figure 2-12	Table 2-15
32	Target Powers 802.11a OFDM	Figure 2-17	Table 2-20
112	Target Powers 5 GHz 802.11n HT20 OFDM	Figure 2-15	Table 2-18
112	Target Powers 5 GHz 802.11n HT40 OFDM	Figure 2-16	Table 2-19
9	CTL Indexes 5 GHz	—	Table 2-21
72	CTL Band Edges 5 GHz	Figure 2-18	Table 2-22
72	CTL Power Data 5 GHz	Figure 2-19	Table 2-22

Miscellaneous Data

Figure 2-5 shows the miscellaneous data fields.

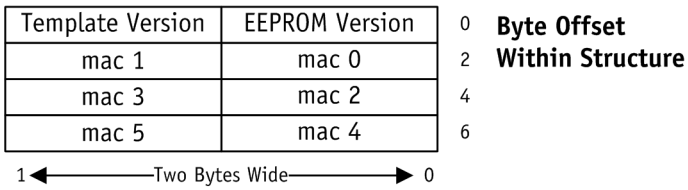

Figure 2-5. **Miscellaneous Data Fields**

Table 2-6 describes the miscellaneous data fields shown in Figure 2-5.

Table 2-6. **Miscellaneous Data Parameter Description**

Parameter Name	Bytes	get/set Command Parameter	Description
eepromVersion	1	Not to be changed by user	Indicates the revision of the calibration structure. Gets updated by software when a large to the calibration structure applies
templateVersion	1	Not to be changed by user	The unique template ID number assigned to each default calibration structure held in the EEPROM. See "Compression Scheme" on page 2-47.
MAC Address	6	`Mac= 00:03:7f:40:a1:87;`	The device's unique WLAN MAC address for APs and clients

Customer Data

Table 2-7 describes the customer data field.

Table 2-7. **Customer Data Parameter Description**

Parameter Name	Bytes	get/set Command Parameter	Description
Customer Data	20	`Customer= hb112-250-n0391;`	These 20 bytes are initially written by the manufacturing flow to contain an Atheros label, but the contents are intended for customers to write their own label or serial number for board tracking purposes.

Base EEPROM Header

Figure 2-6 shows the base EEPROM header board data details.

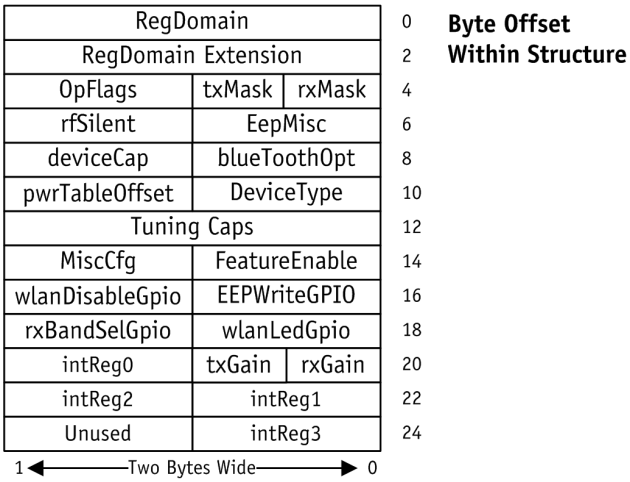

Figure 2-6. **Base EEPROM Header**

Table 2-8 describes the fields shown in Figure 2-6.

Table 2-8. **Base EEPROM Header Parameter Descriptions**

Parameter Name	Bytes	get/set Command Parameter	Description
RegDomain	2	`RegulatoryDomain[0] = 0x0000;`	The first 14 bits (bits [13:0]) of RegulatoryDomain[0] identify the currently selected country or operating domain. The NIC driver or AP software uses it to determine available channels, their operating power, and the operating power at those channels for all data rates (see the support bulletin *Setup for Country or Regulatory Domain*). Most significant bit [15] is the country selector and identifies the 14-bit code. Bit [14] (currently not used) is the world-wide roaming enable bit signifying the card should perform 802.11d regulatory operations.

Table 2-8. **Base EEPROM Header Parameter Descriptions (continued)**

Parameter Name	Bytes	get/set Command Parameter	Description	
RegDomain Extension	2	`RegulatoryDomain[1] =0x001f;`	A 16-bit value used to enable/disable regulatory regions	
			Bit [0]	en_fcc_mid (Must be set to 1) Enables operation in FCC band 5.47–5.7 GHz
			Bit [1]	en_jap_mid (Must be set to 1) Enables operation in Japan band 5.47–5.7 GHz
			Bit [2]	en_fcc_dfs_ht40 (Must be set to 1) Enables operation in FCC band for HT40 support in DFS channels
			Bit [3]	en_jap_ht40 (Must be set to 1) Enables operation in Japan band for HT40 support in 2 and 5 GHz
			Bit [4]	en_jap_dfs_ht40 (Must be set to 1) Enables operation in Japan band for HT40 support in DFS channels
			Bits [15:5]	Reserved; for regulatory setups, or boards that change regulatory during operating lifetime.
txrxMask	4 bits	`Mask.Rx=0x07;`	Bits [3:0] provide a bit mask detailing which device radios are set up to receive. Valid Rx masks are 1, 2, 3, 4, and 7. These bits set a 4-digit hex value to txrxMask.	
			Bit [0]	Setting bit [0] corresponds to physical radio layout 0 being enabled for Rx.
			Bit [2:1]	Bits [1] and [2] of the mask respectively correspond to radio layout 1 and radio layout 2
			Bit [3]	Unused
	4 bits	`Mask.Tx=0x07;`	Bit [7:4] provides a bit mask detailing the radios set up for Tx. Valid Tx masks are 1, 2, 3, 4, and 7. These bits set a 4-digit hex value to txrxMask.	
			Bit [4]	Setting bit [4] corresponds to physical radio layout 0 being enabled for Tx.
			Bit [6:5]	Bits [5] and [6] of the mask respectively correspond to radio layout 1 and radio layout 2
			Bit [7]	Unused
OpFlags	1	`OpFlags=0x03;`	OpFlags describe the modal operational configuration of the board. These bits set an 8-digit hex value to opFlags.	
			Bit [0]	Set if 5 GHz operation allowed
			Bit [1]	Set if 2 GHz operation allowed
			Bit [2]	Set if 5 GHz HT40 operation should be disabled
			Bit [3]	Set if 2 GHz HT40 operation should be disabled
			Bit [4]	Set if 5 GHz HT20 (all 802.11n rates) operation should be disabled (also disables HT40)
			Bit [5]	Set if 2 GHz HT20 (all 802.11n rates) operation should be disabled (also disables HT40)

Table 2-8. **Base EEPROM Header Parameter Descriptions (continued)**

Parameter Name	Bytes	get/set Command Parameter	Description	
RFSilent	1	A collection of RFSilent flags. RFSilent specifies the parameters for configuration of silencing the radio via an externally controlled switch.		
		`rfSilent.HardwareEnable=0x00;`	Bit [0]	If set to 1, a pull up resistor must be placed on the RFSilent GPIO, providing a hardware interface to an external on/off switch allowing manual termination of any RF activity. This bit is only used by a hardware switch. It is ignorned by the hardware if set to 0.
		`RfSilent.Polarity=0x00;`	Bit [1]	Designates the polarity of the RFSilent GPIO switch (0= RFSilent on low, 1= RFSilent on high).
		`RfSilent.Gpio=0x00;`	Bit [4:2]	Allow the EEPROM to select which GPIO acts as the RFSilent GPIO controller
Bluetooth Options	2	`BlueToothOptions=0x00;`	Currently reserved but will be used to set any initial states required by the WLAN's BT coexistence registers.	
device Capabilities	1	`DeviceCapability=0x00;`	A reserved register to change the device capabilities. Should be written as 0.	
DeviceType	1	`DeviceType=0x05;`	Provided for device type/form factor definition. Currently, neither hardware nor software uses these bits, but they should be set for appropriate implementations as future versions of hardware/software may contain device-dependent options. Device type definition:	
			001	CardBus
			010	PCI
			011	Mini PCI
			100	Access Point
			101	PCIE_mini
			110	PCIE_express
			111	PCIE_desktop
pwrTableOffset	1	`PowerTableOffset=+0;`	8-bit value; offset in dB to add to the beginning of the PDADC table during calibration (not currently used).	
tuningCaps	2	`TuningCaps[0]=0x00;` `TuningCaps[1]=0x00;`	(2 Bytes) Values that can be used for tuning of frequency accuracy (ppm) (Feature not fully supported)	

Table 2-8. **Base EEPROM Header Parameter Descriptions (continued)**

Parameter Name	Bytes	get/set Command Parameter	Description	
FeatureEnable	1	A collection of EEPROM FeatureEnable flags:		
		`FeatureEnable.TemperatureCompensation=0x01;`	Bit [0]	Set to 1 to enable temperature compensation for Tx power control
		—	Bit [1]	Reserved. Should be set to 0.
		`FeatureEnable.FastClock=0x01;`	Bit [2]	Set to 1 to enable fast clock (Default = 1)
		`FeatureEnable.Doubling=0x01;`	Bit [3]	Set to 1 to enable doubling (Default = 1)
		`FeatureEnable.SwitchingRegulator=0x00;`	Bit [4]	■ Set to 1 if using AR93xx/AR94xx/AR95xx internal regulator. ■ Set to 0 to use external regulator (must be set on AR93xx/AR94xx/AR95xx reference designs)
		`FeatureEnable.PaPredistortion=0x01;`	Bit [5]	Set to 1 to enable PA predistortion
		`FeatureEnable.TuningCaps=0x00;`	Bit [6]	Set to 1 to enable tuningCaps[] values to take effect
		—	Bit [7]	Reserved. Should be set to 0.
Misc Configuration	1	`Miscellaneous.DriveStrength=0x00;`	Bit [0]	Reserved. Should be set to 0.
		`Miscellaneous.Thermometer=+1;`	Bit [2:1]	Sets which chain's thermometer will be used for calibration and power control.
		`Miscellaneous.Dynamic2x3=0x00;`	Bit [3]	Enables dynamic 2x3 mode when the PCIE current limit is 2.5 W. See the *Adaptive Power Management* application note for more information.
		`Miscellaneous.QuickDropEnable=0x01;`	Bit [4	Parameter used for strong signal.
eepromWrite EnableGPIO	1	`EepromWriteEnableGpio=0x06;`	Specify which GPIO is used to write protect EEPROM (most AR93xx/AR94xx/AR95xx reference designs set this to GPIO6)	
Reserved	1	—	Reserved for GPIO configuration, should not be set	
wlanLedGPIO	1	`WlanLedGpio=0x08;`	Specify GPIO being used for on-board LED (Default = 8)	
rxBandSelect GPIO	1	—	Reserved rxBandSelect GPIO configuration. Do not set.	
txrxGain	4 bits	`GainTable.Rx=0x00;`	Bits [3:0] specify which Rx gain table to use:	
			0	Using board design with an external LNA
			1	Using a board design with no external LNA
	4 bits	`GainTable.Tx=0x01;`	Bits [7:4] specify which Tx gain table to use:	
			0	For client-based designs containing an XPA
			1	For designs containing no XPA
			2	For AP-based (or modules used in an AP) designs containing an XPA
			3	Not currently used
			4	For mixed gain tables, such as boards with internal PA for 2 GHz and external PA for 5 GHz
swRegulator	4	—	Reserved for internal switching regulator configuration	

Modal EEPROM Header

Figure 2-7 shows the modal EEPROM header board data details.

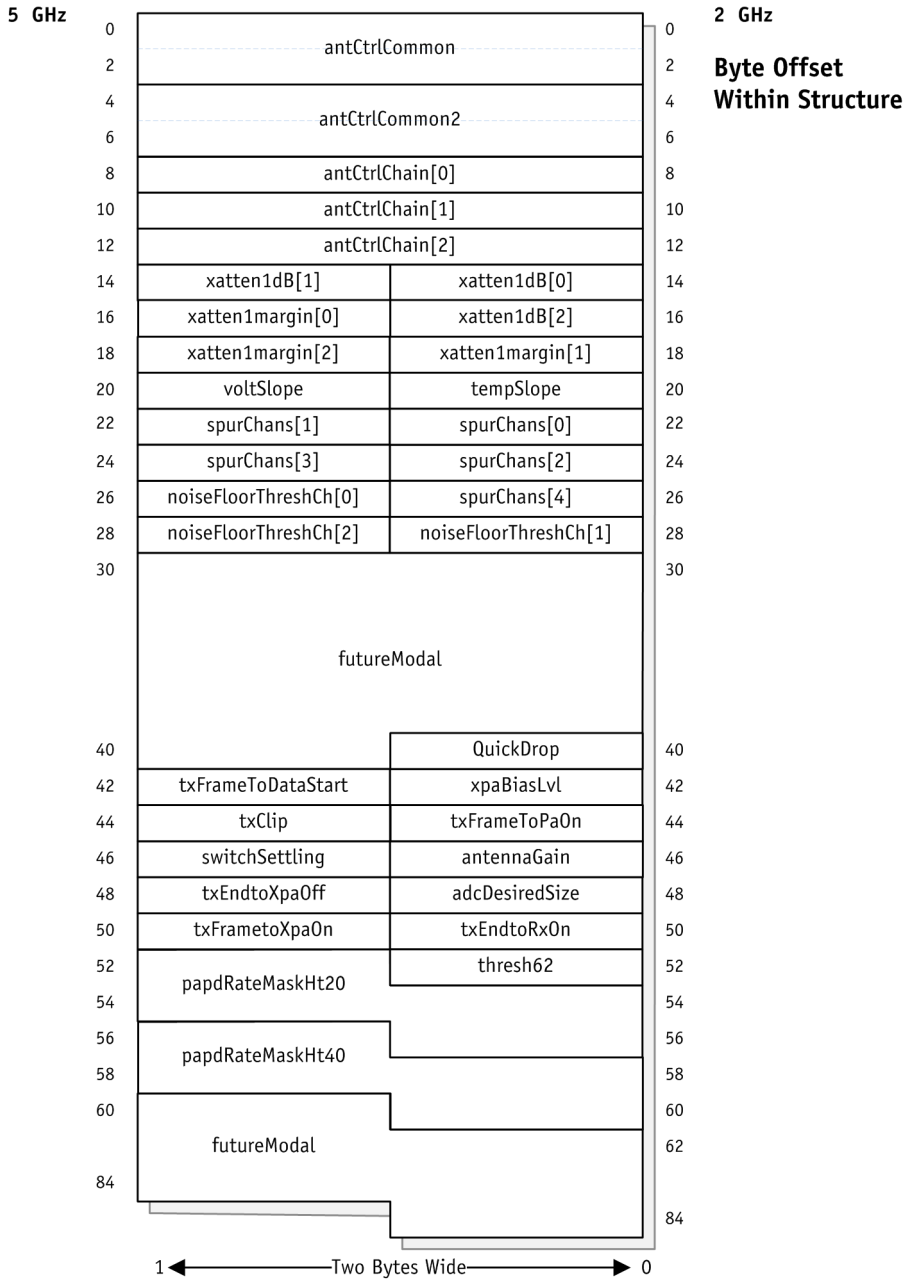

Figure 2-7. **Modal EEPROM Header 2 GHz/5 GHz**

Table 2-9 describes the fields shown in Figure 2-7.

NOTE: In Table 2-9, ART2 set commands should be issued separately to change 2 GHz and 5 GHz parameters, as indicated with the notation [*2*|*5*] within the command name. When entering these commands within ART2, either the 2 or the 5 GHz version of the command should be used. Thus, [*2*|*5*]GHz.AntennaControlChain would be entered as *2*GHz.AntennaControlChain or *5*GHz.AntennaControlChain.

Table 2-9. **Modal EEPROM Header Parameter Descriptions**

Parameter Name	Bytes	get/set Command Parameter	Description[1]
antCtrlCommon	4	`2GHz.AntennaControlCommon=0x00000110;` `5GHz.Antenna.Common=0x00000220;`	Antenna control settings common across chains. A 4-bit setting controlling the output of pins (SW_COM0, SW_COM1, SW_COM2, and SW_COM3) is specified for four possible states. Each word is comprised of a repeated 4-bit setting controlling the output pins (SW_COMX) and specified for idle, Tx, and BT states. Starting at the LSB of the register, these states include: idle, Tx antenna 1, Tx antenna 2, and BT coexistence state. For chips that do not support Tx antenna diversity, set the Tx antenna 1 and antenna 2 to the same value.
antCtrlCommon2	4	`2GHz.AntennaControlCommon2=0x00044444;` `5GHz.Antenna.Common2=0x00044444;`	Antenna control settings common across chains. A 4-bit setting controlling the output of pins (SW_COM0, SW_COM1, SW_COM2, and SW_COM3) is specified for five possible Rx states. Each word is comprised of a repeated 4-bit setting controlling the output pins (SW_COMX) and specified for Rx. Starting at the LSB of the register, these states include: Rx antenna 1 LNA 1, Rx antenna 2 LNA 1, Rx antenna 1 LNA 2, Rx antenna 2 LNA 2, and a combination of Rx antennas 1/2 LNA 1/2. For chips that do not support Rx/LNA antenna diversity, set all five Rx states to the same value.
antCtrlChain[2:0]	12	`2GHz.AntennaControlChain[0,1,2]=0x0010;` `5GHz.Antenna.Chain[0,1,2]=0x0010;`	Antenna control settings for operation on antenna radio chain 0, 1, and 2. This parameter is laid out so that index 0, 1, and 2 correspond to physical chain 0, 1, and 2, respectively. Each word is comprised of a repeated 2-bit setting controlling the output pins (SW_X1 and SW_X0) and specified for six possible Tx/Rx states. Starting at the LSB of the register, these states include: idle, Tx, Rx, Rx with first attenuation addition, Rx with first and second attenuation addition, and BT coexistence state.

Table 2-9. **Modal EEPROM Header Parameter Descriptions (continued)**

Parameter Name	Bytes	get/set Command Parameter	Description[1]
xatten1dB[2:0]	3	`2GHz.Attenuation.Db[0,1,2]=0x18;` `5GHz.Attenuation.Db.Middle[0]=0x1E;`	Specifies the difference in attenuation (in dB) achieved by switching to attenuation Rx stage; should be the attenuation for maximum input reduction provided by the internal or external device controlled by the Rx stage in the switch table. If the switch table does not enable any attenuation for this stage, this value should be 0. Most AR93xx/AR94xx/AR95xx reference devices disable the external LNA in their switch table for attenuation the stage and therefore the value in the EEPROM matches the LNA attenuation when disabled. xatten1margin controls when this stage of attenuation is enabled. xatten1margin and xatten1dB are for frequency 5500–5780.
xatten1margin [2:0]	3	`2GHz.Attenuation.Margin[0,1,2]=0x0B;` `5GHz.Attenuation.Margin.Middle[0,1,2]=0x0B;`	Margin (in dB) that controls when the attenuation stage is enabled. A higher value for xatten1margin means the final attenuation stage enables in at a lower input signal level to attenuate the received signal. The amount of attenuation is specified by the xatten1dB parameter. This parameter is per-chain. xatten1margin and xatten1dB are for frequency 5500–5780.
tempSlope	1	`2GHz.TemperatureSlope=+25;` `5GHz.TemperatureSlope.Middle=+70;`	Characterized temperature compensation slope for 2 GHz and frequency 5500 to 5780 MHz for 5 GHz. This bit sets the 8-digit INT value to the tempSlope of the 2 GHz or 5 GHz field.
voltSlope	1	—	Reserved for voltage compensation.
spurChans[4:0]	5	`[2|5]GHz.Spur[0,1,2,3,4]=ff;`	List of frequency piers requiring application of the spur mitigation algorithm. These bits set freqInMHz values for 5 spur channels to the SpurChans of the 2 or 5 GHz field. 0: 2464 (2 GHz) or 0 (5 GHz) 1, 2, 3, 4: 0
noiseFloorThreshCh[2:0]	3	`[2|5]GHz.NoiseFloorThreshold[0,1,2]=ff;`	A signed 8-bit value in 1-dB steps (that is, a typical value of –85 Noise_Floor_Thresh = –85 dB). The noise floor threshold is written to registers and monitored in software to prevent transmission if the noise floor calibration detects a constant carrier above this threshold. This parameter is per-chain and sets 3 of 8-digit INT values to the NoiseFloorThreshCh of 2 GHz or 5 GHz field for 3 chains. 0: –1 1: +0 2: +0
QuickDrop	1	`2GHz.QuickDrop=-44;` `5GHz.QuickDrop.Middle=-34;`	Gaindrop after detecting saturation in ADC. Used for Rx strong signal tuning. For frequency 5500–5780.
xpaBiasLevel	1	`[2|5]GHz.XpaBiasLevel=0x00;`	Permits a per-solution basis to change the external PA bias level

Table 2-9. **Modal EEPROM Header Parameter Descriptions (continued)**

Parameter Name	Bytes	get/set Command Parameter	Description[1]
txFrameTo DataStart	1	`[2|5]GHz.TxFrameTo DataStart=0x0E;`	Tx frame to data start. Time in 100 ns increments between the Tx frame and the data start. The value must be ≥16 to function correctly. This timing parameter works along with txFrameToPaOn and txFrameToXpaOn to control sequencing and power initialization when transmitting a frame.
txFrameToPaOn	1	`[2|5]GHz.TxFrameTo PaOn=0x0E;`	Time in 100-ns increments between the Tx frame and the internal PA enable. This timing parameter works along with txFrameToDataStart and txFrameToXpaOn to control the sequencing and power initialization when transmitting a frame.
txClip	1	—	Not currently used for AR93xx/AR94xx/AR95xx
antennaGain	1	`[2|5]GHz.Antenna Gain=+0;`	Antenna gain. This antenna gain is added to the calibrated power by the driver to compute the final output power with respect to regulatory domain. Antenna gain is an 8-bit signed quantity in 0.5 dB steps (e.g., +12 Antenna_Gain = +6 dB). This parameter is per-chain.
switchSettling	1	`2GHz.Switch Settling=0x2C;` `5GHz.SwitchSettling= 0x2D;`	Switch settling time. Tx/Rx switch settling time can be set according to the settling time of the external switch. This switch settling time should be the largest of the external Tx/Rx switch, or the largest settling time of any switch controlled by the switch table's attenuation 1 or attenuation 2 settings. The equation to calculate switch settling time register is: (Switch Settling Time Register) = (Switching Settling Time / 25 ns) + 19
adcDesiredSize	1	`[2|5]GHz.AdcSize= -30;`	ADC desired size. Desired amplitude of signal to be presented to the ADC. This signed 8-bit value is used by the automatic gain control stage to output the appropriate signal size to make the best use of ADC range. The value specified is in 0.5 dB steps (e.g, –32 ADC_Desired_Size_11a = –16 dBm).
txEndToXpaOff	1	`[2|5]GHz.TxEndToXpaO ff=0x00;`	Specifies the time difference from when the baseband is finished sending a frame to when the external PA switch is deactivated. Can be adjusted based on the external PA ramp-down time. For example, if the external PA ramp-down time is very fast, it would be desirable to delay deactivating the external PA to ensure that the end of the frame being sent is not prematurely truncated.
txEndToRxOn	1	`[2|5]GHz.TxEndTo RxOn=0x02;`	Specifies the time difference from when the baseband is finished sending a frame to when the external low noise amplifier (LNA) switch is activated. Can be adjusted based on the external LNA ramp-up time. For example, if the external LNA ramp-up time is slow, it would be desirable to turn on the external LNA sooner so that the beginning of the Rx frame is not missed.

Table 2-9. **Modal EEPROM Header Parameter Descriptions (continued)**

Parameter Name	Bytes	get/set Command Parameter	Description[1]
txFrameToXpaOn	1	`[2|5]GHz.TxFrameToXpaOn=0x0E;`	Specifies the time difference in 100ns increments from when the MAC sends the frame to when the external power amplifier switch is activated. Can be adjusted based on the external PA ramp-up time. E.g., if the external PA ramp-up time is fast, it would be desirable to activate the external PA sooner so that the beginning of the frame being sent is not prematurely truncated.
thresh62	1	`[2|5]GHz.Thresh62=0x1C;`	Adjusts CCA sensitivity to meet IEEE 802.11 specifications. Section 17.3.10.5 specifies CCA sensitivity as "A start of a valid OFDM transmission at Rx level equal or greater than minimum 6 Mbps sensitivity (–82 dBm) shall cause CCA to indicate busy with probability > 90% within 4 µs. If the preamble portion of a frame was missed, the receiver shall hold the CS signal Busy for any signal 20 dB above minimum 6 Mbps sensitivity (–62 dBm)". A lower threshold can be chosen for better performance in the presence of collisions by changing the setting in this register.
papdRateMaskHt20	4	`2GHz.PaPredistortion.Ht20=0x0c80c080;` `5GHz.PaPredistortion.Ht20=0x0cf0e0e0;`	Bitmask of HT20 rates that should use PA predistortion. Bits [23:0] are used to enable for rates MCS0...MCS23 respectively.
papdRateMaskHt40	4	`2GHz.PaPredistortion.Ht40=0x0080c080;` `5GHz.PaPredistortion.Ht40=0x6cf0e0e0;`	Bitmask of HT40 rates that should use PA predistortion. Bits [23:0] are used to enable for rates MCS0...MCS23 respectively.
futureModal	10	—	Reserved for future modal

[1] For this table, SW_*x*0 (where *x* is the chain) maps to the XLNABIAS pin in the AR938x/AR958x chips.

Base Extension

Figure 2-8 shows the base extension 1 details.

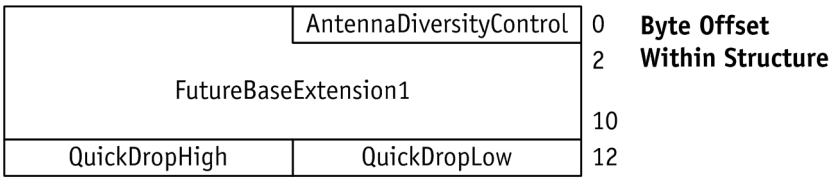

Figure 2-8. **Base Extension 1**

Table 2-10 describes the base extension 1 field.

Table 2-10. **Base Extension 1 Parameter Descriptions**

Parameter Name	Bytes	get/set Command Parameter	Description
AntennaDiversity Control	1	`AntennaDiversity Control=0x00;`	Reserved for future chip sets. Set to 0.
FutureBaseExtension1	11	—	Reserved for future use
QuickDropLow	1	`5GHz.QuickDrop.Low =-34;`	Gaindrop after detecting saturation in ADC. Used for Rx strong signal tuning. For frequency 5180–5490.
QuickDropHigh	1	`5GHz.QuickDrop.High =-34;`	Gaindrop after detecting saturation in ADC. Used for Rx strong signal tuning. For frequency 5785 and up.

Figure 2-9 shows the base extension 2 details.

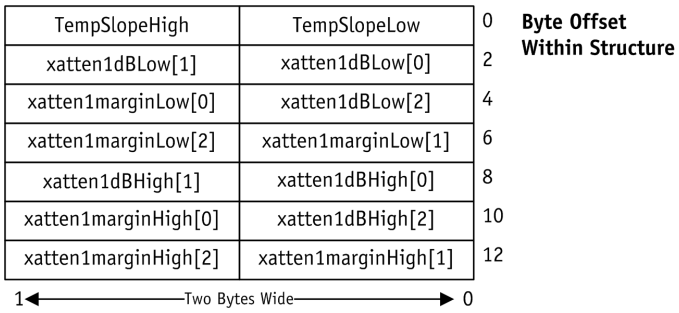

Figure 2-9. **Base Extension 2**

Table 2-11 describes the base extension 2 field.

Table 2-11. **Base Extension 2 Parameter Descriptions**

Parameter Name	Bytes	get/set Command Parameter	Description
TempSlopeLow	1	`5GHz.TemperatureSlope .Low=+35;`	Temperature slope for frequency 5180–5490.
TempSlopeHigh	1	`5GHz.TemperatureSlope .High=+50;`	Temperature slope for frequency 5785 and up.
xatten1dBLow[2:0]	1	`5GHz.Attenuation.Db. Low[0,1,2]=0x1B;`	Specifies the difference in attenuation (in dB) achieved by switching to attenuation Rx stage. For frequency 5180–5490.
xatten1marginLow [2:0]	1	`5GHz.Attenuation.Marg in.Low[0,1,2]=0x10;`	Margin (in dB) that controls when the attenuation stage is enabled. For frequency 5180–5490.
xatten1dBHigh[2:0]	1	`5GHz.Attenuation.Db. High[0,1,2]=0x1E;`	Specifies the difference in attenuation (in dB) achieved by switching to attenuation Rx stage For frequency 5785 and up.
xatten1marginHigh [2:0]	1	`5GHz.Attenuation.Marg in.High[0,1,2]=0x0B;`	Margin (in dB) that controls when the attenuation stage is enabled. For frequency 5785 and up.

Power Calibration Channels

Table 2-12 describes the power calibration channels field.

Table 2-12. **Power Calibration Channels 2 GHz/5 GHz Parameter Descriptions**

Parameter Name	Bytes	get/set Command Parameter	Description	
Power Calibration Channels	3 (2 GHz) 8 (5 GHz)	`2GHz.PowerCalibration. Frequency[0,1,2]=ffff;`	Frequency piers at which calibration should be performed	
			0	2412
			1	2437
			2	2432
		`5GHz.TransmitCalibration .Frequency [0,1,2,3,4,5,6,7]=ffff;`	0	5180
			1	5220
			2	5340
			3	5420
			4	5500
			5	5620
			6	5745
			7	5825

Power Calibration Data

Figure 2-10 shows the power calibration data board data details.

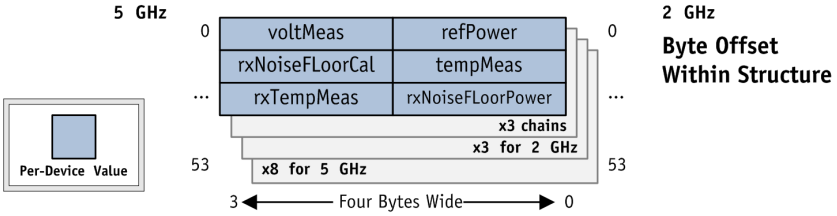

Figure 2-10. **Power Calibration Data 2 GHz/5 GHz**

Table 2-13 describes the fields shown in Figure 2-10.

Table 2-13. **Power Calibration Data 2 GHz/5 GHz Parameter Descriptions**

Parameter Name	Bytes	ART Command	Description
refPower	1	`2GHz.TransmitCalibration.PowerCorrection[x][y]=ffff;` `5GHz.TransmitCalibration.PowerCorrection[x][y]=ffff;`	Calibrated reference power calculation based on power measured and the Txgain index used in calibration; *x* is the chain and *y* is the frequency pier (see Table 2-12).
voltMeas	1	`[2/5]GHz.TransmitCalibration.Voltage[x][y]=0;`	Reserved for voltage compensation; *x* (0–2) is the chain and *y* (0–2 for 2 GHz or 0–7 for 5 GHz) is the frequency pier (see Table 2-12).
tempMeas	1	`2GHz.TransmitCalibration.Temperature[x][y]=ffff;` `5GHz.TransmitCalibration.Temperature[x][y]=ffff;`	Temperature as read back from the chip registers at Tx calibration
rxNoiseFloorCal	1	`[2/5]GHz.ReceiveCalibration.NoiseFloor[x][y]=+0;`	Noise floor calculated by the chip at Rx calibration; *x* (0–2) is the chain and *y* (0–2 for 2 GHz or 0–7 for 5 GHz) is the frequency pier (see Table 2-12).
rxNoiseFloor Power	1	`[2/5].ReceiveCalibration.Power[x][y]=+0;`	Calculated Tx power from known input signal strength (ISS) and RSSI from the chip at Rx calibration; *x* (0–2) is the chain and *y* (0–2 for 2 GHz or 0–7 for 5 GHz) is the frequency pier (see Table 2-12).
rxTempMeas	1	`[2/5]GHz.ReceiveCalibration.Temperature[x][y]=0;`	Temperature as read back from the chip registers at Rx calibration; *x* (0–2) is the chain and *y* (0–2 for 2 GHz or 0–7 for 5 GHz) is the frequency pier (see Table 2-12).

Target Power Channels

Figure 2-11 shows the target power channels 2 GHz board data details.

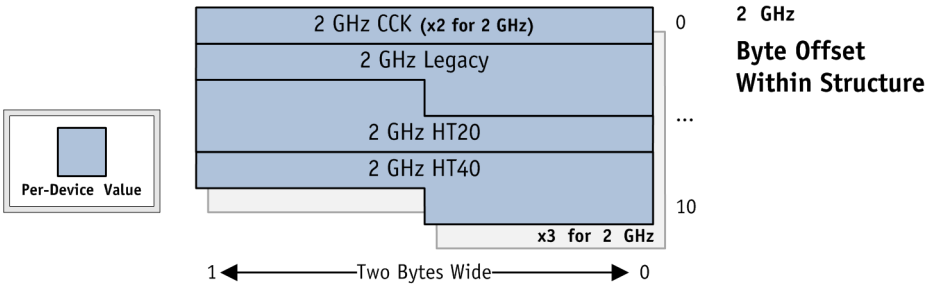

Figure 2-11. **Target Power Channels 2 GHz**

Table 2-14 describes the 2 GHz fields shown in Figure 2-11.

Table 2-14. **Target Power Channels 2 GHz Parameter Descriptions**

Parameter Name	Bytes	get/set Command Parameter	Description	
2 GHz CCK	2	`2GHz.Target.Frequency.Cck[0,1]=ffff`	CCK target power frequency piers	
			0	2412
			1	2472
2 GHz Legacy	3	`2GHz.Target.Frequency.Legacy[0,1,2]=ffff;`	Legacy 2 GHz target power frequency piers	
			0	2412
			1	2437
			2	2472
2 GHz HT20	3	`2GHz.Target.Frequency.Ht20[0,1,2]=ffff;`	2 GHz HT20 target power frequency piers	
			0	2412
			1	2437
			2	2472
2 GHz HT40	3	`2GHz.Target.Frequency.Ht40[0,1,2]=ffff;`	2 GHz HT40 target power frequency piers	
			0	2412
			1	2437
			2	2472

Figure 2-11 shows the target power channels 5 GHz board data details.

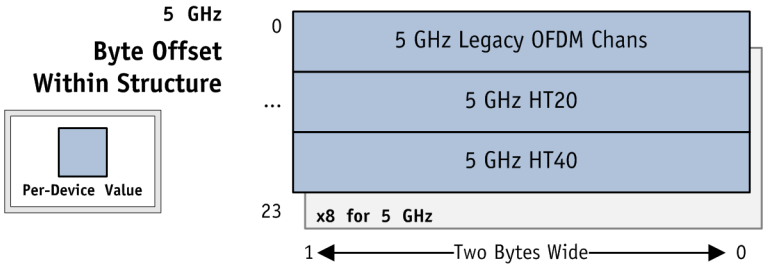

Figure 2-12. **Target Power Channels 5 GHz**

Table 2-15 describes the 5 GHz fields shown in Figure 2-12.

Table 2-15. **Target Power Channels 5 GHz Parameter Descriptions**

Parameter Name	Bytes	get/set Command Parameter	Description	
5 GHz Legacy OFDM	8	`5GHz.Target.Frequency.Legacy[0,1,2,3,4,5,6,7]=ffff;`	Legacy 5 GHz target power frequency piers	
			0	5180
			1	5220
			2	5320
			3	5400
			4	5500
			5	5600
			6	5725
			7	5825
5 GHz HT20	8	`5GHz.Target.Frequency.Ht20[0,1,2,3,4,5,6,7]=ffff;`	HT20 5 GHz target power frequency piers	
			0	5180
			1	5220
			2	5320
			3	5400
			4	5500
			5	5600
			6	5725
			7	5825
5 GHz HT40	8	`5GHz.Target.Frequency.Ht40[0,1,2,3,4,5,6,7]=ffff;`	HT40 5 GHz target power frequency piers	
			0	5180
			1	5220
			2	5320
			3	5400
			4	5500
			5	5600
			6	5725
			7	5825

Target Powers 802.11bg CCK

Figure 2-13 details the target powers 802.11bg CCK board data.

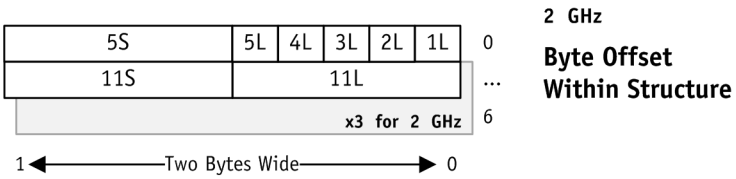

Figure 2-13. **Target Powers 802.11bg CCK**

Table 2-16 describes the fields shown in Figure 2-13.

The bits in Table 2-16 define multiple commands `2GHz.Target.Power.Cck[x][y]`, where *x* is the frequency (0–2 for 2 GHz, 0–7 for 5 GHz) and *y* is the rate (see the `2GHz.Target.Frequency` parameters in Table 2-14)..

NOTE: All target power values are stored in the calibration structure as 2x the target power. Target power values are entered in the ART2 command the power in dBm, with 0.5 dBm accuracy.

Table 2-16. **Target Powers 802.11bg CCK Parameter Descriptions**

Parameter Name	Bytes	get/set Command Parameter	Description
1L–5L	1	`2GHz.Target.Power.Cck[0][0]`	Target power for 1L–5L for frequency [0]
		`2GHz.Target.Power.Cck[1][0]`	Target power for 1L–5L for frequency [1]
5S	1	`2GHz.Target.Power.Cck[0][1]`	Target power for 5S for frequency [0]
		`2GHz.Target.Power.Cck[1][1]`	Target power for 5S for frequency [1]
11L	1	`2GHz.Target.Power.Cck[0][2]`	Target power for 11L for frequency [0]
		`2GHz.Target.Power.Cck[1][2]`	Target power for 11L for frequency [1]
11S	1	`2GHz.Target.Power.Cck[0][3]`	Target power for 11S for frequency [0]
		`2GHz.Target.Power.Cck[1][3]`	Target power for 11S for frequency [1]

Target Powers 802.11g OFDM

Figure 2-14 shows the target powers 802.11g OFDM board data details.

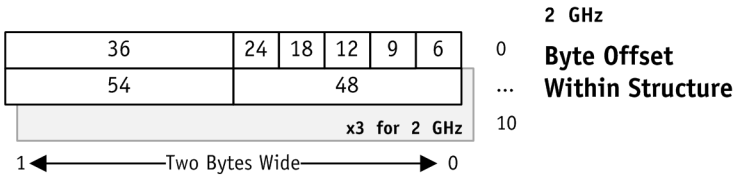

Figure 2-14. **Target Powers 802.11g OFDM**

Table 2-17 describes the fields shown in Figure 2-14.

The bits in Table 2-17 define multiple commands 2GHz.Target.Power.Legacy[*x*][*y*], where *x* is the frequency (0–2) and *y* is the rate (0–3) (see the `2GHz.Target.Frequency` parameters in Table 2-14).

Table 2-17. **Target Powers 802.11g OFDM Parameter Descriptions**

Parameter Name	Bytes	get/set Command Parameter	Description
6–24	1	`2GHz.Target.Power.Legacy[0][0]`	Target power for 2 GHz legacy rates 6, 9, 12, 18, and 24 Mbps for frequency [0]
		`2GHz.Target.Power.Legacy[1][0]`	Target power for 2 GHz legacy rates 6, 9, 12, 18, and 24 Mbps for frequency [1]
		`2GHz.Target.Power.Legacy[2][0]`	Target power for 2 GHz legacy rates 6, 9, 12, 18, and 24 Mbps for frequency [2]
36	1	`2GHz.Target.Power.Legacy[0][1]`	Target power for 2 GHz legacy rate 36 Mbps for frequency [0]
		`2GHz.Target.Power.Legacy[1][1]`	Target power for 2 GHz legacy rate 36 Mbps for frequency [1]
		`2GHz.Target.Power.Legacy[2][1]`	Target power for 2 GHz legacy rate 36 Mbps for frequency [2]
48	1	`2GHz.Target.Power.Legacy[0][2]`	Target power for 2 GHz legacy rate 48 Mbps for frequency [0]
		`2GHz.Target.Power.Legacy[1][2]`	Target power for 2 GHz legacy rate 48 Mbps for frequency [1]
		`2GHz.Target.Power.Legacy[2][2]`	Target power for 2 GHz legacy rate 48 Mbps for frequency [2]
54	1	`2GHz.Target.Power.Legacy[0][3]`	Target power for 2 GHz legacy rate 54 Mbps for frequency [0]
		`2GHz.Target.Power.Legacy[1][3]`	Target power for 2 GHz legacy rate 54 Mbps for frequency [1]
		`2GHz.Target.Power.Legacy[2][3]`	Target power for 2 GHz legacy rate 54 Mbps for frequency [2]

Target Powers 802.11n HT20 OFDM

Figure 2-15 details the target powers 802.11n HT20 OFDM board data.

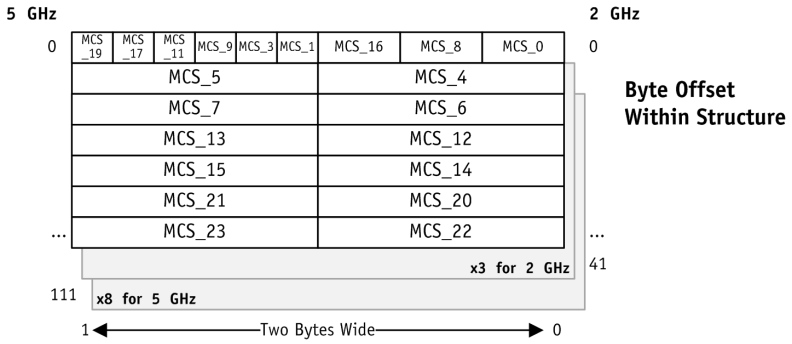

Figure 2-15. **Target Powers 802.11n HT20 OFDM**

Table 2-18 describes the fields shown in Figure 2-15.

These bits define multiple [2 | 5]GHz.Target.Power.Ht20[*x*][*y*] commands, where *x* is the channel (0–2 for 2 GHz, 0–7 for 5 GHz) and *y* is the rate (0–13) (see the `2GHz.Target.Frequency` parameters in Table 2-14 for 2 GHz, see the `5GHz.Target.Frequency` parameters in Table 2-15 for 5 GHz).

Table 2-18. **Target Powers 802.11n HT20 OFDM Parameter Descriptions**

Parameter Name	Bytes	get/set Command Parameter	Description
MCS_0, MCS_8, MCS_16	1	`2GHz.Target.Power.Ht20[0][0]` `5GHz.Target.Power.Ht20[0][0]`	2 | 5 GHz HT20 rates MCS 0, 8, and 16 must be set to the same power value
		`2GHz.Target.Power.Ht20[1][0]` `5GHz.Target.Power.Ht20[1][0]`	
		`2GHz.Target.Power.Ht20[2][0]` `5GHz.Target.Power.Ht20[2][0]`	
		`5GHz.Target.Power.Ht20[3][0]`	
		`5GHz.Target.Power.Ht20[4][0]`	
		`5GHz.Target.Power.Ht20[5][0]`	
		`5GHz.Target.Power.Ht20[6][0]`	
		`5GHz.Target.Power.Ht20[7][0]`	
MCS_1,MCS_2, MCS_3, MCS_9, MCS_10, MCS_11, MCS_17, MCS_18, MCS_19	1	`2GHz.Target.Power.Ht20[0][1]` `5GHz.Target.Power.Ht20[0][1]`	2 | 5 GHz HT20 rates MCS 1, 2, 3, 9, 10, 11, 17, 18, 19 must be set to the same power value
		`2GHz.Target.Power.Ht20[1][1]` `5GHz.Target.Power.Ht20[1][1]`	
		`2GHz.Target.Power.Ht20[2][1]` `5GHz.Target.Power.Ht20[2][1]`	
		`5GHz.Target.Power.Ht20[3][1]`	
		`5GHz.Target.Power.Ht20[4][1]`	
		`5GHz.Target.Power.Ht20[5][1]`	
		`5GHz.Target.Power.Ht20[6][1]`	
		`5GHz.Target.Power.Ht20[7][1]`	

Table 2-18. **Target Powers 802.11n HT20 OFDM Parameter Descriptions (continued)**

Parameter Name	Bytes	get/set Command Parameter	Description
MCS_4	1	2GHz.Target.Power.Ht20[0][2] 5GHz.Target.Power.Ht20[0][2]	Target power value for 2|5 GHz HT20 MCS 4
		2GHz.Target.Power.Ht20[1][2] 5GHz.Target.Power.Ht20[1][2]	
		2GHz.Target.Power.Ht20[2][2] 5GHz.Target.Power.Ht20[2][2]	
		5GHz.Target.Power.Ht20[3][2]	
		5GHz.Target.Power.Ht20[4][2]	
		5GHz.Target.Power.Ht20[5][2]	
		5GHz.Target.Power.Ht20[6][2]	
		5GHz.Target.Power.Ht20[7][2]	
MCS_5	1	2GHz.Target.Power.Ht20[0][3] 5GHz.Target.Power.Ht20[0][3]	Target power value for 2|5 GHz HT20 MCS 5
		2GHz.Target.Power.Ht20[1][3] 5GHz.Target.Power.Ht20[1][3]	
		2GHz.Target.Power.Ht20[2][3] 5GHz.Target.Power.Ht20[2][3]	
		5GHz.Target.Power.Ht20[3][3]	
		5GHz.Target.Power.Ht20[4][3]	
		5GHz.Target.Power.Ht20[5][3]	
		5GHz.Target.Power.Ht20[6][3]	
		5GHz.Target.Power.Ht20[7][3]	
MCS_6	1	2GHz.Target.Power.Ht20[0][4] 5GHz.Target.Power.Ht20[0][4]	Target power value for 2|5 GHz HT20 MCS 6
		2GHz.Target.Power.Ht20[1][4] 5GHz.Target.Power.Ht20[1][4]	
		2GHz.Target.Power.Ht20[2][4] 5GHz.Target.Power.Ht20[2][4]	
		5GHz.Target.Power.Ht20[3][4]	
		5GHz.Target.Power.Ht20[4][4]	
		5GHz.Target.Power.Ht20[5][4]	
		5GHz.Target.Power.Ht20[6][4]	
		5GHz.Target.Power.Ht20[7][4]	
MCS_7	1	2GHz.Target.Power.Ht20[0][5] 5GHz.Target.Power.Ht20[0][5]	Target power value for 2|5 GHz HT20 MCS 7
		2GHz.Target.Power.Ht20[1][5] 5GHz.Target.Power.Ht20[1][5]	
		2GHz.Target.Power.Ht20[2][5] 5GHz.Target.Power.Ht20[2][5]	
		5GHz.Target.Power.Ht20[3][5]	
		5GHz.Target.Power.Ht20[4][5]	
		5GHz.Target.Power.Ht20[5][5]	
		5GHz.Target.Power.Ht20[6][5]	
		5GHz.Target.Power.Ht20[7][5]	

Table 2-18. **Target Powers 802.11n HT20 OFDM Parameter Descriptions (continued)**

Parameter Name	Bytes	get/set Command Parameter	Description
MCS_12	1	2GHz.Target.Power.Ht20[0][6] 5GHz.Target.Power.Ht20[0][6]	Target power value for 2|5 GHz HT20 MCS 12
		2GHz.Target.Power.Ht20[1][6] 5GHz.Target.Power.Ht20[1][6]	
		2GHz.Target.Power.Ht20[2][6] 5GHz.Target.Power.Ht20[2][6]	
		5GHz.Target.Power.Ht20[3][6]	
		5GHz.Target.Power.Ht20[4][6]	
		5GHz.Target.Power.Ht20[5][6]	
		5GHz.Target.Power.Ht20[6][6]	
		5GHz.Target.Power.Ht20[7][6]	
MCS_13	1	2GHz.Target.Power.Ht20[0][7] 5GHz.Target.Power.Ht20[0][7]	Target power value for 2|5 GHz HT20 MCS 13
		2GHz.Target.Power.Ht20[1][7] 5GHz.Target.Power.Ht20[1][7]	
		2GHz.Target.Power.Ht20[2][7] 5GHz.Target.Power.Ht20[2][7]	
		5GHz.Target.Power.Ht20[3][7]	
		5GHz.Target.Power.Ht20[4][7]	
		5GHz.Target.Power.Ht20[5][7]	
		5GHz.Target.Power.Ht20[6][7]	
		5GHz.Target.Power.Ht20[7][7]	
MCS_14	1	2GHz.Target.Power.Ht20[0][8] 5GHz.Target.Power.Ht20[0][8]	Target power value for 2|5 GHz HT20 MCS 14
		2GHz.Target.Power.Ht20[1][8] 5GHz.Target.Power.Ht20[1][8]	
		2GHz.Target.Power.Ht20[2][8] 5GHz.Target.Power.Ht20[2][8]	
		5GHz.Target.Power.Ht20[3][8]	
		5GHz.Target.Power.Ht20[4][8]	
		5GHz.Target.Power.Ht20[5][8]	
		5GHz.Target.Power.Ht20[6][8]	
		5GHz.Target.Power.Ht20[7][8]	
MCS_15	1	2GHz.Target.Power.Ht20[0][9] 5GHz.Target.Power.Ht20[0][9]	Target power value for 2|5 GHz HT20 MCS 15
		2GHz.Target.Power.Ht20[1][9] 5GHz.Target.Power.Ht20[1][9]	
		2GHz.Target.Power.Ht20[2][9] 5GHz.Target.Power.Ht20[2][9]	
		5GHz.Target.Power.Ht20[3][9]	
		5GHz.Target.Power.Ht20[4][9]	
		5GHz.Target.Power.Ht20[5][9]	
		5GHz.Target.Power.Ht20[6][9]	
		5GHz.Target.Power.Ht20[7][9]	

Table 2-18. **Target Powers 802.11n HT20 OFDM Parameter Descriptions (continued)**

Parameter Name	Bytes	get/set Command Parameter	Description
MCS_20	1	2GHz.Target.Power.Ht20[0][10] 5GHz.Target.Power.Ht20[0][10]	Target power value for 2|5 GHz HT20 MCS 20
		2GHz.Target.Power.Ht20[1][10] 5GHz.Target.Power.Ht20[1][10]	
		2GHz.Target.Power.Ht20[2][10] 5GHz.Target.Power.Ht20[2][10]	
		5GHz.Target.Power.Ht20[3][10]	
		5GHz.Target.Power.Ht20[4][10]	
		5GHz.Target.Power.Ht20[5][10]	
		5GHz.Target.Power.Ht20[6][10]	
		5GHz.Target.Power.Ht20[7][10]	
MCS_21	1	2GHz.Target.Power.Ht20[0][11] 5GHz.Target.Power.Ht20[0][11]	Target power value for 2|5 GHz HT20 MCS 21
		2GHz.Target.Power.Ht20[1][11] 5GHz.Target.Power.Ht20[1][11]	
		2GHz.Target.Power.Ht20[2][11] 5GHz.Target.Power.Ht20[2][11]	
		5GHz.Target.Power.Ht20[3][11]	
		5GHz.Target.Power.Ht20[4][11]	
		5GHz.Target.Power.Ht20[5][11]	
		5GHz.Target.Power.Ht20[6][11]	
		5GHz.Target.Power.Ht20[7][11]	
MCS_22	1	2GHz.Target.Power.Ht20[0][12] 5GHz.Target.Power.Ht20[0][12]	Target power value for 2|5 GHz HT20 MCS 22
		2GHz.Target.Power.Ht20[1][12] 5GHz.Target.Power.Ht20[1][12]	
		2GHz.Target.Power.Ht20[2][12] 5GHz.Target.Power.Ht20[2][12]	
		5GHz.Target.Power.Ht20[3][12]	
		5GHz.Target.Power.Ht20[4][12]	
		5GHz.Target.Power.Ht20[5][12]	
		5GHz.Target.Power.Ht20[6][12]	
		5GHz.Target.Power.Ht20[7][12]	
MCS_23	1	2GHz.Target.Power.Ht20[0][13] 5GHz.Target.Power.Ht20[0][13]	Target power value for 2|5 GHz HT20 MCS 23
		2GHz.Target.Power.Ht20[1][13] 5GHz.Target.Power.Ht20[1][13]	
		2GHz.Target.Power.Ht20[2][13] 5GHz.Target.Power.Ht20[2][13]	
		5GHz.Target.Power.Ht20[3][13]	
		5GHz.Target.Power.Ht20[4][13]	
		5GHz.Target.Power.Ht20[5][13]	
		5GHz.Target.Power.Ht20[6][13]	
		5GHz.Target.Power.Ht20[7][13]	

Target Powers 802.11n HT40 OFDM

Figure 2-16 details the target powers 802.11n HT40 OFDM board data.

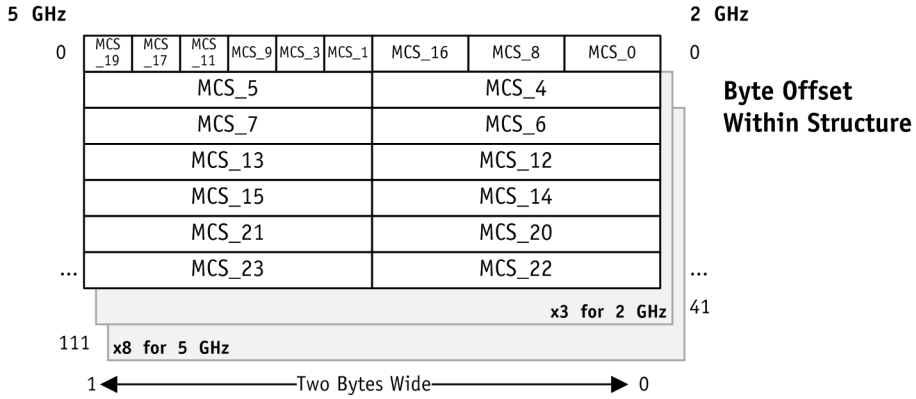

Figure 2-16. **Target Powers 802.11n HT40 OFDM**

Table 2-19 describes the fields shown in Figure 2-16.

These bits define multiple [2 | 5]GHz.TargetPower.Ht40[*x*][*y*] commands, where *x* is the channel (0–2 for 2 GHz, 0–7 for 5 GHz) and *y* is the rate (0–13) (see the `2GHz.Target.Frequency` parameters in Table 2-14 for 2 GHz, see the `5GHz.Target.Frequency` parameters in Table 2-15 for 5 GHz).

Table 2-19. **Target Powers 802.11n HT40 OFDM Parameter Descriptions**

Parameter Name	Bytes	get/set Command Parameter	Description
MCS_0, MCS_8, MCS_16	1	`2GHz.Target.Power.Ht40[0][0]` `5GHz.Target.Power.Ht40[0][0]`	2 | 5 GHz HT40 rates MCS 0, 8, 16 must be set to the same power value
		`2GHz.Target.Power.Ht40[1][0]` `5GHz.Target.Power.Ht40[1][0]`	
		`2GHz.Target.Power.Ht40[2][0]` `5GHz.Target.Power.Ht40[2][0]`	
		`5GHz.Target.Power.Ht40[3][0]`	
		`5GHz.Target.Power.Ht40[4][0]`	
		`5GHz.Target.Power.Ht40[5][0]`	
		`5GHz.Target.Power.Ht40[6][0]`	
		`5GHz.Target.Power.Ht40[7][0]`	
MCS_1, MCS_2, MCS_3, MCS_9, MCS_10, MCS_11, MCS_17, MCS_18, MCS_19	1	`2GHz.Target.Power.Ht40[0][1]` `5GHz.Target.Power.Ht40[0][1]`	2 | 5 GHz HT40 rates MCS 1, 2, 3, 9, 10, 11, 17, 18, 19 must be set to the same power value
		`2GHz.Target.Power.Ht40[1][1]` `5GHz.Target.Power.Ht40[1][1]`	
		`2GHz.Target.Power.Ht40[2][1]` `5GHz.Target.Power.Ht40[2][1]`	
		`5GHz.Target.Power.Ht40[3][1]`	
		`5GHz.Target.Power.Ht40[4][1]`	
		`5GHz.Target.Power.Ht40[5][1]`	
		`5GHz.Target.Power.Ht40[6][1]`	
		`5GHz.Target.Power.Ht40[7][1]`	

Table 2-19. **Target Powers 802.11n HT40 OFDM Parameter Descriptions (continued)**

Parameter Name	Bytes	get/set Command Parameter	Description
MCS_4	1	2GHz.Target.Power.Ht40[0][2] 5GHz.Target.Power.Ht40[0][2]	Target power value for 2|5 GHz HT40 MCS 4
		2GHz.Target.Power.Ht40[1][2] 5GHz.Target.Power.Ht40[1][2]	
		2GHz.Target.Power.Ht40[2][2] 5GHz.Target.Power.Ht40[2][2]	
		5GHz.Target.Power.Ht40[3][2]	
		5GHz.Target.Power.Ht40[4][2]	
		5GHz.Target.Power.Ht40[5][2]	
		5GHz.Target.Power.Ht40[6][2]	
		5GHz.Target.Power.Ht40[7][2]	
MCS_5	1	2GHz.Target.Power.Ht40[0][3] 5GHz.Target.Power.Ht40[0][3]	Target power value for 2|5 GHz HT40 MCS 5
		2GHz.Target.Power.Ht40[1][3] 5GHz.Target.Power.Ht40[1][3]	
		2GHz.Target.Power.Ht40[2][3] 5GHz.Target.Power.Ht40[2][3]	
		5GHz.Target.Power.Ht40[3][3]	
		5GHz.Target.Power.Ht40[4][3]	
		5GHz.Target.Power.Ht40[5][3]	
		5GHz.Target.Power.Ht40[6][3]	
		5GHz.Target.Power.Ht40[7][3]	
MCS_6	1	2GHz.Target.Power.Ht40[0][4] 5GHz.Target.Power.Ht40[0][4]	Target power value for 2|5 GHz HT40 MCS 6
		2GHz.Target.Power.Ht40[1][4] 5GHz.Target.Power.Ht40[1][4]	
		2GHz.Target.Power.Ht40[2][4] 5GHz.Target.Power.Ht40[2][4]	
		5GHz.Target.Power.Ht40[3][4]	
		5GHz.Target.Power.Ht40[4][4]	
		5GHz.Target.Power.Ht40[5][4]	
		5GHz.Target.Power.Ht40[6][4]	
		5GHz.Target.Power.Ht40[7][4]	
MCS_7	1	2GHz.Target.Power.Ht40[0][5] 5GHz.Target.Power.Ht40[0][5]	Target power value for 2|5 GHz HT40 MCS 7
		2GHz.Target.Power.Ht40[1][5] 5GHz.Target.Power.Ht40[1][5]	
		2GHz.Target.Power.Ht40[2][5] 5GHz.Target.Power.Ht40[2][5]	
		5GHz.Target.Power.Ht40[3][5]	
		5GHz.Target.Power.Ht40[4][5]	
		5GHz.Target.Power.Ht40[5][5]	
		5GHz.Target.Power.Ht40[6][5]	
		5GHz.Target.Power.Ht40[7][5]	

Table 2-19. **Target Powers 802.11n HT40 OFDM Parameter Descriptions (continued)**

Parameter Name	Bytes	get/set Command Parameter	Description
MCS_12	1	2GHz.Target.Power.Ht40[0][6] 5GHz.Target.Power.Ht40[0][6]	Target power value for 2|5 GHz HT40 MCS 12
		2GHz.Target.Power.Ht40[1][6] 5GHz.Target.Power.Ht40[1][6]	
		2GHz.Target.Power.Ht40[2][6] 5GHz.Target.Power.Ht40[2][6]	
		5GHz.Target.Power.Ht40[3][6]	
		5GHz.Target.Power.Ht40[4][6]	
		5GHz.Target.Power.Ht40[5][6]	
		5GHz.Target.Power.Ht40[6][6]	
		5GHz.Target.Power.Ht40[7][6]	
MCS_13	1	2GHz.Target.Power.Ht40[0][7] 5GHz.Target.Power.Ht40[0][7]	Target power value for 2|5 GHz HT40 MCS 13
		2GHz.Target.Power.Ht40[1][7] 5GHz.Target.Power.Ht40[1][7]	
		2GHz.Target.Power.Ht40[2][7] 5GHz.Target.Power.Ht40[2][7]	
		5GHz.Target.Power.Ht40[3][7]	
		5GHz.Target.Power.Ht40[4][7]	
		5GHz.Target.Power.Ht40[5][7]	
		5GHz.Target.Power.Ht40[6][7]	
		5GHz.Target.Power.Ht40[7][7]	
MCS_14	1	2GHz.Target.Power.Ht40[0][8] 5GHz.Target.Power.Ht40[0][8]	Target power value for 2|5 GHz HT40 MCS 14
		2GHz.Target.Power.Ht40[1][8] 5GHz.Target.Power.Ht40[1][8]	
		2GHz.Target.Power.Ht40[2][8] 5GHz.Target.Power.Ht40[2][8]	
		5GHz.Target.Power.Ht40[3][8]	
		5GHz.Target.Power.Ht40[4][8]	
		5GHz.Target.Power.Ht40[5][8]	
		5GHz.Target.Power.Ht40[6][8]	
		5GHz.Target.Power.Ht40[7][8]	
MCS_15	1	2GHz.Target.Power.Ht40[0][9] 5GHz.Target.Power.Ht40[0][9]	Target power value for 2|5 GHz HT40 MCS 15
		2GHz.Target.Power.Ht40[1][9] 5GHz.Target.Power.Ht40[1][9]	
		2GHz.Target.Power.Ht40[2][9] 5GHz.Target.Power.Ht40[2][9]	
		5GHz.Target.Power.Ht40[3][9]	
		5GHz.Target.Power.Ht40[4][9]	
		5GHz.Target.Power.Ht40[5][9]	
		5GHz.Target.Power.Ht40[6][9]	
		5GHz.Target.Power.Ht40[7][9]	

Table 2-19. **Target Powers 802.11n HT40 OFDM Parameter Descriptions (continued)**

Parameter Name	Bytes	get/set Command Parameter	Description
MCS_20	1	2GHz.Target.Power.Ht40[0][10] 5GHz.Target.Power.Ht40[0][10]	Target power value for 2|5 GHz HT40 MCS 20
		2GHz.Target.Power.Ht40[1][10] 5GHz.Target.Power.Ht40[1][10]	
		2GHz.Target.Power.Ht40[2][10] 5GHz.Target.Power.Ht40[2][10]	
		5GHz.Target.Power.Ht40[3][10]	
		5GHz.Target.Power.Ht40[4][10]	
		5GHz.Target.Power.Ht40[5][10]	
		5GHz.Target.Power.Ht40[6][10]	
		5GHz.Target.Power.Ht40[7][10]	
MCS_21	1	2GHz.Target.Power.Ht40[0][11] 5GHz.Target.Power.Ht40[0][11]	Target power value for 2|5 GHz HT40 MCS 21
		2GHz.Target.Power.Ht40[1][11] 5GHz.Target.Power.Ht40[1][11]	
		2GHz.Target.Power.Ht40[2][11] 5GHz.Target.Power.Ht40[2][11]	
		5GHz.Target.Power.Ht40[3][11]	
		5GHz.Target.Power.Ht40[4][11]	
		5GHz.Target.Power.Ht40[5][11]	
		5GHz.Target.Power.Ht40[6][11]	
		5GHz.Target.Power.Ht40[7][11]	
MCS_22	1	2GHz.Target.Power.Ht40[0][12] 5GHz.Target.Power.Ht40[0][12]	Target power value for 2|5 GHz HT40 MCS 22
		2GHz.Target.Power.Ht40[1][12] 5GHz.Target.Power.Ht40[1][12]	
		2GHz.Target.Power.Ht40[2][12] 5GHz.Target.Power.Ht40[2][12]	
		5GHz.Target.Power.Ht40[3][12]	
		5GHz.Target.Power.Ht40[4][12]	
		5GHz.Target.Power.Ht40[5][12]	
		5GHz.Target.Power.Ht40[6][12]	
		5GHz.Target.Power.Ht40[7][12]	
MCS_23	1	2GHz.Target.Power.Ht40[0][13] 5GHz.Target.Power.Ht40[0][13]	Target power value for 2|5 GHz HT40 MCS 23
		2GHz.Target.Power.Ht40[1][13] 5GHz.Target.Power.Ht40[1][13]	
		2GHz.Target.Power.Ht40[2][13] 5GHz.Target.Power.Ht40[2][13]	
		5GHz.Target.Power.Ht40[3][13]	
		5GHz.Target.Power.Ht40[4][13]	
		5GHz.Target.Power.Ht40[5][13]	
		5GHz.Target.Power.Ht40[6][13]	
		5GHz.Target.Power.Ht40[7][13]	

Target Powers 802.11a OFDM

Figure 2-17 shows the target powers 802.11a OFDM board data details.

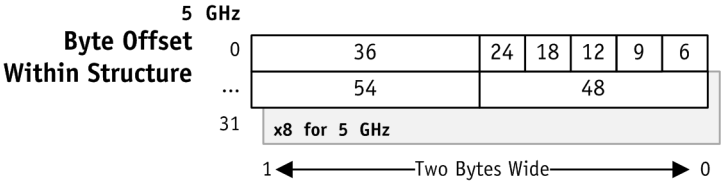

Figure 2-17. **Target Powers 802.11a OFDM**

Table 2-20 describes the fields shown in Figure 2-17.

These bits define multiple [2 | 5]GHz.TargetPower.Legacy[*x*][*y*] commands, where *x* is the channel (0–2 for 2 GHz, 0–7 for 5 GHz) and *y* is the rate (0–3) (see the `2GHz.Target.Frequency` parameters in Table 2-14 for 2 GHz, see the `5GHz.Target.Frequency` parameters in Table 2-15 for 5 GHz).

Table 2-20. **Target Powers 802.11a OFDM Parameter Descriptions**

Parameter Name	Bytes	get/set Command Parameter	Description
6–24	1	`2GHz.Target.Power.Legacy[0][0]` `5GHz.Target.Power.Legacy[0][0]`	Target power for legacy 5 GHz rates 6, 9, 12, 18, and 24 must be set to the same power value.
		`2GHz.Target.Power.Legacy[1][0]` `5GHz.Target.Power.Legacy[1][0]`	
		`2GHz.Target.Power.Legacy[2][0]` `5GHz.Target.Power.Legacy[2][0]`	
		`5GHz.Target.Power.Legacy[3][0]`	
		`5GHz.Target.Power.Legacy[4][0]`	
		`5GHz.Target.Power.Legacy[5][0]`	
		`5GHz.Target.Power.Legacy[6][0]`	
		`5GHz.Target.Power.Legacy[7][0]`	
36	1	`2GHz.Target.Power.Legacy[0][1]` `5GHz.Target.Power.Legacy[0][1]`	Target power for 5 GHz legacy rate 36 Mbps
		`2GHz.Target.Power.Legacy[1][1]` `5GHz.Target.Power.Legacy[1][1]`	
		`2GHz.Target.Power.Legacy[2][1]` `5GHz.Target.Power.Legacy[2][1]`	
		`5GHz.Target.Power.Legacy[3][1]`	
		`5GHz.Target.Power.Legacy[4][1]`	
		`5GHz.Target.Power.Legacy[5][1]`	
		`5GHz.Target.Power.Legacy[6][1]`	
		`5GHz.Target.Power.Legacy[7][1]`	

Table 2-20. **Target Powers 802.11a OFDM Parameter Descriptions (continued)**

Parameter Name	Bytes	get/set Command Parameter	Description
48	1	2GHz.Target.Power.Legacy[0][2] 5GHz.Target.Power.Legacy[0][2]	Target power for 5 GHz legacy rate 48 Mbps
		2GHz.Target.Power.Legacy[1][2] 5GHz.Target.Power.Legacy[1][2]	
		2GHz.Target.Power.Legacy[2][2] 5GHz.Target.Power.Legacy[2][2]	
		5GHz.Target.Power.Legacy[3][2]	
		5GHz.Target.Power.Legacy[4][2]	
		5GHz.Target.Power.Legacy[5][2]	
		5GHz.Target.Power.Legacy[6][2]	
		5GHz.Target.Power.Legacy[7][2]	
54	1	2GHz.Target.Power.Legacy[0][3] 5GHz.Target.Power.Legacy[0][3]	Target power for 5 GHz legacy rate 54 Mbps
		2GHz.Target.Power.Legacy[1][3] 5GHz.Target.Power.Legacy[1][3]	
		2GHz.Target.Power.Legacy[2][3] 5GHz.Target.Power.Legacy[2][3]	
		5GHz.Target.Power.Legacy[3][3]	
		5GHz.Target.Power.Legacy[4][3]	
		5GHz.Target.Power.Legacy[5][3]	
		5GHz.Target.Power.Legacy[6][3]	
		5GHz.Target.Power.Legacy[7][3]	

CTL Indexes

Table 2-21 describes the CTL index fields.

Table 2-21. **CTL Indexes Parameter Descriptions**

<table>
<tr><th>Parameter Name</th><th>Bytes</th><th>get/set Command Parameter</th><th colspan="3">Description</th></tr>
<tr><td rowspan="25">CTL Index</td><td rowspan="25">12 (2 GHz)
9 (5 GHz)</td><td rowspan="13">2GHz.CtlIndex [x]=y;</td><td colspan="3">■ 2GHz.CtlIndex[x] sets the 2.4 GHz field to 8-bit hex values for 12 CTL indexes.
■ 5GHz.CtlIndex[x] sets the 5 GHz field to 8-bit hex values for 9 CTL indexes.
CTL hex codes (for the CTLs this card is calibrated for). The CTL that the driver determines the current country code index belongs to is matched against these codes to figure out the location of correct data to retrieve from the EEPROM.
The lower four bits (bits [3:0]) of hex code identify which operating mode (802.11a/802.11b/802.11g) the CTL pertains to.</td></tr>
<tr><td>x</td><td>y</td><td></td></tr>
<tr><td>0</td><td>0x11</td><td>802.11b (CCK 11g) FCC</td></tr>
<tr><td>1</td><td>0x12</td><td>802.11g Legacy FCC</td></tr>
<tr><td>2</td><td>0x15</td><td>802.11n HT20 FCC</td></tr>
<tr><td>3</td><td>0x17</td><td>802.11n HT40 FCC</td></tr>
<tr><td>4</td><td>0x41</td><td>802.11b (CCK 11g) Japan</td></tr>
<tr><td>5</td><td>0x42</td><td>802.11g Legacy Japan</td></tr>
<tr><td>6</td><td>0x45</td><td>802.11n HT20 Japan</td></tr>
<tr><td>7</td><td>0x47</td><td>802.11n HT40 Japan</td></tr>
<tr><td>8</td><td>0x31</td><td>802.11b (CCK 11g) ETSI</td></tr>
<tr><td>9</td><td>0x32</td><td>802.11g Legacy ETSI</td></tr>
<tr><td>10</td><td>0x35</td><td>802.11n HT20 ETSI</td></tr>
<tr><td>11</td><td>0x37</td><td>802.11n HT40 ETSI</td></tr>
<tr><td rowspan="12">5GHz.CtlIndex [x]=y;</td><td>0</td><td>0x10</td><td>802.11a Legacy FCC</td></tr>
<tr><td>1</td><td>0x16</td><td>802.11n HT20 FCC</td></tr>
<tr><td>2</td><td>0x18</td><td>802.11n HT40 FCC</td></tr>
<tr><td>3</td><td>0x40</td><td>802.11a Legacy Japan</td></tr>
<tr><td>4</td><td>0x46</td><td>802.11n HT20 Japan</td></tr>
<tr><td>5</td><td>0x48</td><td>802.11n HT40 Japan</td></tr>
<tr><td>6</td><td>0x30</td><td>802.11a Legacy ETSI</td></tr>
<tr><td>7</td><td>0x36</td><td>802.11n HT20 ETSI</td></tr>
<tr><td>8</td><td>0x38</td><td>802.11n HT40 ETSI</td></tr>
</table>

CTL Band Edges and CTL Power Data

Figure 2-18 shows the target powers CTL band edges board data details.

NOTE: For 2 GHz, these tables contain only four power entries.

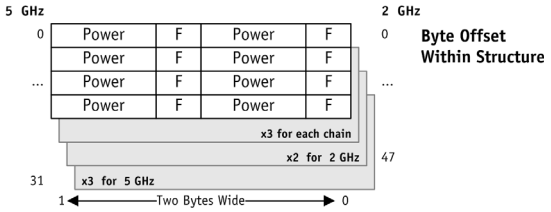

Figure 2-18. **CTL Band Edges**

Figure 2-19 shows the target powers CTL power board data details.

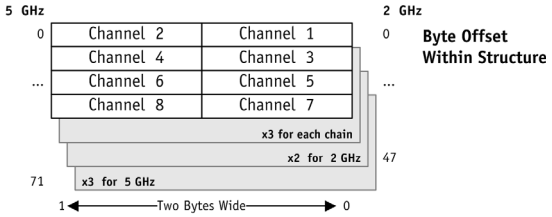

Figure 2-19. **CTL Power Data**

Table 2-22 describes the fields shown in Figure 2-18 and Figure 2-19. These bits define multiple commands ([2|5]*Command*[*x*][*y*]), where *x is* the x value in Table 2-21 (0–11 for 2 GHz, 0–8 for 5 GHz) and *y* the pier (0–3 for 2 GHz, 0–7 for 5 GHz).

Table 2-22. **CTL Band Edges and Power Data Parameter Descriptions**

Parameter Name	Bytes	get/set Command Parameter	Description

Channel 1–8 (2 GHz)	1 per channel	`2GHz.Ctl.Frequency[0][0]=2412;`
		`2GHz.Ctl.Frequency[0][1]=2417;`
		`2GHz.Ctl.Frequency[0][2]=2457;`
		`2GHz.Ctl.Frequency[0][3]=2462;`
			The encoded band edge channel for 802.11b (CCK 11g) FCC

		`2GHz.Ctl.Frequency[1][0]=2412;`
		`2GHz.Ctl.Frequency[1][1]=2417;`
		`2GHz.Ctl.Frequency[1][2]=2462;`
		`2GHz.Ctl.Frequency[1][3]=2555;`
			The encoded band edge channel for 802.11g Legacy FCC

		`2GHz.Ctl.Frequency[2][0]=2412;`
		`2GHz.Ctl.Frequency[2][1]=2417;`
		`2GHz.Ctl.Frequency[2][2]=2462;`
		`2GHz.Ctl.Frequency[2][3]=2555;`
			The encoded band edge channel for 802.11n HT20 FCC

		`2GHz.Ctl.Frequency[3][0]=2422;`
		`2GHz.Ctl.Frequency[3][1]=2427;`
		`2GHz.Ctl.Frequency[3][2]=2447;`
		`2GHz.Ctl.Frequency[3][3]=2452;`
			The encoded band edge channel for 802.11n HT40 FCC

		`2GHz.Ctl.Frequency[4][0]=2412;`
		`2GHz.Ctl.Frequency[4][1]=2417;`
		`2GHz.Ctl.Frequency[4][2]=2472;`
		`2GHz.Ctl.Frequency[4][3]=2484;`
			The encoded band edge channel for 802.11b (CCK 11g) Japan

Table 2-22. **CTL Band Edges and Power Data Parameter Descriptions (continued)**

Parameter Name	Bytes	get/set Command Parameter	Description
Channel 1–8 (2 GHz) Cont.	1 per channel	2GHz.Ctl.Frequency[5][0]=2412; 2GHz.Ctl.Frequency[5][1]=2417; 2GHz.Ctl.Frequency[5][2]=2472; 2GHz.Ctl.Frequency[5][3]=0;	The encoded band edge channel for 802.11g Legacy Japan
		2GHz.Ctl.Frequency[6][0]=2412; 2GHz.Ctl.Frequency[6][1]=2417; 2GHz.Ctl.Frequency[6][2]=2472; 2GHz.Ctl.Frequency[6][3]=0;	The encoded band edge channel for 802.11n HT20 Japan
		2GHz.Ctl.Frequency[7][0]=2422; 2GHz.Ctl.Frequency[7][1]=2427; 2GHz.Ctl.Frequency[7][2]=2447; 2GHz.Ctl.Frequency[7][3]=2462;	The encoded band edge channel for 802.11n HT40 Japan
		2GHz.Ctl.Frequency[8][0]=2412; 2GHz.Ctl.Frequency[8][1]=2417; 2GHz.Ctl.Frequency[8][2]=2472; 2GHz.Ctl.Frequency[8][3]=0;	The encoded band edge channel for 802.11b (CCK 11g) ETSI
		2GHz.Ctl.Frequency[9][0]=2412; 2GHz.Ctl.Frequency[9][1]=2417; 2GHz.Ctl.Frequency[9][2]=2472; 2GHz.Ctl.Frequency[9][3]=0;	The encoded band edge channel for 802.11g Legacy ETSI
		2GHz.Ctl.Frequency[10][0]=2412; 2GHz.Ctl.Frequency[10][1]=2417; 2GHz.Ctl.Frequency[10][2]=2472; 2GHz.Ctl.Frequency[10][3]=0;	The encoded band edge channel for 802.11n HT20 ETSI
		2GHz.Ctl.Frequency[11][0]=2422; 2GHz.Ctl.Frequency[11][1]=2427; 2GHz.Ctl.Frequency[11][2]=2447; 2GHz.Ctl.Frequency[11][3]=2462;	The encoded band edge channel for 802.11n HT40 ETSI
Channel 1–8 (5 GHz)	1 per channel	5GHz.Ctl.Frequency[0][0]=5180; 5GHz.Ctl.Frequency[0][1]=5260; 5GHz.Ctl.Frequency[0][2]=5280; 5GHz.Ctl.Frequency[0][3]=5500; 5GHz.Ctl.Frequency[0][4]=5600; 5GHz.Ctl.Frequency[0][5]=5700; 5GHz.Ctl.Frequency[0][6]=5745; 5GHz.Ctl.Frequency[0][7]=5825;	The encoded band edge channel for 802.11a Legacy FCC
		5GHz.Ctl.Frequency[1][0]=5180; 5GHz.Ctl.Frequency[1][1]=5260; 5GHz.Ctl.Frequency[1][2]=5280; 5GHz.Ctl.Frequency[1][3]=5500; 5GHz.Ctl.Frequency[1][4]=5520; 5GHz.Ctl.Frequency[1][5]=5700; 5GHz.Ctl.Frequency[1][6]=5745; 5GHz.Ctl.Frequency[1][7]=5825;	The encoded band edge channel for 802.11n HT20 FCC
		5GHz.Ctl.Frequency[2][0]=5190; 5GHz.Ctl.Frequency[2][1]=5230; 5GHz.Ctl.Frequency[2][2]=5270; 5GHz.Ctl.Frequency[2][3]=5310; 5GHz.Ctl.Frequency[2][4]=5510; 5GHz.Ctl.Frequency[2][5]=5550; 5GHz.Ctl.Frequency[2][6]=5670; 5GHz.Ctl.Frequency[2][7]=5755;	The encoded band edge channel for 802.11n HT40 FCC

Table 2-22. **CTL Band Edges and Power Data Parameter Descriptions (continued)**

Parameter Name	Bytes	get/set Command Parameter	Description
Channel 1–8 (5 GHz) Cont.	1 per channel	5GHz.Ctl.Frequency[3][0]=5180; 5GHz.Ctl.Frequency[3][1]=5200; 5GHz.Ctl.Frequency[3][2]=5260; 5GHz.Ctl.Frequency[3][3]=5320; 5GHz.Ctl.Frequency[3][4]=5500; 5GHz.Ctl.Frequency[3][5]=5700; 5GHz.Ctl.Frequency[3][6]=6075; 5GHz.Ctl.Frequency[3][7]=6075;	The encoded band edge channel for 802.11a Legacy Japan
		5GHz.Ctl.Frequency[4][0]=5180; 5GHz.Ctl.Frequency[4][1]=5260; 5GHz.Ctl.Frequency[4][2]=5500; 5GHz.Ctl.Frequency[4][3]=5700; 5GHz.Ctl.Frequency[4][4]=6075; 5GHz.Ctl.Frequency[4][5]=6075; 5GHz.Ctl.Frequency[4][6]=6075; 5GHz.Ctl.Frequency[4][7]=6075;	The encoded band edge channel for 802.11n HT20 Japan
		5GHz.Ctl.Frequency[5][0]=5190; 5GHz.Ctl.Frequency[5][1]=5270; 5GHz.Ctl.Frequency[5][2]=5310; 5GHz.Ctl.Frequency[5][3]=5510; 5GHz.Ctl.Frequency[5][4]=5590; 5GHz.Ctl.Frequency[5][5]=5670; 5GHz.Ctl.Frequency[5][6]=6075; 5GHz.Ctl.Frequency[5][7]=6075;	The encoded band edge channel for 802.11n HT40 Japan
		5GHz.Ctl.Frequency[6][0]=5180; 5GHz.Ctl.Frequency[6][1]=5200; 5GHz.Ctl.Frequency[6][2]=5220; 5GHz.Ctl.Frequency[6][3]=5260; 5GHz.Ctl.Frequency[6][4]=5500; 5GHz.Ctl.Frequency[6][5]=5600; 5GHz.Ctl.Frequency[6][6]=5700; 5GHz.Ctl.Frequency[6][7]=5745;	The encoded band edge channel for 802.11a Legacy ETSI
		5GHz.Ctl.Frequency[7][0]=5180; 5GHz.Ctl.Frequency[7][1]=5260; 5GHz.Ctl.Frequency[7][2]=5320; 5GHz.Ctl.Frequency[7][3]=5500; 5GHz.Ctl.Frequency[7][4]=5560; 5GHz.Ctl.Frequency[7][5]=5700; 5GHz.Ctl.Frequency[7][6]=5745; 5GHz.Ctl.Frequency[7][7]=5825;	The encoded band edge channel for 802.11n HT20 ETSI
		5GHz.Ctl.Frequency[8][0]=5190; 5GHz.Ctl.Frequency[8][1]=5230; 5GHz.Ctl.Frequency[8][2]=5270; 5GHz.Ctl.Frequency[8][3]=5510; 5GHz.Ctl.Frequency[8][4]=5550; 5GHz.Ctl.Frequency[8][5]=5670; 5GHz.Ctl.Frequency[8][6]=5755; 5GHz.Ctl.Frequency[8][7]=5795;	The encoded band edge channel for 802.11n HT40 ETSI

Table 2-22. **CTL Band Edges and Power Data Parameter Descriptions (continued)**

Parameter Name	Bytes	get/set Command Parameter	Description
Power (2 GHz)	(6 bits)	2GHz.Ctl.Power[0][0]=30.0; 2GHz.Ctl.Power[0][1]=30.0; 2GHz.Ctl.Power[0][2]=30.0; 2GHz.Ctl.Power[0][3]=30.0;	Band edge power (specified in half dB) for 802.11b (CCK 11g) FCC
		2GHz.Ctl.Power[1][0]=30.0; 2GHz.Ctl.Power[1][1]=30.0; 2GHz.Ctl.Power[1][2]=30.0; 2GHz.Ctl.Power[1][3]=30.0;	Band edge power (specified in half dB) for 802.11g Legacy FCC
		2GHz.Ctl.Power[2][0]=30.0; 2GHz.Ctl.Power[2][1]=30.0; 2GHz.Ctl.Power[2][2]=30.0; 2GHz.Ctl.Power[2][3]=30.0;	Band edge power (specified in half dB) for 802.11n HT20 FCC
		2GHz.Ctl.Power[3][0]=30.0; 2GHz.Ctl.Power[3][1]=30.0; 2GHz.Ctl.Power[3][2]=0.0; 2GHz.Ctl.Power[3][3]=0.0;	Band edge power (specified in half dB) for 802.11n HT40 FCC
		2GHz.Ctl.Power[4][0]=30.0; 2GHz.Ctl.Power[4][1]=30.0; 2GHz.Ctl.Power[4][2]=30.0; 2GHz.Ctl.Power[4][3]=30.0;	Band edge power (specified in half dB) for 802.11b (CCK 11g) Japan
		2GHz.Ctl.Power[5][0]=30.0; 2GHz.Ctl.Power[5][1]=30.0; 2GHz.Ctl.Power[5][2]=30.0; 2GHz.Ctl.Power[5][3]=30.0;	Band edge power (specified in half dB) for 802.11g Legacy Japan
		2GHz.Ctl.Power[6][0]=30.0; 2GHz.Ctl.Power[6][1]=30.0; 2GHz.Ctl.Power[6][2]=30.0; 2GHz.Ctl.Power[6][3]=30.0;	Band edge power (specified in half dB) for 802.11n HT20 Japan
		2GHz.Ctl.Power[7][0]=30.0; 2GHz.Ctl.Power[7][1]=30.0; 2GHz.Ctl.Power[7][2]=30.0; 2GHz.Ctl.Power[7][3]=30.0;	Band edge power (specified in half dB) for 802.11n HT40 Japan
		2GHz.Ctl.Power[8][0]=30.0; 2GHz.Ctl.Power[8][1]=30.0; 2GHz.Ctl.Power[8][2]=30.0; 2GHz.Ctl.Power[8][3]=30.0;	Band edge power (specified in half dB) for 802.11b (CCK 11g) ETSI
		2GHz.Ctl.Power[9][0]=30.0; 2GHz.Ctl.Power[9][1]=30.0; 2GHz.Ctl.Power[9][2]=30.0; 2GHz.Ctl.Power[9][3]=30.0;	Band edge power (specified in half dB) for 802.11g Legacy ETSI
		2GHz.Ctl.Power[10][0]=30.0; 2GHz.Ctl.Power[10][1]=30.0; 2GHz.Ctl.Power[10][2]=30.0; 2GHz.Ctl.Power[10][3]=30.0;	Band edge power (specified in half dB) for 802.11n HT20 ETSI
		2GHz.Ctl.Power[11][0]=30.0; 2GHz.Ctl.Power[11][1]=30.0; 2GHz.Ctl.Power[11][2]=30.0; 2GHz.Ctl.Power[11][3]=30.0;	Band edge power (specified in half dB) for 802.11n HT40 ETSI

Table 2-22. **CTL Band Edges and Power Data Parameter Descriptions (continued)**

Parameter Name	Bytes	get/set Command Parameter	Description
Channel 1–8 (2 GHz) Cont.	1 per channel	2GHz.Ctl.Frequency[5][0]=2412; 2GHz.Ctl.Frequency[5][1]=2417; 2GHz.Ctl.Frequency[5][2]=2472; 2GHz.Ctl.Frequency[5][3]=0;	The encoded band edge channel for 802.11g Legacy Japan
		2GHz.Ctl.Frequency[6][0]=2412; 2GHz.Ctl.Frequency[6][1]=2417; 2GHz.Ctl.Frequency[6][2]=2472; 2GHz.Ctl.Frequency[6][3]=0;	The encoded band edge channel for 802.11n HT20 Japan
		2GHz.Ctl.Frequency[7][0]=2422; 2GHz.Ctl.Frequency[7][1]=2427; 2GHz.Ctl.Frequency[7][2]=2447; 2GHz.Ctl.Frequency[7][3]=2462;	The encoded band edge channel for 802.11n HT40 Japan
		2GHz.Ctl.Frequency[8][0]=2412; 2GHz.Ctl.Frequency[8][1]=2417; 2GHz.Ctl.Frequency[8][2]=2472; 2GHz.Ctl.Frequency[8][3]=0;	The encoded band edge channel for 802.11b (CCK 11g) ETSI
		2GHz.Ctl.Frequency[9][0]=2412; 2GHz.Ctl.Frequency[9][1]=2417; 2GHz.Ctl.Frequency[9][2]=2472; 2GHz.Ctl.Frequency[9][3]=0;	The encoded band edge channel for 802.11g Legacy ETSI
		2GHz.Ctl.Frequency[10][0]=2412; 2GHz.Ctl.Frequency[10][1]=2417; 2GHz.Ctl.Frequency[10][2]=2472; 2GHz.Ctl.Frequency[10][3]=0;	The encoded band edge channel for 802.11n HT20 ETSI
		2GHz.Ctl.Frequency[11][0]=2422; 2GHz.Ctl.Frequency[11][1]=2427; 2GHz.Ctl.Frequency[11][2]=2447; 2GHz.Ctl.Frequency[11][3]=2462;	The encoded band edge channel for 802.11n HT40 ETSI
Channel 1–8 (5 GHz)	1 per channel	5GHz.Ctl.Frequency[0][0]=5180; 5GHz.Ctl.Frequency[0][1]=5260; 5GHz.Ctl.Frequency[0][2]=5280; 5GHz.Ctl.Frequency[0][3]=5500; 5GHz.Ctl.Frequency[0][4]=5600; 5GHz.Ctl.Frequency[0][5]=5700; 5GHz.Ctl.Frequency[0][6]=5745; 5GHz.Ctl.Frequency[0][7]=5825;	The encoded band edge channel for 802.11a Legacy FCC
		5GHz.Ctl.Frequency[1][0]=5180; 5GHz.Ctl.Frequency[1][1]=5260; 5GHz.Ctl.Frequency[1][2]=5280; 5GHz.Ctl.Frequency[1][3]=5500; 5GHz.Ctl.Frequency[1][4]=5520; 5GHz.Ctl.Frequency[1][5]=5700; 5GHz.Ctl.Frequency[1][6]=5745; 5GHz.Ctl.Frequency[1][7]=5825;	The encoded band edge channel for 802.11n HT20 FCC
		5GHz.Ctl.Frequency[2][0]=5190; 5GHz.Ctl.Frequency[2][1]=5230; 5GHz.Ctl.Frequency[2][2]=5270; 5GHz.Ctl.Frequency[2][3]=5310; 5GHz.Ctl.Frequency[2][4]=5510; 5GHz.Ctl.Frequency[2][5]=5550; 5GHz.Ctl.Frequency[2][6]=5670; 5GHz.Ctl.Frequency[2][7]=5755;	The encoded band edge channel for 802.11n HT40 FCC

Table 2-22. **CTL Band Edges and Power Data Parameter Descriptions (continued)**

Parameter Name	Bytes	get/set Command Parameter	Description
Channel 1–8 (5 GHz) Cont.	1 per channel	5GHz.Ctl.Frequency[3][0]=5180; 5GHz.Ctl.Frequency[3][1]=5200; 5GHz.Ctl.Frequency[3][2]=5260; 5GHz.Ctl.Frequency[3][3]=5320; 5GHz.Ctl.Frequency[3][4]=5500; 5GHz.Ctl.Frequency[3][5]=5700; 5GHz.Ctl.Frequency[3][6]=6075; 5GHz.Ctl.Frequency[3][7]=6075;	The encoded band edge channel for 802.11a Legacy Japan
		5GHz.Ctl.Frequency[4][0]=5180; 5GHz.Ctl.Frequency[4][1]=5260; 5GHz.Ctl.Frequency[4][2]=5500; 5GHz.Ctl.Frequency[4][3]=5700; 5GHz.Ctl.Frequency[4][4]=6075; 5GHz.Ctl.Frequency[4][5]=6075; 5GHz.Ctl.Frequency[4][6]=6075; 5GHz.Ctl.Frequency[4][7]=6075;	The encoded band edge channel for 802.11n HT20 Japan
		5GHz.Ctl.Frequency[5][0]=5190; 5GHz.Ctl.Frequency[5][1]=5270; 5GHz.Ctl.Frequency[5][2]=5310; 5GHz.Ctl.Frequency[5][3]=5510; 5GHz.Ctl.Frequency[5][4]=5590; 5GHz.Ctl.Frequency[5][5]=5670; 5GHz.Ctl.Frequency[5][6]=6075; 5GHz.Ctl.Frequency[5][7]=6075;	The encoded band edge channel for 802.11n HT40 Japan
		5GHz.Ctl.Frequency[6][0]=5180; 5GHz.Ctl.Frequency[6][1]=5200; 5GHz.Ctl.Frequency[6][2]=5220; 5GHz.Ctl.Frequency[6][3]=5260; 5GHz.Ctl.Frequency[6][4]=5500; 5GHz.Ctl.Frequency[6][5]=5600; 5GHz.Ctl.Frequency[6][6]=5700; 5GHz.Ctl.Frequency[6][7]=5745;	The encoded band edge channel for 802.11a Legacy ETSI
		5GHz.Ctl.Frequency[7][0]=5180; 5GHz.Ctl.Frequency[7][1]=5260; 5GHz.Ctl.Frequency[7][2]=5320; 5GHz.Ctl.Frequency[7][3]=5500; 5GHz.Ctl.Frequency[7][4]=5560; 5GHz.Ctl.Frequency[7][5]=5700; 5GHz.Ctl.Frequency[7][6]=5745; 5GHz.Ctl.Frequency[7][7]=5825;	The encoded band edge channel for 802.11n HT20 ETSI
		5GHz.Ctl.Frequency[8][0]=5190; 5GHz.Ctl.Frequency[8][1]=5230; 5GHz.Ctl.Frequency[8][2]=5270; 5GHz.Ctl.Frequency[8][3]=5510; 5GHz.Ctl.Frequency[8][4]=5550; 5GHz.Ctl.Frequency[8][5]=5670; 5GHz.Ctl.Frequency[8][6]=5755; 5GHz.Ctl.Frequency[8][7]=5795;	The encoded band edge channel for 802.11n HT40 ETSI

Table 2-22. **CTL Band Edges and Power Data Parameter Descriptions (continued)**

Parameter Name	Bytes	get/set Command Parameter	Description
Power (2 GHz)	(6 bits)	2GHz.Ctl.Power[0][0]=30.0; 2GHz.Ctl.Power[0][1]=30.0; 2GHz.Ctl.Power[0][2]=30.0; 2GHz.Ctl.Power[0][3]=30.0;	Band edge power (specified in half dB) for 802.11b (CCK 11g) FCC
		2GHz.Ctl.Power[1][0]=30.0; 2GHz.Ctl.Power[1][1]=30.0; 2GHz.Ctl.Power[1][2]=30.0; 2GHz.Ctl.Power[1][3]=30.0;	Band edge power (specified in half dB) for 802.11g Legacy FCC
		2GHz.Ctl.Power[2][0]=30.0; 2GHz.Ctl.Power[2][1]=30.0; 2GHz.Ctl.Power[2][2]=30.0; 2GHz.Ctl.Power[2][3]=30.0;	Band edge power (specified in half dB) for 802.11n HT20 FCC
		2GHz.Ctl.Power[3][0]=30.0; 2GHz.Ctl.Power[3][1]=30.0; 2GHz.Ctl.Power[3][2]=0.0; 2GHz.Ctl.Power[3][3]=0.0;	Band edge power (specified in half dB) for 802.11n HT40 FCC
		2GHz.Ctl.Power[4][0]=30.0; 2GHz.Ctl.Power[4][1]=30.0; 2GHz.Ctl.Power[4][2]=30.0; 2GHz.Ctl.Power[4][3]=30.0;	Band edge power (specified in half dB) for 802.11b (CCK 11g) Japan
		2GHz.Ctl.Power[5][0]=30.0; 2GHz.Ctl.Power[5][1]=30.0; 2GHz.Ctl.Power[5][2]=30.0; 2GHz.Ctl.Power[5][3]=30.0;	Band edge power (specified in half dB) for 802.11g Legacy Japan
		2GHz.Ctl.Power[6][0]=30.0; 2GHz.Ctl.Power[6][1]=30.0; 2GHz.Ctl.Power[6][2]=30.0; 2GHz.Ctl.Power[6][3]=30.0;	Band edge power (specified in half dB) for 802.11n HT20 Japan
		2GHz.Ctl.Power[7][0]=30.0; 2GHz.Ctl.Power[7][1]=30.0; 2GHz.Ctl.Power[7][2]=30.0; 2GHz.Ctl.Power[7][3]=30.0;	Band edge power (specified in half dB) for 802.11n HT40 Japan
		2GHz.Ctl.Power[8][0]=30.0; 2GHz.Ctl.Power[8][1]=30.0; 2GHz.Ctl.Power[8][2]=30.0; 2GHz.Ctl.Power[8][3]=30.0;	Band edge power (specified in half dB) for 802.11b (CCK 11g) ETSI
		2GHz.Ctl.Power[9][0]=30.0; 2GHz.Ctl.Power[9][1]=30.0; 2GHz.Ctl.Power[9][2]=30.0; 2GHz.Ctl.Power[9][3]=30.0;	Band edge power (specified in half dB) for 802.11g Legacy ETSI
		2GHz.Ctl.Power[10][0]=30.0; 2GHz.Ctl.Power[10][1]=30.0; 2GHz.Ctl.Power[10][2]=30.0; 2GHz.Ctl.Power[10][3]=30.0;	Band edge power (specified in half dB) for 802.11n HT20 ETSI
		2GHz.Ctl.Power[11][0]=30.0; 2GHz.Ctl.Power[11][1]=30.0; 2GHz.Ctl.Power[11][2]=30.0; 2GHz.Ctl.Power[11][3]=30.0;	Band edge power (specified in half dB) for 802.11n HT40 ETSI

Table 2-22. **CTL Band Edges and Power Data Parameter Descriptions (continued)**

Parameter Name	Bytes	get/set Command Parameter	Description
Power (5 GHz)	(6 bits)	5GHz.Ctl.Power[0][0]=30.0; 5GHz.Ctl.Power[0][1]=30.0; 5GHz.Ctl.Power[0][2]=30.0; 5GHz.Ctl.Power[0][3]=30.0; 5GHz.Ctl.Power[0][4]=30.0; 5GHz.Ctl.Power[0][5]=30.0; 5GHz.Ctl.Power[0][6]=30.0; 5GHz.Ctl.Power[0][7]=30.0;	Band edge power (specified in half dB) for 802.11a Legacy FCC
		5GHz.Ctl.Power[1][0]=30.0; 5GHz.Ctl.Power[1][1]=30.0; 5GHz.Ctl.Power[1][2]=30.0; 5GHz.Ctl.Power[1][3]=30.0; 5GHz.Ctl.Power[1][4]=30.0; 5GHz.Ctl.Power[1][5]=30.0; 5GHz.Ctl.Power[1][6]=30.0; 5GHz.Ctl.Power[1][7]=30.0;	Band edge power (specified in half dB) for 802.11n HT20 FCC
		5GHz.Ctl.Power[2][0]=30.0; 5GHz.Ctl.Power[2][1]=30.0; 5GHz.Ctl.Power[2][2]=30.0; 5GHz.Ctl.Power[2][3]=30.0; 5GHz.Ctl.Power[2][4]=30.0; 5GHz.Ctl.Power[2][5]=30.0; 5GHz.Ctl.Power[2][6]=30.0; 5GHz.Ctl.Power[2][7]=30.0;	Band edge power (specified in half dB) for 802.11n HT40 FCC
		5GHz.Ctl.Power[3][0]=30.0; 5GHz.Ctl.Power[3][1]=30.0; 5GHz.Ctl.Power[3][2]=30.0; 5GHz.Ctl.Power[3][3]=30.0; 5GHz.Ctl.Power[3][4]=30.0; 5GHz.Ctl.Power[3][5]=30.0; 5GHz.Ctl.Power[3][6]=30.0; 5GHz.Ctl.Power[3][7]=30.0;	Band edge power (specified in half dB) for 802.11a Legacy Japan
		5GHz.Ctl.Power[4][0]=30.0; 5GHz.Ctl.Power[4][1]=30.0; 5GHz.Ctl.Power[4][2]=30.0; 5GHz.Ctl.Power[4][3]=30.0; 5GHz.Ctl.Power[4][4]=30.0; 5GHz.Ctl.Power[4][5]=30.0; 5GHz.Ctl.Power[4][6]=30.0; 5GHz.Ctl.Power[4][7]=30.0;	Band edge power (specified in half dB) for 802.11n HT20 Japan
		5GHz.Ctl.Power[5][0]=30.0; 5GHz.Ctl.Power[5][1]=30.0; 5GHz.Ctl.Power[5][2]=30.0; 5GHz.Ctl.Power[5][3]=30.0; 5GHz.Ctl.Power[5][4]=30.0; 5GHz.Ctl.Power[5][5]=30.0; 5GHz.Ctl.Power[5][6]=30.0; 5GHz.Ctl.Power[5][7]=30.0;	Band edge power (specified in half dB) for 802.11n HT40 Japan

Table 2-22. **CTL Band Edges and Power Data Parameter Descriptions (continued)**

Parameter Name	Bytes	get/set Command Parameter	Description
Power (5 GHz) Cont.	(6 bits)	5GHz.Ctl.Power[6][0]=30.0; 5GHz.Ctl.Power[6][1]=30.0; 5GHz.Ctl.Power[6][2]=30.0; 5GHz.Ctl.Power[6][3]=30.0; 5GHz.Ctl.Power[6][4]=30.0; 5GHz.Ctl.Power[6][5]=30.0; 5GHz.Ctl.Power[6][6]=30.0; 5GHz.Ctl.Power[6][7]=30.0;	Band edge power (specified in half dB) for 802.11a Legacy ETSI
		5GHz.Ctl.Power[7][0]=30.0; 5GHz.Ctl.Power[7][1]=30.0; 5GHz.Ctl.Power[7][2]=30.0; 5GHz.Ctl.Power[7][3]=30.0; 5GHz.Ctl.Power[7][4]=30.0; 5GHz.Ctl.Power[7][5]=30.0; 5GHz.Ctl.Power[7][6]=30.0; 5GHz.Ctl.Power[7][7]=30.0;	Band edge power (specified in half dB) for 802.11n HT20 ETSI
		5GHz.Ctl.Power[8][0]=30.0; 5GHz.Ctl.Power[8][1]=30.0; 5GHz.Ctl.Power[8][2]=30.0; 5GHz.Ctl.Power[8][3]=30.0; 5GHz.Ctl.Power[8][4]=30.0; 5GHz.Ctl.Power[8][5]=30.0; 5GHz.Ctl.Power[8][6]=30.0; 5GHz.Ctl.Power[8][7]=30.0;	Band edge power (specified in half dB) for 802.11n HT40 ETSI
F (2 GHz)	(2 bits)	2GHz.Ctl.BandEdge[0][0]=0x00; 2GHz.Ctl.BandEdge[0][1]=0x01; 2GHz.Ctl.BandEdge[0][2]=0x00; 2GHz.Ctl.BandEdge[0][3]=0x00;	Band edge flag specifying edge (0x01) or inband (0x00) channel for 802.11b (CCK 11g) FCC
		2GHz.Ctl.BandEdge[1][0]=0x00; 2GHz.Ctl.BandEdge[1][1]=0x01; 2GHz.Ctl.BandEdge[1][2]=0x00; 2GHz.Ctl.BandEdge[1][3]=0x00;	Band edge flag specifying edge (0x01) or inband (0x00) channel for 802.11g Legacy FCC
		2GHz.Ctl.BandEdge[2][0]=0x01; 2GHz.Ctl.BandEdge[2][1]=0x00; 2GHz.Ctl.BandEdge[2][2]=0x00; 2GHz.Ctl.BandEdge[2][3]=0x01;	Band edge flag specifying edge (0x01) or inband (0x00) channel for 802.11n HT20 FCC
		2GHz.Ctl.BandEdge[3][0]=0x01; 2GHz.Ctl.BandEdge[3][1]=0x00; 2GHz.Ctl.BandEdge[3][2]=0x00; 2GHz.Ctl.BandEdge[3][3]=0x00;	Band edge flag specifying edge (0x01) or inband (0x00) channel for 802.11n HT40 FCC
		2GHz.Ctl.BandEdge[4][0]=0x00; 2GHz.Ctl.BandEdge[4][1]=0x01; 2GHz.Ctl.BandEdge[4][2]=0x00; 2GHz.Ctl.BandEdge[4][3]=0x00;	Band edge flag specifying edge (0x01) or inband (0x00) channel for 802.11b (CCK 11g) Japan
		2GHz.Ctl.BandEdge[5][0]=0x00; 2GHz.Ctl.BandEdge[5][1]=0x01; 2GHz.Ctl.BandEdge[5][2]=0x00; 2GHz.Ctl.BandEdge[5][3]=0x00;	Band edge flag specifying edge (0x01) or inband (0x00) channel for 802.11g Legacy Japan
		2GHz.Ctl.BandEdge[6][0]=0x00; 2GHz.Ctl.BandEdge[6][1]=0x01; 2GHz.Ctl.BandEdge[6][2]=0x01; 2GHz.Ctl.BandEdge[6][3]=0x00;	Band edge flag specifying edge (0x01) or inband (0x00) channel for 802.11n HT20 Japan

Table 2-22. **CTL Band Edges and Power Data Parameter Descriptions (continued)**

Parameter Name	Bytes	get/set Command Parameter	Description
F (2 GHz) Cont.	(2 bits)	2GHz.Ctl.BandEdge[7][0]=0x00; 2GHz.Ctl.BandEdge[7][1]=0x01; 2GHz.Ctl.BandEdge[7][2]=0x00; 2GHz.Ctl.BandEdge[7][3]=0x00;	Band edge flag specifying edge (0x01) or inband (0x00) channel for 802.11n HT40 Japan
		2GHz.Ctl.BandEdge[8][0]=0x00; 2GHz.Ctl.BandEdge[8][1]=0x01; 2GHz.Ctl.BandEdge[8][2]=0x00; 2GHz.Ctl.BandEdge[8][3]=0x00;	Band edge flag specifying edge (0x01) or inband (0x00) channel for 802.11b (CCK 11g) ETSI
		2GHz.Ctl.BandEdge[9][0]=0x00; 2GHz.Ctl.BandEdge[9][1]=0x01; 2GHz.Ctl.BandEdge[9][2]=0x00; 2GHz.Ctl.BandEdge[9][3]=0x00;	Band edge flag specifying edge (0x01) or inband (0x00) channel for 802.11g Legacy ETSI
		2GHz.Ctl.BandEdge[10][0]=0x00; 2GHz.Ctl.BandEdge[10][1]=0x01; 2GHz.Ctl.BandEdge[10][2]=0x01; 2GHz.Ctl.BandEdge[10][3]=0x01;	Band edge flag specifying edge (0x01) or inband (0x00) channel for 802.11n HT20 ETSI
		2GHz.Ctl.BandEdge[11][0]=0x00; 2GHz.Ctl.BandEdge[11][1]=0x01; 2GHz.Ctl.BandEdge[11][2]=0x01; 2GHz.Ctl.BandEdge[11][3]=0x01;	Band edge flag specifying edge (0x01) or inband (0x00) channel for 802.11n HT40 ETSI
F (5 GHz)	(2 bits)	5GHz.Ctl.BandEdge[0][0]=0x01; 5GHz.Ctl.BandEdge[0][1]=0x01; 5GHz.Ctl.BandEdge[0][2]=0x01; 5GHz.Ctl.BandEdge[0][3]=0x01; 5GHz.Ctl.BandEdge[0][4]=0x01; 5GHz.Ctl.BandEdge[0][5]=0x01; 5GHz.Ctl.BandEdge[0][6]=0x01; 5GHz.Ctl.BandEdge[0][7]=0x00;	Band edge flag specifying edge (0x01) or inband (0x00) channel for 802.11a Legacy FCC
		5GHz.Ctl.BandEdge[1][0]=0x01; 5GHz.Ctl.BandEdge[1][1]=0x01; 5GHz.Ctl.BandEdge[1][2]=0x01; 5GHz.Ctl.BandEdge[1][3]=0x01; 5GHz.Ctl.BandEdge[1][4]=0x01; 5GHz.Ctl.BandEdge[1][5]=0x01; 5GHz.Ctl.BandEdge[1][6]=0x01; 5GHz.Ctl.BandEdge[1][7]=0x00;	Band edge flag specifying edge (0x01) or inband (0x00) channel for 802.11n HT20 FCC
		5GHz.Ctl.BandEdge[2][0]=0x00; 5GHz.Ctl.BandEdge[2][1]=0x01; 5GHz.Ctl.BandEdge[2][2]=0x00; 5GHz.Ctl.BandEdge[2][3]=0x01; 5GHz.Ctl.BandEdge[2][4]=0x01; 5GHz.Ctl.BandEdge[2][5]=0x01; 5GHz.Ctl.BandEdge[2][6]=0x01; 5GHz.Ctl.BandEdge[2][7]=0x01;	Band edge flag specifying edge (0x01) or inband (0x00) channel for 802.11n HT40 FCC
		5GHz.Ctl.BandEdge[3][0]=0x00; 5GHz.Ctl.BandEdge[3][1]=0x01; 5GHz.Ctl.BandEdge[3][2]=0x01; 5GHz.Ctl.BandEdge[3][3]=0x00; 5GHz.Ctl.BandEdge[3][4]=0x01; 5GHz.Ctl.BandEdge[3][5]=0x00; 5GHz.Ctl.BandEdge[3][6]=0x00; 5GHz.Ctl.BandEdge[3][7]=0x00;	Band edge flag specifying edge (0x01) or inband (0x00) channel for 802.11a Legacy Japan

Table 2-22. **CTL Band Edges and Power Data Parameter Descriptions (continued)**

Parameter Name	Bytes	get/set Command Parameter	Description
F (5 GHz) Cont.	(2 bits)	5GHz.Ctl.BandEdge[4][0]=0x01; 5GHz.Ctl.BandEdge[4][1]=0x01; 5GHz.Ctl.BandEdge[4][2]=0x01; 5GHz.Ctl.BandEdge[4][3]=0x00; 5GHz.Ctl.BandEdge[4][4]=0x00; 5GHz.Ctl.BandEdge[4][5]=0x00; 5GHz.Ctl.BandEdge[4][6]=0x00; 5GHz.Ctl.BandEdge[4][7]=0x00;	Band edge flag specifying edge (0x01) or inband (0x00) channel for 802.11n HT20 Japan
		5GHz.Ctl.BandEdge[5][0]=0x01; 5GHz.Ctl.BandEdge[5][1]=0x01; 5GHz.Ctl.BandEdge[5][2]=0x01; 5GHz.Ctl.BandEdge[5][3]=0x01; 5GHz.Ctl.BandEdge[5][4]=0x01; 5GHz.Ctl.BandEdge[5][5]=0x00; 5GHz.Ctl.BandEdge[5][6]=0x00; 5GHz.Ctl.BandEdge[5][7]=0x00;	Band edge flag specifying edge (0x01) or inband (0x00) channel for 802.11n HT40 Japan
		5GHz.Ctl.BandEdge[6][0]=0x01; 5GHz.Ctl.BandEdge[6][1]=0x01; 5GHz.Ctl.BandEdge[6][2]=0x01; 5GHz.Ctl.BandEdge[6][3]=0x01; 5GHz.Ctl.BandEdge[6][4]=0x01; 5GHz.Ctl.BandEdge[6][5]=0x01; 5GHz.Ctl.BandEdge[6][6]=0x01; 5GHz.Ctl.BandEdge[6][7]=0x01;	Band edge flag specifying edge (0x01) or inband (0x00) channel for 802.11a Legacy ETSI
		5GHz.Ctl.BandEdge[7][0]=0x01; 5GHz.Ctl.BandEdge[7][1]=0x01; 5GHz.Ctl.BandEdge[7][2]=0x00; 5GHz.Ctl.BandEdge[7][3]=0x01; 5GHz.Ctl.BandEdge[7][4]=0x01; 5GHz.Ctl.BandEdge[7][5]=0x01; 5GHz.Ctl.BandEdge[7][6]=0x01; 5GHz.Ctl.BandEdge[7][7]=0x00;	Band edge flag specifying edge (0x01) or inband (0x00) channel for 802.11n HT20 ETSI
		5GHz.Ctl.BandEdge[8][0]=0x01; 5GHz.Ctl.BandEdge[8][1]=0x00; 5GHz.Ctl.BandEdge[8][2]=0x01; 5GHz.Ctl.BandEdge[8][3]=0x01; 5GHz.Ctl.BandEdge[8][4]=0x01; 5GHz.Ctl.BandEdge[8][5]=0x01; 5GHz.Ctl.BandEdge[8][6]=0x00; 5GHz.Ctl.BandEdge[8][7]=0x01;	Band edge flag specifying edge (0x01) or inband (0x00) channel for 802.11n HT40 ETSI

Storage of Calibration Structure

To be able to fit the calibration structure on the on-chip OTP and in smaller EEPROMs, the calibration structure is stored in a compressed manner in these cases. Even on larger EEPROMs that could support storage of the calibration structure uncompressed, the ART2 user has an option of storing the data in compressed format and to specify how much of the EEPROM can be used.

NOTE: By default, Atheros has been storing the calibration data on all of the AR93xx/AR94xx/AR95xx module reference designs in a compressed manor using only the first 8 Kbits to prepare for switching to storing data on OTP.

This section provides some information about the implementation of the compression mechanism for reference only. It does not provide the necessary information to be able to decode the raw contents of EEPROM or OTP. To read or change the parameters, use the provided **get** and **set** commands.

Default Structures Held in Software

Compression Scheme

The compression scheme relies on storing default structures in software and then comparing the desired calibration structure against the defaults structure and storing the difference in EEPROM or OTP. Figure 2-20 shows how the scheme works and this section describes how it works for various scenarios.

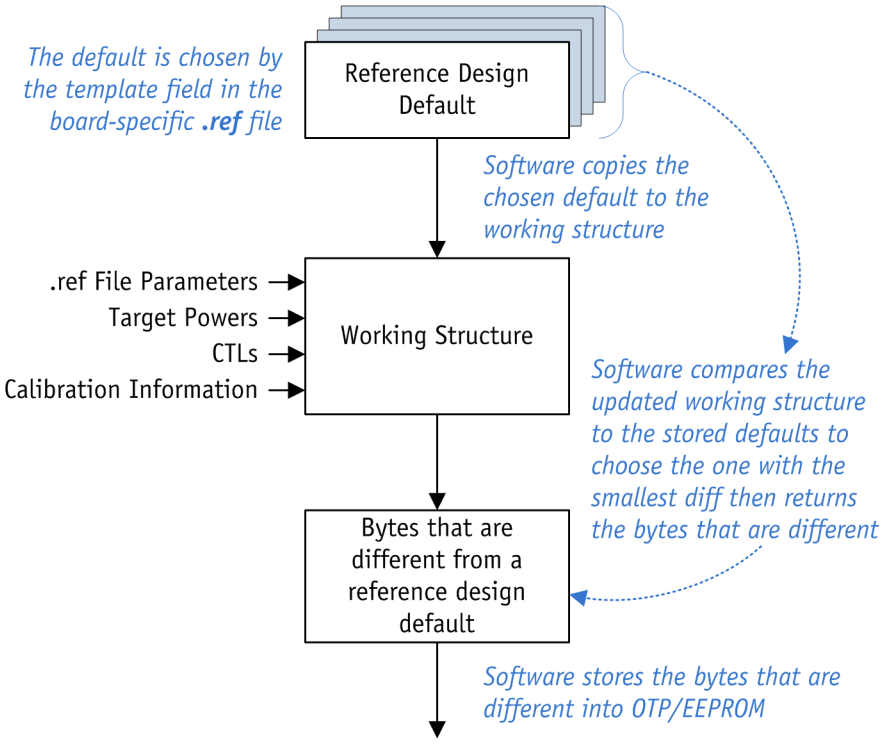

Figure 2-20. **Default Structure Comparison**

Blank Board

For a blank board, the calibration and storage scheme works as follows:

1. The ref file specifies the best default structure for the board being calibrated. If none is specified a generic default structure is selected.
2. Software makes a copy of this structure to be used as a working structure.
3. Set commands from the ref file, target power file and any additional set commands issued on the command line update to the working structure.
4. Calibration results update to the working structure.
5. When the **commit** command is issued, ART2 software compares the current working structure to all allowed default structures to see which diff results in the smallest number of bytes.
6. The difference in bytes between the default structure selected and the working structure is stored in EEPROM/OTP, along with details of which default was used.

NOTE: Although software contains multiple defaults, it is possible to limit step 5 to only select from a single default or a subset list, enabling the number of default structures stored in the production software to be reduced.

Reading Compressed Calibration Structure Back from EEPROM or OTP

Figure 2-21 depicts how the calibration structure is recreated in software from reading the compressed contents of the EEPROM or OTP.

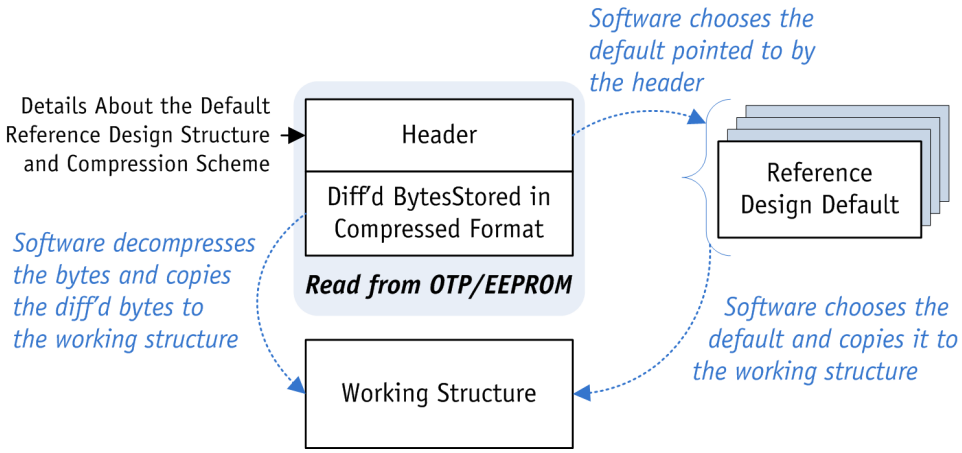

Figure 2-21. **Read Back Diagram**

1. Software reads the header in EEPTOM/OTP to find out which default structure should be selected.
2. This default structure is copied to the working structure.
3. Software uses the decompression scheme identified in the header to update the relevant bytes within the working structure.

Updating Parameters in EEPROM or OTP

Because in OTP, a block of data can only be updated once, subsequent calibrations or updates will be appended on the end of existing blocks as shown in Figure 2-22.

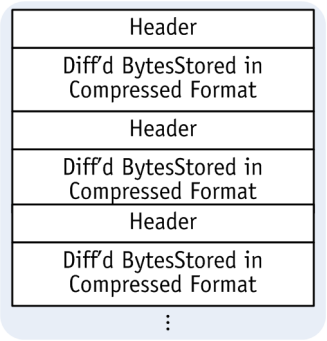

Figure 2-22. **subsequent Updates Append on the End of Existing Blocks**

For small updates, e.g., a MAC address update, the second block is an added difference from the first block and is decompressed by the sequence:

1. Software reads the first header in EEPROM/OTP to find out which default structure should be selected.
2. This default structure is copied to the working structure.
3. Software uses the decompression scheme identified in the header to update the relevant fields within the working structure.
4. Software finds the second header in the EEPROM/OTP and identifies that it is an addition to the first block.
5. Software updates the working structure from step 3 with the additional bytes and inserts into the relevant fields within the structure.

For larger updates, the compress scheme may find that a smaller number of bytes would be stored if a different default structure were used. So the second header would have details that points to a new default. In this case, the contents would be decompressed by the sequence:

1. Software reads the first header in EEPROM/OTP to find out which default structure should be selected.
2. This default structure is copied to the working structure.
3. Software uses the decompression scheme identified in the header to update the relevant fields within the working structure.
4. Software finds the second header and finds it is a difference based on a new default structure.
5. Software clears the working structure and copies in the new default structure.
6. Software uses the decompression scheme identified in the second header to update the fields within the working structure.

Because the on-chip OTP can only be written once, it has a limit to the number of times new structures can be appended, therefore care should be taken to update the boards only a limited number of times. Software gives an error if a new block of data cannot fit onto OTP. If using EEPROM but using the compressed storage scheme, it is recommended that the contents of the EEPROM be erased and recalibrated if multiple calibrations are required.

A

EEPROM Board Data Structure File

This appendix shows the board data structure file.

Format Description

```
/*
 * Copyright (c) 2008-2010, Atheros Communications Inc.
 *
 * Permission to use, copy, modify, and/or distribute this software
 * for any purpose with or without fee is hereby granted, provided
 * that the above copyright notice and this permission notice appear
 * in all copies. *
 * THE SOFTWARE IS PROVIDED "AS IS" AND THE AUTHOR DISCLAIMS ALL
 * WARRANTIES WITH REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED
 * WARRANTIES OF MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL THE
 * AUTHOR BE LIABLE FOR ANY SPECIAL, DIRECT, INDIRECT, OR
 * CONSEQUENTIAL DAMAGES OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS
 * OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT,
 * NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN
 * CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.
 */

#define AR5416_EEP_VER                0xE
#define AR5416_EEP_START_LOC          256
#define AR5416_NUM_5G_CAL_PIERS         8
#define AR5416_NUM_2G_CAL_PIERS         4
#define AR5416_NUM_5G_20_TARGET_POWERS  8
#define AR5416_NUM_5G_40_TARGET_POWERS  8
#define AR5416_NUM_2G_CCK_TARGET_POWERS 3
#define AR5416_NUM_2G_20_TARGET_POWERS  4
#define AR5416_NUM_2G_40_TARGET_POWERS  4
#define AR5416_NUM_CTLS                24
#define AR5416_NUM_BAND_EDGES           8
#define AR5416_NUM_PD_GAINS             4
#define AR5416_PD_GAIN_ICEPTS           5
#define AR5416_EEPROM_MODAL_SPURS       5
#define AR5416_MAX_CHAINS               3
```

```
#ifndef _ATH_AR9300_EEP_H_
#define _ATH_AR9300_EEP_H_

#include "opt_ah.h"

#if defined(WIN32) || defined(WIN64)
#pragma pack (push, ar9300, 1)
#endif

#define MSTATE 100
#define MOUTPUT 2048
#define MDEFAULT 15
#define MVALUE 100

enum CompressAlgorithm
{
    _CompressNone=0,
    _CompressLzma,
    _CompressPairs,
    _CompressBlock,
    _Compress4,
    _Compress5,
    _Compress6,
    _Compress7,
};

enum
{
    CalibrationDataNone=0,
    CalibrationDataFlash,
    CalibrationDataEeprom,
    CalibrationDataOtp,
};

//
// DO NOT CHANGE THE DEFINTIONS OF THESE SYMBOLS.
// Add additional definitions to the end.
// Yes, the first one is 2. Do not use 0 or 1.
//
enum Ar9300EepromTemplate
{
    Ar9300EepromTemplateGeneric=2,
    Ar9300EepromTemplateHB112=3,
    Ar9300EepromTemplateHB116=4,
    Ar9300EepromTemplateXB112=5,
    Ar9300EepromTemplateXB113=6,
    Ar9300EepromTemplateXB114=7,
    Ar9300EepromTemplateTB417=8,
    Ar9300EepromTemplateAP111=9,
    Ar9300EepromTemplateAP121=10,
    Ar9300EepromTemplateHornetGeneric=11,
};

#define Ar9300EepromTemplateDefault Ar9300EepromTemplateGeneric
#define Ar9300EepromFormatDefault 2

#define ReferenceCurrent 0
#define CompressionHeaderLength 4
#define CompressionChecksumLength 2
```

```
#define OSPREY_EEP_VER               0xD000
#define OSPREY_EEP_VER_MINOR_MASK    0xFFF
#define OSPREY_EEP_MINOR_VER_1       0x1
#define OSPREY_EEP_MINOR_VER         OSPREY_EEP_MINOR_VER_1

// 16-bit offset location start of calibration struct
#define OSPREY_EEP_START_LOC         256
#define OSPREY_NUM_5G_CAL_PIERS      8
#define OSPREY_NUM_2G_CAL_PIERS      3
#define OSPREY_NUM_5G_20_TARGET_POWERS  8
#define OSPREY_NUM_5G_40_TARGET_POWERS  8
#define OSPREY_NUM_2G_CCK_TARGET_POWERS 2
#define OSPREY_NUM_2G_20_TARGET_POWERS  3
#define OSPREY_NUM_2G_40_TARGET_POWERS  3
//#define OSPREY_NUM_CTLS              21
#define OSPREY_NUM_CTLS_5G           9
#define OSPREY_NUM_CTLS_2G           12
#define OSPREY_CTL_MODE_M            0xF
#define OSPREY_NUM_BAND_EDGES_5G     8
#define OSPREY_NUM_BAND_EDGES_2G     4
#define OSPREY_NUM_PD_GAINS          4
#define OSPREY_PD_GAINS_IN_MASK      4
#define OSPREY_PD_GAIN_ICEPTS        5
#define OSPREY_EEPROM_MODAL_SPURS    5
#define OSPREY_MAX_RATE_POWER        63
#define OSPREY_NUM_PDADC_VALUES      128
#define OSPREY_NUM_RATES             16
#define OSPREY_BCHAN_UNUSED          0xFF
#define OSPREY_MAX_PWR_RANGE_IN_HALF_DB 64
#define OSPREY_OPFLAGS_11A           0x01
#define OSPREY_OPFLAGS_11G           0x02
#define OSPREY_OPFLAGS_5G_HT40       0x04
#define OSPREY_OPFLAGS_2G_HT40       0x08
#define OSPREY_OPFLAGS_5G_HT20       0x10
#define OSPREY_OPFLAGS_2G_HT20       0x20
#define OSPREY_EEPMISC_BIG_ENDIAN    0x01
#define OSPREY_EEPMISC_WOW           0x02
#define OSPREY_CUSTOMER_DATA_SIZE    20

#define FREQ2FBIN(x,y) ((y) ? ((x) - 2300) : (((x) - 4800) / 5))
#define FBIN2FREQ(x,y) ((y) ? (2300 + x) : (4800 + 5 * x))
#define OSPREY_MAX_CHAINS            3
#define OSPREY_ANT_16S               25
#define OSPREY_FUTURE_MODAL_SZ       6
#define OSPREY_NUM_ANT_CHAIN_FIELDS     7
#define OSPREY_NUM_ANT_COMMON_FIELDS    4
#define OSPREY_SIZE_ANT_CHAIN_FIELD     3
#define OSPREY_SIZE_ANT_COMMON_FIELD    4
#define OSPREY_ANT_CHAIN_MASK           0x7
#define OSPREY_ANT_COMMON_MASK          0xf
#define OSPREY_CHAIN_0_IDX              0
#define OSPREY_CHAIN_1_IDX              1
#define OSPREY_CHAIN_2_IDX              2
#define OSPREY_1_CHAINMASK              1
#define OSPREY_2LOHI_CHAINMASK          5
#define OSPREY_2LOMID_CHAINMASK         3
#define OSPREY_3_CHAINMASK              7
#define OSPREY_PWR_TABLE_OFFSET_DB     -5 /* value must be tuned */
#define AR928X_NUM_ANT_CHAIN_FIELDS     6
#define AR928X_SIZE_ANT_CHAIN_FIELD     2
#define AR928X_ANT_CHAIN_MASK           0x3
```

```
/* Delta from which to start power to pdadc table */
/* This offset is used in both open loop and closed loop power control
 * schemes. In open loop power control, it is not really needed, but
 * for the "sake of consistency" it was kept.
 * For certain AP designs, this value is overwritten by the value
 * in the flag "pwrTableOffset" just before writing the pdadc vs pwr
 * into the chip registers.
 */
#define OSPREY_PWR_TABLE_OFFSET  0

//enable flags for voltage and temp compensation
#define ENABLE_TEMP_COMPENSATION 0x01
#define ENABLE_VOLT_COMPENSATION 0x02

#define FLASH_BASE_CALDATA_OFFSET  0x1000
#define AR9300_EEPROM_SIZE 16*1024  // byte addressable
#define FIXED_CCA_THRESHOLD 15

typedef struct eepFlags {
    u_int8_t  opFlags;
    u_int8_t  eepMisc;
} __packed EEP_FLAGS;

typedef enum targetPowerHTRates {
    HT_TARGET_RATE_0_8_16,
    HT_TARGET_RATE_1_3_9_11_17_19,
    HT_TARGET_RATE_4,
    HT_TARGET_RATE_5,
    HT_TARGET_RATE_6,
    HT_TARGET_RATE_7,
    HT_TARGET_RATE_12,
    HT_TARGET_RATE_13,
    HT_TARGET_RATE_14,
    HT_TARGET_RATE_15,
    HT_TARGET_RATE_20,
    HT_TARGET_RATE_21,
    HT_TARGET_RATE_22,
    HT_TARGET_RATE_23
}TARGET_POWER_HT_RATES;

const static int mapRate2Index[24]=
{
    0,1,1,1,2,
    3,4,5,0,1,
    1,1,6,7,8,
    9,0,1,1,1,
    10,11,12,13
};

typedef enum targetPowerLegacyRates {
    LEGACY_TARGET_RATE_6_24,
    LEGACY_TARGET_RATE_36,
    LEGACY_TARGET_RATE_48,
    LEGACY_TARGET_RATE_54
}TARGET_POWER_LEGACY_RATES;
```

```
typedef enum targetPowerCckRates {
    LEGACY_TARGET_RATE_1L_5L,
    LEGACY_TARGET_RATE_5S,
    LEGACY_TARGET_RATE_11L,
    LEGACY_TARGET_RATE_11S
}TARGET_POWER_CCK_RATES;

#define OSPREY_CHECKSUM_LOCATION (OSPREY_EEP_START_LOC + 1)

typedef struct osprey_BaseEepHeader {
    u_int16_t  regDmn[2]; //Does this need to be outside of this structure, if it gets
written after calibration
    u_int8_t   txrxMask;  //4 bits tx and 4 bits rx
    EEP_FLAGS  opCapFlags;
    u_int8_t   rfSilent;
    u_int8_t   blueToothOptions;
    u_int8_t   deviceCap;
    u_int8_t   deviceType; // takes lower byte in eeprom location
    int8_t     pwrTableOffset; // offset in dB to be added to beginning of pdadc table
in calibration
    u_int8_t   params_for_tuning_caps[2];
    u_int8_t   featureEnable; //bit0 - enable tx temp comp
                              //bit1 - enable tx volt comp
                              //bit2 - enable fastClock - default to 1
                              //bit3 - enable doubling - default to 1
                          //bit4 - enable internal regulator default 1
    u_int8_t   miscConfiguration; //misc flags: bit0 - turn down
                      drivestrength
    u_int8_t   eepromWriteEnableGpio;
    u_int8_t   wlanDisableGpio;
    u_int8_t   wlanLedGpio;
    u_int8_t   rxBandSelectGpio;
    u_int8_t   txrxgain;
    u_int32_t   swreg;    // SW controlled internal regulator fields
} __packed OSPREY_BASE_EEP_HEADER;

typedef struct osprey_BaseExtension_1 {
    u_int8_t    future[14];
} __packed OSPREY_BASE_EXTENSION_1;

typedef struct osprey_BaseExtension_2 {
    int8_t    tempSlopeLow;
    int8_t    tempSlopeHigh;
    u_int8_t   xatten1DBLow[OSPREY_MAX_CHAINS];           // 3  //xatten1_db for merlin
(0xa20c/b20c 5:0)
    u_int8_t   xatten1MarginLow[OSPREY_MAX_CHAINS];          // 3  //xatten1_margin for
merlin (0xa20c/b20c 16:12
    u_int8_t   xatten1DBHigh[OSPREY_MAX_CHAINS];          // 3  //xatten1_db for merlin
(0xa20c/b20c 5:0)
    u_int8_t   xatten1MarginHigh[OSPREY_MAX_CHAINS];          // 3  //xatten1_margin for
merlin (0xa20c/b20c 16:12
} __packed OSPREY_BASE_EXTENSION_2;

typedef struct spurChanStruct {
    u_int16_t spurChan;
    u_int8_t  spurRangeLow;
    u_int8_t  spurRangeHigh;
} __packed SPUR_CHAN;
```

```
//Note the order of the fields in this structure has been optimized to put all fields
likely to change together
typedef struct ospreyModalEepHeader {
    u_int32_t  antCtrlCommon; // 4   idle, t1, t2, b (4 bits/setting)
    u_int32_t  antCtrlCommon2; // 4    ra1l1, ra2l1, ra1l2, ra2l2,
                    ra12
    u_int16_t  antCtrlChain[OSPREY_MAX_CHAINS]; // 6   idle, t, r, rx1, rx12, b (2 bits
each)
    u_int8_t   xatten1DB[OSPREY_MAX_CHAINS];           // 3  //xatten1_db for merlin
(0xa20c/b20c 5:0)
    u_int8_t   xatten1Margin[OSPREY_MAX_CHAINS];          // 3  //xatten1_margin for
merlin (0xa20c/b20c 16:12
    int8_t    tempSlope;
    int8_t    voltSlope;
    u_int8_t spurChans[OSPREY_EEPROM_MODAL_SPURS];  // spur channels in usual fbin
coding format
    int8_t    noiseFloorThreshCh[OSPREY_MAX_CHAINS]; // 3    //Check if the register is
per chain
    u_int8_t   ob[OSPREY_MAX_CHAINS];                  // 3
    u_int8_t   db_stage2[OSPREY_MAX_CHAINS];           // 3
    u_int8_t   db_stage3[OSPREY_MAX_CHAINS];           // 3
    u_int8_t   db_stage4[OSPREY_MAX_CHAINS];           // 3
    u_int8_t   xpaBiasLvl;                             // 1
    u_int8_t   txFrameToDataStart;                     // 1
    u_int8_t   txFrameToPaOn;                          // 1
    u_int8_t   txClip;                                      // 4 bits tx_clip, 4 bits
dac_scale_cck
    int8_t    antennaGain;                            // 1
    u_int8_t   switchSettling;                         // 1
    int8_t    adcDesiredSize;                         // 1
    u_int8_t   txEndToXpaOff;                          // 1
    u_int8_t   txEndToRxOn;                            // 1
    u_int8_t   txFrameToXpaOn;                         // 1
    u_int8_t   thresh62;                               // 1
   u_int32_t papdRateMaskHt20;
   u_int32_t papdRateMaskHt40;
   u_int8_t    futureModal[10];
   // last 12 bytes stolen and moved to newly created base extension structure
} __packed OSPREY_MODAL_EEP_HEADER;                     // == 100 B

typedef struct ospCalDataPerFreqOpLoop {
    int8_t refPower;    /*   */
    u_int8_t voltMeas; /* pdadc voltage at power measurement */
    u_int8_t tempMeas;  /* pcdac used for power measurement   */
    int8_t rxNoisefloorCal; /*range is -60 to -127 create a mapping equation 1db
resolution */
    int8_t rxNoisefloorPower; /*range is same as noisefloor */
    u_int8_t rxTempMeas; /*temp measured when noisefloor cal was performed */
} __packed OSP_CAL_DATA_PER_FREQ_OP_LOOP;

typedef struct CalTargetPowerLegacy {
    u_int8_t  tPow2x[4];
} __packed CAL_TARGET_POWER_LEG;

typedef struct ospCalTargetPowerHt {
    u_int8_t  tPow2x[14];
} __packed OSP_CAL_TARGET_POWER_HT;
```

```
#if AH_BYTE_ORDER == AH_BIG_ENDIAN
typedef struct CalCtlEdgePwr {
    u_int8_t  flag  :2,
              tPower :6;
} __packed CAL_CTL_EDGE_PWR;
#else
typedef struct CalCtlEdgePwr {
    u_int8_t  tPower :6,
             flag   :2;
} __packed CAL_CTL_EDGE_PWR;
#endif

typedef struct ospCalCtlData_5G {
    CAL_CTL_EDGE_PWR  ctlEdges[OSPREY_NUM_BAND_EDGES_5G];
} __packed OSP_CAL_CTL_DATA_5G;

typedef struct ospCalCtlData_2G {
    CAL_CTL_EDGE_PWR  ctlEdges[OSPREY_NUM_BAND_EDGES_2G];
} __packed OSP_CAL_CTL_DATA_2G;


typedef struct ospreyEeprom {
    u_int8_t  eepromVersion;
    u_int8_t  templateVersion;
    u_int8_t  macAddr[6];
    u_int8_t  custData[OSPREY_CUSTOMER_DATA_SIZE];

    OSPREY_BASE_EEP_HEADER    baseEepHeader;

    OSPREY_MODAL_EEP_HEADER   modalHeader2G;
    OSPREY_BASE_EXTENSION_1 base_ext1;
    u_int8_t            calFreqPier2G[OSPREY_NUM_2G_CAL_PIERS];
    OSP_CAL_DATA_PER_FREQ_OP_LOOP
calPierData2G[OSPREY_MAX_CHAINS][OSPREY_NUM_2G_CAL_PIERS];
    u_int8_t calTarget_freqbin_Cck[OSPREY_NUM_2G_CCK_TARGET_POWERS];
    u_int8_t calTarget_freqbin_2G[OSPREY_NUM_2G_20_TARGET_POWERS];
    u_int8_t calTarget_freqbin_2GHT20[OSPREY_NUM_2G_20_TARGET_POWERS];
    u_int8_t calTarget_freqbin_2GHT40[OSPREY_NUM_2G_40_TARGET_POWERS];
    CAL_TARGET_POWER_LEG calTargetPowerCck[OSPREY_NUM_2G_CCK_TARGET_POWERS];
    CAL_TARGET_POWER_LEG calTargetPower2G[OSPREY_NUM_2G_20_TARGET_POWERS];
    OSP_CAL_TARGET_POWER_HT  calTargetPower2GHT20[OSPREY_NUM_2G_20_TARGET_POWERS];
    OSP_CAL_TARGET_POWER_HT  calTargetPower2GHT40[OSPREY_NUM_2G_40_TARGET_POWERS];
    u_int8_t   ctlIndex_2G[OSPREY_NUM_CTLS_2G];
    u_int8_t   ctl_freqbin_2G[OSPREY_NUM_CTLS_2G][OSPREY_NUM_BAND_EDGES_2G];
    OSP_CAL_CTL_DATA_2G   ctlPowerData_2G[OSPREY_NUM_CTLS_2G];

    OSPREY_MODAL_EEP_HEADER   modalHeader5G;
    OSPREY_BASE_EXTENSION_2 base_ext2;
    u_int8_t            calFreqPier5G[OSPREY_NUM_5G_CAL_PIERS];
    OSP_CAL_DATA_PER_FREQ_OP_LOOP
calPierData5G[OSPREY_MAX_CHAINS][OSPREY_NUM_5G_CAL_PIERS];
    u_int8_t calTarget_freqbin_5G[OSPREY_NUM_5G_20_TARGET_POWERS];
    u_int8_t calTarget_freqbin_5GHT20[OSPREY_NUM_5G_20_TARGET_POWERS];
    u_int8_t calTarget_freqbin_5GHT40[OSPREY_NUM_5G_40_TARGET_POWERS];
    CAL_TARGET_POWER_LEG calTargetPower5G[OSPREY_NUM_5G_20_TARGET_POWERS];
    OSP_CAL_TARGET_POWER_HT  calTargetPower5GHT20[OSPREY_NUM_5G_20_TARGET_POWERS];
    OSP_CAL_TARGET_POWER_HT  calTargetPower5GHT40[OSPREY_NUM_5G_40_TARGET_POWERS];
    u_int8_t   ctlIndex_5G[OSPREY_NUM_CTLS_5G];
    u_int8_t   ctl_freqbin_5G[OSPREY_NUM_CTLS_5G][OSPREY_NUM_BAND_EDGES_5G];
    OSP_CAL_CTL_DATA_5G   ctlPowerData_5G[OSPREY_NUM_CTLS_5G];
} __packed ar9300_eeprom_t;
```

```
/*
** SWAP Functions
** used to read EEPROM data, which is apparently stored in little
** endian form.  We have included both forms of the swap functions,
** one for big endian and one for little endian.  The indices of the
** array elements are the differences
*/
#if AH_BYTE_ORDER == AH_BIG_ENDIAN

#define AR9300_EEPROM_MAGIC         0x5aa5
#define SWAP16(_x) ( (u_int16_t)( (((const u_int8_t *)(&_x))[0] ) |\
                     ( ( (const u_int8_t *)( &_x ) )[1]<< 8) ) )

#define SWAP32(_x) ((u_int32_t)(                        \
                    (((const u_int8_t *)(&_x))[0]) |        \
                    (((const u_int8_t *)(&_x))[1]<< 8) |    \
                    (((const u_int8_t *)(&_x))[2]<<16) |    \
                    (((const u_int8_t *)(&_x))[3]<<24)))

#else // AH_BYTE_ORDER

#define AR9300_EEPROM_MAGIC         0xa55a
#define    SWAP16(_x) ( (u_int16_t)( (((const u_int8_t *)(&_x))[1] ) |\
                       ( ( (const u_int8_t *)( &_x ) )[0]<< 8) ) )

#define SWAP32(_x) ((u_int32_t)(                        \
                    (((const u_int8_t *)(&_x))[3]) |        \
                    (((const u_int8_t *)(&_x))[2]<< 8) |    \
                    (((const u_int8_t *)(&_x))[1]<<16) |    \
                    (((const u_int8_t *)(&_x))[0]<<24)))

#endif // AH_BYTE_ORDER

// OTP registers for OSPREY

#define AR_GPIO_IN_OUT            0x4048 // GPIO input / output register
#define OTP_MEM_START_ADDRESS     0x14000
#define OTP_STATUS0_OTP_SM_BUSY   0x00015f18
#define OTP_STATUS1_EFUSE_READ_DATA 0x00015f1c

#define OTP_LDO_CONTROL_ENABLE    0x00015f24
#define OTP_LDO_STATUS_POWER_ON   0x00015f2c
#define OTP_INTF0_EFUSE_WR_ENABLE_REG_V 0x00015f00

#define AR9300_EEPROM_MAGIC_OFFSET  0x0
/* reg_off = 4 * (eep_off) */
#define AR9300_EEPROM_S             2
#define AR9300_EEPROM_OFFSET        0x2000
#ifdef AR9100
#define AR9300_EEPROM_START_ADDR    0x1fff1000
#else
#define AR9300_EEPROM_START_ADDR    0x503f1200
#endif
#define AR9300_FLASH_CAL_START_OFFSET    0x1000
#define AR9300_EEPROM_MAX           0xae0
#define IS_EEP_MINOR_V3(_ahp) (ar9300EepromGet((_ahp), EEP_MINOR_REV)  >=
AR9300_EEP_MINOR_VER_3)
```

```
/* RF silent fields in \ */
#define EEP_RFSILENT_ENABLED        0x0001  /* bit 0: enabled/disabled */
#define EEP_RFSILENT_ENABLED_S      0       /* bit 0: enabled/disabled */
#define EEP_RFSILENT_POLARITY       0x0002  /* bit 1: polarity */
#define EEP_RFSILENT_POLARITY_S     1       /* bit 1: polarity */
#define EEP_RFSILENT_GPIO_SEL       0x001c  /* bits 2..4: gpio PIN */
#define EEP_RFSILENT_GPIO_SEL_S     2       /* bits 2..4: gpio PIN */
#define AR9300_EEP_VER               0xE
#define AR9300_BCHAN_UNUSED          0xFF
#define AR9300_MAX_RATE_POWER        63

typedef enum {
    CALDATA_AUTO=0,
    CALDATA_EEPROM,
    CALDATA_FLASH,
    CALDATA_OTP
} CALDATA_TYPE;

typedef enum {
    EEP_NFTHRESH_5,
    EEP_NFTHRESH_2,
    EEP_MAC_MSW,
    EEP_MAC_MID,
    EEP_MAC_LSW,
    EEP_REG_0,
    EEP_REG_1,
    EEP_OP_CAP,
    EEP_OP_MODE,
    EEP_RF_SILENT,
    EEP_OB_5,
    EEP_DB_5,
    EEP_OB_2,
    EEP_DB_2,
    EEP_MINOR_REV,
    EEP_TX_MASK,
    EEP_RX_MASK,
    EEP_FSTCLK_5G,
    EEP_RXGAIN_TYPE,
    EEP_OL_PWRCTRL,
    EEP_TXGAIN_TYPE,
    EEP_RC_CHAIN_MASK,
    EEP_DAC_HPWR_5G,
    EEP_FRAC_N_5G,
    EEP_DEV_TYPE,
    EEP_TEMPSENSE_SLOPE,
    EEP_TEMPSENSE_SLOPE_PAL_ON,
    EEP_PWR_TABLE_OFFSET,
    EEP_DRIVE_STRENGTH,
    EEP_INTERNAL_REGULATOR,
    EEP_SWREG,
   EEP_PAPRD_ENABLED
} EEPROM_PARAM;
```

```
typedef enum ar9300_Rates {
    ALL_TARGET_LEGACY_6_24,
    ALL_TARGET_LEGACY_36,
    ALL_TARGET_LEGACY_48,
    ALL_TARGET_LEGACY_54,
    ALL_TARGET_LEGACY_1L_5L,
    ALL_TARGET_LEGACY_5S,
    ALL_TARGET_LEGACY_11L,
    ALL_TARGET_LEGACY_11S,
    ALL_TARGET_HT20_0_8_16,
    ALL_TARGET_HT20_1_3_9_11_17_19,
    ALL_TARGET_HT20_4,
    ALL_TARGET_HT20_5,
    ALL_TARGET_HT20_6,
    ALL_TARGET_HT20_7,
    ALL_TARGET_HT20_12,
    ALL_TARGET_HT20_13,
    ALL_TARGET_HT20_14,
    ALL_TARGET_HT20_15,
    ALL_TARGET_HT20_20,
    ALL_TARGET_HT20_21,
    ALL_TARGET_HT20_22,
    ALL_TARGET_HT20_23,
    ALL_TARGET_HT40_0_8_16,
    ALL_TARGET_HT40_1_3_9_11_17_19,
    ALL_TARGET_HT40_4,
    ALL_TARGET_HT40_5,
    ALL_TARGET_HT40_6,
    ALL_TARGET_HT40_7,
    ALL_TARGET_HT40_12,
    ALL_TARGET_HT40_13,
    ALL_TARGET_HT40_14,
    ALL_TARGET_HT40_15,
    ALL_TARGET_HT40_20,
    ALL_TARGET_HT40_21,
    ALL_TARGET_HT40_22,
    ALL_TARGET_HT40_23,
    ar9300RateSize
} AR9300_RATES;


/**************************************************************************
 * fbin2freq
 *
 * Get channel value from binary representation held in eeprom
 * RETURNS: the frequency in MHz
 */
static inline u_int16_t
fbin2freq(u_int8_t fbin, HAL_BOOL is2GHz)
{
    /*
    * Reserved value 0xFF provides an empty definition both as
    * an fbin and as a frequency - do not convert
    */
    if (fbin == AR9300_BCHAN_UNUSED)
    {
        return fbin;
    }

    return (u_int16_t)((is2GHz) ? (2300 + fbin) : (4800 + 5 * fbin));
}
```

```
extern int CompressionHeaderUnpack(u_int8_t *best, int *code, int *reference, int
*length, int *major, int *minor);
extern void Ar9300EepromFormatConvert(ar9300_eeprom_t *mptr);
extern HAL_BOOL ar9300EepromRestore(struct ath_hal *ah);
extern int ar9300EepromRestoreInternal(struct ath_hal *ah, ar9300_eeprom_t *mptr, int /
*msize*/);
extern int ar9300EepromBaseAddress(struct ath_hal *ah);
extern int ar9300EepromVolatile(struct ath_hal *ah);
extern int ar9300EepromLowLimit(struct ath_hal *ah);
extern u_int16_t ar9300CompressionChecksum(u_int8_t *data, int dsize);
extern int ar9300CompressionHeaderUnpack(u_int8_t *best, int *code, int *reference, int
*length, int *major, int *minor);

extern u_int16_t ar9300EepromStructSize(void);
extern ar9300_eeprom_t *ar9300EepromStructInit(int defaultIndex);
extern ar9300_eeprom_t *ar9300EepromStructGet(void);
extern ar9300_eeprom_t *ar9300EepromStructDefault(int defaultIndex);
extern ar9300_eeprom_t *ar9300EepromStructDefaultFindById(int ver);
extern int ar9300EepromStructDefaultMany(void);
extern int ar9300EepromUpdateCalPier(int pierIdx, int freq, int chain,
                        int pwrCorrection, int voltMeas, int tempMeas);
extern int ar9300PowerControlOverride(struct ath_hal *ah, int frequency, int
*correction, int *voltage, int *temperature);

extern void ar9300EepromDisplayCalData(int for2GHz);
extern void ar9300EepromDisplayAll(void);
extern void ar9300SetTargetPowerFromEeprom(struct ath_hal *ah, u_int16_t freq);
extern int ar9300TransmitPowerRegWrite(struct ath_hal *ah, u_int8_t *pPwrArray);
extern int ar9300CompressionHeaderUnpack(u_int8_t *best, int *code, int *reference, int
*length, int *major, int *minor);

extern u_int8_t ar9300EepromGetLegacyTrgtPwr(struct ath_hal *ah, u_int16_t rateIndex,
u_int16_t freq, HAL_BOOL is2GHz);
extern u_int8_t ar9300EepromGetHT20TrgtPwr(struct ath_hal *ah, u_int16_t rateIndex,
u_int16_t freq, HAL_BOOL is2GHz);
extern u_int8_t ar9300EepromGetHT40TrgtPwr(struct ath_hal *ah, u_int16_t rateIndex,
u_int16_t freq, HAL_BOOL is2GHz);
extern u_int8_t ar9300EepromGetCckTrgtPwr(struct ath_hal *ah, u_int16_t rateIndex,
u_int16_t freq);
extern HAL_BOOL ar9300InternalRegulatorApply(struct ath_hal *ah);
extern HAL_BOOL ar9300DriveStrengthApply(struct ath_hal *ah);
extern HAL_BOOL ar9300AttenuationApply(struct ath_hal *ah, u_int16_t channel);


extern int32_t ar9300MacAdressGet(u_int8_t *mac);
extern int32_t ar9300CustomerDataGet(u_int8_t *data, int32_t len);
extern int32_t ar9300ReconfigDriveStrengthGet(void);
extern int32_t ar9300EnableTempCompensationGet(void);
extern int32_t ar9300EnableVoltCompensationGet(void);
extern int32_t ar9300FastClockEnableGet(void);
extern int32_t ar9300EnableDoublingGet(void);
```

```
extern u_int16_t *ar9300RegulatoryDomainGet(struct ath_hal *ah);
extern int32_t ar9300EepromWriteEnableGpioGet(struct ath_hal *ah);
extern int32_t ar9300WlanLedGpioGet(struct ath_hal *ah);
extern int32_t ar9300WlanDisableGpioGet(struct ath_hal *ah);
extern int32_t ar9300RxBandSelectGpioGet(struct ath_hal *ah);
extern int32_t ar9300RxGainIndexGet(struct ath_hal *ah);
extern int32_t ar9300TxGainIndexGet(struct ath_hal *ah);
extern int32_t ar9300XpaBiasLevelGet(struct ath_hal *ah, HAL_BOOL is2GHz);
extern HAL_BOOL ar9300XpaBiasLevelApply(struct ath_hal *ah, HAL_BOOL is2GHz);
extern u_int32_t ar9300AntCtrlCommonGet(struct ath_hal *ah, HAL_BOOL is2GHz);
extern u_int32_t ar9300AntCtrlCommon2Get(struct ath_hal *ah, HAL_BOOL is2GHz);
extern u_int16_t ar9300AntCtrlChainGet(struct ath_hal *ah, int chain, HAL_BOOL is2GHz);
extern HAL_BOOL ar9300AntCtrlApply(struct ath_hal *ah, HAL_BOOL is2GHz);
/* since valid noise floor values are negative, returns 1 on error */
extern int32_t ar9300NoiseFloorCalOrPowerGet(
    struct ath_hal *ah, int32_t frequency, int32_t ichain, HAL_BOOL use_cal);
#define ar9300NoiseFloorGet(ah, frequency, ichain) \
    ar9300NoiseFloorCalOrPowerGet(ah, frequency, ichain, 1/*use_cal*/)
#define ar9300NoiseFloorPowerGet(ah, frequency, ichain) \
    ar9300NoiseFloorCalOrPowerGet(ah, frequency, ichain, 0/*use_cal*/)
extern void ar9300EepromTemplatePreference(int32_t value);
extern void ar9300CalibrationDataSet(struct ath_hal *ah, int32_t source);
extern int32_t ar9300CalibrationDataGet(struct ath_hal *ah);
extern HAL_BOOL ar9300CalibrationDataReadFlash(struct ath_hal *ah, long address,
u_int8_t *buffer, int many);
extern HAL_BOOL ar9300CalibrationDataReadEeprom(struct ath_hal *ah, long address,
u_int8_t *buffer, int many);
extern HAL_BOOL ar9300CalibrationDataReadOtp(struct ath_hal *ah, long address, u_int8_t
*buffer, int many);
extern HAL_BOOL ar9300CalibrationDataRead(struct ath_hal *ah, long address, u_int8_t
*buffer, int many);
extern int32_t ar9300EepromSize(struct ath_hal *ah);
extern HAL_BOOL ar9300CalibrationDataReadArray(struct ath_hal *ah, int address, u_int8_t
*buffer, int many);



#if defined(WIN32) || defined(WIN64)
#pragma pack (pop, ar9300)
#endif

#endif  /* _ATH_AR9300_EEP_H_ */
```

B

Characterizing Temperature Slope

Introduction

The parameter tempSlope characterizes the RF power dependence on temperature. It is an important part of power control. Stored in EEPROM, it can take different values for different module designs.

TempSlope may vary with channel frequency. NART supports storing one value for the 2G band, and three values for low-, mid-, and high-5 G band.

TempSlope measurement involves measuring RF output power at various temperatures and calculating the slope of the power-temperature curve. RF power is measured at the module output with power meter; temperature is measured by an on-chip temperature sensor that reports a digitized reading. The reading is available at the baseband register 0xA454[7:0].

It is stressed that tempSlope is conceptually the slope of a 2-D line, with the y-axis being power and the x-axis being temperature. Its unit is:

```
(2^8) * gain_change / temp_change
```

The (2^8) is a scaling factor for digital processing. The unit of gain_change is half-dB; that of temp_change is LSB, because the temperature is read from an on-chip sensor with digitized output. This example of tempSlope calculation assumes output powers are measured at two temperatures and gets the data:

Temp	Power (dBm)
110	8
100	7

Then, tempSlope is calculated as:

```
(2^8) *
((7 - 8) / 0.5)    # Unit is half-dB
(100 - 110)
```

Which equals:

51.2

Detailed Procedure

1. Put the module in a temperature chamber.
2. Configure the chip to be at 2437 MHz.
3. Set the chamber temperature to a low value (e.g.,–20 °C)
4. Let the module continuously transmit for at least 10 minutes to let the module heat up and let the temperature stabilize.
5. Record output power.
6. Record temperature by reading the sensor output at register 0xA454[7:0] using the command:
   fr a=BB_therm_adc_3.latest_therm_value
   or
   rr a=a454;
   Increase the temperature and repeat steps 4 through 6. Atheros recommends taking at least four measurements between a change of 80 °C in temperature to ensure enough coverage.
7. Calculate tempSlope. Because the two-dimensional plane contains more than two points, the slope may be computed with linear regression. Remember the scaling factor (2^8) and the half-dB unit for power.
8. Repeat steps 2 to 8 to obtain the tempSlopes for 5180, 5500, and 5795 MHz.

Atheros Communications, Incorporated
1700 Technology Drive
San Jose, CA 95110
tel: 408.773.5200
fax: 408.773.9940
www.atheros.com