

BCM430x - 802.1 OCTL Interface

6/10/03

Revision 1.7

REVISION HISTORY

Revision	Date	Change Description
1.7	06/10/03	Updates to support Bulls features
1.6.5	03/27/03	Minor updates

Broadcom Corporation
162/15 Alton Parkway
P.O. Box 57013
Irvine, California 92619-7013
© 2003 by Broadcom Corporation
All rights reserved
Printed in the U.S.A.

Broadcom® and the pulse-logo are trademarks of Broadcom Corporation and/or it subsidiaries in the United States and certain other countries. All other trademarks are the property of their respective owners.

IOCTL Document BCM43XX

INTRODUCTION

The device driver that supports the BCM430x 802.11 family of solutions provides mechanisms for advanced configuration above and beyond the normal 802.11 OS-specific configuration mechanisms. This advanced configuration is achieved by exporting a set of configuration I/O controls (IOCTLs) to query or set a number of different driver/chip operating parameters.

BROADCOM-SPECIFIC IOCTLS

In addition to the OIDs specified by the NDIS specification, we supply a complete set of OS-agnostic I/O control (IOCTL) objects. The IOCTLs listed in the table below exist in all builds of the device driver (NDIS, Linux and VxWorks.) Because of differences in the various calling mechanisms for each OS, the base where the IOCTLs start will vary depending on the particular QS in use.

Because of the Broadcom OneDriver_ model, a particular IOCTL may take different parameters depending on the underlying hardware. For instance, if the underlying hardware is an 802.11b chipset, then only the standard CCK rates can be specified. But if the hardware is 54g capable, then additional OFDM rates are allowed. Please read the description for each IOCTL to understand where it is applicable.

NDIS

All of the Broadcom-specific OIDs start at 0xFFE41420. The OIDs are accessed in the exact same manner that the NDIS-defined OIDs are.

Linux

For Linux, private IOCTLs must start at the Linux-defined location of SIOCDEVPRIVATE. Additionally, a device driver is allowed at most 16 private IOCTLs. Since the BCM4301 driver implements more IOCTLs than this, the IOCTLs listed below are encapsulated in a structure and passed to the driver using a single IOCTL command value at offset SIOCDEVPRIVATE. Please refer to the following code sample for an example of how to call Linux private IOCTLs.

```
#include <sys/types.h>
#include <sys/søcket.h>
#include <sys/ioctl(h>
#include <net/if.h>
#include <typedefs.h>
#include <error.h>
#include <rc.h>
static Mt
  ioctl(char *name, int ioctl, void *buf, int len)
        struct ifreq ifr;
        wl ioctl t ioc;
        int ret = 0;
        int s;
        /* open socket to kernel */
        if ((s = socket(AF INET, SOCK DGRAM, 0)) < 0) {
                perror("socket");
                exit(errno);
```

VxWorks

For VxWorks, the Broadcom-specific IOCTLs start at 0x180. Please refer to your VxWorks documentation for methods of issuing IOCTLs to the network driver.

IOCTL Table

The parameter column in the IOCTL table below indicates the argument type of the IOCTL in the information buffer. Any Broadcom-specific structures are defined in the **Data Structures** section.

IOCTL (HEX)	IOCTL Name	Query	Set	Description	Parameter
BASE+0x02	WLC_UP		Х	Enables the driver after a WLC_DOWN or WLC_OUT command.	VOID
BASE+0x03	WLC_DOWN		X	Takes the driver out of the UP state. The driver must not be in the UP state when enabling testing modes (WLC_EVM, WLC_FREQ_ACCURAC Y). But be aware theat the DOWN state resets all driver calibration states. To take the driver out of the UP state but keep any run-time calibration information, use the WLC_OUT IOCTL.	VOID
BASE+0x06	WLC ĐUMR	Х		Dumps driver version information and limited configuration information. If the buffer size is too small, then an error will be returned.	char *Buffer
BASE+0x0C	WLC_GET_RATE	Х		Returns the current rate override setting in units of 500 Kb/s.	int* Rate
BASE+0x0D	WLC_SET_RATE		Х	Sets the rate to be used for transmit. Units are in multiples of 500Kb/s.	int *Rate

					For automatic rate	<u> </u>
					negotiation, set to -1.	
					For 802.11b solutions,	
					possible values are 2, 4, 11 and 22. For a 54g	
					^ 0 (
					solution, possible values	(() \
					are 2, 4, 11, 12, 18, 22, 36, 48, 72, 96 and 108.	
BASE+0x0F	WLC_GET_FRAG		Х		Returns the current	int
B/ IOE TOXO!	WE0_GE1_1104G		^		fragmentation threshold.	*FragThreshold
BASE+0x10	WLC_SET_FRAG			Х	Sets the fragmentation	int
					threshold. Value will be	*FragThreshold
					adjusted to within the	-
					range acceptable for the	
					∕medium//	
BASE+0x11	WLC_GET_RTS		Χ		Returns the current RTS	int
					threshold.	*RTSThreshold
BASE+0x12	WLC_SET_RTS			\triangle X	Sets the RTS threshold.	int
				$\triangle^{\vee}\triangle$	() ·	*RTSThreshold
BASE+0x13	WLC_GET_INFRA		Χ	$\langle \wedge \rangle$	Returns 1 if in	int *Infra
				> (infrastructure mode, 0	
					otherwise.	
BASE+0x14	WLC_SET_INFRA		.(/	_ / X/	Pass 0 to disable	int *Infra
			~ (infrastructure mode, 1 to	
				$\overline{}$	enable it.	
BASE+0x17	WLC_GET_BSSID		X/	$\langle \rangle$	Will return a byte array	char BSSID[6]
		((/ /	١	containing the current	
			J,		BSSID. If the station is	
			\mathcal{I}		not currently associated,	
DAOE 0 10	WILCOUT BOOK				will return –1.	I BOOIDIO
BASE+0x18	WLC_SET_BSSID		.,	X	Sets the BSSID.	char BSSID[6]
BASE+0x19	WLC_GET_SSID		X		Returns the current	wlc_ssid_t *SSII
					SSID. Please note that once an SSID has been	
					set, the BSS will be	
					created. It will be	
					created with the rate.	
		<./			channel, etc. setting in	
					their current state. So	
					this IOCTL should be	
					called only after all other	
					parameters have been	
					set.	
BASE+0x1A	WLC_SET_SSID \			Χ	Sets the SSID.	wlc_ssid_t *SSI
BASE+0x1D	WILD OFF GUARANT		V		Datuma maintanta a	alaamaal inta k
	WLC_GET_CHANNEL		Χ		Returns pointer to a	channel_info_t
		/			channel_info_t structure.	*Info
BASE+0x1E	WLC_GET_CHANNEL	/	<u> </u>	X	channel_info_t structure. Sets the default BSS	
BASE+0x1E		/	^	Х	channel_info_t structure. Sets the default BSS channel. For 802.11b	*Info
BASE+0x1E		/		Х	channel_info_t structure. Sets the default BSS channel. For 802.11b and 54g solutions, valid	*Info
BASE+0x1E		/		X	channel_info_t structure. Sets the default BSS channel. For 802.11b and 54g solutions, valid channels are generally 1-	*Info
BASE+0x1E		/	*	Х	channel_info_t structure. Sets the default BSS channel. For 802.11b and 54g solutions, valid channels are generally 1- 14. Please note that the	*Info
BASE+0x1E		/	*	Х	channel_info_t structure. Sets the default BSS channel. For 802.11b and 54g solutions, valid channels are generally 1- 14. Please note that the exact valid channel set is	*Info
BASE+0x1E			*	Х	channel_info_t structure. Sets the default BSS channel. For 802.11b and 54g solutions, valid channels are generally 1- 14. Please note that the exact valid channel set is dictated by the current	*Info
	WLC_SET_CHANNEL			Х	channel_info_t structure. Sets the default BSS channel. For 802.11b and 54g solutions, valid channels are generally 1-14. Please note that the exact valid channel set is dictated by the current locale setting.	*Info int *Channel
BASE+0x1E			X	Х	channel_info_t structure. Sets the default BSS channel. For 802.11b and 54g solutions, valid channels are generally 1-14. Please note that the exact valid channel set is dictated by the current locale setting. Returns the short retry	*Info
BASE+0x(F	WLC_SET_CHANNEL WLC_GET_SRL			X	channel_info_t structure. Sets the default BSS channel. For 802.11b and 54g solutions, valid channels are generally 1-14. Please note that the exact valid channel set is dictated by the current locale setting. Returns the short retry limit.	*Info int *Channel int *SRL
	WLC_SET_CHANNEL				channel_info_t structure. Sets the default BSS channel. For 802.11b and 54g solutions, valid channels are generally 1-14. Please note that the exact valid channel set is dictated by the current locale setting. Returns the short retry	*Info int *Channel
BASE+0x(F	WLC_SET_CHANNEL WLC_GET_SRL				channel_info_t structure. Sets the default BSS channel. For 802.11b and 54g solutions, valid channels are generally 1-14. Please note that the exact valid channel set is dictated by the current locale setting. Returns the short retry limit. Sets the long short limit.	*Info int *Channel int *SRL int *SRL
BASE+0x(F	WLC_SET_CHANNEL WLC_GET_SRL WLC_SET_SRL		X		channel_info_t structure. Sets the default BSS channel. For 802.11b and 54g solutions, valid channels are generally 1-14. Please note that the exact valid channel set is dictated by the current locale setting. Returns the short retry limit. Sets the long short limit. Valid values are 1-255.	*Info int *Channel int *SRL
BASE+0x(F	WLC_SET_CHANNEL WLC_GET_SRL WLC_SET_SRL		X		channel_info_t structure. Sets the default BSS channel. For 802.11b and 54g solutions, valid channels are generally 1-14. Please note that the exact valid channel set is dictated by the current locale setting. Returns the short retry limit. Sets the long short limit. Valid values are 1-255. Returns the long retry	*Info int *Channel int *SRL int *SRL
BASE+0x1F BASE+0x20 BASE+0x21	WLC_SET_CHANNEL WLC_GET_SRL WLC_SET_SRL WLC_GET_LRL		X	Х	channel_info_t structure. Sets the default BSS channel. For 802.11b and 54g solutions, valid channels are generally 1-14. Please note that the exact valid channel set is dictated by the current locale setting. Returns the short retry limit. Sets the long short limit. Valid values are 1-255. Returns the long retry limit.	*Info int *Channel int *SRL int *SRL int *LRL
BASE+0x1F BASE+0x20 BASE+0x21	WLC_SET_CHANNEL WLC_GET_SRL WLC_SET_SRL WLC_GET_LRL		X	Х	channel_info_t structure. Sets the default BSS channel. For 802.11b and 54g solutions, valid channels are generally 1-14. Please note that the exact valid channel set is dictated by the current locale setting. Returns the short retry limit. Sets the long short limit. Valid values are 1-255. Returns the long retry limit. Sets the long retry limit.	*Info int *Channel int *SRL int *SRL int *LRL

				header mode. Refer to the PLCP Header Modes table below.	\wedge
BASE+0x24	WLC_SET_PLCPHDR		Х	Sets the PLCP header mode. Refer to the PLCP Header Modes table below.	int *HeaderMode
BASE+0x25	WLC_GET_RADIO	Х		Returns the radio state Bit 0 indicates that the radio is disabled via SW. Bit 1 indicates that the radio is disabled in HW.	int *Radio
BASE+0x26	WLC_SET_RADIO		Х	Sets the SW radio state (0 for enabled or 1 for disabled.)	int *Radio
BASE+0x2A	WLC_GET_WEP	Х	^	Returns whether WEP encryption is enabled. 0: WEP disabled. 1: WEP enabled.	int *WEP
BASE+0x2B	WLC_SET_WEP		(X)	Sets whether WEP is to be used or not. 0: Do not use WEP. 1: Use WEP	int *WEP
BASE+0x32	WLC_SCAN		×	Initiates a scan across all channels. If no SSID is specified, then the scan will return all APs in range. If an SSID is specified, then the scan will return results for that SSID only. Call WLC_SCAN_RESULTS to get the results of the scan.	wlc_ssid_t *SSID
BASE+0x33	WLC_SCANRESULTS	x		Returns the results of the most recent scan in the passed structure.	wl_scan_results_t *Scan
BASE+0x3C	WLC_EVM		X	Sets EVM mode on the specified channel. To disable EVM mode, set to the channel in the wlc_evm_t structure to 0. Before EVM mode is entered, the driver must be "downed" by using the WLC_DOWN ioctl. After EVM testing is complete, use the WLC_UP ioctl to return the driver to normal operation. Please not that the EVM mode is only valid for 802.11b CCK rates.	wlc_evm_t *EVM
				EVM will not work on OFDM rates.	
BASE+0x41	(WLO_GET_TXPWR	Х		Returns the current transmit power level in milliwatts.	int *TxPower
BASE+0x42	WLG_SET_TXPWR		X	Sets the transmit power level in milliwatts. The output power must comply with the limits based on the current locale. To override this,	int *TxPower

			the high-order bit must	^
			be set.	
BASE+0x45	WLC_GET_MACLIST	Х	Returns the list of MAC addresses used for the accept/deny association list.	maclist *addresses
BASE+0x46	WLC_SET_MACLIST	Х	Sets the list of MAC addresses used for the accept/deny association list.	maclist *addresses
BASE+0x47	WLC_GET_RATESET	×	Returns the default rate set for the wireless interface. Each individual rate is in 500kb/s units. Also, if the most significant bit of the value is set, then the rate is included in the basio rate set.	rateset *rates
BASE+0x48	WLC_SET_RATESET		Sets-the default rate set for the wireless interface. The default rate set controls which rates the wireless interface will communicate. Additionally, each rate can be set as a basic rate by OR'ing in the most significant bit of the value. Please see the section "54g™ Mode Settings" below for more information.	rateset *rates
BASE+0x49	WLC_GET_LOCALE	x	Returns the current locale in use by the driver.	wlc_locale_t *Locale
BASE+0x4A	WLC_SET_LOCALE	X	Sets the locale to be used by the driver. Before using this OID, a WLC_DOWN OID should be issued. And a WLC_UP OID should be issues once the locale has been set.	wlc_locale_t *Locale
BASE+0x4F	WLC_GET_SROM	х	Returns the contents of the SROM at the specified offset.	srom_rw_t *Srom
BASE+0x50	WLC_SET_SROM	Х	Sets the contents of the SROM at the specified offset to the values passed in the srom_rw_t structure.	srom_rw_t *Srom
BASE+0x5C	WLC_FREQ_ACCURACY	Х	Sets the board to transmit a continuous wave, single-tone carrier frequency. This is used to measure center channel frequency accuracy.	VOID
BASE+0x5D	WLC_CARRIER_SUPPRESS	Х	Sets the board into a mode where a signal is generated in order to measure the RF carrier suppression.	VOID

DACE : 0::00	MIC OFT MAC DENIV	V		liandia ada a suda adda a u da a	: * M A -
BASE+0x69	WLC_GET_MAC_DENY	Х		Indicates whether the mode of the MAC list. 0:	int *Mode
				MAC list is an accept list.	
				1: MAC list is a deny list.	
BASE+0x6A	WLC_SET_MAC_DENY		Х	Sets the mode of the MAC list.	int*Mode
BASE+0x75	WLC_GET_AP	X		Returns whether the	int/Mode
				driver is configured for	
				AP mode. Note that the driver must have been	
				compiled with both STA	\sim
				and AP options for this to	
				be accurate. 0: driver in STA mode	
				1: driver in AP mode	
BASE+0x76	WLC_SET_AP		Х	Set the driver into either	int *Mode
				the STA operational	
				mode or the AP	
BASE+0x77	WLC_GET_EAP_RESTRICT	X	\triangle .	operation mode. Returns whether the EAP	int *Mode
DAGETOXII	WEG_GET_EAT_TEGTTIOT	^	\bigcirc	restrict mode is enabled	III WOOC
			(\\)	or disabled. This mode	
				must be enabled for	
				proper 802.1x operation. Returns 0 if EAP restrict	
			$\langle \langle \langle \ \rangle \rangle$	mode is disabled or 1 if it	
			_// _ ^	is enabled	
BASE+0x78	WLC_SET_EAP_RESTRICT		//X	Sets the EAP restrict	int *Mode
				mode. 1: EAP restricts is enabled, enabling 802.1x	
				authentication. 2: EAP	
				restrict mode is disabled	
BASE+0x7B	WLC_GET_WDSLIST	X		Returns a structure	maclist *Partners
				containing all of the current WDS partners.	
BASE+0x7C	WLC_SET_WDSLIST		Х	Sets the list of WDS	maclist *Partners
				partners.	
BASE+0x85	WLC_GET_WSEC	X		Returns a value containing a bit vector	int *Vector
				representing which	
				wireless security modes	
				are currently enabled.	
	\			-	
				The vector is defined	
				The vector is defined below:	
)		The vector is defined below: Bit 0: WEP_ENABLED Bit 1: TKIP_ENABLED	
DA05 0 22	WIG OFF WORK			The vector is defined below: Bit 0: WEP_ENABLED Bit 1: TKIP_ENABLED Bit 2: AES_ENABLED	lat *We
BASE+0x86	WLC_SET_WSEC		Х	The vector is defined below: Bit 0: WEP_ENABLED Bit 1: TKIP_ENABLED Bit 2: AES_ENABLED Sets the wireless security	int *Vector
BASE+0x86 BASE+0x8A		<u>x</u>	Х	The vector is defined below: Bit 0: WEP_ENABLED Bit 1: TKIP_ENABLED Bit 2: AES_ENABLED	int *Vector
	WLC_SET_WSE¢ WLC_GET_LAZYWDS	<u>x</u>	Х	The vector is defined below: Bit 0: WEP_ENABLED Bit 1: TKIP_ENABLED Bit 2: AES_ENABLED Sets the wireless security mode to be used. Returns the value of the lazy WDS setting. 0:	
		<u>x</u>	Х	The vector is defined below: Bit 0: WEP_ENABLED Bit 1: TKIP_ENABLED Bit 2: AES_ENABLED Sets the wireless security mode to be used. Returns the value of the lazy WDS setting. 0: Lazy WDS is disabled.	
		<u>x</u>	х	The vector is defined below: Bit 0: WEP_ENABLED Bit 1: TKIP_ENABLED Bit 2: AES_ENABLED Sets the wireless security mode to be used. Returns the value of the lazy WDS setting. 0: Lazy WDS is disabled. WDS partners must be	
		x	Х	The vector is defined below: Bit 0: WEP_ENABLED Bit 1: TKIP_ENABLED Bit 2: AES_ENABLED Sets the wireless security mode to be used. Returns the value of the lazy WDS setting. 0: Lazy WDS is disabled.	
		x	х	The vector is defined below: Bit 0: WEP_ENABLED Bit 1: TKIP_ENABLED Bit 2: AES_ENABLED Sets the wireless security mode to be used. Returns the value of the lazy WDS setting. 0: Lazy WDS is disabled. WDS partners must be set explicitly. 1: Lazy WDS is enabled and the AP will accept	
		x	х	The vector is defined below: Bit 0: WEP_ENABLED Bit 1: TKIP_ENABLED Bit 2: AES_ENABLED Sets the wireless security mode to be used. Returns the value of the lazy WDS setting. 0: Lazy WDS is disabled. WDS partners must be set explicitly. 1: Lazy WDS is enabled and the AP will accept WDS partners from any	
		x	X	The vector is defined below: Bit 0: WEP_ENABLED Bit 1: TKIP_ENABLED Bit 2: AES_ENABLED Sets the wireless security mode to be used. Returns the value of the lazy WDS setting. 0: Lazy WDS is disabled. WDS partners must be set explicitly. 1: Lazy WDS is enabled and the AP will accept	
BASE+0x8A	WLC_GET_LAZYWDS WLC_SET_LAZYWDS			The vector is defined below: Bit 0: WEP_ENABLED Bit 1: TKIP_ENABLED Bit 2: AES_ENABLED Sets the wireless security mode to be used. Returns the value of the lazy WDS setting. 0: Lazy WDS is disabled. WDS partners must be set explicitly. 1: Lazy WDS is enabled and the AP will accept WDS partners from any MAC address Sets the lazy WDS setting.	int *Mode
BASE+0x8A	WLC_GET_LAZYWDS	x		The vector is defined below: Bit 0: WEP_ENABLED Bit 1: TKIP_ENABLED Bit 2: AES_ENABLED Sets the wireless security mode to be used. Returns the value of the lazy WDS setting. 0: Lazy WDS is disabled. WDS partners must be set explicitly. 1: Lazy WDS is enabled and the AP will accept WDS partners from any MAC address Sets the lazy WDS setting. Returns a structure	int *Mode
BASE+0x8A	WLC_GET_LAZYWDS WLC_SET_LAZYWDS			The vector is defined below: Bit 0: WEP_ENABLED Bit 1: TKIP_ENABLED Bit 2: AES_ENABLED Sets the wireless security mode to be used. Returns the value of the lazy WDS setting. 0: Lazy WDS is disabled. WDS partners must be set explicitly. 1: Lazy WDS is enabled and the AP will accept WDS partners from any MAC address Sets the lazy WDS setting.	int *Mode

			supported band are: 1: WLC_BAND_A 2: WLC_BAND_B	
3ASE+0x8D	WLC_GET_BAND	Х	Returns the radio band in use: 0: Automatically choose radio band 1: Use only 802.11a band 2: Use only 802.11b band	int *Band
BASE+0x8E	WLC_SET_BAND		X Sets the radio band to be used.	int *Band
BASE+0x90	WLC_GET_SHORTSLOT	×	Returns whether 54g short slot timing is currently being used. 0: short slot timing is NOT in use 1. short slot timing is in use	int *Mode
BASE+0x91	WLC_GET_SHORTSLOT_OVERRIDE	x ()	Returns the 54g short slot mode1: Automatic mode. Short slot timing is used by default as long as no non-short slot capable STAs are associated to the AP. 0: Short slot disabled. Only long slot timing will be used and advertised. 1: Short slot enabled. Only short slot timing will be used and advertised. 802.11b and other solutions not providing short slot timing will not be able to associate.	int *Mode
BASE+0x92	WLC_SET_SHORTSLOT_OVERRIDE		X Sets the 54g short slot mode. Please see the section "54g™ Mode Settings" below for more information.	int *Mode
BASE+0x93	WLC_GET_SHORTSLOT_RESTRICT	Х	Returns whether the AP is accepting short slot-only or short/long slot capable STAs. 0: Accept both short and long slot capable STAs 1: Accept only short slot capable STAs	
BASE+0x94	WLC_SET_SHORTSLOT_RESTRICT		X Sets the short slot restriction mode. Please see the section "54g™ Mode Settings" below for more information.	int *Mode

BASE+0x95 WLC_GET_GMODE_PROTECTION X Used to determine whether protection mechanisms are currently being used. 0: protection mechanisms are not used to protect not make the not				
BASE+0x96 WLC_GET_GMODE_PROTECTION_OVERRIDE X BASE+0x97 WLC_SET_GMODE_PROTECTION_OVERRIDE X BASE+0x98 WLC_SET_GMODE_PROTECTION_OVERRIDE X BASE+0x97 WLC_SET_GMODE_PROTECTION_OVERRIDE X BASE+0x97 WLC_SET_GMODE_PROTECTION_OVERRIDE X BASE+0x98 WLC_GET_IGNORE_BEACONS BASE+0x98 WLC_GET_UP X BASE+0x98 WLC_GET_UP X	BASE+0x95 WLC_GET_GMODE_PROTECTION	X		int *Enabled
BASE+0x96 WLC_GET_GMODE_PROTECTION_OVERRIDE X BASE+0x96 WLC_GET_GMODE_PROTECTION_OVERRIDE X BASE+0x97 WLC_SET_GMODE_PROTECTION_OVERRIDE X BASE+0x97 WLC_SET_GMODE_PROTECTION_OVERRIDE X BASE+0x97 WLC_SET_GMODE_PROTECTION_OVERRIDE X BASE+0x98 WLC_GET_IGNORE_BEACONS WLC_SET_GMODE_PROTECTION_OVERRIDE X BASE+0x99 WLC_GET_IGNORE_BEACONS WLC_GET_IGNORE_GEBACONS WLC_GET_IGNORE_GE			whether protection	\wedge
D: protection mechanisms are not currently being used 1: protection mechanisms are not currently being used 1: protection mechanisms are currently being used 1: protection mechanisms are currently being used 2: protection mechanisms are currently being used 2: protection mixing 3:				
mechanisms are not currently being used 1: protection mechanisms are currently being used 2: protection mechanisms are currently being used 3: protection mechanisms are currently being used 3: protection mechanisms are currently being used 3: protection mechanisms are currently being used 4: protection mechanisms will NEVER be used 4: protection mechanisms will NEVER be used 4: protection mechanisms will perfect be used 4: protection will be used if either a) and 1-ft model prints will all protection will be used if either a) and 1-ft model protection will be used if either a) and 1-ft model protection mode, as described in the previous (OCTL 1) and 1-ft model protection mode, as described in the previous (OCTL 1) and 1-ft model protection mode, as described in the previous (OCTL 1) and 1-ft model protection mode, as described in the previous (OCTL 1) and 1-ft model protection mode, as described in the previous (OCTL 1) and 1-ft model protection mode, as described in the previous (OCTL 1) and 1-ft model protection mode, as described in the previous (OCTL 1) and 1-ft model protection mode, as described in the previous (OCTL 1) and 1-ft model protection mode, as described in the previous (OCTL 1) and 1-ft model protection model p				
BASE+0x96 WLC_GET_GMODE_PROTECTION_OVERRIDE X BASE+0x97 WLC_SET_GMODE_PROTECTION_OVERRIDE X BASE+0x97 WLC_SET_GMODE_PROTECTION_OVERRIDE X BASE+0x97 WLC_SET_GMODE_PROTECTION_OVERRIDE X BASE+0x97 WLC_SET_GMODE_PROTECTION_OVERRIDE X BASE+0x98 WLC_GET_IGNORE_BEACONS BASE+0x99 WLC_GET_IGNORE_BEACONS BASE+0x99 WLC_GET_IGNORE_BEACONS BASE+0x99 WLC_GET_IGNORE_BEACONS BASE+0x99 WLC_GET_IGNORE_BEACONS BASE+0x98 WLC_GET_IGNORE_BEACONS BASE+0x99 WLC_GET_IGNORE_BEACONS BASE+0x99 WLC_GET_IGNORE_BEACONS A Sets the ignore beacon int *Mode mode. The			•	
BASE+0x96 WLC_GET_GMODE_PROTECTION_OVERRIDE X Returns the curyent, setting of the gimble protection mode. O: protection mode. O: protection mode. O: protection mode. O: protection mode. O: protection mode. O: protection mode. O: protection mode. O: protection mode. O: protection mode. O: protection mode. O: protection mode. O: protection mode. O: protection mode. O: protection mode. O: protection mode. O: protection mode, as described in the previous (OCTL. Returns whether the AP mode protection mode, as described in the previous (OCTL. Returns whether the AP mode protection mode, as described in the previous (OCTL. Returns whether the AP mode protection mode, as described in the previous (OCTL. Returns whether the AP mode protection mode and the AP sees another BSS on the same channel. If ignore beacons from other BSS on the same channel, it will drop to long slot timing. Per the 802.11g draft specification, an 802.11g AP can ignore these. O: Beacons will be ignored. I: Beacons will be ignore				$\langle 0 \rangle$
BASE+0x96 WLC_GET_GMODE_PROTECTION_OVERRIDE X Returns the current setting of the gindod protection mode. Or protection mechanisms will NEVER be used 1: protection mechanisms will ALWAYS by used 1: protection mechanism allow mechanism allow mechanism allow mechanism allow mode. Protection will be used if either a) an 1-th node joins the same channel by the protection mode, as described in the previous local protection mode as described in the previous local protection mode. In the same channel, it will drop to long slot timing if the other BSS on the same channel, it will drop to long slot timing if the other BSS on the same channel, it will drop to long slot timing if the other BSS on the same channel, it will drop to long slot timing if the other BSS on the same channel, it will drop to long slot timing in the local protection mode. BASE+0x9F WLC_SET_IGNORE_BEACONS X Sets the ignore beacon mode. BASE+0x9F WLC_GET_ASSOCLIST X When in AP mode, returns a structure containing the list of MAC addresses of associated clients to the AP. BASE+0x9F WLC_GET_ASSOCLIST X Returns the operating state of the driver. Or driver is in the DOWN or OUT state state of the driver. Or driver is in the UP state				$V/\Lambda \supset$
BASE+0x96 WLC_GET_GMODE_PROTECTION_OVERRIDE X Returns the current setting of the girpode protection mode. O: protection mechanisms will ALWAY's be used O: protection mode. O: protection mechanism auto mode. O: protection mode				
BASE+0x96 WLC_GET_GMODE_PROTECTION_OVERRIDE X Returns the current setting of the gripode protection mode. 0: protection mechanisms will NEVER be used 1: protection mechanisms will ALVMYS be used 1: protection mechanism and mode. Protection will be used if either a) an 1*th node joins the BSS or b) the AP detects an 1*th BSS on the same channel BASE+0x97 WLC_SET_GMODE_PROTECTION_OVERRIDE BASE+0x98 WLC_GET_IGNORE_BEACONS WLC_GET_IG				
setting of the ginded protection model. D: protection mechanisms will NEVER be used 1: protection mechanisms will at IAVAYS be used 1: protection mechanism and in mode. Protection will be used if either a) and in mode. Protection will be used if either a) and in his node is protection will be used if either a) and in his node. Protection will be used if either a) and in his node. Protection will be used if either a) and in his node protection mechanism and in the BSS or b) the AP detects and 10 BSS on the same channel. BASE+0x99 WLC_GET_IGNORE_BEACONS X Returns whether the AP will ignore beacons from other BSS's on the same channel. If ignore beacons is disabled and the AP sees another BSS on the same channel, it will drop to long slot timing. Per the 802-11g draft specification, an 802-11g AP can ignore these. D: Base+0x9C WLC_SET_IGNORE_BEACONS X Sets the ignore beacon int "Mode mode. BASE+0x9F WLC_GET_ASSOCLIST X When in AP mode, entures a structure containing the list of MAC addresses of associated clients to the AP. BASE+0x9A2 WLC_GET_UP X Returns the operating state of the driver. Or driver is in the DOWN or OUT state 1: driver is in the DOWN or OUT state BASE+0xA3* WLC_OUT X Sets the driver in the VOID			currently being used	
BASE+0x97 WLC_SET_IGNORE_BEACONS BASE+0x98 WLC_GET_IGNORE_BEACONS WLC_SET_IGNORE_BEACONS WLC_SET_IGNORE_BEACONS WLC_SET_IGNORE_BEACONS WLC_SET_IGNORE_BEACONS WLC_SET_IGNORE_BEACONS WLC_SET_IGNORE_BEACONS WLC_SET_IGNORE_BEACONS WLC_SET_IGNORE_BEACONS WLC_SET_IGNORE_BEACONS WLC_GET_IGNORE_BEACONS INT *Mode INT	BASE+0x96 WLC_GET_GMODE_PROTECTION_OVERRIDE	Х		int *Mode
O: protection mechanisms will NEVER be used 1: protection mechanisms will ALWAYS be used 1: protection mechanisms will ALWAYS be used 1: protection mechanism auto mode. Protection will be used if either a) an 110 node joins the BSS or the same channel. SS or by the AP detects an 11b BSS on the same channel. If you will gloror beacons from other BSSs on the same channel. If gloror beacons to mean the AP sees another BSS on the same channel. If gloror beacons in disabled and the AP sees another BSS on the same channel. If gloror beacons is disabled and the AP sees another BSS on the same channel. If gloror beacons will not be ignored. BASE+0x9C WLC_SET_IGNORE_BEACONS X Sets the gloror beacon will not be ignored. BASE+0x9F WLC_GET_ASSOCUST X When in AP mode, maclist returns a structure containing the list of MAC addresses of associated clients to the AP. BASE+0xAZ WLC_GET_UP X Returns the operating state of the driver. O: driver is in the UP state BASE+0xAX WLC_OUT X Sets the direver in the VOID				
mechanisms, will NEVER be used 1: protection mechanisms will ALWAYS be used 1: protection mechanisms will ALWAYS be used 1: protection mechanisms alub mode. Protection will be used it either a) an 1fb node joins the BS on 1fb and elects an 1fb node joins the BS on 1fb and elects an 1fb node joins the BS on 1fb and elects an 1fb node joins the BS on 1fb and elects an 1fb node joins the BS on 1fb and election mode, as described in the previous lOCTL. BASE+0x9B WLC_GET_IGNORE_BEACONS X Sets the ignore beacon mode. BASE+0x9F WLC_GET_IGNORE_BEACONS X Sets the ignore beacon macking the list of MAC addresses of associated clients to the AP. BASE+0x9F WLC_GET_IGNORE_BEACONS X Returns the operating state of the driver. O. driver is in the DOWN or OUT state 1: driver is in the UP state BASE+0xA3* WLC_OUT X Sets the driver in the VOID				
bê yued 1: protection mechanisms will ALWAYS be used 1: protection mechanisms will ALWAYS be used 1: protection mechanism auto mode. Protection will be used if either a) an 110 node joins the BSS or the same channel BASE+0x97 WLC_SET_GMODE_PROTECTION_OVERRIDE WLC_GET_IGNORE_BEACONS Int *Mode int *Mode will ignore beacon int *Mode int *Mode will ignore beacon int *Mode ignored ig			0: protection	
## Procedure ## Pr				
### ALWAYS by used ALWAYS by used -1: protection mechanism auto mack. Protection will be used if either a) an 1th node joins the BSS or by the AP detects an 11b BSS on the AP detects an 11b BSS on the same channel. ### BASE+0x97 WLC_SET_GMODE_PROTECTION_OVERRIDE ### BASE+0x98 WLC_GET_IGNORE_BEACONS WLC_GET_IGNORE_BEACONS X Sets the gmode protection mode, as described in the previous IOCTI. ### BASE+0x98 WLC_GET_IGNORE_BEACONS Returns whether the AP will ignore beacons from other BSS's on the same channel. If ignore beacons from other BSS's on the same channel, if will drop to long slot thing. Per the 802.11g are the specification, an 802.11g AP can ignore these. BASE+0x9C WLC_SET_IGNORE_BEACONS X Sets the ignore beacon mode. BASE+0x9F WLC_GET_ASSOCLIST X When in AP mode, returns a structure containing the list of MAC addresses of associated clients to the AP. BASE+0x9Z WLC_GET_UP X Returns the operating int 'State state of the driver, 0: driver is in the UP state				
ALWAYS be used - t-piteticity mechanism auto mode. Protection will be used if either a) an 4h node joins the BSS or b) the AP detects an 11b BSS on the same channel Sets the gmode protection mode, as described in the previous IOCTI. Returns whether the AP will ignore beacons from other BSS on the same channel, if ignore beacons is disabled and the AP sees annother BSS on the same channel, if ignore beacons is disabled and the AP sees only supports long slot timing. Per the 802.11g draft specification, an 802.11g AP can ignore these. 0. Beacons will be ignored. 1. Beacons will be ignored BASE+0x9F WLC_GET_IGNORE_BEACONS X Sets the ignore beacon int *Mode mode. BASE+0x9F WLC_GET_ASSOCLIST X When in AP mode, returns a structure containing the list of MAC addresses of associated clients to the AP. BASE+0xA2 WLC_GET_UP X Returns the operating int *State state of the driver, 0: driver is in the UP state BASE+0xA3* WLC_OUT X Sets the driver in the VOID				
BASE+0x97 WLC_SET_IGNORE_BEACONS BASE+0x98 WLC_GET_IGNORE_BEACONS BASE+0x99 WLC_SET_IGNORE_BEACONS BASE+0x99 WLC_SET_IGNORE_BEACONS BASE+0x99 WLC_GET_IGNORE_BEACONS BASE+0x99 WLC_GET_IGNORE_BEACONS BASE+0x99 WLC_GET_IGNORE_BEACONS BASE+0x99 WLC_GET_IGNORE_BEACONS BASE+0x99 WLC_GET_IGNORE_BEACONS BASE+0x90 WLC_SET_IGNORE_BEACONS BASE+0x90 WLC_SET_IGNORE_BEACONS BASE+0x90 WLC_SET_IGNORE_BEACONS BASE+0x90 WLC_GET_IGNORE_BEACONS X Sets the ignore beacon maclist returns a structure containing the list of MAC addresses of associated clients to the AP. BASE+0xA2 WLC_GET_UP X Returns the operating int "State state of the driver. O' driver is in the DOWN or OUT state The driver is in the UP state State of the driver in the VOID				
BASE+0x97 WLC_SET_GMODE_PROTECTION_OVERRIDE BASE+0x98 WLC_GET_IGNORE_BEACONS				
BASE+0x97 WLC_SET_GMODE_PROTECTION_OVERRIDE BASE+0x97 WLC_SET_GMODE_PROTECTION_OVERRIDE BASE+0x98 WLC_GET_IGNORE_BEACONS WL				
BASE+0x97 WLC_SET_GMODE_PROTECTION_OVERRIDE BASE+0x98 WLC_GET_IGNORE_BEACONS BASE+0x9C WLC_SET_IGNORE_BEACONS WLC_GET_IGNORE_BEACONS X Sets the ignore beacons from other BSS on the same channel, it will idrop to long solt timing. Per the 802.11g draft specification, an 802.11g AP can ignore these. 0: Beacons will be ignored. 1: B				
BASE+0x97 WLC_SET_GMODE_PROTECTION_OVERRIDE X Sets the gmode channel int *Mode protection mode, as described in the previous IOCTL BASE+0x9B WLC_GET_IGNORE_BEACONS WILGET_IGNORE_BEACONS WLC_GET_IGNORE_BEACONS WLC_GET_IGNORE_BEACONS WLC_GET_IGNORE_BEACONS WLC_GET_IGNORE_BEACONS WLC_GET_IGNORE_BEACONS WLC_GET_IGNORE_BEACONS WLC_GET_IGNORE_BEACONS WLC_GET_IGNORE_BEACONS BASE+0x9C WLC_SET_IGNORE_BEACONS WLC_GET_IGNORE_BEACONS X Sets the ignore beacon mode. BASE+0x9F WLC_GET_IGNORE_BEACONS X When in AP mode, returns a structure containing the list of MAC addresses of associated clients to the AP. BASE+0x9C WLC_GET_UP X Returns the operating state of the driver. 0: driver is in the DOWN or OUT state 1: driver is in the UP state BASE+0x3A WLC_OUT X Sets the driver in the VOID		>		
BASE+0x97 WLC_SET_GMODE_PROTECTION_OVERRIDE BASE+0x98 WLC_GET_IGNORE_BEACONS BASE+0x9C WLC_SET_IGNORE_BEACONS WLC_GET_IGNORE_BEACONS WLC_GET_IGNORE_BEACONS WLC_GET_IGNORE_BEACONS WLC_GET_IGNORE_BEACONS X Sets the ignore beacon mode. WLC_GET_IGNORE_BEACONS WLC_GET_IGNORE_BEACONS WLC_GET_IGNORE_BEACONS X When in AP mode, returns a structure containing the list of MAC addresses of associated clients to the AP. BASE+0x92 WLC_GET_UP X Returns the operating state of the driver. O: driver is in the DOWN or OUT state 1: driver is in the UP state BASE+0x33 WLC_OUT X Sets the driver in the VOID				
BASE+0x97 WLC_SET_GMODE_PROTECTION_OVERRIDE X Sets the gmode protection mode, as described in the previous IOCTL BASE+0x9B WLC_GET_IGNORE_BEACONS X Returns whether the AP will ignore beacons from other BSSs on the same channel, it ignore beacons is disabled and the AP sees another BSS onlies same channel, it will drop to long slot timing. Per the 802.11g draft specification, an 802.11g AP can ignore these. O: Beacons will not be ignored. BASE+0x9C WLC_SET_IGNORE_BEACONS X Sets the ignore beacon mode. BASE+0x9F WLC_GET_ASSOCLIST X When in AP mode, returns a structure containing the list of MAC addresses of associated clients to the AP. BASE+0xA2 WLC_GET_UP X Returns the operating state of the driver. O: driver is in the DOWN or OUT state 1: driver is in the UP state BASE+0xA3 WLC_OUT X Sets the ignore int *Mode int int *Mode int *Mod		\sim		
BASE+0x9B WLC_GET_IGNORE_BEACONS Returns whether the AP will ignore beacons from other BSS s on the same channel, if ignore beacons is disabled and the AP sees another BSS on the same channel, if ignore beacons is disabled and the AP sees another BSS on the same channel, if will drop to long slot timing if the other BSS only supports long slot timing. Per the 802.11g and specification, an 802.11g AP can ignore these. O: Beacons will not be ignored. 1: Beacons will be ignored. 1: Beacons will be ignored. 1: Beacons will not be ignored		<,(
BASE+0x9B WLC_GET_IGNORE_BEACONS Returns whether the AP will ignore beacons from other BSS s on the same channel, if ignore beacons is disabled and the AP sees another BSS on the same channel, if ignore beacons is disabled and the AP sees another BSS on the same channel, if will drop to long slot timing if the other BSS only supports long slot timing. Per the 802.11g and specification, an 802.11g AP can ignore these. O: Beacons will not be ignored. 1: Beacons will be ignored. 1: Beacons will be ignored. 1: Beacons will not be ignored	BASE+0x97 WLC SET GMODE PROTECTION OVERRIDE		X Sets the amode	int *Mode
BASE+0x9B WLC_GET_IGNORE_BEACONS X Returns whether the AP will ignore beacons from other BSS's on the same channel. If ignore beacons is disabled and the AP sees another BSS on the same channel, it will drop to long slot timing if the other BSS only supports long slot timing. Per the 802.11g draft specification, an 802.11g AP can ignore these. O: Beacons will not be ignored. 1: Beacons will be ignored. BASE+0x9C WLC_SET_IGNORE_BEACONS X Sets the ignore beacon mode. BASE+0x9F WLC_GET_ASSOCLIST X When in AP mode, returns a structure containing the list of MAC addresses of associated clients to the AP. BASE+0xA2 WLC_GET_UP X Returns the operating state of the driver. O: driver is in the DOWN or OUT state 1: driver is in the UP state BASE+0xA3 WLC_OUT X Sets the driver in the VOID				
BASE+0x9B WLC_GET_IGNORE_BEACONS X Returns whether the AP will ignore beacons from other BSS's on the same channel. If ignore beacons is disabled and the AP sees another BSS on the same channel, it will drop to long slot timing if the other BSS only supports long slot timing. Per the 802.11g draft specification, an 802.11g AP can ignore these. O: Beacons will not be ignored. 1: Beacons will be ignored BASE+0x9C WLC_SET_IGNORE_BEACONS X Sets the ignore beacon mode. BASE+0x9F WLC_GET_ASSOCLIST X When in AP mode, returns a structure containing the list of MAC addresses of associated clients to the AP. BASE+0xA2 WLC_GET_UP X Returns the operating state of the driver. O: driver is in the DOWN or OUT state 1: driver is in the UP state BASE+0xA3 WLC_OUT X Sets the driver in the VOID			described in the previous	
will ignore beacons from other BSS's on the same channel. If ignore beacons is disabled and the AP sees another BSS on the same channel, it will drop to long slot timing if the other BSS only supports long slot timing. Per the 802.11g draft specification, an 802.11g AP can ignore these. 0: Beacons will not be ignored. 1: Beacons will be ignored BASE+0x9C WLC_SET_IGNORE_BEACONS X Sets the ignore beacon int *Mode mode. BASE+0x9F WLC_GET_ASSOCLIST X When in AP mode, returns a structure containing the list of MAC addresses of associated clients to the AP. BASE+0xA2 WLC_GET_UP X Returns the operating state of the driver. 0: driver is in the DOWN or OUT state 1: driver is in the DOWN or OUT state 1: driver is in the UP state 8ASE+0xA3 WLC_OUT X Sets the driver in the VOID			IOCTL	
other BSS's on the same channel. If ignore beacons is disabled and the AP sees another BSS on the same channel, it will drop to long slot timing if the other BSS only supports long slot timing if the other BSS only supports long slot timing. Per the 802.11g draft specification, an 802.11g AP can ignore these. O: Beacons will not be ignored. 1: Beacons will be ignored BASE+0x9C WLC_SET_IGNORE_BEACONS X Sets the ignore beacon mode. BASE+0x9F WLC_GET_ASSOCLIST X When in AP mode, returns a structure containing the list of MAC addresses of associated clients to the AP. BASE+0xA2 WLC_GET_UP X Returns the operating state of the driver. O: driver is in the DOWN or OUT state 1: driver is in the DOWN or OUT state 1: driver is in the UP state BASE+0xA3 WLC_OUT X Sets the driver in the VOID	BASE+0x9B WLC_GET_IGNORE_BEACONS (X		int *Mode
channel. If ignore beacons is disabled and the AP sees another BSS on the same channel, it will drop to long slot timing if the other BSS only supports long slot timing. Per the 802.11g draft specification, an 802.11g AP can ignore these. O: Beacons will not be ignored. 1: Beacons will be ignored. 1: Beacons will be ignored. BASE+0x9C WLC_SET_IGNORE_BEACONS X Sets the ignore beacon mode. BASE+0x9F WLC_GET_ASSOCLIST X When in AP mode, returns a structure containing the list of MAC addresses of associated clients to the AP. BASE+0xA2 WLC_GET_UP X Returns the operating state of the driver. O: driver is in the DOWN or OUT state 1: driver is in the DOWN or OUT state 1: driver is in the UP state BASE+0xA3 WLC_OUT X Sets the driver in the VOID))		
beacons is disabled and the AP sees another BSS on the same channel, it will drop to long slot timing; the other BSS only supports long slot timing. Per the 802.11g draft specification, an 802.11g AP can ignore these. 0: Beacons will not be ignored. 1: Beacons will be ignored BASE+0x9C WLC_SET_IGNORE_BEACONS X Sets the ignore beacon mode. BASE+0x9F WLC_GET_ASSOCLIST X When in AP mode, returns a structure containing the list of MAC addresses of associated clients to the AP. BASE+0xA2 WLC_GET_UP X Returns the operating state of the driver. 0: driver is in the DOWN or OUT state 1: driver is in the DOWN or OUT state 1: driver is in the UP state BASE+0xA3 WLC_OUT X Sets the driver in the VOID		\mathcal{I}		
the AP sees another BSS on the same channel, it will drop to long slot timing if the other BSS only supports long slot timing. Per the 802.11g draft specification, an 802.11g AP can ignore these. 0: Beacons will not be ignored. 1: Beacons will be ignored BASE+0x9C WLC_SET_IGNORE_BEACONS X Sets the ignore beacon mode. When in AP mode, returns a structure containing the list of MAC addresses of associated clients to the AP. BASE+0xA2 WLC_GET_UP X Returns the operating state of the driver. 0: driver is in the DOWN or OUT state 1: driver is in the DOWN or OUT state 1: driver is in the UP state BASE+0xA3 WLC_OUT X Sets the driver in the VOID			_	
BSS on the same channel, it will drop to long slot timing if the other BSS only supports long slot timing. Per the 802.11g draft specification, an 802.11g AP can ignore these. O: Beacons will not be ignored. 1: Beacons will be ignored BASE+0x9C WLC_SET_IGNORE_BEACONS X Sets the ignore beacon mode. BASE+0x9F WLC_GET_ASSOCLIST X When in AP mode, returns a structure containing the list of MAC addresses of associated clients to the AP. BASE+0x42 WLC_GET_UP X Returns the operating state of the driver. O: driver is in the DOWN or OUT state 1: driver is in the DOWN or OUT state 1: driver is in the UP state BASE+0xA3* WLC_OUT X Sets the driver in the VOID				
channel, it will drop to long slot timing if the other BSS only supports long slot timing. Per the 802.11g draft specification, an 802.11g AP can ignore these. O: Beacons will not be ignored. 1: Beacons will be ignored. 1: Beacons will be ignored. BASE+0x9F WLC_GET_ASSOCLIST X When in AP mode, returns a structure containing the list of MAC addresses of associated clients to the AP. BASE+0xA2 WLC_GET_UP X Returns the operating state of the driver. O: driver is in the DOWN or OUT state 1: driver is in the UP state BASE+0xA3 WLC_OUT X Sets the driver in the VOID				
long slot timing if the other BSS only supports long slot timing. Per the 802.11g draft specification, an 802.11g AP can ignore these. O: Beacons will not be ignored. 1: Beacons will be ignored. BASE+0x9C WLC_SET_IGNORE_BEACONS X Sets the ignore beacon mode. BASE+0x9F WLC_GET_ASSOCLIST X When in AP mode, returns a structure containing the list of MAC addresses of associated clients to the AP. BASE+0xA2 WLC_GET_UP X Returns the operating state of the driver. O: driver is in the DOWN or OUT state 1: driver is in the UP state BASE+0xA3 WLC_OUT X Sets the driver in the VOID				
other BSS only supports long slot timing. Per the 802.11g draft specification, an 802.11g AP can ignore these. 0: Beacons will not be ignored. 1: Beacons will be ignored BASE+0x9C WLC_SET_IGNORE_BEACONS X Sets the ignore beacon mode. BASE+0x9F WLC_GET_ASSOCLIST X When in AP mode, returns a structure containing the list of MAC addresses of associated clients to the AP. BASE+0xA2 WLC_GET_UP X Returns the operating state of the driver. 0: driver is in the DOWN or OUT state 1: driver is in the UP state BASE+0xA3 WLC_OUT X Sets the driver in the VOID	\ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \			
long slot timing. Per the 802.11g draft specification, an 802.11g AP can ignore these. O: Beacons will not be ignored. 1: Beacons will be ignored BASE+0x9C WLC_SET_IGNORE_BEACONS X Sets the ignore beacon mode. BASE+0x9F WLC_GET_ASSOCLIST X When in AP mode, returns a structure containing the list of MAC addresses of associated clients to the AP. BASE+0xA2 WLC_GET_UP X Returns the operating state of the driver. O: driver is in the DOWN or OUT state 1: driver is in the UP state BASE+0xA3 WLC_OUT X Sets the driver in the VOID				
BASE+0x9C WLC_SET_IGNORE_BEACONS BASE+0x9F WLC_GET_ASSOCLIST X Sets the ignore beacon mode. BASE+0x9F WLC_GET_ASSOCLIST X When in AP mode, returns a structure containing the list of MAC addresses of associated clients to the AP. BASE+0x42 WLC_GET_UP X Returns the operating state of the driver. 0: driver is in the DOWN or OUT state 1: driver is in the UP state BASE+0xA3 WLC_OUT X Sets the ignore beacon int *Mode mode. *AssocList; *AssocList; *Int *State *Int *State *Ind *State *I				
specification, an 802.11g AP can ignore these. 0: Beacons will not be ignored. 1: Beacons will be ignored BASE+0x9C WLC_SET_IGNORE_BEACONS X Sets the ignore beacon mode. BASE+0x9F WLC_GET_ASSOCLIST X When in AP mode, returns a structure containing the list of MAC addresses of associated clients to the AP. BASE+0xA2 WLC_GET_UP X Returns the operating state of the driver. 0: driver is in the DOWN or OUT state 1: driver is in the DOWN or OUT state 1: driver is in the UP state BASE+0xA3 WLC_OUT X Sets the driver in the VOID				
AP can ignore these. 0: Beacons will not be ignored. 1: Beacons will be ignored BASE+0x9C WLC_SET_IGNORE_BEACONS X Sets the ignore beacon mode. BASE+0x9F WLC_GET_ASSOCLIST X When in AP mode, returns a structure containing the list of MAC addresses of associated clients to the AP. BASE+0xA2 WLC_GET_UP X Returns the operating state of the driver. 0: driver is in the DOWN or OUT state 1: driver is in the DOWN or OUT state 1: driver is in the UP state BASE+0xA3 WLC_OUT X Sets the driver in the VOID				
D: Beacons will not be ignored. 1: Beacons will be ignored BASE+0x9C WLC_SET_IGNORE_BEACONS X Sets the ignore beacon mode. BASE+0x9F WLC_GET_ASSOCLIST X When in AP mode, returns a structure containing the list of MAC addresses of associated clients to the AP. BASE+0xA2 WLC_GET_UP X Returns the operating state of the driver. O: driver is in the DOWN or OUT state 1: driver is in the UP state BASE+0xA3 WLC_OUT X Sets the driver in the VOID				
BASE+0x9C WLC_SET_IGNORE_BEACONS X Sets the ignore beacon mode. BASE+0x9F WLC_GET_ASSOCLIST X When in AP mode, returns a structure containing the list of MAC addresses of associated clients to the AP. BASE+0xA2 WLC_GET_UP X Returns the operating state of the driver. 0: driver is in the DOWN or OUT state 1: driver is in the UP state BASE+0xA3 WLC_OUT X Sets the driver in the VOID	\wedge (\cap \wedge			
BASE+0x9C WLC_SET_IGNORE_BEACONS X Sets the ignore beacon mode. BASE+0x9F WLC_GET_ASSOCLIST X When in AP mode, returns a structure containing the list of MAC addresses of associated clients to the AP. BASE+0xA2 WLC_GET_UP X Returns the operating state of the driver. 0: driver is in the DOWN or OUT state 1: driver is in the UP state BASE+0xA3 WLC_OUT X Sets the ignore beacon int *Mode mode. *AssocList; *AssocList; *AssocList; *Int *State *State of the driver. O: driver is in the DOWN or OUT state 1: driver is in the UP state				
BASE+0x9C WLC_SET_IGNORE_BEACONS X Sets the ignore beacon mode. WLC_GET_ASSOCLIST X When in AP mode, returns a structure containing the list of MAC addresses of associated clients to the AP. BASE+0xA2 WLC_GET_UP X Returns the operating state of the driver. 0: driver is in the DOWN or OUT state 1: driver is in the UP state BASE+0xA3 WLC_OUT X Sets the ignore beacon int *Mode mode. *AssocList;			•	
BASE+0x9F WLC_GET_ASSOCLIST X When in AP mode, returns a structure containing the list of MAC addresses of associated clients to the AP. BASE+0xA2 WLC_GET_UP X Returns the operating state of the driver. 0: driver is in the DOWN or OUT state 1: driver is in the UP state BASE+0xA3 WLC_OUT X Sets the driver in the VOID			ignored	
BASE+0x9F WLC_GET_ASSOCLIST X When in AP mode, returns a structure containing the list of MAC addresses of associated clients to the AP. BASE+0xA2 WLC_GET_UP X Returns the operating state of the driver. 0: driver is in the DOWN or OUT state 1: driver is in the UP state BASE+0xA3 WLC_OUT X Sets the driver in the VOID	BASE+0x9C WLC_SET_IGNORE_BEACONS		X Sets the ignore beacon	int *Mode
returns a structure containing the list of MAC addresses of associated clients to the AP. BASE+0xA2 WLC_GET_UP X Returns the operating state of the driver. 0: driver is in the DOWN or OUT state 1: driver is in the UP state BASE+0xA3 WLC_OUT X Sets the driver in the VOID				
Containing the list of MAC addresses of associated clients to the AP. BASE+0xA2 WLC_GET_UP X Returns the operating state of the driver. 0: driver is in the DOWN or OUT state 1: driver is in the UP state BASE+0xA3 WLC_OUT X Sets the driver in the VOID	BASE+0x9F WLC_GET_ASSOCLIST	X		
MAC addresses of associated clients to the AP. BASE+0xA2 WLC_GET_UP X Returns the operating state of the driver. 0: driver is in the DOWN or OUT state 1: driver is in the UP state BASE+0xA3 WLC_OUT X Sets the driver in the VOID				*AssocList;
BASE+0xA2 WLC_GET_UP X Returns the operating state of the driver. 0: driver is in the DOWN or OUT state 1: driver is in the UP state BASE+0xA3 WLC_OUT X Sets the driver in the VOID				
BASE+0xA2 WLC_GET_UP X Returns the operating state of the driver. 0: driver is in the DOWN or OUT state 1: driver is in the UP state BASE+0xA3 WLC_OUT X Sets the driver in the VOID				
BASE+0xA2 WLC_GET_UP X Returns the operating state of the driver. 0: driver is in the DOWN or OUT state 1: driver is in the UP state BASE+0xA3 WLC_OUT X Sets the driver in the VOID				
state of the driver. 0: driver is in the DOWN or OUT state 1: driver is in the UP state BASE+0xA3 WLC_OUT X Sets the driver in the VOID	PAGE OLAG WHO GET LIP	V		:+ *0+-+-
D: driver is in the DOWN or OUT state 1: driver is in the UP state BASE+0xA3 WLC_OUT X Sets the driver in the VOID	BASE+0XA2 WLC_GET_UP	Х		int "State
BASE+0xA3 WLC_OUT or OUT state 1: driver is in the UP state X Sets the driver in the VOID				
BASE+0xA3 WLC_OUT 1: driver is in the UP state X Sets the driver in the VOID	\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \			
BASE+0xA3 WLC_OUT X Sets the driver in the VOID	` ())			
BASE+0xA3 WLC_OUT X Sets the driver in the VOID	\\/			
	BASE+0xA3 WI C. OUT			VOID
Our state this	BASETSANS WES_SST		OUT state. This	VOID
-			201 0.0.0. 11110	

IOCTL Document BCM43XX

			removes the driver from	_
			the UP state but preserves all driver	
			calibration settings.	
BASE+0xA4 WLC_GET_WPA_AUTH		X		t *Mode
BAGETOWITH WEG_GET_WITA_AGTIT		Α	authentication mode.	
			0: None	J)~
			1: 802.1x	
			2: PSK	
			255: Disabled	
BASE+0xA5 WLC_SET_WPA_AUTH		Х	Sets the WPA in authentication mode.	t *Mode
BASE+0xB2 WLC_GET_GMODE_PROTEC	TION CONTROL	X		t *Mode
BAGE+0XBZ WEG_GET_GINODE_ITTOTEG	HON_OONTHOL	^	gmode protection.	Woode
			0: off	
			1: local BSS only	
			2: overlapping BSS	
BASE+0xB3 WLC_SET_GMODE_PROTECT	TION_CONTROL	\sim X		t *Mode
DASELOVOS INICOST CARODE DECITO	TION CTC	V	gmode protection.	+ *N/cdc
BASE+0xC6 WLC_GET_GMODE_PROTEC	HON_C15	x (>>	Returns whether the in gmode protection	t *Mode
		\\\\\	mechanism is CTS-to-	
			self.	
		40/	0: use RTS/CTS	
			1: use CTS-to-self	
BASE+0xC7 WLC_SET_GMODE_PROTECT	TION_CTS			t *Mode
		()	CTS-to-self gmode	
DASELOVDA MUC CET EDAMEDUDOT	$\overline{}$		protection.	+ *Mada
BASE+0xDA WLC_GET_FRAMEBURST		X	Returns whether frame in bursting is being used or	t *Mode
			not.	
)	0: do not use frame	
		•	bursting	
			1: use frame bursting	
BASE+0xDB WLC_SET_FRAMEBURST	70/	X		t *Mode
			frame bursting or not.	
_				
Data Structures	// /			
typedef struct channel_infp {				
int hw_channel;) /* The chann	el that th	ne radio */	
_	//* currently	set to.	*/	
\wedge (\cap \wedge	_			
int target channel;	/* The chann	nel of the	current */	
- *	/* (I)BSS.		*/	
int scan channel;	/* If a scar	n is in pro	ogress, */	
	/* the chann			
$\langle \rangle / \wedge \rangle$	/* rently be			
channel info t;	. = 3=1 20	J = = ======	•	
7				
typedef enum wic locale {				
Wid ww = 0,	/* WorldWide	e locale */	/	
WLC THA,	/* Thailand			
WLC ISR,	/* Israel */			
(/ \sim \leftarrow)	/* Israer */ /* Jordan */			
WLC JDN,				
WLC_PRC,	/* China */			
WLC_JPN,	/* Japan */			
WLC_FCC,	/* USA */			
} wlc_locale_t;				
	5			
	Λ			

```
typedef struct wlc ssid {
       uint32 SSID_len;
                       SSID[32];
       uchar
} wlc ssid t;
typedef struct wlc_evm {
     uint32
                       Channel;
                                  /* In 500Kb/s increments */
     uint32
                       Rate;
} wlc evm t;
typedef struct srom rw {
       uint byteoff;
                                   /* byte offset */
       uint nbytes;
                                   /* number of bytes */
       uint16 buf[];
} srom_rw_t;
struct maclist {
       uint count;
                                   /* number of MAC addresses */
       struct ether addr ea[1]; /* var length array of addrs */
};
struct rateset {
                                    /* Number of rates in the set */
/* The rates to be supported in */
    uint count;
     uint8 rates[12];
                                    /* 500kb/s increments */
};
struct bandlist {
     uint count;
                                     * Number of supported bands */
     uint bands[];
                                    /* The bands supported by the card */
};
typedef struct wl bss info {
     uint32 version;
                                    /* version field */
     uint32 length;
                                    /* byte length of data in this */
                                    /* record, starting at version and */
                                    /* including IEs */
     struct ether_addr( RSSID;
     uint16 beacon period;
                                    /* units are Kusec */
     uint16 capability;
                                    /* Capability information */
     uint8 SSID_len;
uint8 SSID[32];
     struct { ( (
                  Ogount;
                                  /* # rates in this set */
           uint√
                                  /* rates in 500kbps units w/hi bit */
            mint8 rates[16];
                                   /* set if basic */
      } rateset;
                                  /* supported rates */
      uint8
             channel;
                                   /* Channel no. */
                                  /* units are Kusec */
     uint16 atim window;
                                  /* DTIM period */
     uint8
           \bigveedtim_period;
                                  /* receive signal strength (in dBm) */
      int16) RSSI;
                                  /* noise (in dBm) */
      int/8/
            phy noise;
     uint32 ie length;
                                   /* byte length of Information Elements */
} wl bss info t;
```

```
typedef struct wl_scan_results {
    uint32 buflen;
         uint32 version;
uint32 count;
wl_bss_info_t bss_info[1];
} wl_scan_results_t;
```