RW-WLAN-nX MAC HW STA

User Manual

RW-WLAN-nX-MAC-HW-STA-UM

Version 4.08

2019-04-10

RivieraWaves
http://www.rivierawaves.com/

Revision History

Version	Date	Revision Description	Author
1.00	2011-06-07	Initial release of document	Jerome Vanthournout
1.01	2012-04-05	Changed lifetime field definition (#1499) Changed acceptErrorFrame definition (#1505) Added software profiling register (#1278) Added debugPHYReg register (#1504)	Jerome Vanthournout
1.02	2012-08-06	Added baFrameReceived in THD (#1649) Added lstpReg register (#1668) Moved stateCntrlReg (#1752) Added rxPayloadCount Interrupt (#1546) Added rxdecryption Error filter (#1953) Changed reset value of encryption Key register (#1954) Added acceptNotExpectedBA filter (#1963)	Jerome Vanthournout
2.00	2012-12-10	Added 11ac support Changed rate selection mechanism for the retries (#1502) Swapped Header Control Information field and Payload Length field in Rx Header Descriptor (#1543) Update Tx/Rx Header Descritpor and Policy Table to support 11ac (#2077) Added dedicated macTxPower for DSSS/CCK and OFDM packet (#1548) Increase size of nBlankDelimiter (#2156) Added report of current AC for all response frames (#2320)	Jerome Vanthournout
2.01	2013-01-31	Updated for bandwidth management (#2432) Added rx Flow Control (#2411) Added Flag in RHD for respFrame (#2488) Diagnostic port update Cleaning	Jerome Vanthournout
2.02	2013-04-04	Increased number of absolute timers (#2647) Added acceptFrame filter for Beacon and Probe Resp (#2671) Increased max length allowed value (#2668)	Jerome Vanthournout
2.03	2013-09-20	Added AMPDU information in RHD (#2991) Added bandwidth statistic in MIB (#2989) Added medium occupancy measurement on secondary channel (#2423) Allows to add blank delimiters in front of the first MPDU in an AMPDU (#2913)	Jerome Vanthournout
2.04	2013-12-18	Added unique pattern on Transmit Payload Descriptor (#3152) Added a status bit and the possibility to generate an interrupt on the Transmit Payload Descriptor (#2312) Added tsfMgtDisable register to support multi STA mode (#3519)	Jerome Vanthournout
2.05	2014-03-20	Added possibility to halt the Beacon queue. (#3668) Clarify clen usage	Jerome Vanthournout
2.06	2014-11-03	Added specific transmit power level per rate. (#4002) Added optional frame length in THD used in case of bandwidth drop. (#3975) Added Medium time used reported in THD (#3976) Added WAPI Support (#4305) Added Coex Support (#3356)	Jerome Vanthournout
2.07	2015-02-15	Added Beamformee Support (#4635) Added Beamformer Support (#4636) Added MAC-PHY anticipated Tx Data Request (#4637) Added MAC Address search SW API (#3962)	Jerome Vanthournout

Version	Date	Revision Description	Author
2.08	2015-06-20	Updated description of Beamformee Support (#4635)	Jerome Vanthournout
3.00	2015-10-30	Added MU-MIMO TX Support (#5000) Added rxEndForTiming_p error recovery (#5681)	Jerome Vanthournout
3.01	2016-03-04	Added control of wtClk ratio by register (#5890) Increased rxStartDelayShort and rxStartDelayLong default values (#5897)	Jerome Vanthournout
3.02	2016-05-25	Added support of 32.768kHz LowPower Clock (#6127)	Jerome Vanthournout
3.03	2016-09-10	Updated port map with MU-MIMO TX additional signals	Jerome Vanthournout
3.04	2016-10-21	Aligned registers description with RTL	Jerome Vanthournout
3.05	2016-11-15	Removed deprecated defines (#6726) Added support of Timing Advertisement frame (#6719)	Jerome Vanthournout
4.00	2017-08-29	Updated THD & Policy Table for 11ax-20MHz Support (#) Added BSSColor Support (#) Added Trigger Based Transmission (#) Added Dual Nav Support (#) Removed MU-MIMO Tx support Removed support of start TXOP from register (#8085) Removed ATIM Wake-up support (#8090) Removed CFP support (#8093) Remove transmit RIFS provision (#8100) Remove quiet interval 2 (#8133) Remove provision for JIT (#8120) Remove unused HCCA interrupt (#8127)	Jerome Vanthournout
4.03	2018-04-24	Move insertion of Blank delimiter after the MPDU instead of before the MPDU delimiter. (#8646)	Jerome Vanthournout
4.04	2018-12-21	Several update in encryption, power control for trigger frame	Jerome Vanthournout
4.05	2019-01-16	Updated size of DEFAULT_PE_DURATION (#9538) Added doppler and midamble support in PolicyTable (#9546) Added DCM support in PolicyTable (#9551) Updated trigger frame filtering (#9476) Updated MIB with trigger based transmission (#9482)	Jerome Vanthournout
4.06	2019-02-06	Updated TxPower computation in case of HE_TB (#9678). Added Spatial Reuse Registers (#8789) Added Transmit Trigger Based Information Registers (#9681) Changed Rx Descriptor Management (#9210) Added interrupt to the SW when a programmed HE_TB transmission is cancelled. (#9710) Added registers providing parameters extracted from Common and User Info fields of trigger frame (#9786) Add register to disable CS in trigger based for debug purpose (#9709) Add possibility to accept all trigger frame in monitor mode (#9791)	Jerome Vanthournout

Version	Date	Revision Description	Author
		Add possibility to disable response generation of HE TB frames (#9792)	
4.07	2019-03-05	Added primaryChPosition field in scanCntrlReg Clarified payloadLengthRx in RHD (#9855) Removed Encryption Rx buffer (#9850) Replaced DualPort SRAM for WEP/TKIP SBOX by a Two Port RAM (#9866) Added unwrap-able and partially wrap-able payload feature (#9871) Added possibility to store the MAC Header along with the Payload in the RPD (#9873) Added UORA Support (#9906) Replaced baRxTrigger interrupt by rxBuffer2Trigger (#9922)	Jerome Vanthournout
4.08	2019-04-08	Changed reported error and encryption type in RHD status (#9896) Allow disabling the storage in Rx Buffer 2 (#9958) Add possibility to get and force OCW LFSR value (#9976)	Jerome Vanthournout

Table of Contents

List of Figures

List of Tables

1 Overview

1.1 Document overview

1.1.1 Purpose

This document gives information on the external interfaces of RivieraWaves's RW_WLAN-nX HW core along with the detailed register set description. The intention of this document is to give sufficient information to enable software developers to develop the MAC software for the core and system designers to be able to integrate this core into an 802.11 system effectively

The hardware portion of the RW_WLAN-nX MAC (henceforth referred to as MAC-HW, or the MAC core) is implemented as a synthesizable core in Verilog. Along with the RivieraWaves's MAC firmware it implements the complete 802.11 MAC specification. This core can be easily interfaced to an 802.11 PHY and a host CPU to form a complete solution for wireless LAN applications.

1.1.2 Scope

The document provides information about the interfaces of the MAC HW and the programming description. It does not address the internal architecture and design of HW blocks, their interaction and their implementation issues.

1.1.3 Document conventions

All registers, fields, parameter values are written using the notation *special*.

In descriptions that refer to specific register fields/bits, the notation *register.field* is used, where *register* refers to the register name and *field* refers to a named field within that register. In a 32-bit register, bit 31 is the most significant bit (MSB) and bit 0 the least significant bit (LSB).

1.1.4 Abbreviations and Acronyms

Acronym	Expansion
ACK	Acknowledgment
AP	Access Point
CF	Contention Free
CTS	Clear To Send
DCF	Distributed Coordination Function
EDCA	Enhanced Distributed Channel Access
HCCA	HCF Controlled Channel Access
HT	High Throughput
HE	High Efficienty
L-SIG	Legacy (Non-HT) Signal Field
MAC	Medium Access Control
MIB	Management Information Base
MIMO	Multiple Input Multiple Output
MU-MIMO	Multiple User Multiple Input Multiple Output

SU-MIMO	Single User Multiple Input Multiple Output
MSDU	MAC service data unit
NAV	Network Allocation Vector
OS	Operating System
PC	Point Coordinator
PHY	Physical layer
PSMP	Power Save Multi-Poll
QoS	Quality of Service
RD	Reverse Direction
RSNA	Robust Security Network Association
STA	Station
STBC	Space-Time Block Coding
TID	Traffic Identifier
VHT	Very High Throughput
WAPI	WLAN Authentication and Privacy Infrastructure
WEP	Wired Equivalent Privacy
WLAN	Wireless LAN

2 Functional description

The following sections describe the software-hardware handshake involved in controlling the HW functionality from the SW.

2.1 MAC core states

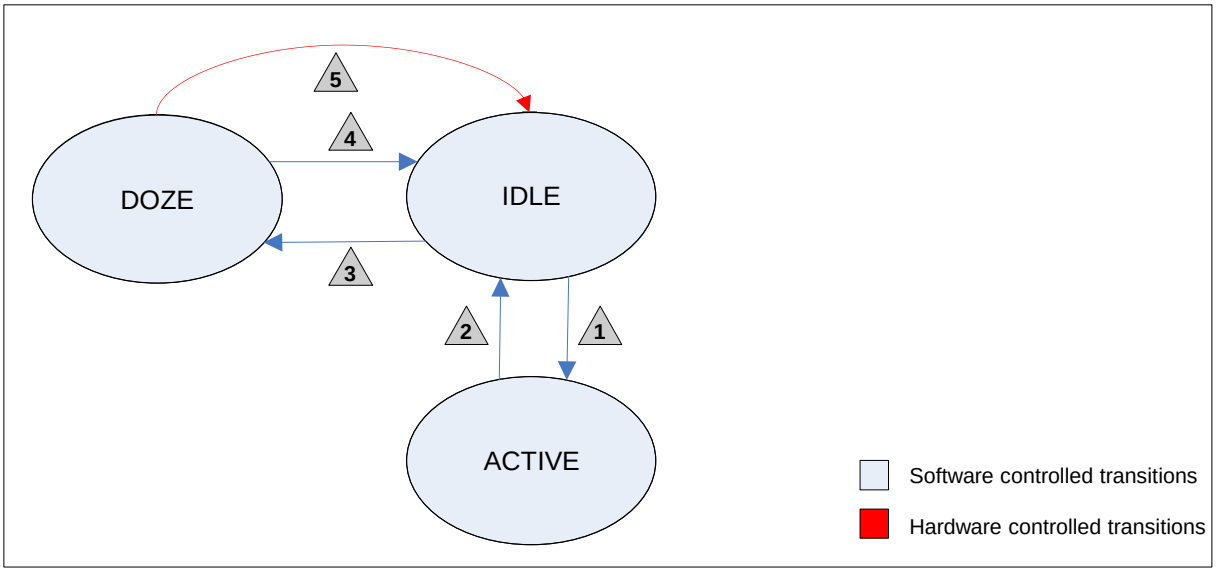


Figure 1: MAC core state and transitions

The MAC HW has three states as shown above. The HW always starts up in IDLE and can be moved between the states by software by programming the *State Control Register (stateCntrlReg)*. Any state transition performed by SW has to pass through the IDLE state. This ensures that the MAC HW internal state machines move in sync when transiting to the next state.

Transition	Driver	Remark
1	SW	SW moves the HW to ACTIVE state for the purpose of transmitting and receiving frames.
2	SW	SW moves the state of the HW to IDLE in order to stop transmit and receive operation. If the HW is actively transmitting or receiving a frame and SW programs the next state to IDLE, the HW will finish the current atomic frame exchange sequence and only then change its state to IDLE. The atomic frame exchange being currently received or transmitted is NOT aborted. An atomic frame exchange sequence defined in section *2.2.4.2.6*, *Determining frame exchange sequences and error handling*.
3	SW	SW moves the HW to DOZE state to go into a low power state.
4	SW	SW moves the HW to IDLE state in order to subsequently move into the ACTIVE state.
5	HW	HW moves its state from DOZE to IDLE based on an absTimer interrupt, or just before the DTIM Beacon is expected.

Table 1: MAC HW state transitions

2.1.1 Scanning

Passive and Active Scanning procedure is supported from SW. However, hooks in HW make it easier for the SW to perform the scanning procedure.

2.1.1.1 Passive scanning

To perform Passive Scanning:

- ✓ The SW moves the HW to IDLE if it is not already in IDLE state.
- ✓ With the HW in IDLE, the SW programs the channel number into the appropriate baseband register.
- ✓ Note that if the channel need not be changed, it is not necessary to move the HW to the IDLE state.
- ✓ The SW moves the HW from IDLE to ACTIVE, and does not program any frames for transmission.
- ✓ For the period of the Passive Scan, the SW can request the HW to pass to it only Beacons or Probe Response frames directed to this device by programming the relevant bits of the *Receive Control Register (rxCntrlReg)*. The SW can also request the HW to pass to it received Probe Response frames not directed to this device by programming the relevant bits of the *rxCntrlReg*. Note that the HW does not perform an SSID check. It is the responsibility of software to read the received Beacon and Probe Response frames and filter out the ones whose SSID do not match the desired SSID or the broadcast SSID.
- ✓ The scan timer is maintained by the SW. When it expires, the SW may perform the same steps again, on the same or different channel.

2.1.1.2 Active scanning

To perform Active Scanning:

- ✓ The SW moves the HW to IDLE if it is not already in IDLE state.
- ✓ With the HW in IDLE, the SW programs the channel number into the appropriate baseband register.
- ✓ Note that if the channel need not be changed, it is not necessary to move the HW to the IDLE state.
- ✓ The SW moves the HW from IDLE to ACTIVE, creates and links a Probe Request frame to the *Transmit AC3 Head Pointer Register (txAC3HeadPtrReg).txAC3HeadPtr* for AC_VO.
- ✓ The HW loads ProbeDelay duration into the NAV counter from the *Scan Control Register (scanCntrlReg).probeDelay*, which ensures that this device does not contend on the medium for this period. A *PHY-RXSTART.indication* in this period cancels the NAV.
- ✓ The HW performs the backoff function and transmits the chained Probe Request frame.
- ✓ For the period of the Active Scan, the SW can request the HW to pass to it only Beacons or Probe Response frames directed to this device by programming the relevant bits of the *Receive Control Register (rxCntrlReg)*. The SW can also request the HW to pass any Probe Response frame directed to any device to it by programming the relevant bits of the *rxCntrlReg*. Note that the HW does not perform an SSID check. It is the responsibility of software to read the received Beacon and Probe Response frames and filter out the ones whose SSID do not match the desired SSID or the broadcast SSID.
- ✓ The scan timer is maintained by the SW. When it expires, the SW may perform the same steps again, on the same or different channel. The SW may switch from the *MinChannelTime* to the *MaxChannelTime* by using the *General Interrupt Event Register (genIntEventReg).phyRxStart* interrupt.
- ✓ In order to reuse the Probe Request frame, the SW can point the *Next Atomic Fame Exchange Sequence Pointer* of the DMA Header Descriptor of the Probe Request frame to the starting location of the same DMA Header Descriptor.
- ✓ When the active scanning process is over, software may remove the chained Probe Request frame from the linked list to prevent it from being transmitted again.

2.1.2 Active mode

The ACTIVE state is used to perform any kind of frame transmission and reception, and for the Scanning function. To move the core into the ACTIVE state:

- ✓ The SW moves the HW to IDLE if it is not already in IDLE state.
- ✓ With the HW in IDLE, it programs the channel number into the appropriate baseband register.
- ✓ The SW programs the various registers that control active mode transmission and reception, like the Control1, Control2, Beacon Interval register, etc.
- ✓ The SW moves the HW from IDLE to ACTIVE.
- ✓ Depending on the configuration bits, the device can start up as an AP, as a STA in an infrastructure network, or as a STA in an independent network. It transmits and receives any frames in the active mode.

2.1.3 Power save (Doze) support

The core supports Power Save in AP as well as STA mode.

2.1.3.1 Power Save support as an AP

It supports associated STAs in the DOZE state. After transmission of a DTIM Beacon frame, the SW transmits the buffered group addressed frames through the required DMA channel.

The core as an AP will respond to a PS-Poll frame with an ACK frame. The PS-Poll frame is passed to SW. The SW subsequently sends the required frame to the STA from which the PS-Poll was received.

2.1.3.2 Power Save support as a STA

2.1.3.2.1 Protocol based power save

2.1.3.2.1.1 Power save mode

The core enters power save mode when the software programs *State Control Register (stateCntrlReg).nextState* = DOZE.

The MAC HW follows the following procedure when it is programmed into the DOZE state:

1. It first turns both the receive channels to HALTED to ensure that it does not try to move any received frames into system memory when it comes out of DOZE, until the platform clocks have been turned on.
2. If the six secondary clock domains (*macCoreTxClk*, *macCoreRxClk*, macCryptClk, *macPITxClk* and *macPIRxClk* and *macWTClk*) are not already off, it explicitly turns them off using the individual enable signals.
3. It asserts the *macLPClkSwitch* signal to switch the *macLPClk* from *macCoreClk* frequency to 32kHz.
4. Finally, it de-asserts the *macPriClkEn* signal to the platform clock controller to turn off the following three primary clock domains: *macPIClk*, *macCoreClk*, and *mpIFClk*.

At this point, only the 32kHz *macLPClk* domain in the MAC is active, and no registers or memories on the host interface are accessible. If SW needs access to the available registers[1] on the *macPIClk* domain it should explicitly request the platform clock controller to turn on this clock domain.

2.1.3.2.1.2 Auto wake-up mode - legacy

The core follows different behavior depending on the value of the *Doze Control 2 Register (dozeCntrl2Reg).wakeUpSW* bit.

[1] Access is permitted to all registers in the *macPISlaveClk* domain in the DOZE state.

If the wakeUpSW bit is set:

A wake-up signal (*platformWakeUp*) is asserted to the external clock unit for turning on the platform clocks and subsequently moving the CPU out of its low power state. The HW remains in the DOZE state. Once the SW is woken up, subsequent behavior is as described in section *2.1.3.2.1.3*, *Explicit wake-up from SW*.

If the wakeUpSW bit is not set:

The core transitions to the IDLE state (*stateCntrlReg.currentState* = IDLE and *stateCntrlReg.nextState* = IDLE) when any of the following conditions is true:

1. The core has been in DOZE state for *Doze Control 1 Register (dozeCntrl1Reg).listenInterval* number of beacon intervals. The core will transition to ACTIVE at the next TBTT.
2. The core will transition to IDLE at the next TBTT if the next beacon is expected to be a DTIM beacon and *dozeCntrl1Reg.wakeUpDTIM* bit is set.
3. One of the absolute timers expired.

The following procedure is followed by the MAC HW when it moves out of the DOZE state:

1. It first deasserts the *macLPClkSwitch* signal to turn the 32 kHz clock back to the *macCoreClk* frequency.
2. Next, it asserts the *macPriClkEn* signal to the platform clock controller to turn on the three primary clock domains: *macPIClk*, *macCoreClk* and *mpIFClk*.
3. The six secondary clock domains (*macCoreTxClk*, *macCoreRxClk*, *macCryptclk*, *macPITxClk*, *macPIRxClk* and *macWTClk*) are turned on as required.

After the HW wakes up, it generates an interrupt to the SW and stays in IDLE state.

In this case, the only way to wake up the HW is by programming *Doze Control 2 Register (dozeCntrl2Reg).wakeUpFromDoze*. When asserted, the HW re-enables the clocks and moves out of DOZE to IDLE as previsously described. This is explained in the following section *2.1.3.2.1.3*, *Explicit wake-up from SW*.

2.1.3.2.1.3 Explicit wake-up from SW

SW has the ability to explicitly wake up the HW.

To perform explicit wake-up:

1. Request the platform clock controller to turn on *macCoreClk* clock domain, so that it has access to the *State Control Register (stateCntrlReg)*.
2. Program the HW to move to IDLE state. The MAC HW follows the procedure given below when it moves out of the DOZE state.
3. Program the HW to move to ACTIVE state. After the HW wakes up, it starts passively listening to the medium

The following procedure is followed by the MAC HW when it moves out of the DOZE state:

1. First, it asserts the *macPriClkEn* signal to the platform clock controller to turn on the three primary clock domains: *macPIClk*, *macCoreClk* and *mpIFClk*.
2. Next, it deasserts the *macLPClkSwitch* signal to turn the 32kHz clock back to the *macCoreClk* frequency.
3. The five secondary clock domains (*macCoreTxClk*, *macCoreRxClk*, *macCryptClk*, *macPITxClk*, *macPIRxClk* and *macWTClk*) are turned on as required.

2.1.3.2.2 Design based power save

Portions of the MAC HW logic are turned off whenever possible to achieve lower active state power consumption for battery-powered devices.

Refer to section *8.2.3*, *Clock domains and MAC HW functional blocks*. Different functional blocks of the MAC HW are supplied with different primary and secondary clocks. The secondary clocks are turned off in ACTIVE state when the functional block being supplied by that clock domain is not required to be running.

This feature can be disabled at run time by unsetting the *MAC Control 1 Register (macCntrl1Reg).activeClkGating* bit.

2.2 Transmit operation

2.2.1 Introduction

High-level frame transmission control is in SW, while the fine-grained transmission control is in HW. When a *protocol trigger* or *transmission trigger* is generated from the HW, the SW is responsible for chaining frames to the required DMA queue. An *early protocol trigger* can be generated to SW if required to take care of latency in performing SW operations. When the DMA channel is triggered, the MAC HW reads frames from the system memory and moves then into the MAC HW Transmit FIFO.

The HW has six logical DMA channels on transmission. Four are used for EDCA, and map to the four EDCA ACs. One is used for Trigger Based Transmission and the last is used for Beacon/Broadcast frames transmission.

Only one channel is in use at any time, i.e. there is no contention or arbitration between the individual channels. The required channel is selected by the MAC HW depending on the protocol on air. All the DMA channel or queues feed a single FIFO RAM in the MAC HW, hereafter called the Transmit FIFO. The Transmit FIFO acts as an intermediate stage of buffering to smoothen out the bursty flow (if any) when reading from the system memory on the internal interconnect, and transmitting it at a constant rate to the baseband.

The HW reads frames from the required DMA queue, performs transmission checks (Refer section *2.2.4*, *MAC core transmission procedure*) and controls the transmission, and writes back the transmission status of each transmission in the descriptor for the frame. The frame format is as described in the section *6.1*, *Transmit MPDU template*, while the DMA descriptor structure is described in section *3*, *DMA descriptors*.

2.2.2 Transmit DMA channel states

A HW DMA channel can be in four states: HALTED, PASSIVE, ACTIVE and DEAD. The *DMA Status 1 Register (dmaStatus1Reg).txState* bits indicate the state of the HW Transmit DMA channel.

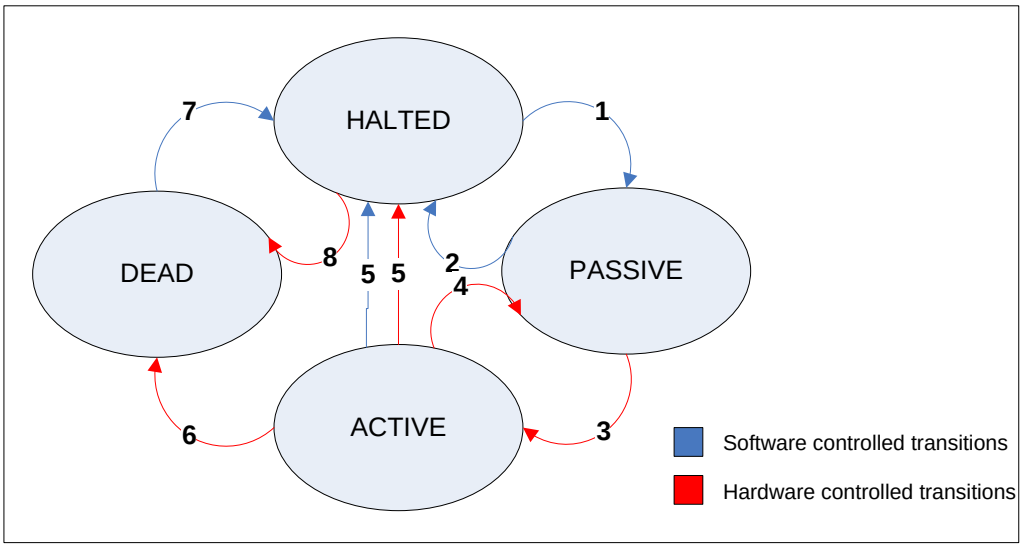


Figure 2: Transmit DMA channel states

State Name	State Description
HALTED	A DMA channel starts up in this state after reset. It may also transition to this state under other conditions. The DMA channel waits in this state if it does not have a valid linked list to handle. Multiple channels can be in HALTED simultaneously.
PASSIVE	In this state, the DMA channel has a valid linked list to handle, but does not need to actively process it. It waits for a trigger from the MAC Controller to move into the ACTIVE state and start DMA operations. Multiple channels can be in PASSIVE simultaneously.
ACTIVE	In this state, the DMA channel follows the transmit linked list, reads descriptors and moves the data from memory buffers to the Transmit FIFO. Only one channel can be in ACTIVE at any time.
DEAD	In this state, the DMA channel has encountered an error and waits for error recovery from SW.

Transition No.	Transition Description
1	HALTED to PASSIVE This transition occurs if the *DMA Control Register (dmaCntrlReg).newHead* bit for that channel is set and the *queueHeadPointer* for that channel has a valid address. This transition also occurs if the *newTail* bit for that channel is set. Refer section *2.2.3*, *Transmit DMA channel control and procedure* for more details.
2	PASSIVE to HALTED This transition occurs when the SW sets the *newHead* bit.
3	PASSIVE to ACTIVE This transition occurs when triggered by the MAC Controller to start (or continue) the DMA channel.
4	ACTIVE to PASSIVE This transition occurs when the MAC Controller ends the current transmission operation and the *DMA Control Register (dmaCntrlReg).haltACxAfterTXOP* bit for that channel is not set. This transition also occurs when the MAC Controller issues a retry request.
5	ACTIVE to HALTED This transition occurs under the following conditions: 1. The current THD is the last descriptor of the linked list, indicating that no more MPDUs are queued 2. *DMA Control Register (dmaCntrlReg).haltACxAfterTXOP* bit is set 3. The SW sets the *newHead* bit **Condition 2**: Normally, a DMA channel moves from ACTIVE to PASSIVE when the MAC core ends the current transmission. Optionally, it may move from ACTIVE to HALTED if the *DMA Control Register (dmaCntrlReg).haltACxAfterTXOP* bit[2] is set. If the *TXOPLimit* for the AC that has won contention is zero, then the channel moves to HALTED state after attempting one frame exchange sequence. If the *TXOPLimit* for the AC that has won contention has a non-zero value, then the DMA channel moves to HALTED state when the current TXOP ends. This feature allows the SW to perform one of the following: 1. Remove the pending frames (if any) and chain a new set of frames for the list. In this case, it sets the *newHead* bit. 2. Chain new descriptors at the end of the queue. In this case, it sets the *newTail* bit. 3. If frames are still pending in the queue, SW can also set the *newTail* bit without actually chaining any new descriptors at the end of the queue. This allows the HW to continue with the pending frames.

[2] The *haltACxAfterTXOP* is not expected to be set for AC queues which have the TXOPLimit parameter set to zero to prevent the DMA channel from unnecessarily halting every time one frame exchange sequence is attempted.

6	ACTIVE to DEAD This transition occurs when an error occurs. The following errors can occur during the Transmit DMA operation: 1. Length mismatch, i.e. there is less data in the MPDU buffers than the *frameLengthTx* indicated. 2. The *uPatternTx* field does not contain the expected pattern. 3. If the *Next Atomic Frame Exchange Sequence Pointer* or the *Next MPDU Descriptor Pointer* is not valid (i.e. the 2 LSBs of these address fields are not equal to 2'b00). 4. If the Policy Table Address in a DMA Descriptor is not valid (i.e. the address field is null or the 2 LSBs are not equal to 2'b00). 5. There is a bus error while performing any DMA operations. Refer to section *9.3, Terminal DMA errors* for details.
7	DEAD to HALTED This transition occurs when the state machine is reset, as part of the error recovery from SW.
8	HALTED to DEAD This transition occurs if the *DMA Control Register (dmaCntrlReg)*.*newHead* bit for that channel is set and the *queueHeadPointer* for that channel does not have a valid address (i.e. the address field is null or the 2 LSBs are not equal to 2'b00). Refer to section *9.3, Terminal DMA errors* for details.

Table 2: Transmit DMA state table

2.2.3 Transmit DMA channel control and procedure

Each logical DMA channel has its own linked list that is prepared by SW and consumed by HW. Each DMA channel has the starting address of the first *Transmit DMA Header Descriptor* of the queue programmed in a *queueHeadPointer* register (e.g. *Transmit AC0 Head Pointer Register (txAC0HeadPtrReg)*) in the MAC HW. There are five Queue Head Pointer registers corresponding to the five logical queues.

Two semaphore bits per queue are used to control synchronization between SW and HW: *newHead* and *newTail*.

2.2.3.1 Creating descriptor trees from SW

The below diagrams show all the possible ways in which Transmit Header Descriptors can be linked by SW for HW to understand. Each logical column link of descriptors contains a single unit of transmission (or an atomic frame exchange sequence as explained in section *2.2.4.2.6, Determining frame exchange sequences and error handling*) and must end with a *Next MPDU Descriptor Pointer* = 0. The logical row link of starting descriptors starts an atomic frame exchange sequence. An RTS or CTS frame can only be chained by SW at the start of the atomic frame exchange sequence, not inside the atomic frame exchange sequence. An atomic frame exchange sequence is so called since it is discarded in toto if any intermediate frame fails.

Figure 3: Possible descriptor lists shows the three possible scenarios.

1. The first column link shows a fragmented MSDU which is linked behind an RTS frame. Note that the RTS frame chained from SW is optional. If the RTS frame is discarded for any reason, the HW updates the status of the entire unit in the Header Descriptor of the RTS frame only and uses the *Next Atomic Frame Exchange Sequence Pointer* to jump to the start of the next atomic frame exchange sequence. It does not traverse the column list. Similarly, if a particular fragment fails, the status is updated in the Header Descriptor of that fragment and the HW uses the *Next Atomic Frame Exchange Sequence Pointer* to jump to the start of the next transmission unit.

2. The second column link shows an A-MPDU which is linked behind a self-CTS frame. Note that the self-CTS frame chained from SW is optional. An explicit BAR should be chained at the end of the A-MPDU to allow HW to transmit it if the A-MPDU does not get its expected response.
3. The third column link shows a single unfragmented MSDU. This may optionally have an RTS or CTS frame chained before it as in scenario #1 or #2.

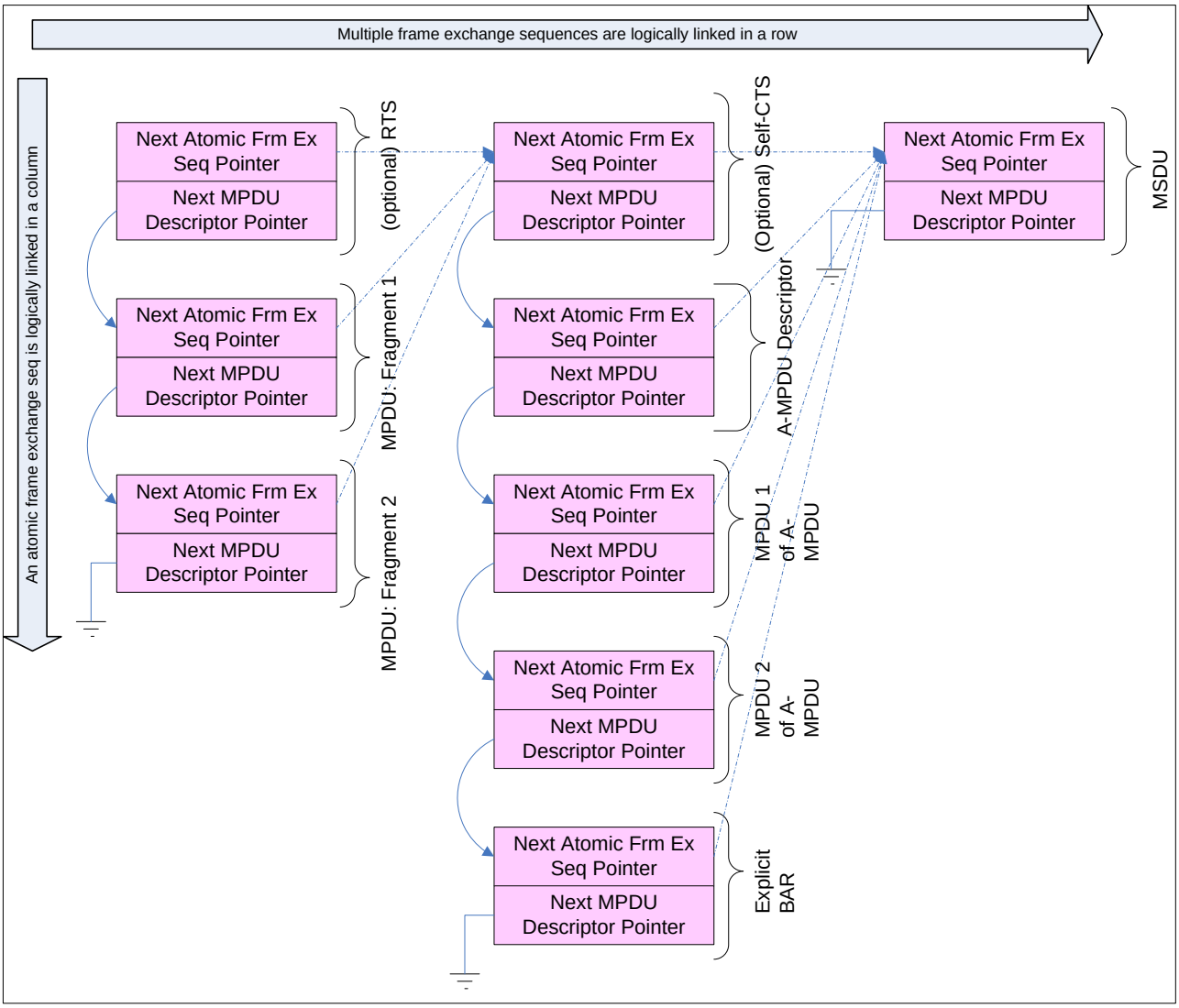


Figure 3: Possible descriptor lists

2.2.3.2 Setting the *newHead* bit

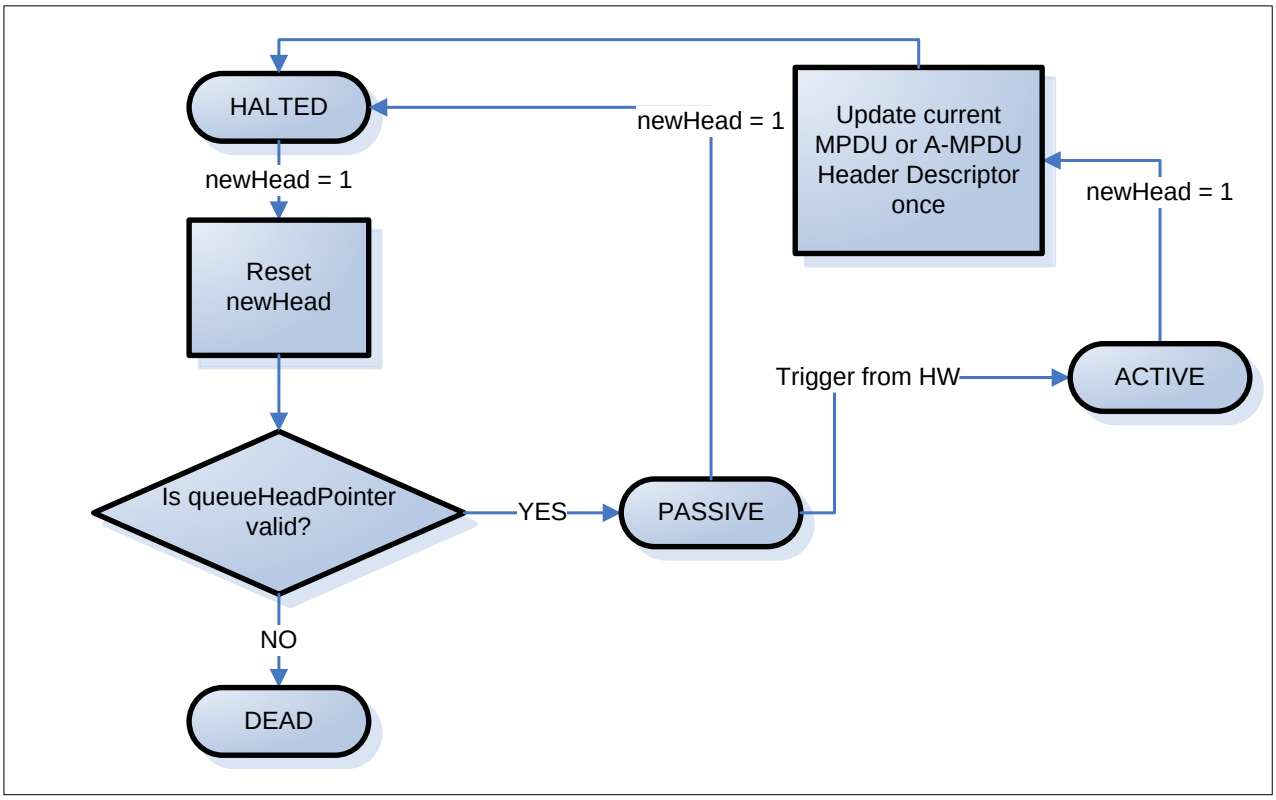
When SW adds a new linked list to a *queueHeadPointer*, it sets the corresponding *newHead* bit for that DMA channel. It is recommended that the *queueHeadPointer* is not changed if the SW finds that the *newHead* bit is already set. The effect of the *newHead* bit is different depending on the current state of the DMA channel.

1. If the DMA channel is HALTED, the HW reads the *newHead* bit immediately and resets it, and reads the *queueHeadPointer* register. If the *queueHeadPointer* contains a valid address, the DMA channel moves to PASSIVE and copies the contents of the *queueHeadPointer* to the *statusPointer* register for that channel; else that DMA channel moves to DEAD state.
2. If the DMA channel is PASSIVE, then the DMA channel first moves to HALTED and subsequent procedure is as per #1.

3. If the DMA channel is ACTIVE, then the DMA channel moves to HALTED state after the current MPDU or A-MPDU Header Descriptor status has been updated once. Subsequent procedure is as per #1.

The *statusPointer* register for a Transmit DMA channel indicates the address of the non-A-MPDU Header Descriptor or A-MPDU Header Descriptor for which the status has not been updated yet, or the last Header Descriptor of the list in case the *Next Atomic Frame Exchange Sequence Pointer* is null. It is controlled only by the HW.

SW is recommended to set the *newHead* bit for a DMA channel only if the state of the channel is HALTED. Provision is given for setting the *newHead* bit even when a DMA channel is ACTIVE or PASSIVE, but should be used sparingly under specific error recovery options or to implement an unforeseen feature in the future that cannot be supported otherwise. Note that SW is free to change the *queueHeadPointer* anytime, in order to keep the address of a new linked list programmed in HW and may hold off setting the *newHead* bit until the current channel operation is complete and it has moved to HALTED or PASSIVE state.



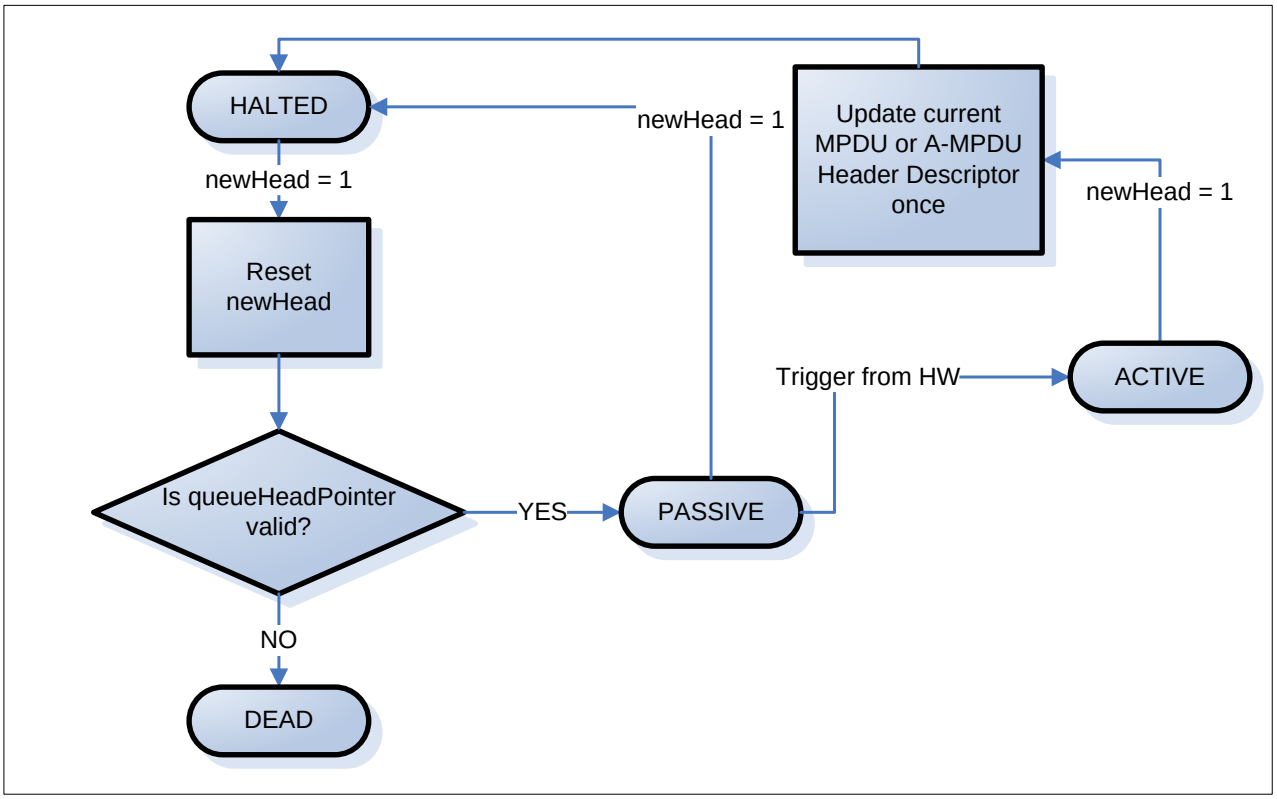


Figure 4: Effect of setting the newHead bit

2.2.3.3 Setting the *newTail* bit

Irrespective of the state of the HW DMA channel, SW can attached descriptors for new Atmic Frame Exchange Sequences to HW by adding them to the end of the existing queue. Different descriptors need to be updated when chaining additional descriptors to HW, depending on whether the last Atomic Frame Exchange Sequence in the list is a singleton MSDU, fragmented MSDU, A-MPDU and whether it contains a control frame chained from SW.

Case of Singleton MSDU: If the last Atomic Frame Exchange Sequence is a singleton MSDU, the *Next Atomic Frame Exchange Sequence Pointer* will not be populated in its single Transmit Header Descriptor. When SW wants to add a new Atomic Frame Exchange Sequence to the list, it has to update the *Next Atomic Frame Exchange Sequence Pointer* in this Transmit Header Descriptor.

Case of Fragmented MSDU: If the last Atomic Frame Exchange Sequence is a fragmented MSDU, the *Next Atomic Frame Exchange Sequence Pointer* will not be populated in any of the Transmit Header Descriptors describing the fragments. When SW wants to add a new Atomic Frame Exchange Sequence to the list, it has to update the *Next Atomic Frame Exchange Sequence Pointer* in ALL the Transmit Header Descriptors describing the fragments. If SW updates only the last fragment, and HW finds an error in one of the intermediate fragments, then the HW will halt since it will think that there is no next Atomic Frame Exchange Sequence chained.

Case of A-MPDU: If the last Atomic Frame Exchange Sequence is an A-MPDU, the *Next Atomic Frame Exchange Sequence Pointer* will not be populated in the Transmit Header Descriptors describing the A-MPDU, the constituent MPDUs, or the Explicit BAR. When SW wants to add a new Atomic Frame Exchange Sequence to the list, it has to update the *Next Atomic Frame Exchange Sequence Pointer* in the Transmit Header Descriptors of the A-MPDU and the Explicit BAR frame. It does not need to update the *Next Atomic Frame Exchange Sequence Pointer* in the Transmit Header Descriptors of the constituent MPDUs of the A-MPDU.

Case of SW formed control frame: If the last Atomic Frame Exchange Sequence contains a SW formed control frame, the *Next Atomic Frame Exchange Sequence Pointer* will not be populated in its single Transmit Header Descriptor. When SW wants to add a new Atomic Frame Exchange Sequence to the list, it has to update the *Next Atomic Frame Exchange Sequence Pointer* in this Transmit Header Descriptor.

newTail procedure: After updating the Transmit Header Descriptors of the last atomic frame exchange sequence, SW sets the *newTail* bit of that DMA channel to inform the HW that new descriptors have been added at the end of the queue. The previous state of the *newTail* bit need not be checked by the SW. The effect of the *newTail* bit is different depending on the current state of the DMA channel.

1. If the DMA channel is in PASSIVE, the *newTail* bit is reset immediately.
2. If the DMA channel is in HALTED, then the DMA channel moves to PASSIVE and the *newTail* bit is reset.
3. If the DMA channel is in ACTIVE, and the last known Next Atomic Frame Exchange Sequence Pointer is Valid, then the *newTail* bit is reset immediately.
4. If the DMA channel is in ACTIVE, and the last known Next Atomic Frame Exchange Sequence Pointer is null, then the *newTail* bit is not reset. The DMA channel first moves to HALTED at the end of the linked list (by its default transition), and subsequent procedure is as per #2.

Behavior of HW when a channel moves from PASSIVE to ACTIVE is described in section *2.2.3.5*, *Transmit DMA channel HW procedure*.

2.2.3.4 Retransmitting unsuccessful MPDUs of an A-MPDU

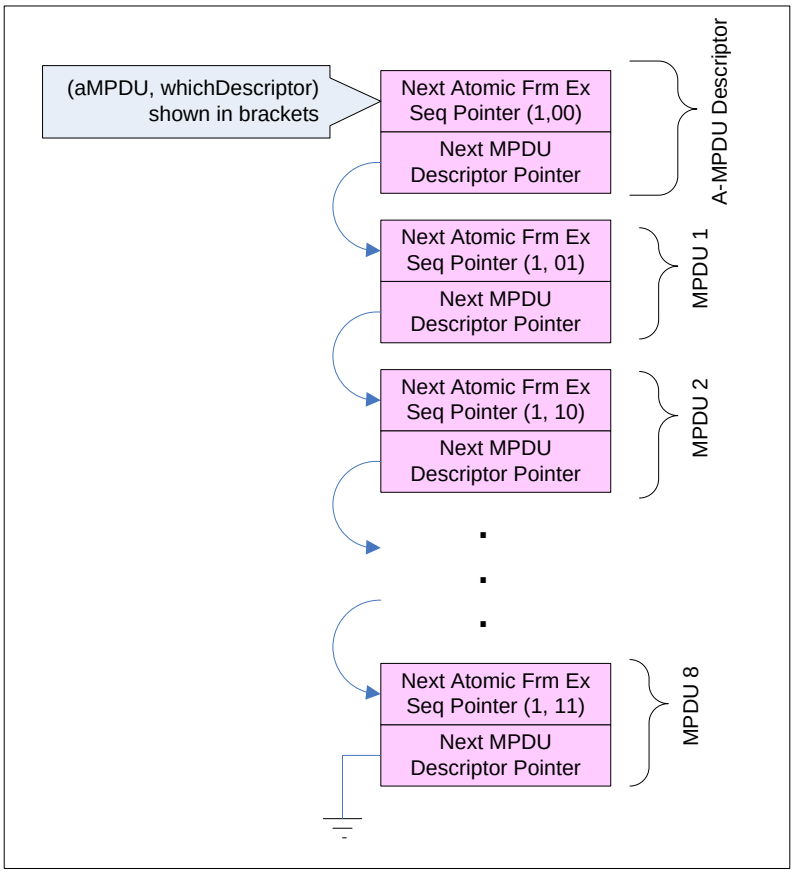


Figure 5: A-MPDU descriptor list

An A-MPDU is handed to HW linked in a logical column forming 1 atomic frame exchange sequence, with the A-MPDU THD + multiple MPDUs THDs chained in a column, as shown in *Figure 5: A-MPDU descriptor list*.

If the BA for this A-MPDU indicates that some MPDUs were unsuccessfully delivered, SW may retransmit the individual MPDUs of the A-MPDU as singleton frames, or as a new A-MPDU. Note that if all frames of an A-MPDU are unsuccessful, the same A-MPDU should be retransmitted without any changes.

2.2.3.4.1 Retransmitting as singleton frames

If unsuccessful MPDUs of the A-MPDU need to be retransmitted as singleton frames, then the linked list given to HW should resemble the singleton MPDU linked list structure, as described in section *2.2.3.1*, *Creating descriptor trees from SW*. Each MPDU of the original A-MPDU that needs to be transmitted as singleton MPDUs should be in a separate "column", i.e. should be treated as an atomic frame exchange sequence.

From the A-MPDU described in *Figure 5: A-MPDU descriptor list*, say, MPDUs 1 and 2 were successful, MPDU 3 was unsuccessful, MPDUs 4 and 5 were successful, MPDU 6 was unsuccessful and MPDUs 7 and 8 were successful. SW should chain back to HW a list starting with the first unsuccessful frame, in this case, MPDU 3.

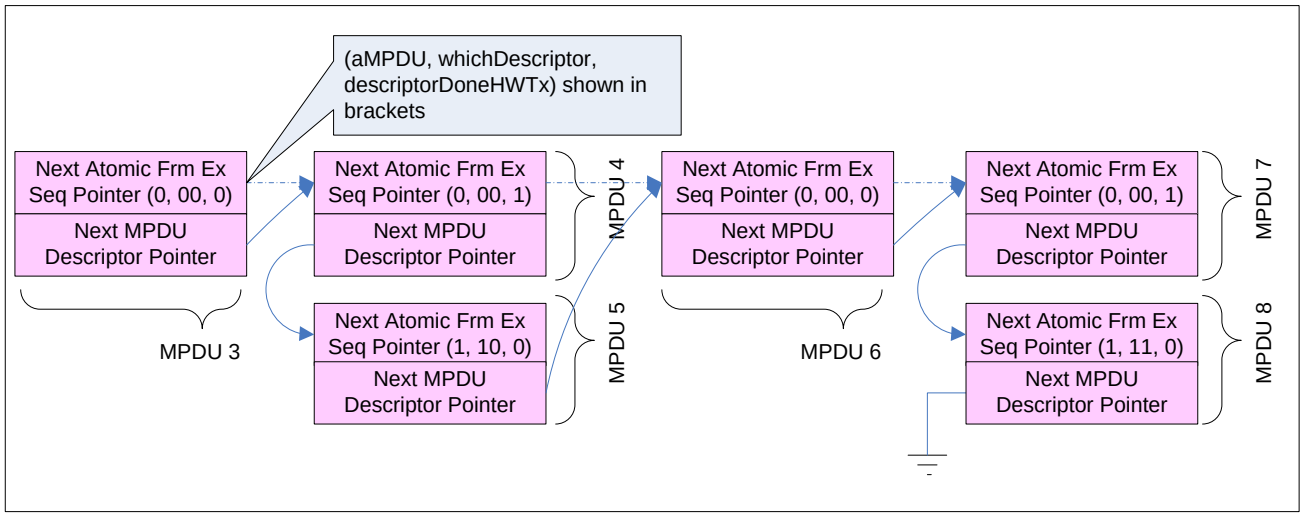


Figure 6: Linked list when retransmitting MPDUs of an A-MPDU as singleton MPDUs

MPDU 3 should be passed to HW as an atomic frame exchange sequence. The original setting of {*aMPDU*, *whichDescriptor*, *descriptorDoneHWTx*} must be changed from {1, 10, 0} to {0, 00, 0}. The Next Atomic Frame Exchange Sequence Pointer should be corrected updated.

MPDU 4 should be passed to HW as an atomic frame exchange sequence to be skipped. The original setting of {*aMPDU*, *whichDescriptor*, *descriptorDoneHWTx*} must be changed from {1, 10, 0} to {0, 00, 1}. Next Atomic Frame Exchange Sequence Pointer should be corrected updated. The HW finding the *descriptorDoneHWTx* = 1 and the *whichDescriptor* = 00, will skip the entire atomic frame exchange sequence and start at the next column.

For MPDU 5, nothing needs to be done. The HW will discard what it thinks is an “atomic frame exchange sequence” and hence it will skip over MPDUs 4 and 5 and will transmit 6 next.

MPDU 6 needs similar SW processing as MPDU 3, while MPDU 7 and 8 need similar SW processing as MPDUs 4 and 5.

2.2.3.4.2 Retransmitting as an A-MPDU

If unsuccessful MPDUs of the A-MPDU need to be retransmitted as a new A-MPDU, then the linked list given to HW should resemble the A-MPDU linked list structure, as described in section *2.2.3.1*, *Creating descriptor trees from SW*. Successful MPDUs which need not be retransmitted can remain in a "column" without any change, except that each MPDU should have the *descriptorDoneHWTx* set, and HW will skip them.

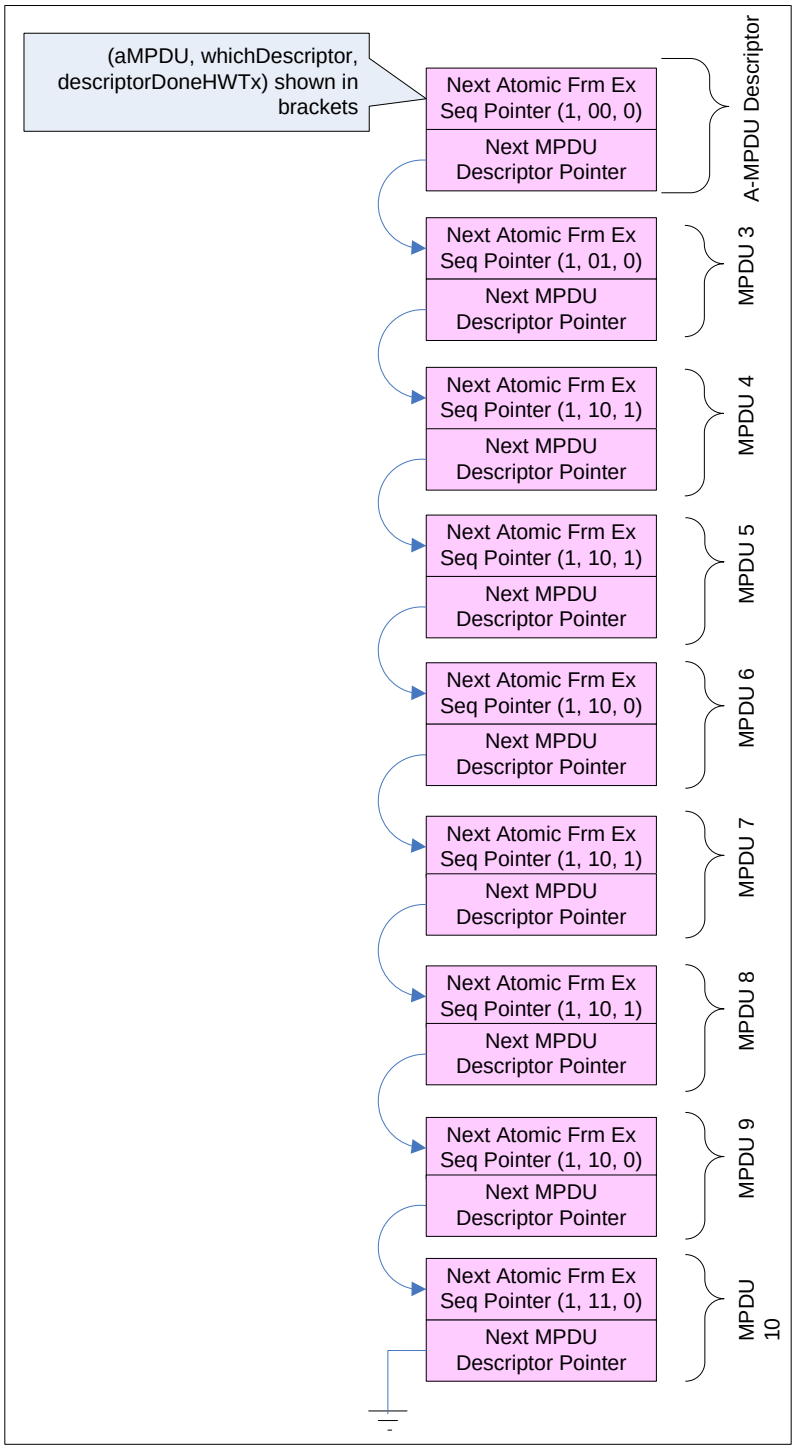


Figure 7: Linked list when retransmitting MPDUs of an A-MPDU as a new A-MPDU

From the A-MPDU described in *Figure 5: A-MPDU descriptor list*, say, MPDUs 1 and 2 were successful, MPDU 3 was unsuccessful, MPDUs 4 and 5 were successful, MPDU 6 was unsuccessful and MPDUs 7 and 8 were successful. If the number of BA receive buffers at the receiver are 8, SW can form an A-MPDU of old MPDUs 3 and 6 and two new MPDUs 9 and 10. SW should chain back to HW a list starting with the first unsuccessful frame, in this case, MPDU 3.

MPDU 3 should be passed to HW as the first MPDU of an A-MPDU. The original setting of {*aMPDU*, *whichDescriptor*, *descriptorDoneHWTx*} must be changed from {1, 10, 0} to {1, 01, 0}.

MPDUs 4 and 5 should be passed to HW as intermediate MPDUs of an A-MPDU to be skipped. The original setting of {*aMPDU*, *whichDescriptor*, *descriptorDoneHWTx*} must be changed from {1, 10, 0} to {1, 10, 1}. The HW finding the *descriptorDoneHWTx* = 1 and the *whichDescriptor* = 10, will skip that particular MPDU Transmit Header Descriptor start at the next one.

For MPDU 6 nothing needs to be done. MPDUs 7 and 8 needs similar SW processing as MPDUs 4 and 5.

2.2.3.5 Transmit DMA channel HW procedure

The DMA channel moves from PASSIVE to ACTIVE when some transmission needs to be made from it. Operation starts at the descriptor pointed to by the current *statusPointer* register for that channel.

In the ACTIVE state, the *descriptorDoneHWTx* bit is checked for any THD that is fetched. This bit is set by the HW to indicate that the descriptor has been handled by the HW. This bit may be set by SW in some conditions, to indicate that this descriptor (and maybe other descriptors) should be skipped by HW, and the MPDUs associated with the skipped descriptors should not be transmitted.

- ✓ If the *descriptorDoneHWTx* bit = 0, then the frame associated with this THD is fetched. After it has been used:
  - If the *Next MPDU Descriptor Pointer* field is valid, then the HW uses the *Next MPDU Descriptor Pointer* to fetch the next MPDU in the same atomic frame exchange sequence.
  - If the *Next MPDU Descriptor Pointer* field is null:
    - If the last known *Next Atomic Frame Exchange Sequence Pointer* is valid, then the HW uses the last known *Next Atomic Frame Exchange Sequence Pointer* to jump to the start of the next atomic frame exchange sequence. It does not traverse the remaining column list.
    - If the last known *Next Atomic Frame Exchange Sequence Pointer* is null, then the HW moves to the HALTED state when the DMA channel is stopped by the MAC Controller[3].
  - Irrespective of the value of the *Next MPDU Descriptor Pointer*, if the current atomic frame exchange sequence needs to be skipped over[4]:
    - If the *Next Atomic Frame Exchange Sequence Pointer* is valid, then the HW uses the *Next Atomic Frame Exchange Sequence Pointer* to jump to the start of the next atomic frame exchange sequence. It does not traverse the remaining column list.
    - If the *Next Atomic Frame Exchange Sequence Pointer* is null, then the HW moves to the HALTED state when the when the DMA channel is stopped by the MAC Controller.
- ✓ If the *descriptorDoneHWTx* bit = 1, then the frame associated with this THD is NOT fetched. Subsequent behavior depends on the setting of the *whichDescriptor* field:
  - If the *whichDescriptor* = 0:
    - If the *Next Atomic Frame Exchange Sequence Pointer* is valid, then the HW uses the *Next Atomic Frame Exchange Sequence Pointer* to jump to the start of the next atomic frame exchange sequence. It does not traverse the remaining column list.
    - If the *Next Atomic Frame Exchange Sequence Pointer* is null, then the HW moves to the HALTED state when the DMA channel is stopped by the MAC Controller.
  - If the *whichDescriptor* != 0:
    - If the *Next MPDU Descriptor Pointer* is valid, then the HW uses the *Next MPDU Descriptor Pointer* field to fetch the next MPDU in the same atomic frame exchange sequence.

[3] The way this works is as follows: The DMA puts frame 1 in the FIFO then fetches frame 2. After fetching frame 2, it discovers that the queue has ended. It should not immediately move to HALTED at this point, since the MAC Controller may issue a retry request for frame 1 or frame 2 which needs to be handled by the DMA engine. The MAC Controller may also issue a status update request which needs to be handled by the DMA engine.

[4] Like for an explicit BAR attached at the end of the queue.

- If the *Next MPDU Descriptor Pointer* is null, then the HW moves to the HALTED state when the DMA channel is stopped by the MAC Controller.

2.2.4 MAC core transmission procedure

2.2.4.1 Beacon transmission

The core supports beacon transmission as an AP in a BSS or as a STA in an IBSS. A beacon is attempted to be transmitted at the primary and secondary TBTTs. The primary TBTT is calculated as: TSF mod *beaconInterval* = 0. The secondary TBTT is calculated as: TSF mod *beaconInterval* = 0 + ½ BI.

The core replaces the Timestamp and DTIM Count fields of the beacon frame with the latest values before transmission.

A *General Interrupt Event Register (genIntEventReg).impPriTBTT* interrupt is generated *Beacon Control 1 Register (bcnCntrl1Reg).impTBTTPeriod* microseconds before the Primary TBTT to indicate to the SW to keep the Primary Beacon frame ready. Similarly for the Secondary TBTT.

At any TBTT, if the Beacon DMA channel is PASSIVE, the HW reads the descriptor pointed to by the *bcnStatusPointer* register. In case of multiple-BSSID support where multiple Beacon frames need to be transmitted, the SW can chain multiple Beacons in the Beacon channel linked list. The HW will transmit all the chained beacons one after another, separated by SIFS. Note that SW can advertise different set of parameters in different Beacon, except parameters which have an effect on the MAC HW, for instance: BSS Basic Rate Set, BSS Basic MCS Set, Beacon Interval, DTIM Interval and Slot duration (short or long slot). In addition, since HW maintains the network synchronization, the networks belonging to each virtual AP (BSSID) will remain network synchronized, i.e. will have TSF and DTIM Count in synch.

If the Beacon DMA channel is HALTED, then the Beacon frame transmission is skipped and normal EDCA transmission carries on.

2.2.4.2 EDCA transmission

Software programs the EDCA TXOP registers (TXOP Limit, AIFSN, CWMin and CWMax) with the corresponding parameters from the last received beacon, when the core is a QSTA or from the parameters it has advertised in the beacons when the core is a QAP. The SW is free to use a different set of parameters that what is advertised in the beacon, when the core is a QAP.

The AIFSN, CWMin and CWMax values are used by the core to control the backoff for these ACs, while the TXOP Limit is used by the core to control the medium occupancy timer when an AC has won contention to the medium.

The MAC HW generates an early protocol trigger as explained in section *2.2.4.2.1*, *Generating the early protocol trigger*. When the MAC HW wins contention, it determines whether SW has scheduled a transmission. This is explained in section *2.2.4.2.2*, *Acquiring an EDCA TXOP*. If HW finds that SW has scheduled a frame transmission, then it needs to determine whether a 20 MHz, 40 MHz, 80MHz or 160MHz TXOP needs to be acquired as described in section *2.2.4.2.3*, *Determining 20/40/80/160* MHz TXOP.

The required protection method and frames is determined as described in section *2.2.4.2.5*, *Protection methods*. The HW next determines the required atomic frame exchange sequence as described in section *2.2.4.2.6*, *Determining frame exchange sequences and error handling*.

If the TXOPLimit is zero in EDCA, an atomic frame exchange sequences is attempted on air, subject to the next scheduled Quiet interval as described in section *2.2.4.4*, *Keeping track of Quiet intervals*.

If the TXOPLimit != 0, a time calculation is performed to decide if the atomic frame exchange sequence will fit in the remaining TXOP, subject to the next scheduled Quiet interval as described in section *2.2.4.4*, *Keeping track of Quiet intervals*. If the time on air for the atomic frame exchange sequence is less than (or equal to) the remaining TXOP, only then is the frame exchange sequence started.

When a frame exchange sequence starts, the HW determines which fields it should transmit for each frame formed from SW as described in section *2.2.4.2.7, MAC Header generation logic in HW* The fields that should be transmitted

may be updated by the HW before transmission as described in section *2.2.4.2.8*, *Updating MAC Header fields in HW when transmitting*.

After the transmission of a frame (singleton MPDU or A-MPDU), the status is updated as explained in the section *2.2.4.2.9*, *Transmit DMA channel status updation*.

After an atomic sequence is completed successfully, the HW performs these steps again.

A TXOP is terminated as described in section *2.2.4.2.10*, *Terminating a TXOP*.

2.2.4.2.1 Generating the early protocol trigger

In order to provide more time to the LMAC SW for performing its operations inside the JIT context, an interrupt to SW is raised some time before the protocol event starts on air. This interrupt is the "early" *protocol trigger*.

The MAC HW predicts which AC will win contention next and raises an early protocol trigger in the *Transmit / Receive Interrupt Event Register (txRxIntEventReg)*. The interrupt is raised for the highest AC which win contention.

2.2.4.2.2 Acquiring an EDCA TXOP

2.2.4.2.2.1 Single AC wins contention, i.e. internal collision doesn't occur

When the HW wins actual contention on the medium for a particular AC and internal collision doesn't occur, it determines whether it should attempt to acquire an EDCA TXOP. This is done as follows:

1. The HW checks the DMA channel state of the AC which won contention.
2. If the DMA channel is in the DEAD state, the HW does nothing.
3. If the DMA channel is in the PASSIVE state, it is moved to the ACTIVE state and the HW attempts to acquire an EDCA TXOP for that AC by transmitting frames from this DMA channel.
4. If the DMA channel is in the HALTED state, the procedure as described in section *2.2.4.2.2, Acquiring an EDCA TXOP* is followed again when the HW next wins contention[5].

The PHY is not put out of receive mode until the HW starts acquisition of an EDCA TXOP. Hence, if a receive operation starts in the meanwhile, the MAC is ready to accept the frame(s) and respond with relevant acknowledgement frames as required.

2.2.4.2.2.2 Multiple ACs win contention, i.e. internal collision occurs

When the HW wins actual contention on the medium for multiple ACs, i.e. internal collision occurs, it determines whether it should attempt to acquire an EDCA TXOP. In this case, the HW tries to acquire an EDCA TXOP for the highest priority AC which won contention and which has data to transmit (DMA channel is PASSIVE state) or for which SW has programmed control frame transmission. This is done as follows:

1. The HW checks the DMA channel state of the highest priority AC which won contention.
2. If the DMA channel is in the DEAD or HALTED state, the HW checks the DMA channel state of the next lower priority AC which won contention.

[5] Note that the backoff counter will free at 0 in this condition, and this AC will win contention again at every subsequent slot boundary. Subsequently, if there is data to be transmitted from SW, the HW will attempt to acquire a TXOP. If the medium becomes busy before there is data to be transmitted from SW, the backoff procedure is reinvoked.

3. If a DMA channel is in the PASSIVE state, it is moved to the ACTIVE state and the HW attempts to acquire an EDCA TXOP for its corresponding AC by transmitting frames from this DMA channel. Note that the HW increments the Contention Window of the lower ACs which also won contention..
4. If no DMA channel is in the PASSIVE state, the procedure as described in section *2.2.4.2.2, Acquiring an EDCA TXOP* is followed again when the HW next wins contention.

The PHY is not put out of receive mode until the HW starts acquisition of an EDCA TXOP. Hence, if a receive operation starts in the meanwhile, the MAC is ready to accept the frame(s) and respond with relevant acknowledgement frames as required.

2.2.4.2.3 Determining 20/40/80/160 MHz TXOP

	default BW TXOPV	defaultBWTXOP	Does DMA channel have data	bwProtTx or bwTx	numTry*BW* Acquisition != 0?	HW behaviour
1.	0	X	0	X	X	Nothing to do
2.	0	X	1	0	X	Acquire 20 MHz TXOP
3.	0	X	1	1/2/3	0	Only acquire 40/80/160 MHz TXOP
4.	0	X	1	1/2/3	1	Acquire 40/80/160 MHz TXOP, but drop to a lower bandwidth if unsuccessful for some number of times
5.	1	0	X	X	X	Acquire 20 MHz TXOP
6.	1	1/2/3	X	X	0	Only acquire 40/80/160 MHz TXOP
7.	1	1/2/3	X	X	1	Acquire 40/80/160 MHz TXOP, but drop to a lower bandwidth if unsuccessful for some number of times

Table 3: 20/40/80/160 MHz transmission truth table

HW makes a decision to acquire a 20 MHz, 40 MHz, 80 MHz or 160MHz TXOP as described in the above truth table.

When the HW wins contention on the primary channel, it first checks the fields of the *Transmission Bandwidth Control Register (txBWCntrlReg)* register.

If the *defaultBWTXOPV* bit is not set, the HW uses the bandwidth information contained in the *Transmit DMA Header Descriptor* or the Start Transmission registers to make a decision.

If no data has been programmed by SW to the DMA linked list, no decision on 20/40/80/160 MHz TXOP needs to be taken (Row 1).

If data has been programmed by the SW to the DMA linked list, the HW uses the information contained in the *Transmit DMA Header Descriptor* to make a decision.

If the Transmit DMA Header Descriptor indicates that a 20 MHz TXOP should be acquired, then the HW has no further decision to take for the purposes of this clause (Row 5).

If the Transmit DMA Header Descriptor indicates that a 40/80/160 MHz TXOP should be acquired, then HW checks if the corresponding secondary channels have been free for PIFS duration. If the corresponding secondary channels have been free for PIFS duration, then a 40/80/160 MHz TXOP has been acquired. If the one of the secondary channel has not been free for PIFS duration, further HW behavior depends on the *numTryBWAcquisition* parameter (Row 6 or 7).

40/80/160 MHz TXOP decision tree and fast adaptation[6]: In rows 4, 6 and 7 when SW has indicated to HW to acquire a 40/80/160 MHz TXOP, the HW checks the value of *numTryBWAcquisition* field. The *numTryBWAcquision* value allows HW to perform a fast adaptation to the latest channel condition (CCASecondary information) and it switches over to the highest bandwidth available using the same MCS and other PHY parameters for the PPDU.

If the *numTryBWAcquisition* field is set to 0, then the HW restarts the contention process by loading a new random number. This setting indicates that the 40/80/160 MHz fast adaptation feature is disabled i.e. the HW will keep on trying to acquire a 40/80/160 MHz TXOP until it succeeds.

If the *numTryBWAcquisition* field has a non-zero value, and the number of previous attempts to acquire a 40/80/160 MHz TXOP for this AC is less than *numTryBWAcquisition*, then the HW restarts the contention process by loading a new random number.

If the *numTryBWAcquisition* field has a non-zero value, and the number of previous attempts to acquire a 40/80/160 MHz TXOP for this AC is equal to *numTryBWAcquisition* – 1, then the HW switches over to a lower bandwidth MHz TXOP. 40/80/160 MHz PPDUs are transmitted at a lower bandwidth using the same MCS and other PHY parameters, if the frame exchange sequence fits into the remaining TXOP at the lower rate and the PPDU at the lower rate does not exceed *aPPDUMaxTime* value. Once this decision has been taken at the start of the TXOP, all 40/80/160MHz PPDUs are transmitted using the smaller bandwidth between the requested and the selected one for the duration of the TXOP. The number of attempts made to acquire a 40/80/160MHz TXOP are reported to SW in the *MIB Table*. The BW used to transmit a particular frame is reported to SW in the *Status Information* field of *Transmit DMA Header Descriptor*. If HW finds that a 40/80/160MHz PPDU has been scheduled for transmission in a TXOP with a lower bandwidth, the HW behavior is dependant on the *dropToLowerBW* setting. When set, the HW will transmit the 40/80/160MHz PPDU in lower bandwidth if the frame exchange sequence at the lower rate will fit into the remaining TXOP and the PPDU at the lower rate does not exceed *aPPDUMaxTime* value. When reset, the HW will release the TXOP without attempting to transmit the frame.

20 MHz TXOP decision tree: In rows 2 and 5, SW indicates to HW to acquire a 20 MHz TXOP. In this case, SW should try to ensure the PPDUs of the TXOP are for 20 MHz. If HW finds that a 40/80/160MHz PPDU has been scheduled for transmission in a 20 MHz TXOP, the HW behaviour is dependant on *dropToLowerBW* setting. When set, the HW will transmit the 40/80/160MHz PPDU in 20 MHz if the frame exchange sequence at the lower rate will fit into the remaining TXOP and the PPDU at the lower rate does not exceed *aPPDUMaxTime* value. When reset, the HW will release the TXOP without attempting to transmit the frame.

[6] Not to be confused with FLA

2.2.4.2.4 Frame Length adaptation

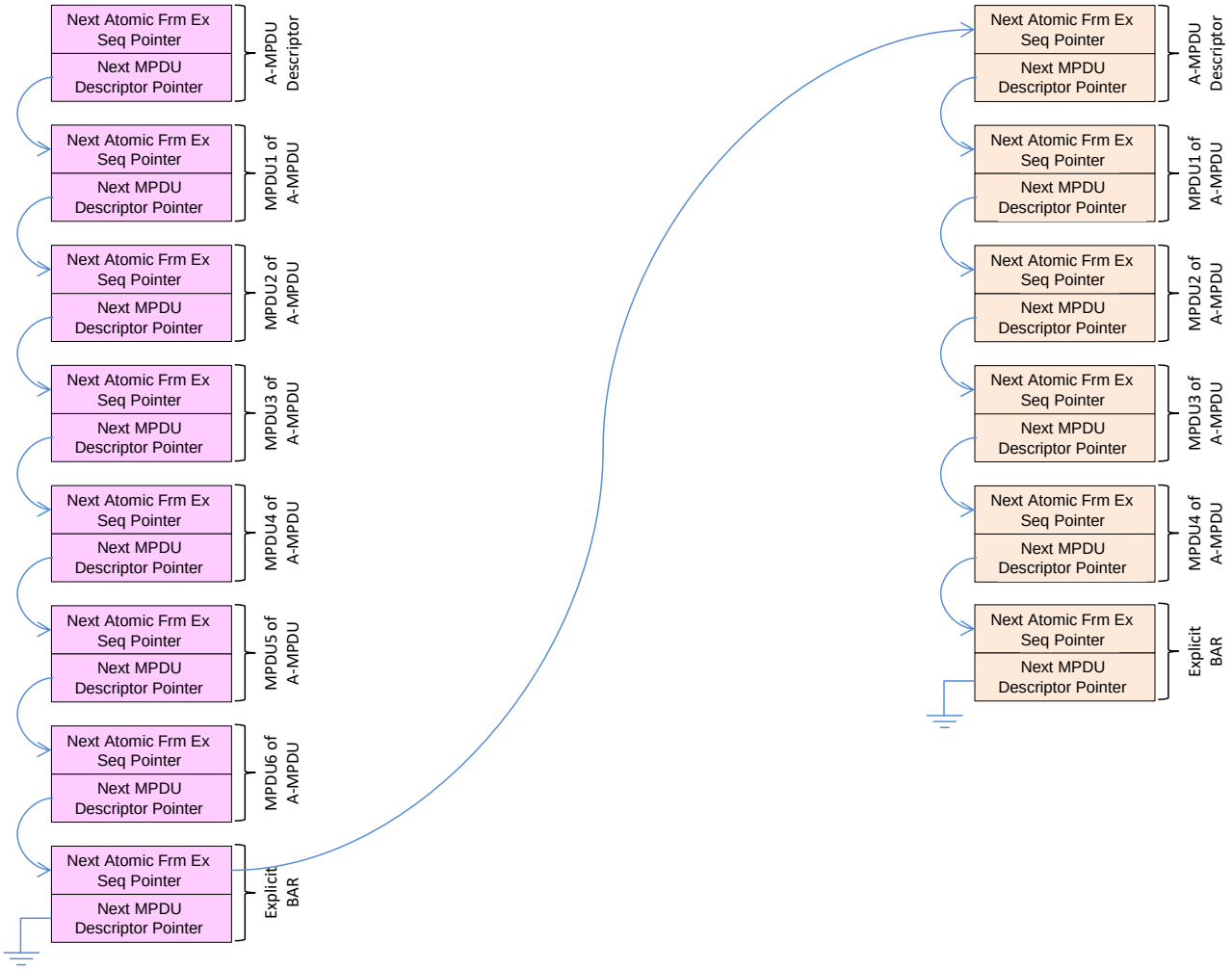
As described in 2.2.4.2.3, the MACCore may change the transmission bandwidth requested by the SW. This change is done is two cases, either one of the secondary channel is busy and the BW need to be reduced or in case of Bandwidth Signaling in dynamic mode. Upon reception of the CTS, the requested bandwidth can be reduced.

The bandwidth drop impacts the frame duration which may be longer than 5.3ms and the LSIG length might not be correct. To avoid that, the SW provides the frame lengths for each lower bandwidth in the *Optional Frame Length* fields of *Transmit DMA Header Descriptor.* When the bandwidth drop is decided, the HW uses the optional frame length corresponding to the decided bandwidth. If case of bandwidth drop, the HW checks if the optional frame length is equal or not with the Frame Length, it generates a *acXBWDropTrigger* interrupt in *Transmit / Receive Interrupt Event Register (txRxIntEventReg)* indicating to the SW that the transmission bandwidth has been reduced and the AMPDU needs to be re-arranged. The *acXBWDropTrigger* interrupt is not generated if the optional Frame Length is equal to the Frame Length, even if the bandwidth has been dropped.

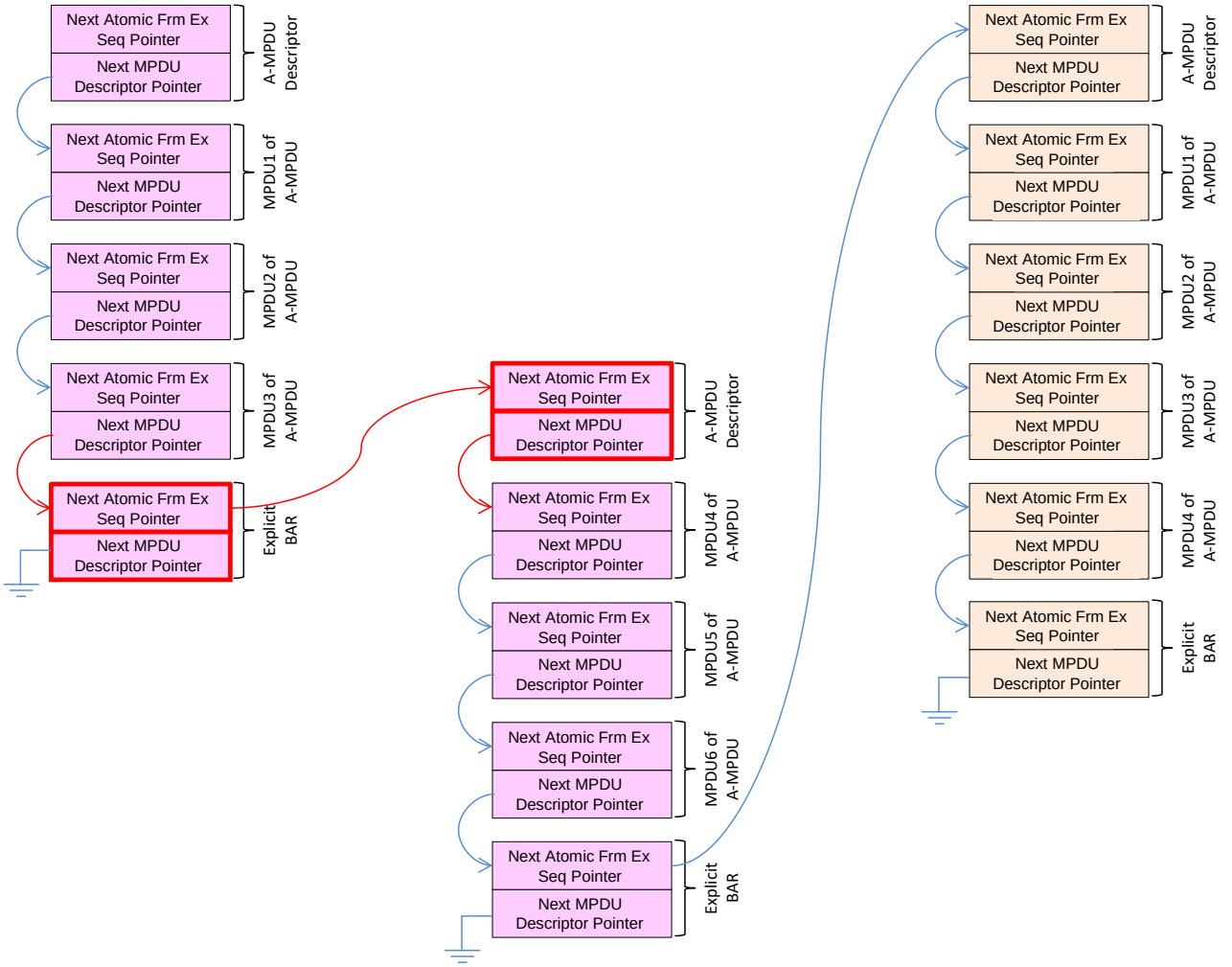
In case of *acXBWDropTrigger* interrupt, the software shall split the programmed A-MPDU in two knowing that the length of the programmed A-MPDU has been reduced. To do so, the SW gets from the *txBWDropInfoReg* register the new transmission bandwidth and then it splits the A-MPDU as follow:

- Create a new BAR descriptor,
- Modifie the *whichDescriptor* of the new last MDPU to 2'b11
- Clear the Next MPDU Descriptor Pointer
- Link the new BAR THD to the new last MDPU
- Creates a new A-MPDU THD based on the remaining MPDU (which have not going to be transmited as the A-MPDU has been split.) and link this new A-MPDU THD to the inserted BAR.

The following figures illustrate this mechanism. On the first figure, the SW has prepared and chained two A-MPDUs.



When starting the transmission of the first A-MPDU, a bandwidth drop occurs. Thus, the HW uses one of the optional frame lengths and generates a Protocol trigger interrupt. On this interrupt, the SW splits the A-MPDU in two as shown on the following figure. It first creates two new THD (highlighted in red), one for the BAR and one for the new A-MPDU containing the remaining MPDUs. Then it inserts these two new THD into the chain list (new connection highlighted in red).



2.2.4.2.5 Protection methods

The kind of protection frame required is selected and triggered from SW and this information can be passed to the HW in three ways:

- ✓ Indicated in the *Transmit DMA Header Descriptor fields*, protection frame formed by HW
- ✓ Chained in the linked list, protection frame formed by SW

SW can indicate to HW to perform the following types of protection in the *navProtFrmEx* field of the *Transmit DMA Header Descriptor fields*:

- ✓ 3'b000: No protection
- ✓ 3'b001: Self-CTS i.e. a CTS frame directed to this device
- ✓ 3'b010: RTS/CTS with intended receiver of the frame, i.e. a RTS frame directed to the MAC address to which the PPDU is directed.
- ✓ 3'b011: RTS/CTS with QAP. This feature is used for better protection when using DLS.
- ✓ 3'b100: As a non-AP STA - RTS/Dual-CTS directed to the AP. This feature is used when Dual-CTS protection needs to be turned on.
- ✓ 3'b100: As an AP: CTS frame directed to the MAC address to which the PPDU is directed. This feature is used when Dual-CTS protection needs to be turned on.

Conversely, the SW may chain an RTS or CTS frame to achieve NAV protection without using the HW mechanism. In that case, the *navProtFrmEx* field must not be set. It is recommended that SW chain a protection frame only if the HW formed mechanism does not suffice, for instance to solicit an MFB by attaching +HTC = MRQ to an RTS frame. The *frameLifetime* of the SW formed protection frame must be the same as the *frameLifetime* field of the frame that it belongs to.

SW should ensure that SW and HW formed protection mechanisms are not mixed together. It should not chain an RTS frame in the linked list before a PPDU that is enabled for HW formed RTS. Similarly, an RTS frame chained from SW should not itself be enabled for HW formed RTS protection.

If HW finds that a protection frame has been chained by SW or the *navProtFrmEx* field indicates that a protection frame needs to be transmitted, but NAV coverage already exists, then the protection frame is suppressed by HW. Note that this ensures that a protection frame is transmitted only at the start of a frame exchange sequence after winning contention.

Bandwidth Signaling

The core supports the bandwidth signaling introduced in the 802.11ac specification. The SW, depending on the type of device addressed by the transmitted frame, can enable the bandwidth signaling feature by setting the bit *useBWSignalingTx* in the *Transmit DMA Header Descriptor*. The core supports the static bandwidth signaling as well as the dynamic bandwidth signaling as it is able to adjust the transmitted BW and re-compute all the durations within SIFS. The selection between these two modes is done by the SW by setting the bit *dynBWTx* in the *Transmit DMA Header Descriptor.* As the core supports both modes, the SW enables the dynamic badwidth signaling by default as soon as the addressed device is VHT which provides a better bandwidth usage and optimizes the throughput.

The bandwidth signaling informations are automatically injected in the transmited RTS. The core changes the source address to set the bandwidth signaling bit and update the service field of the txVector with the bandwidth information. Then, it checks if the received CTS contains the bandwidth signaling information and uses the channel bandwith (*chBWInNonHT* from *rxVector1*) extracted by the PHY from the scrambler initialization value. Based on the returned bandwidth, the core re-computes the frame duration as well as the Legacy length of the PPDU before being transmitted. In this case, the bandwidth of the acquired TXOP will be the bandwidth received from the peer device and all the transmission fone inside this TXOP will be using either this TXOP bandwidth or the requested one whichever is lower.

Note that this procedure is only enabled if the selected protection is RTS/CTS. For all the other value of *navProtFrmEx*, the bit *useBWSignalingTx* is not considered by the core.

From SW point of view, the *useBWSignalingTx* bit is set for all frames transmited to VHT devices; else it is forced to 0. Then, as the core is able to supports the dynamic bandwidth signaling, the SW should always set the bit *dynBWTx.*

2.2.4.2.6 Determining frame exchange sequences and error handling

2.2.4.2.6.1 Case 1: TXOP Limit = 0 with unfragmented MSDU or MMPDU, A-MSDU, PS-Poll and BAR

The atomic frame exchange sequences are controlled from SW and are as follows:

- ✓ {RTS/CTS or RTS/Dual-CTS or CTS} – unfragmented MSDU or MMPDU {- ACK}
- ✓ {RTS/CTS or RTS/Dual-CTS or CTS} – A-MSDU {- ACK}
- ✓ {RTS/CTS or RTS/Dual-CTS or CTS} – BAR {- ACK or BA}
- ✓ {RTS/CTS or RTS/Dual-CTS or CTS} – PS-Poll {- ACK or Data}

Each atomic frame exchange sequence forms a single notional “column” in the DMA descriptor linked list structure. Refer section *2.2.3.1, Creating descriptor trees from SW*.

The unfragmented MSDU or MMPDU, A-MSDU, PS-Poll and BAR is referred to as MPDU.

If a protection frame has been chained by the SW, the HW checks the *frameLifetime* field of the protection frame; else it checks the *frameLifetime* field of the MPDU. If the *frameLifetime* has expired, the HW uses the *Next Atomic Frame Exchange Sequence Pointer* to find the start of the next "column", thus discarding the entire "column" consisting of the above frame exchange sequence.

If there is no protection frame chained from SW, HW checks the *navProtFrmEx* field in the DMA Descriptor of the MPDU to determine if the HW needs to create a protection frame.

The HW first transmits the required protection frame (if any). If the protection frame is unsuccessfully transmitted (e.g. an RTS did not receive a successful CTS frame in response), the medium is released, and the process starts again at the next contention win with the same "column".

If the protection frame is successfully transmitted, the HW transmits the MPDU.

If the MPDU is unsuccessful, the medium is released, and the process starts again at the next contention win with the same "column".

This process continues until the chained protection frame reaches the *shortRetryLimit* or the *frameLifetime*, or the MPDU reaches the *shortRetryLimit* / *longRetryLimit* or the *frameLifetime*. In that case, the HW uses the *Next Atomic Frame Exchange Sequence Pointer* to find the start of the next "column", thus discarding the entire "column" consisting of the above frame exchange sequence.

If the MPDU is successful, the medium is released, and the process starts again at the next contention win with the next "column".

The SW behavior to control HW is as follows:

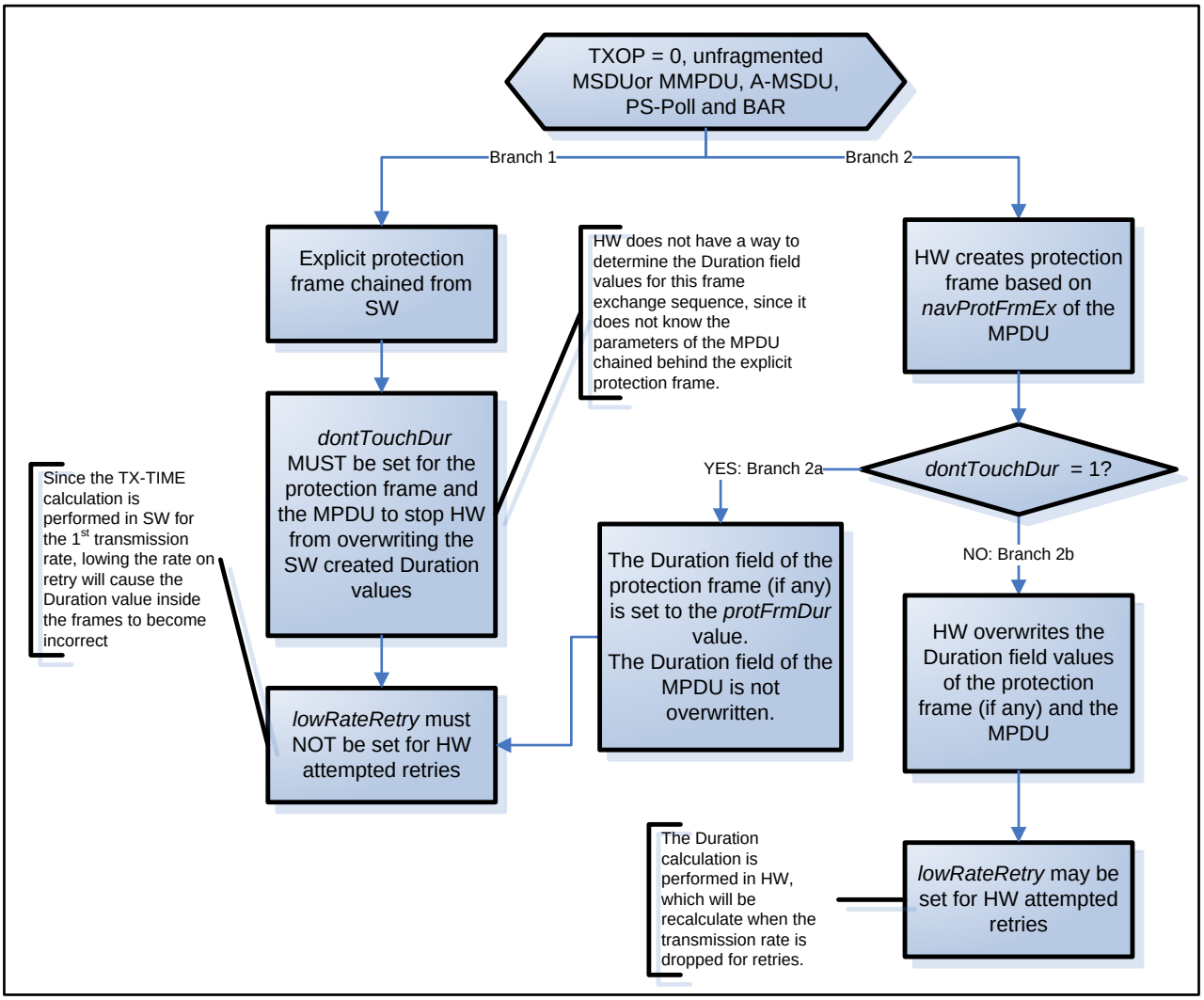


Figure 8: Case 1 - TXOP Limit = 0 with unfragmented MSDU, MMPDU, A-MSDU or BAR

In Branch 1, the frame exchange sequence is controlled by SW and frames are created by SW. In this case, a protection frame is created and chained from SW. HW does not use the *protFrmDur* field. Since the Duration field for all the frames of the frame exchange is prepared in SW, HW cannot be allowed to reduce the transmission rate when performing frame retries.

In Branch 2, the frame exchange sequence is controlled by SW and frames are created by HW and SW. In this case, the *navProtFrmEx* field is set to allow HW to create the protection frame.

In Branch 2a, the Duration field inside the MPDU is created by SW. HW updates the Duration field inside the protection frame to the *protFrmDur* value. Since the Duration field for all the frames of the frame exchange is prepared in SW, HW cannot be allowed to reduce the transmission rate when performing frame retries.

In Branch 2b, the Duration field inside the protection frame and the MPDU is created by HW. HW does not use the *protFrmDur* field. Since the Duration field for all the frames of the frame exchange is prepared in HW, HW can be allowed to reduce the transmission rate when performing frame retries.

The Duration field inside the protection frame and the MPDU, whether created by SW or HW, is the estimated time required for the transmission of the remaining frame exchange sequence. For example, in the sequence RTS – CTS – unfragmented MSDU – ACK, the RTS carries duration equal to the time on air for transmitting the CTS, unfragmented MSDU and ACK frames plus 3 SIFS.

2.2.4.2.6.2 Case 2: TXOP Limit = 0 with A-MPDU

The atomic frame exchange sequence is controlled from SW and HW and is as follows:

✓ {RTS/CTS or RTS/Dual-CTS or CTS} – A-MPDU {- BA} – backoff {- RTS/CTS or RTS/Dual-CTS or CTS} – BAR {- BA}

It forms a single notional "column" in the DMA descriptor linked list structure. Refer section *2.2.3.1, Creating descriptor trees from SW*.

If a protection frame has been chained by the SW, the HW checks the *frameLifetime* field of that frame; else it checks the *frameLifetime* field of the A-MPDU. If the *frameLifetime* has expired, the HW uses the *Next Atomic Frame Exchange Sequence Pointer* to find the start of the next "column", thus discarding the entire "column" consisting of the above frame exchange sequence.

If there is no protection frame chained from SW, HW checks the *navProtFrmEx* field in the DMA Descriptor of the A-MPDU to determine if the HW needs to create a protection frame.

The HW first transmits the required protection frame (if any). If the protection frame is unsuccessfully transmitted (e.g. an RTS didn't receive a successful CTS frame in response), the medium is released, and the process starts again at the next contention win with the same "column".

This process continues until the chained protection frame reaches the *shortRetryLimit* or the *frameLifetime* or the A-MPDU reaches the *frameLifetime*. In that case, the HW uses the *Next Atomic Frame Exchange Sequence Pointer* to find the start of the next "column", thus discarding the entire "column" consisting of the above frame exchange sequence.

If the protection frame is successfully transmitted, the HW transmits the A-MPDU.

If the A-MPDU is not successfully transmitted (i.e. it didn't receive a successful Compressed BA frame in response) the MAC HW does not retry the A-MPDU. The process starts again at the next contention win with the chained explicit BAR frame behind the A-MPDU. Note that the explicit BAR frame must be chained using the *Next MPDU Descriptor Pointer* and not the *Next Atomic Frame Exchange Sequence Pointer* to distinguish the BAR frame attached to the A-MPDU from a separate HT-Delayed explicit BAR frame. Subsequent behavior is similar to the case of the BAR in section *2.2.4.2.6.1*, *Case 1: TXOP Limit = 0 with unfragmented MSDU or MMPDU, A-MSDU, PS-Poll and BAR*.

If the A-MPDU is successful, the medium is released, and the process starts again at the next contention win with the next "column", automatically suppressing the chained explicit BAR frame.

The SW behavior to control HW is similar to *Figure 8: Case 1 - TXOP Limit = 0 with unfragmented MSDU, MMPDU, A-MSDU or BAR*.

2.2.4.2.6.3 Case 3: TXOP Limit = 0 with fragmented MSDU or MMPDU

The atomic frame exchange sequence is controlled from SW and HW and is as follows:

✓ {RTS/CTS or RTS/Dual-CTS or CTS} – fragment {- ACK} - fragment {- ACK} {- fragment } {- ACK}

It forms a single notional "column" in the DMA descriptor linked list structure. Refer section *2.2.3.1, Creating descriptor trees from SW*.

If a protection frame has been chained by the SW, the HW checks the *frameLifetime* field of that frame; else it checks the *frameLifetime* field of the fragment. If the *frameLifetime* has expired, the HW uses the *Next Atomic Frame Exchange Sequence Pointer* to find the start of the next "column", thus discarding the entire "column" consisting of the above frame exchange sequence.

If there is no protection frame chained from SW, HW checks the *navProtFrmEx* field in the DMA Descriptor of the MPDU to determine if the HW needs to create a protection frame.

The HW first transmits the required protection frame (if any). If the protection frame is unsuccessfully transmitted (e.g. an RTS didn't receive a successful CTS frame in response), the medium is released, and the process starts again at the next contention win with the same "column".

If the protection frame is successfully transmitted, the HW transmits the pending fragments one after another. Note that the SW may set the *navProtFrmEx* field to enable protection for all fragments. The HW suppresses the protection frame for all fragments except for the first fragment that is transmitted after winning contention.

If a fragment is unsuccessful, the medium is released, and the process starts again at the next contention win with the same “column”. Note that the start of the “column” may have moved to an intermediate fragment if previous fragments were successfully transmitted.

This process continues until the chained protection frame from SW reaches the *shortRetryLimit* or the *frameLifetime*, or any fragment reaches the *shortRetryLimit* / *longRetryLimit* or the *frameLifetime*. In that case, the HW uses the *Next Atomic Frame Exchange Sequence Pointer* to find the start of the next “column”, thus discarding the entire “column” consisting of the above frame exchange sequence.

If the MSDU is successful, the medium is released, and the process starts again at the next contention win with the next “column”.

The SW behavior to control HW is as follows:

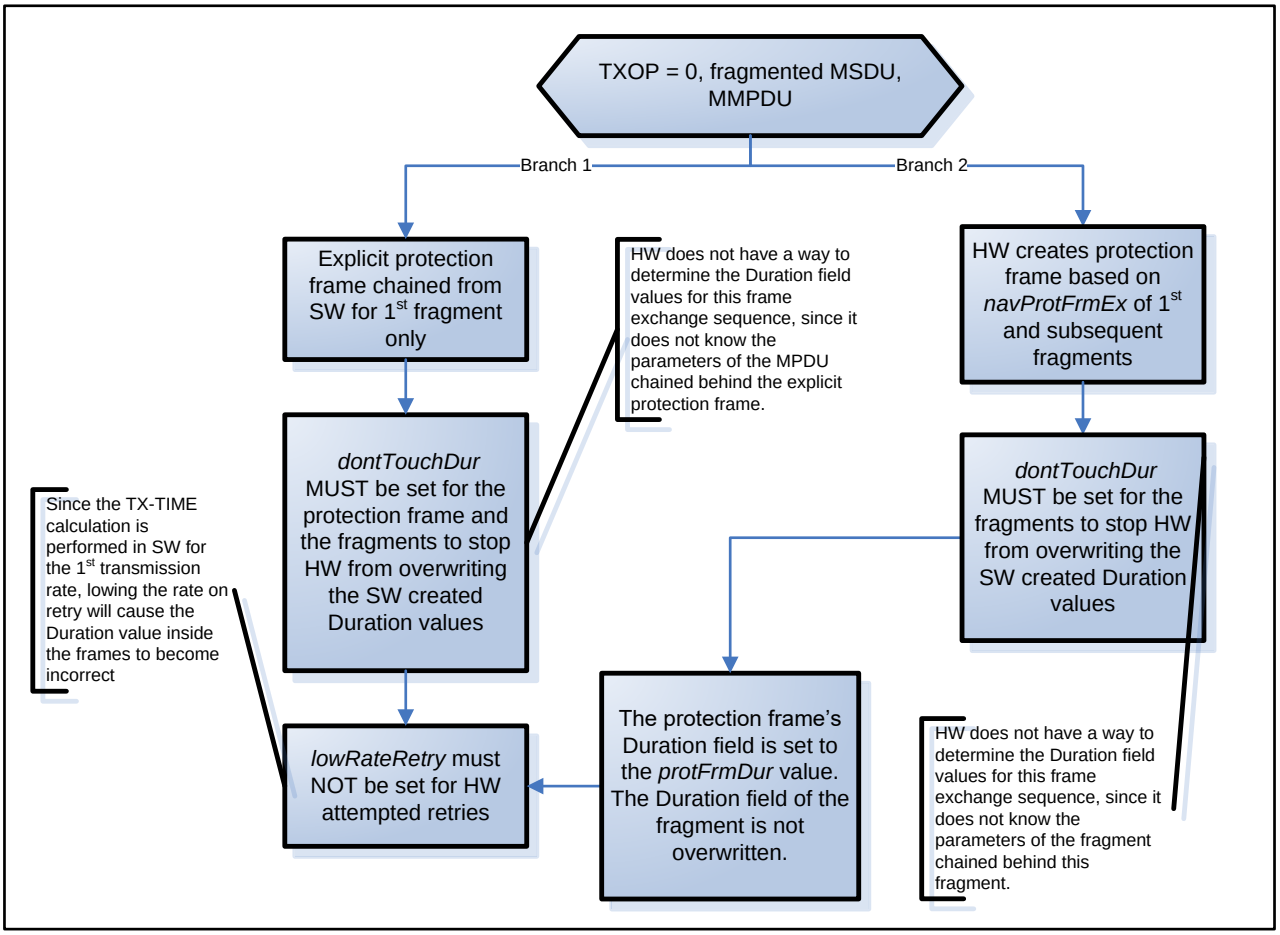


Figure 9: Case 3 - TXOP Limit = 0 with fragmented MSDU, MMPDU

In Branch 1, the frame exchange sequence is controlled by SW and frames are created by SW. In this case, a protection frame is chained from SW at the start of the fragmented sequence. The Duration field inside the protection frame and the fragments is created by SW. The *protFrmDur* field is not used by HW. Since the Duration field for all the frames of the frame exchange is prepared in SW, HW cannot be allowed to reduce the transmission rate when performing frame retries.

In Branch 2, the frame exchange sequence is controlled by SW and HW and frames are created by HW and SW. In this case, the *navProtFrmEx* field is set to allow HW to create the protection frame at the start of the fragmented sequence. The Duration field inside the fragements is created by SW. The Duration field inside the protection frame is updated by HW to the *protFrmDur* value. Since the Duration field for all the frames of the frame exchange is prepared in SW, HW cannot be allowed to reduce the transmission rate when performing frame retries.

2.2.4.2.6.4 Case 4: TXOP Limit != 0

The frame exchange sequence is controlled from SW and HW and is made up of one or more atomic frame exchange sequences described in the previous sections. The first frame exchange can be one of the following:

- ✓ {RTS/CTS or RTS/Dual-CTS or CTS} – unfragmented MSDU or MMPDU {- ACK}
- ✓ {RTS/CTS or RTS/Dual-CTS or CTS} – A-MSDU {- ACK}
- ✓ {RTS/CTS or RTS/Dual-CTS or CTS} – BAR {- ACK or BA}
- ✓ {RTS/CTS or RTS/Dual-CTS or CTS} – PS-Poll {- ACK or Data}
- ✓ {RTS/CTS or RTS/Dual-CTS or CTS} – A-MPDU {- BA}

- ✓ {RTS/CTS or RTS/Dual-CTS or CTS} – fragment {- ACK} - fragment {- ACK} {- fragment } {- ACK}

Followed by more atomic frame exchange sequences without the protection frame[7] as long as NAV coverage lasts:

- ✓ Unfragmented MSDU or MMPDU {- ACK}
- ✓ A-MSDU {- ACK}
- ✓ BAR {- ACK or BA}
- ✓ PS-Poll {- ACK or Data}
- ✓ A-MPDU {- BA}
- ✓ Fragment {- ACK} - fragment {- ACK} {- fragment } {- ACK}

Each atomic frame exchange sequence forms a single notional "column" in the DMA descriptor linked list structure. Refer section *2.2.3.1, Creating descriptor trees from SW*.

If a protection frame has been chained by the SW, the HW checks the *frameLifetime* field of that frame; else it checks the *frameLifetime* field of the first pending frame. Note that in this case, the frame can be an unfragmented MSDU or MMPDU, A-MSDU, A-MPDU or fragmented MSDU or MMPDU. If the *frameLifetime* has expired, the HW uses the *Next Atomic Frame Exchange Sequence Pointer* to find the start of the next "column", thus discarding the entire "column" consisting of a frame exchange sequence.

If there is no protection frame chained from SW, HW checks the *navProtFrmEx* field in the DMA Descriptor of the first pending frame to determine if the HW needs to create a protection frame.

The HW first transmits the required protection frame (if any). If the protection frame is unsuccessfully transmitted (e.g. an RTS didn't receive a successful CTS frame in response), the medium is released, and the process starts again at the next contention win with the same "column".

If the protection frame is successfully transmitted, the HW transmits the pending frames one after another until TXOP expires. Note that the SW may set the *navProtFrmEx* field to enable protection for all frames. The HW suppresses the protection frame for all frames except for the first frame that is transmitted after winning contention.

If a frame (other than an A-MPDU) is unsuccessful, it is retried by the HW if NAV coverage exists. If NAV coverage does not exist, the medium is released, and the process starts again at the next contention win with the same "column". Note that the start of the "column" may have moved to an intermediate fragment if this "column" consists of a fragmented MSDU or MMPDU and previous fragments were successfully transmitted.

This process continues until the chained protection frame from SW reaches the *shortRetryLimit* or the *frameLifetime*, or any frame reaches the *shortRetryLimit* / *longRetryLimit* or the *frameLifetime*. In that case, the HW uses the *Next Atomic Frame Exchange Sequence Pointer* to find the start of the next "column", thus discarding the entire "column" consisting of the above frame exchange sequence.

If an A-MPDU is not successfully transmitted (i.e. it didn't receive a successful Compressed BA frame in response) the MAC HW does not retry the A-MPDU. If NAV coverage exists, the explicitly chained BAR frame behind the A-MPDU is transmitted. Note that the explicit BAR frame must be chained using the *Next MPDU Descriptor Pointer* and not the *Next Atomic Frame Exchange Sequence Pointer* to distinguish the BAR frame attached to the A-MPDU from a separate explicit BAR frame.

If the frame is successful, the process continues with the next "column".

It is recommended that all frames transmitted using an AC for which TXOPLimit != 0 should have *navProtFrmEx* != 0. This ensures that every TXOP mandatorily starts with "Long NAV" protection, even when a sequence breaks and is restarted after backoff. The HW suppresses the protection frame for all frames except for the first frame which is transmitted after winning contention.

The SW behavior to control HW is as follows:

[7] If SW indicates that a protection frame should be transmitted, it is suppressed by HW.

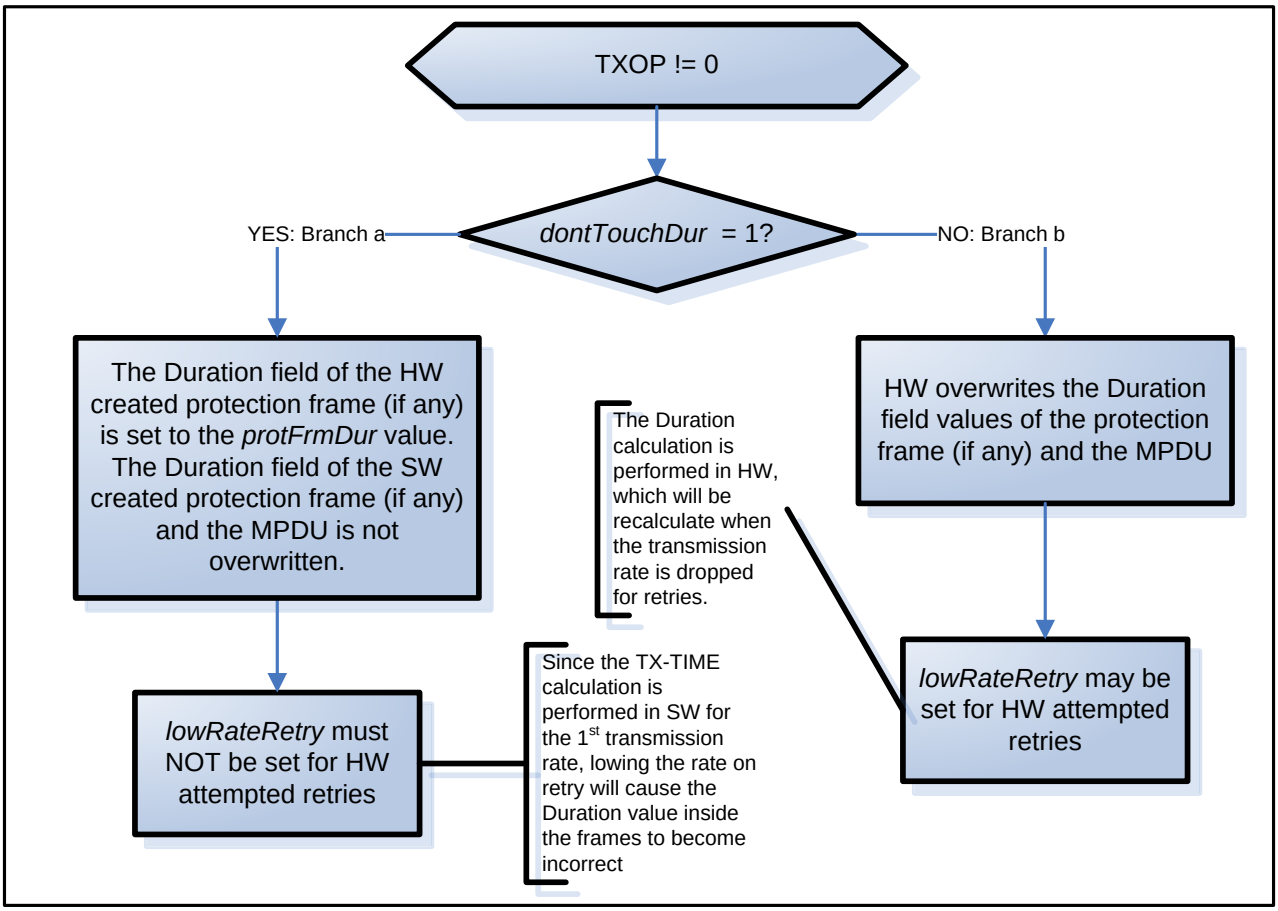


Figure 10: Case 4 – TXOP Limit != 0

In Branch A, the Duration fields are created by SW. If the *navProtFrmEx* field is set to allow HW to create the protection frame, then the Duration field inside the protection frame is updated by HW to the *protFrmDur* value. The Duration field of any protection frame or MPDU transmitted from SW is not touched by HW. Since the Duration field for all the frames of the frame exchange is prepared in SW, HW cannot be allowed to reduce the transmission rate when performing frame retries.

In Branch B, the Duration field inside any protection frame and MPDU is created by HW. The *protFrmDur* field is not used by HW. Since the Duration field for all the frames of the frame exchange is prepared in HW, HW can be allowed to reduce the transmission rate when performing frame retries.

2.2.4.2.7 MAC Header generation logic in HW

The MAC HW decides whether certain MAC Header fields from the *Transmit MPDU template* will be transmitted or not depending on other MAC Header fields as explained in the table below. The fields marked with "x" are not checked by HW when deciding to include a certain field, in other words they are always included. The HW does NOT perform error checking to ensure that a field is not included in a frame that should not carry it. For example, the *Protected Frame* bit should not be set for Control frames. However, if SW does set it, the HW *will* attempt to encrypt the control frame with unpredictable results. Hence it is SW's responsibility to set the various bits correctly.

Note that SW can bypass the MAC Header generation logic in HW by setting the *dontGenerateMH* field in the *MAC Control Information 2* field of the *Transmit DMA Header Descriptor*. This allows SW to transmit any MAC Header without any modification from HW.

This field is included	Based on these fields						
	Order	**From DS**	**To DS**	**Protected Frame**	**Ext IV**	**Type**	**Subtype**
Frame Control	x	x	x	x	x	x	x
Duration/ID	x	x	x	x	x	x	x
Address 1	x	x	x	x	x	x	x
Address 2	x	x	x	x	x	Control	RTS PS-Poll CF-End BAR BA
Address 2	x	x	x	x	x	Data	x
Address 2	x	x	x	x	x	Management	x
Address 3	x	x	x	x	x	Data	x
Address 3	x	x	x	x	x	Management	x
Sequence Control	x	x	x	x	x	Data	x
Sequence Control	x	x	x	x	x	Management	x
Address 4	x	1	1	x	x	x	x
QoS Control	x	x	x	x	x	Data	QoS
Carried Frame Control	x	x	x	x	x	Control	Control Wrapper
HTC	x	x	x	x	x	Control	Control Wrapper
HTC	1	x	x	x	x	x	x
IV	x	x	x	1	x	x	x
Extended IV	x	x	x	1	1	x	x
ICV	x	x	x	1	x	x	If the encryption algorithm is TKIP and the *dontEncrypt* bit is not set
CCMP MIC	x	x	x	1	1	x	If the encryption algorithm is CCMP and the *dontEncrypt* bit is not set

This field is included	Based on these fields						
	Order	**From DS**	**To DS**	**Protected Frame**	**Ext IV**	**Type**	**Subtype**
FCS	x	x	x	x	x	x	x

Table 4: MAC Header generation logic

2.2.4.2.8 Updating MAC Header fields in HW when transmitting

The MAC HW decides whether certain MAC Header fields of frames which have been prepared from SW will be updated or not depending on the *dontTouchFC*, *dontTouchDur*, *dontTouchQoS*, *dontTouchHTC*, *dontTouchFCS*, *dontTouchTSF*, *dontTouchDTIM* fields in the MPDU Header Descriptor. The setting of these fields is explained in the section *3.1.3*, *Transmit DMA Header Descriptor fields*. The HW operation is explained in the table below:

Field	Sub-field	Remarks
Frame Control	Protocol Version	The HW does not modify this field.
	Type	The HW does not modify this field.
	Subtype	The HW does not modify this field.
	To DS	The HW does not modify this field.
	From DS	The HW does not modify this field.
	More Frag	The HW does not modify this field.
	Retry	HW will set this bit if it retransmits a frame which has an *expectedACK* setting of 2'b01, and *dontTouchFC* for this frame is set to 0.
	Pwr Mgt	This bit is set by the HW to the value programmed in the *MAC Control 1 Register (macCntrl1Reg)*.*pwrMgt* field when the *dontTouchFC* for this frame is set to 0.
	More Data	The HW does not modify this field.
	Protected Frame	The HW does not modify this field.
	Order	The HW does not modify this field.
Duration		This field is updated by HW with the Long NAV duration value in EDCA when the *dontTouchDur* for this frame is set to 0.
QoS Control Field	TID	The HW does not modify this field.
	EOSP	?
	ACK Policy	The HW does not modify this field.
	A-MSDU present	The HW does not modify this field.
	TXOP Limit	The HW does not modify this field.
	Queue Size	?
	TXOP Duration requested	?
	QAP PS Buffer State	?

Field	Sub-field	Remarks
HT Control field		The HW does not modify this field.
VHT Control field		The HW does not modify this field.
HE Control field		The HW does not modify this field.
TSF		This field is updated by the HW with the latest (current) TSF value when the *dontTouchTSF* for this frame is set to 0. (Value for Beacon, Probe Response and Timing Advertisement frames).
DTIM Count		This field is updated by the HW with the latest (current) DTIM Count value when the *dontTouchDTIM* for this frame is set to 0. (Value for Beacon and Probe Response frames).
FCS		This field is updated by the HW with the CRC-32 calculated over the MPDU when the *dontTouchFCS* for this frame is set to 0.

Table 5: MAC Header fields which are updated by MAC HW

2.2.4.2.9 Transmit DMA channel status updation

If the frame is a singleton MPDU, then the status of the transmission is updated in the *Status Information* field of the Header Descriptor for the MPDU. If the frame is an A-MPDU, then the status of the transmission is updated in the *Status Information* field of the A-MPDU Header Descriptor for the entire A-MPDU. The individual Header Descriptors for each constituent MPDU of the A-MPDU is not updated.

When HW sets the *descriptorDoneHWTx* bit in the Header Descriptor of any frame, it checks the *interruptEnTx* bit. If the *interruptEnTx* bit is set, the HW raises a *transmission trigger* interrupt to SW.

If the *interruptEnTx* bit is set for a constituent MPDU of an A-MPDU, the HW raises a *transmission trigger* interrupt to SW.

Before an MPDU or A-MPDU transmission occurs, the only condition that can occur is LT Expired.

After an MPDU or A-MPDU transmission is complete, five conditions can occur: RTS Success, MPDU Success, A-MPDU success, Partial failure and RT Limit Reached.

The table summarizes the various events that can occur in HW and the status fields that are updated in each case.

Event	Defined as	Status fields updated
RTS formed by HW is successful	RTS Success	The *numRTSRetries* field of the Header Descriptor of the RTS is updated.
RTS formed by SW is successful	RTS Success	The *numRTSRetries* field of the Header Descriptor of the RTS is updated. The *frmSuccessfulTx* bit is set. The *descriptorDoneHWTx* bit is NOT set in this case.
MPDU is successful	Normal Success	The *numMPDURetries* field of the Header Descriptor of the MPDU is updated. The *frmSuccessfulTx* bit is set. The *descriptorDoneHWTx* bit is set.
A-MPDU is successful	Normal Success	The *frmSuccessfulTx* bit is set. The *descriptorDoneHWTx* bit is set.
A-MPDU is unsuccessful	A-MPDU Failure	The *descriptorDoneHWTx* bit is set.

RTS formed by HW is unsuccessful	Partial Failure	The *numRTSRetries* field of the Header Descriptor of the MPDU or A-MPDU is updated.
RTS formed by SW is unsuccessful	Partial Failure	The *numRTSRetries* field of the Header Descriptor of the RTS is updated.
MPDU is unsuccessful	Partial Failure	The *numMPDURetries* field of the Header Descriptor of the MPDU is updated.
RTS formed by HW fails	RT Limit Reached	The *numRTSRetries* field of the Header Descriptor of the MPDU or A-MPDU is updated. The *retryLimitReached* bit is set. The *descriptorDoneHWTx* bit in set.
RTS formed by SW fails	RT Limit Reached	The *numRTSRetries* field of the Header Descriptor of the RTS is updated. The *retryLimitReached* bit is set. The *descriptorDoneHWTx* bit is set.
MPDU fails	RT Limit Reached	The *numMPDURetries* field of the Header Descriptor of the MPDU is updated. The *retryLimitReached* bit is set. The *descriptorDoneHWTx* bit in set.
RTS formed by SW expires	LT Expired	The *lifetimeExpired* bit of the Header Descriptor of the RTS is set. The *descriptorDoneHWTx* bit is set.
MPDU or A-MPDU expires	LT Expired	The *lifetimeExpired* bit of the Header Descriptor of the MPDU is set. The *descriptorDoneHWTx* bit is set.

Table 6: Protocol events and DMA status

2.2.4.2.9.1 RTS Success

A transmission RTS Success results when an RTS frame receives the CTS frame successfully after 1 or more transmission attempts on air. The status of the transmission is updated in the *Status Information* field of the Header Descriptor of the frame. The *numRTSRetries* field is updated with the cumulative retries.

If the RTS has been formed in SW, then the *frmSuccessfulTx* bit is set. The *descriptorDoneHWTx* bit is NOT set in this case. This is to allow for a subsequent retransmission of the RTS frame if required.

2.2.4.2.9.2 Normal Success

A transmission Normal Success results when:

1. a unicast unaggregated data or management MPDU which expects an immediate ACK receives the ACK frame successfully after 1 or more transmission attempts on air.
2. a PS-Poll frame which expects an immediate ACK or Data frame receives the response successfully after 1 or more transmission attempts on air.
3. a unicast A-MPDU which expects an immediate Compressed Block ACK receives the BA frame successfully after 1 transmission attempt on air.
4. a BAR frame which expects an immediate Block ACK receives the BA frame successfully after 1 or more transmission attempts on air.
5. a BAR frame which expects an immediate ACK receives the ACK frame successful after 1 or more transmission attempts on air.
6. a unicast unaggregated MPDU which doesn't expect an immediate response is transmitted without any error.
7. a group addressed MPDU is transmitted without any error.

If the frame is an MPDU, then the status of the transmission is updated in the *Status Information* field of the Header Descriptor of the MPDU. The *numMPDURetries* field is updated.

If the frame is an A-MPDU, then the status of the transmission is updated in the *Status Information* field of the A-MPDU Header Descriptor for the entire A-MPDU. The individual Header Descriptors for each MPDU within the A-MPDU are updated with the *descriptorDoneHWTx* bit set. Since the A-MPDU Header Descriptor indicates that the BA was successfully received, the SW examines the received BA to determine whether the individual MPDUs of the A-MPDU were successfully delivered. In this case, the Explicit BAR frame chained behind the A-MPDU must not be checked by SW.

The *frmSuccessfulTx* bit and the *descriptorDoneHWTx* bit is set for the MPDU, MMPDU and the A-MPDU.

2.2.4.2.9.3 A-MPDU Failure

A transmission A-MPDU Failure results when a unicast A-MPDU which expects an immediate Compressed Block ACK does not receive the BA frame successfully after 1 transmission attempt on air.

The status of the transmission is updated in the *Status Information* field of the Header Descriptor of the A-MPDU Header Descriptor for the entire A-MPDU. The individual Header Descriptors for each MPDU within the A-MPDU are updated with the *descriptorDoneHWTx* bit set. Since the A-MPDU Header Descriptor indicates that the BA was not successfully received, the SW examines the status of the BAR frame chained behind this A-MPDU. It may so happen that the explicit BAR frame chained by SW behind the A-MPDU was successfully delivered.

The *descriptorDoneHWTx* bit in set in this case.

2.2.4.2.9.4 Partial Failure

A transmission Partial Failure results when:

1. a RTS frame transmitted to the same STA to which the unicast MPDU, MMPDU or A-MPDU is directed does not receive the CTS successfully, but the MSDU Short Retry Limit is not reached.
2. a unicast data or management MPDU which expects an immediate ACK does not receive the ACK frame successfully, but the MSDU Long/Short Retry Limit is not reached.
3. a PS-Poll frame which expects an immediate ACK or Data frame does not receive the response successfully, but the MSDU Short Retry Limit is not reached.
4. a BAR frame which expects an immediate Block ACK does not receive the BA frame successfully, but the MSDU Short Retry Limit is not reached.
5. a BAR frame which expects an immediate ACK does not receive the ACK frame successfully, but the MSDU Short Retry Limit is not reached.

If the frame is an MPDU, then the status of the transmission is updated in the *Status Information* field of the Header Descriptor of the MPDU or MMPDU. The *numMPDURetries* or the *numRTSRetries* field is updated for the MPDU or MMPDU.

If the frame is an A-MPDU, then the status of the transmission is updated in the *Status Information* field of the A-MPDU Header Descriptor for the entire A-MPDU. The individual Header Descriptors for each MPDU within the A-MPDU are not updated. The *numRTSRetries* field is updated.

2.2.4.2.9.5 RT Limit Reached

A transmission RT Limit Reached results when:

1. a RTS frame transmitted to the same STA to which the unicast MPDU, MMPDU or A-MPDU is directed does not receive the CTS successfully, and the MSDU Short Retry Limit is reached.
2. a unicast data or management MPDU which expects an immediate ACK does not receive the ACK frame successfully, and the MSDU Long/Short Retry Limit is reached.

3. a PS-Poll frame which expects an immediate ACK or Data frame does not receive the response successfully, and the MSDU Short Retry Limit is reached.
4. a BAR frame which expects an immediate Block ACK does not receive the BA frame successfully, and the MSDU Short Retry Limit is reached.
5. a BAR frame which expects an immediate ACK does not receive the ACK frame successfully, and the MSDU Short Retry Limit is reached.

If the frame is an MPDU, then the status of the transmission is updated in the *Status Information* field of the Header Descriptor of the MPDU or MMPDU. The *numMPDURetries* or the *numRTSRetries* field is updated for the MPDU or MMPDU.

If the frame is an A-MPDU, then the status of the transmission is updated in the *Status Information* field of the A-MPDU Header Descriptor for the entire A-MPDU. The individual Header Descriptors for each MPDU within the A-MPDU are not updated. The *numRTSRetries* field is updated.

The *retryLimitReached* bit is set. The *descriptorDoneHWTx* bit in set. In all these cases, the MAC HW starts transmission at the next Header Descriptor pointed to by the *Next Atomic Frame Exchange Sequence Pointer*.

2.2.4.2.9.6 LT Expired

A transmission LifeTime Expired erro results when the *frameLifetime* field of an MPDU, MMPDu or A-MPDU indicates that the frame has expired.

If the frame is an MPDU, then the status of the transmission is updated in the *Status Information* field of the Header Descriptor for the MPDU.

If the frame is an A-MPDU, then the status of the transmission is updated in the *Status Information* field of the A-MPDU Header Descriptor for the entire A-MPDU. The individual Header Descriptors for each MPDU within the A-MPDU are not updated.

The *lifetimeExpired* bit is set. The *descriptorDoneHWTx* bit in set in this case. The MAC HW starts transmission at the next Header Descriptor pointed to by the *Next Atomic Frame Exchange Sequence Pointer*.

2.2.4.2.10 Terminating a TXOP

A TXOP can be terminated by the sending of a CF-End frame. This is controlled from SW in three ways:

- ✓ Case1 :Default HW behavior indicated in the *EDCA Control Register (edcaCntrlReg)*, CF-End formed by HW
- ✓ Case2 :Immediate transmission of a CF-End frame indicated in the *EDCA Control Register (edcaCntrlReg)*, CF-End formed by HW
- ✓ Case 3: Chained in the linked list, formed by SW

In these three cases, if the HW finds that SW has enabled the transmission of a CF-End frame, it checks if the CF-End frame procedure will fit in the remaining NAV that has been set at other device as follows:

- ✓ If this is a non-AP STA and the TXOP was obtained using Dual-CTS protection, then the HW checks if the NAV at other devices is greater than the time it takes to transmit a CF-End frame with the same parameters as was used for the initial RTS frame + SIFS + *STBC Control Register (stbcCntrlReg).cfEndSTBCDur*. If this condition evaluates to true, the CF-End frame is transmitted.
- ✓ If this is an AP and the TXOP was obtained using Dual-CTS protection, then the HW checks if the NAV at other devices is greater than the time programmed in *STBC Control Register (stbcCntrlReg).cfEndSTBCDur*. If this condition evaluates to true, the required CF-End frames are transmitted.
- ✓ If the TXOP was not obtained using Dual-CTS protection, then the HW checks if the NAV at other devices is greater than the time it takes to transmit the CF-End using the highest BSS Basic Rate. If this condition evaluates to true, the CF-End frame is transmitted.

Case1: If the HW runs out of frames to transmit, the HW checks the *EDCA Control Register (edcaCntrlReg).sendCFEnd* bit. If the *sendCFEnd* bit is set, HW checks if the CF-End frame can be transmitted as described above.

Case2: If the bit is not set, the HW waits for the SW to chain a CF-End frame through the DMA channel, or to set the *EDCA Control Register (edcaCntrlReg).sendCFEndNow* bit. If the SW sets the *sendCFEndNow* bit, HW checks if the CF-End frame can be transmitted as described above. The HW immediately resets the *sendCFEndNow* bit.

Case 3: SW can transmit a CF-End frame "normally" through the DMA by chaining it in the linked list. While transmitting from a particular queue, if the HW finds that the next frame that has been chained to the DMA linked list is a CF-End frame, it checks if the CF-End frame can be transmitted as described above. If it cannot, the HW releases the TXOP. If the *Next Atomic Frame Exchange Sequence Pointer* is valid, the HW uses it to find the start of the next "column", else it moves to HALTED.

2.2.4.3 Trigger Based transmission

The Core supports transmission of trigger based frames as defined in 802.11ax.

When receiving a trigger frame, the Core checks if this frame is directed to the device either using addr1 or AID (in case of Trigger frame).

After FCS checks, depending on the Trigger frame variant, either the triggered response is automatically generated or the SW is requested to generate the expected response. The *Table 7: Core behavior based on trigger frame variant* indicates how each Trigger frame variant is handled by the MAC Core.

Trigger frame variant	Core Behavior
Basic	Triggered frame is prepared by SW (see *2.2.4.3.2 Trigger Based transmission prepared by SW*)
Beamforming Report Poll (BRP)	Fully handled by the Core without SW interaction (see *2.2.4.3.1 Trigger Based transmission prepared by HW*)
MU-BAR	Fully handled by the Core without SW interaction (see *2.2.4.3.1 Trigger Based transmission prepared by HW*)
MU-RTS	Fully handled by the Core without SW interaction (see *2.2.4.3.1 Trigger Based transmission prepared by HW*)
Buffer Status Report Poll (BSRP)	Triggered frame is prepared by SW (see *2.2.4.3.2 Trigger Based transmission prepared by SW*)
GCR MU-BAR	Fully handled by the Core without SW interaction (see *2.2.4.3.1 Trigger Based transmission prepared by HW*)
Bandwidth Query Report Poll (BQRP)	Fully handled by the Core without SW interaction (see *2.2.4.3.1 Trigger Based transmission prepared by HW*)
NDP Feedback Report Poll (NFRP)	Not supported yet.
Reserved	

Table 7: Core behavior based on trigger frame variant

The Core extacts from the Common and User field of the trigger frame the parameters required to prepare the txVector of the HE_TB response. In case of TB frame prepared by SW, it provides some of the extracted Common and User Information fields to the SW through the registers *Receive HE Trigger Common Information Register (rxHETrigCommonInfoReg)* and *Receive HE Trigger User Information Register (rxHETrigUserInfoReg).* This information can be used by the SW to compute the maximum length allowed in the Trigger Based frame.

The generation of HE_TB upon reception of a valid Trigger frame addressed to our device can be disabled by the SW by setting the bit *disableTBResp* in *HE Configuration Register (HEConfigReg)*. Note that this field does not control whether a valid trigger frame is passed to SW. This is controlled by a set of fields (*acceptTriggerHWFrames*, *acceptTriggerSWFrames*, *acceptAllTriggerFrames*) available in *HE Configuration Register (HEConfigReg)*. The *acceptTriggerHWFrames* enables the transmission to the SW of all the valid trigger frames addressed to our device for which the response is fully handled by the HW. The *acceptTriggerSWFrames* enables the transmission to the SW of all the valid trigger frames addressed to our device for which the response is handled by the SW. The *acceptAllTriggerFrames* enables the transmission to the SW of all the Trigger frames whaterver theirs type and even if they are not addressed to the device.

2.2.4.3.1 Trigger Based transmission prepared by HW

Upon reception of a valid Beamforming Report Poll (BRP), MU-BAR, MU-RTS, GCR MU-BAR or Bandwidth Query Report Poll (BQRP) Trigger frame, the Core generates automatically the immediate response.

As for a classic immediate response, the Core computes the differents TxVector parameters to be provided to the PHY. It is able to generate the following Trigger Based Response: CTS, BA and Bandwidth Query report.

For the CTS, the immediate reponse is provided in NON-HT whereas for BA and Bandwitdh Query report, the immediate reponse is provided in HE_TB format.

All the TX vector parameters are contained into the Trigger frame expect in case of TRS, the legLength is computed by the Core using the HE TB PPDU Length field of the TRS Control field.

Another parameter which is computed by the Core is the TxPower.

- In case of Trigger frame :

  TX Power = -20 + **AP Tx Power** - 110 + **UL Target RSSI** – **Rx RSSI Legacy**

  With :

  - **AP Tx Power** subfield extracted from Trigger Frame
  - **UL Target RSSI** subfield extracted from Trigger Frame
  - **Rx RSSI Legacy** in dB from Trigger Frame RxVector (in 2's complement)
- In case of TRS Control field

  TX Power = -20 + 2***DL Tx Power** -90 + 2 * **UL Target RSSI** – **Rx RSSI Legacy**

  With :

  - **DL Tx Power** subfield extracted from TRS Control field
  - **UL Target RSSI** subfield extracted from TRS Control field
  - **Rx RSSI Legacy** in dB from Frame containing TRS Control Field RxVector (in 2's complement)
- In both case, the computed Tx Power will be between *ofdmMinPwrLevel* and *ofdmMaxPwrLevel* defined in *Maximum Power Level Register (maxPowerLevelReg)*.

For the CTS and BA, the Core behaves as for a standard CTS and BA transmission in reponses to a RTS or BAR.

In case of Bandwitdh Query Report Poll, the Core checks the sub channel availability and creates the Available Channel Bitmap. Then, the Core builds a QoSNull frame with HE variant HT Control field in the MAC Header containing the BQR Control Information subfield. Note that for a 20MHz STA, this bitmap is forced to 0.

2.2.4.3.2 Trigger Based transmission prepared by SW

Upon reception of a valid basic trigger or Buffer Status Report frame, the Core stores in *Receive Header Trigger Frame Pointer Register (rxHeaderTFPtrReg).rxHeaderTFPtrReg* the address of the Rx Header Descriptor of the Trigger frame

and marks it valid by setting the *rxHeaderTFPtrValid* bit. Then, an interrupt is generated to the SW (*Transmit / Receive Interrupt Event Register (txRxIntEventReg).tbProtTrigger*).

Upon reception of this interrupt, the SW builds the Trigger Based response frame which can be either a singleton or an A-MPDU, then links it to the dedicated DMA channel (*Transmit TB Head Pointer Register (txtbHeadPtrReg)*) and finally indicates it to the Core by setting *DMA Control Register (dmaCntrlReg).txTBNewHead*. As all other THD, a Policy Table shall be linked containing the MAC Control Information required for encryption. The PHY Control information as well as Rate and Power Control fields shall be forced to 0.

In this case, no EDCA is performed as this frame is an immediate reponse.

When programming the transmission, the SW shall consider the PHY parameters extracted from the Trigger Frame (available in the registers *Receive HE Trigger Common Information Register (rxHETrigCommonInfoReg)* and *Receive HE Trigger User Information Register (rxHETrigUserInfoReg)*) in order to compute the maximum data length. In case of Data length bigger than the maximum data length, the PHY may generate a phyErr.

The SW programs the HE_TB transmission as a standard A-MPDU knowing that all the PHY parameters provided in the Policy Table will not be used. In case of HE_TB, the TxVector is fully built by the MAC Core except for the HE_Length which comes from the A-MPDU THD. In case of Singleton MPDU transmission, the Core automatically inserts the AMPDU delimiter and padding to the MPDU during HE_TB transmission. This flow is equivalent to a standard VHT transmission.

To leave more time to the SW for building the HE_TB frame, the Core starts the transmission of TxVector at the SIFS boundary but inserts pause before the first byte of LENGTH field until a valid HT length from THD. If the THD is not fetch on time and the inserted pause becomes too long, the PHY will generate a *phyErr*. To avoid *phyErr* situation, the MAC HW provides to the SW the remaining time before the underrun situation. This remaining time, provided in *Transmit Trigger Based Information Register (txHETBInfoReg)*, is a downcounter which starts upon assertion of txReq from a defined value (field *txHETBMaxDur*). When it reaches zero, the underrun situation will occur. During the preparation of the Trigger Based A-MPDU, the SW can regularely check the remaining duration (*txHETBRemDur*) and finalizes the transmission before it reaches 0.

If the SW has nothing to transmit, it shall link a Null frame.

Upon reception of trigger frame with CS Required bit set in Common Info field, the HW checks the Carrier Sense before starting a HE TB transmission and may have to cancel the transmission. As the *Transmit / Receive Interrupt Event Register (txRxIntEventReg).tbProtTrigger* interrupt has already been generated when the decision to cancel the transmission is taken, the HW indicates it to the SW by asserting the *Transmit / Receive Interrupt Event Register (txRxIntEventReg).tbTxCancelled* interrupt.

For debug purpose, the Carrier Sense checks before Trigger Based transmission can be disabled by SW by setting *HE Configuration Register (HEConfigReg).disableTBCS*.

2.2.4.3.3 UORA

The MAC HW supports the Uplink OFDMA Random Access which allows associated and non-associated STAs to use unallocated RU using an OFDMA back-off procedure. As for a classical Trigger Based transmission, the decision to transmit is taken by the HW.

First of all, the SW shall indicate if Random Access traffic is pending for the AP and its type using *raRUEnable* and *raRUType* fields of *Transmit Trigger Based Information Register (txHETBInfoReg).*

When enabled, the Core searches in the Trigger Frame the User Info fields with AID = 0 (if *raRUType* = 0 : the device is associated) or for the AID = 2045 (if *raRUType* = 1 : the device is not associated). Note that this is done along with the user AID search and the User AID matches

If one of the AID matched the Device AID (indicated in *Beacon Control 2 Register (bcnCntrl2Reg)*), this RU is selected, the Random Access procedure stops and the normal Trigger Based transmission described in *2.2.4.3 Trigger Based transmission* is followed.

If not, the Core follows the OFDMA backoff procedure to select or not an RU in the eligible RUs. The Core is in charge of generating a Random Number and selecting an RA-RU based on current OCW value and the number of eligible RUs.

To ease the selection of eligible RUs, the SW shall indicate to the Core the supported PHY features: *maxMCSInHETB*, *dopplerSupport*, *dcmSupport* information. Furthermore, the Core needs also to know if the device is associated or not. The SW provides this information using *raRUType* field.

The generation of the random OCW is based on an LFSR which can be sed by writing in *Debug UORA Register (debugUORAReg).ocwLFSR*. This register allows also to set the next value of OCW and to read it for debug purpose.

If a RA-RU has been selected, the Core updates the registers *Receive HE Trigger Common Information Register (rxHETrigCommonInfoReg)* and *Receive HE Trigger User Information Register (rxHETrigUserInfoReg)* as for a normal HE-TB transmission, sets the field *Receive HE Trigger User Information Register (rxHETrigUserInfoReg).ulRUType* to 1 indicating a Random-Access RU.

Then, depending on the Trigger Frame type (Basic, BQRP or BSRP), the Core follows either the procedure described in *2.2.4.3.1 Trigger Based transmission prepared by HW* or in *2.2.4.3.2 Trigger Based transmission prepared by SW*.

If the trigger frame was handled by the SW, it needs also to manage the OCW (OFDMA Contention Window) based on the reception or not of the acknowledgement of the transmitted frame. If an MPDU part of the HE-TB transmission has been acknowledged, the SW shall reset the *eOCW* field of *Transmit Trigger Based Information Register (txHETBInfoReg)* to eOCWMin indicated by the AP. If not, the SW shall increment the *eOCW*. Note that *eOCW* shall remain in the range of eOCWmin to eOCWmax.

If the Trigger frame was fully generated by the HW (BQRP trigger frame), the Core resets the *eOCW* with the value provided in *eOCWMin*.

If the SW has nothing else to transmit, it shall reset the *raRUEnable* indicating to the Core that only the trigger frame directly addressed to the device (addr1 or AID matches) shall be handled.

2.2.4.4 Keeping track of Quiet intervals

The MAC HW as STA keeps track of one scheduled Quiet Intervals. Every received Beacon and Probe Response from a matching BSSID is parsed and the Quiet interval parameters extracted from them are written into the *Quiet Element 1a Register (quietElement1aReg)* and the *Quiet Element 1b Register (quietElement1bReg)*. HW keeps track of the next scheduled Quiet interval.

- ✓ For an AC with a *TXOPLimit* = 0, before starting transmission, the HW checks whether the next atomic frame exchange sequence will cross into the next scheduled Quiet interval. If so, the next atomic frame exchange sequence is not attempted. The current contention win on the medium is not used.
- ✓ For an AC with a *TXOPLimit* != 0, the relevant MOT register (e.g. *Medium Occupancy Timer 1 Register (mot1Reg)*) is loaded with a value of *TXOPLimit* for the AC capped by the start of the next scheduled Quiet interval.

2.2.5 Transmit Vector preparation

The Transmit Vector is prepared by the MAC HW for every transmission. Transmission of a frame can be triggered in the following ways:

1. Case 1: Data, Management and Control frames created by SW, chained in a linked list
2. Case 2: Control frames (RTS or CTS only) created by HW, triggered from SW using the *navProtFrmEx* bits in the DMA Header Descriptor of a Data, Management or Control frame chained in a linked list
3. Case 3: Control frames (CTS, ACK, BA, CF-End only) created autonomously from HW, or CF-End triggered from SW using the *EDCA Control Register (edcaCntrlReg)*
4. Case 4: CF-End transmitted autonomously by HW

2.2.5.1 Case 1: Data, management or control frame created by SW

In this scenario a Data, Management or Control frame is created by the SW and chained in a linked list. The Tx Vector parameters are taken from fields of the *Transmit DMA Header Descriptor fields*, the *Policy Table fields* and registers.

Tx Vector Parameter	Filled from
txPwrLevel	Policy Table: *txPwrLevelPTRCX*
chBW	Policy Table: *bwTxRCX*
smoothing	DMA Header Descriptor: *smoothingTx*
antennaSet	Policy Table: *antennaSetPT*
smmIndex	Policy Table: *smmIndexPT*
mcs	Policy Table: *mcsIndexTxRCX*
preType	Policy Table: *preTypeTxRCX*
formatMod	Policy Table: *formatModTxRCX*
numExtnSS	Policy Table: *numExtnSSPT*
stbc	Policy Table: *stbcPT*
fecCoding	Policy Table: *fecCodingPT*
sounding	DMA Header Descriptor: *soundingTx*
legLength	DMA Header Descriptor: *frameLengthTx* or prepared by HW for HT-MF
legRate	Policy Table: *mcsIndexTxRCX* or set to 6 Mbps for HT-MF
service	Baseband Service Register: *bbServiceA* or *bbServiceB*
htLength	DMA Header Descriptor: *frameLengthTx*
nTx	Policy Table: *nTxPT*
shortGI	Policy Table: *shortGIPT*
aggregation	DMA Header Descriptor: *aMPDU*

Table 8: Case 1 of TxVector preparation

2.2.5.2 Case 2: RTS & CTS transmission triggered from SW in DMA Header Descriptor

In this scenario SW triggers the HW to transmit a Control frame (RTS or CTS only) using the *navProtFrmEx* bits in the DMA Header Descriptor of a Data, Management or Control frame chained in a linked list. The Tx Vector parameters are taken from fields of the *Transmit DMA Header Descriptor fields*, the *Policy Table fields* and registers, as described below:

Tx Vector Parameter	Filled from
txPwrLevel	Policy Table: *txPwrLevelProtPTRCX*
chBW	Policy Table: *bwProtTxRCX*
smoothing	DMA Header Descriptor: *smoothingProtTx*
antennaSet	Policy Table: *antennaSetPT*
smmIndex	Policy Table: *smmIndexPT*

Tx Vector Parameter	Filled from
mcs	Policy Table: *mcsIndexProtTxRCX* and registers
preType	Policy Table: *preTypeProtTxRCX*
formatMod	Policy Table: *formatModProtTxRCX*
numExtnSS	Policy Table: *numExtnSSPT*
stbc	Policy Table: *stbcPT*
fecCoding	Policy Table: *fecCodingPT*
sounding	NA
legLength	Prepared by HW
legRate	Policy Table: *mcsIndexProtTxRCX* and registers or set to 6 Mbps for HT-MF
service	Baseband Service Register: *bbServiceA* or *bbServiceB*
htLength	Prepared by HW
nTx	Policy Table: *nTxPT*
shortGI	Policy Table: *shortGIPT*
aggregation	NA

Table 9: Case 2 of TxVector preparation

2.2.5.3 Case 3: Control response frame transmitted autonomously by HW

In this scenario, the HW autonomously transmits a control response frame in response to a valid received frame:

- ✓ CTS in response to RTS
- ✓ BA in response to Implicit or Explicit BAR
- ✓ ACK in response to MPDU

The Tx Vector parameters are prepared by HW based on the Rx Vector of the frame being responded to, and the settings in various HW registers, as described in *[2]* and *[5]*.

2.2.5.4 Case 4: CF-End transmitted autonomously by HW

In this scenario, the HW transmits one or more CF-End frames when programmed from SW in the *EDCA Control Register (edcaCntrlReg)*, or after receiving a CF-End frame.

If a TXOP was obtained using Dual-CTS, then the CF-End(s) to terminate the TXOP is transmitted using PHY parameters that were used for the "matching control" frame at the start of the TXOP. The "matching control frame" is defined in *[2]* and *[5]*.

If the TXOP was not obtained using Dual-CTS, then the Tx Vector parameters are prepared by HW based on the settings in various HW resgisters, as described below:

Tx Vector Parameter	Filled from
txPwrLevel	Baseband Service Register: *maxPwrLevel*
chBW	NON_HT_CBW20 or NON_HT_CBW40
smoothing	NA

Tx Vector Parameter	Filled from
antennaSet	NA
smmIndex	NA
mcs	NA
preType	Long Preamble or NA
formatMod	NON-HT or NON-HT-DUP-OFDM
numExtnSS	NA
stbc	NA
fecCoding	NA
sounding	NA
legLength	Prepared by HW
legRate	Rates Register: *bssBasicRateSet*
service	Baseband Service Register: *bbServiceA* or *bbServiceB*
htLength	NA
nTx	NA
shortGI	NA
aggregation	NA

Table 10: Case 5 of TxVector preparation

2.3 Receive operation

2.3.1 Introduction

The HW has one logical DMA channels on reception which move the received frame and the differents descriptor in Rx ring buffers. The chapter *2.3.2 Rx Ring Buffer Management* describes how the RX ring buffers are managed.

The RX DMA Channel is fed from a FIFO RAM in the MAC HW, called the Receive FIFO. The Receive FIFO acts as an intermediate stage of buffering when receiving a frame from the baseband at a constant rate and writing into the system memory using a bursty flow (if any) on the internal interconnect.

2.3.2 Rx Ring Buffer Management

2.3.2.1.1 Pointer management

The Rx Buffer management is done based on two sets of registers *rxBuf1/2WrPtr*, *rxBuf1/2RdPtr*, *rxBuf1/2StartPtr* and *rxBuf1/2EndPtr*.

In IDLE state, the software defines the location and the size of the Rx Buffers by setting the *rxBuf1StartPtr*, *rxBuf2StartPtr*, *rxBuf1EndPtr* and *rxBuf2EndPtr*. It also sets the HW Read & Write pointers *rxBuf1RdPtr*, *rxBuf2RdPtr,* *rxBuf1WrPtr* and *rxBuf2WrPtr* to *rxBuf1StartPtr* and *rxBuf2StartPtr* respectively.

The Software can ask the Core to insert blank words before and after each descriptor (RHD and RPD) using the *rxBufRHDHeader*, *rxBufRHDFooter*, *rxBufRPDHeader* and *rxBufRPDFooter*.

These pointers shall be quadlets aligned.

The two Rx buffers (1 & 2) are used to prioritize the reception based on the frame type/traffic type (see *2.3.3.2 Frame Priority rules*). The DMA Engine gets the priority along with the MPDU and stored it to the right Rx buffer.

Then, when the Rx DMA Engine is triggered due to a reception, it starts to write the RHD, RPD and the associated data at the location indicated in *rxBufXWrPtr*. Once all the related information of the received MPDU have been moved in the SRAM, the HW updates the value of *rxBufXWrPtr* with the address of the next quadlet. It also updates the pointers of the RHD (Next Header Descriptor Pointer, First Payload Buffer Descriptor Pointer, Data Start Pointer and Data End Pointer) and RPD (Next Payload Buffer Descriptor Pointer, Data Start Pointera and Data End Pointer). It also write the new value of the *rxBufXWrPtr* in Next Write Pointer field of RHD.

The Software manages the *rxBufXRdPtr* which indicates the current read pointer. It shall guaranty that the *rxBufXRdPtr* is kept within the *rxBufXStartPtr* and rxBufXEndPtr and does not cross the *rxBufXWrPtr* boundary.

The *rxBufXRdPtr* indicates also to the HW the start address of the data which have not been processed yet by the SW. As a consequence, the HW shall not write at the location indicated in *rxBufXRdPtr*. Before writing a RHD, the DMA Engine checks if the MPDU and its associated descriptors fit in the remaining part of the rxBuffer. If not, the rxFlow Control mechanism is activated.

The DMA Engine guaranties that the RHD and RPD (including their header and footer) are aligned on 32-bits and not splitted in the memory due to the Rx Buffer wrap. Before writing a RHD or a RPD in the rxBuffer, the DMA Engine check if *rxBufXEndPtr* – *rxBufXWrPtr* < Descriptor size. If a potential wrap of the descriptor is detected, the DMA Engine writes the full descriptor starting at the address *rxBufXStartPtr* and leaves the remaining space empty.

When the MAC Header is written in the Rx Buffer, the DMA engine guanranty that it is written in one block and does nto wrap (see *2.3.2.1.2 MAC Header location*).

Each time a RHD is inserted, the DMA Engine updates the Next Header Descriptor Pointer of the previous one to build the chained list. Note that when written, the new RHD has its field Next Header Descriptor Pointer written to 0.

During the payload storage in the Rx buffer, it may occur that the write pointer reaches *rxBufXEndPtr*. In this case, the DMA Engine inserts a new RPD at the address *rxBufXStartPtr* and moves forward with the payload. However, the Core offers the possibility to guaranty that the payload or a part of the payload will not wrap due to the end of the RX Buffer (see *2.3.2.1.3 Payload location*).

The *Figure 11: Different case of wrap* summarizes the different cases when the DMA reaches the end of the RX Buffer.

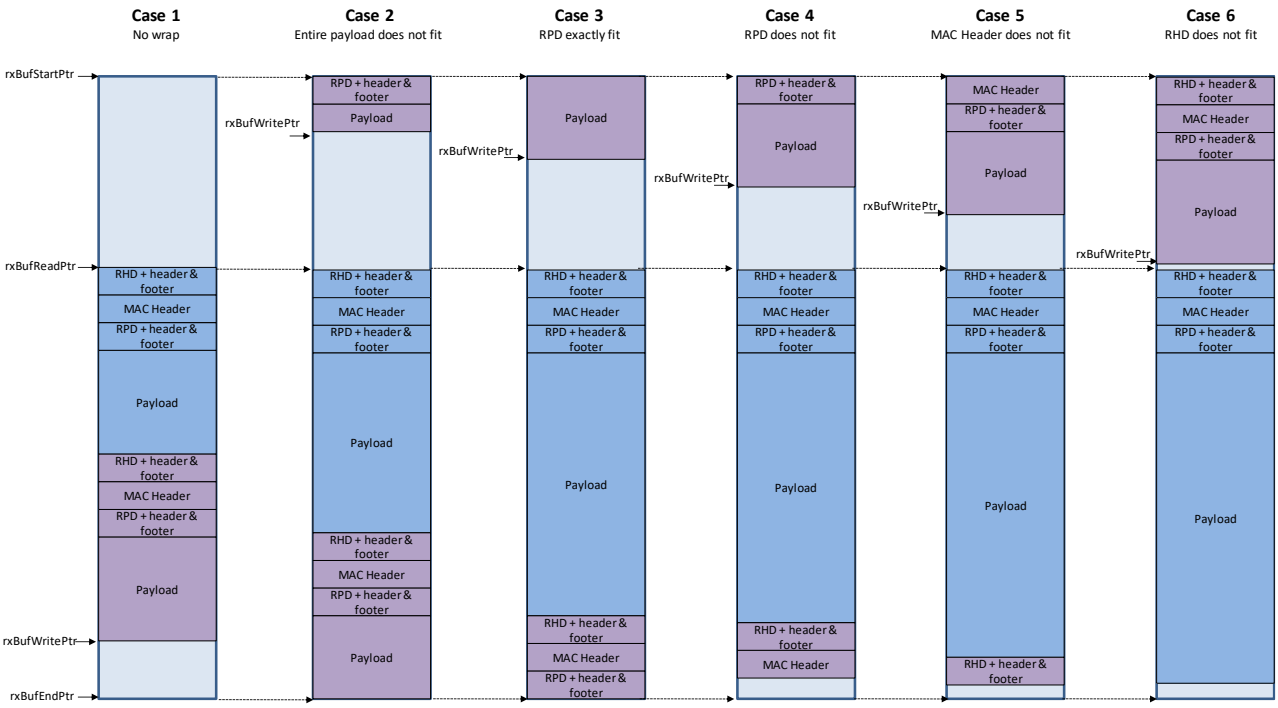


Figure 11: Different case of wrap

2.3.2.1.2 MAC Header location

By default, the MAC Header of the received frames are stored after the footer of the RHD and is location is indicated using *dataStart* and *dataEnd* pointers of the RHD. But for some types of frame such as Control or Management frames, it may be more convenient for the SW to get the MAC Header and the Payload in one piece in the RPD. It is also the case with Data frame when the device behaves as monitor.

This feature allows selecting where the MAC Header is stored: Either in the RHD or along with the Payload in the RPD. In the second case, the RHD *dataStart* and *dataEnd* pointers are set to 0 indicating a null length and the RPD contains the full frame.

The *ctrlMHStoredwithPld*, *mgtMHStoredwithPld* and *dataMHStoredwithPld* fields of *Receive Controller 2 (rxCntrl2Reg)* allowing to select the location of the MAC Header for respectively Control, Management or Data frame.

The *Figure 12: Different locations of MAC Header in Rx Buffer* shows the different memory organizations depending on *xxxMHStoredwithPld*.

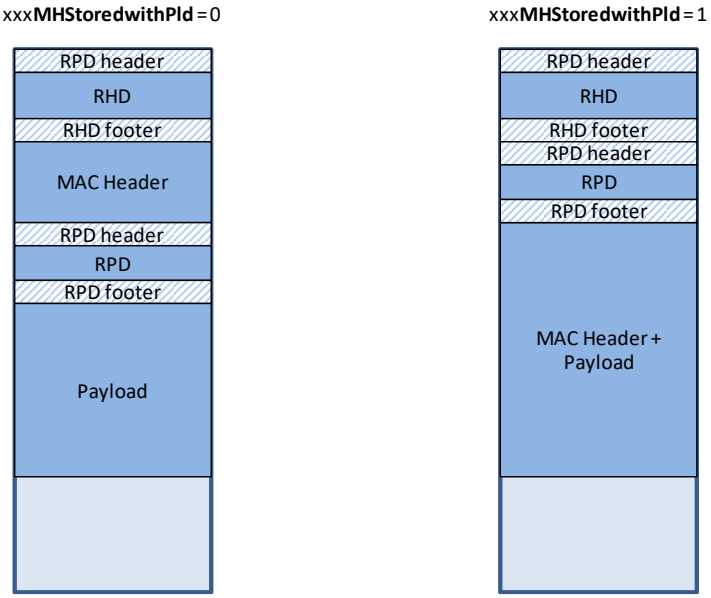


Figure 12: Different locations of MAC Header in Rx Buffer

2.3.2.1.3 Payload location

By default, the payload is written in the RX buffer at after the footer part of the RPD as long as one word remain available in the Rx Buffer before wrapping. However, for some types of frame, it may be convenient for the software to keep the entire payload or at least the beginning of the payload in one block. This is why the “wrap-able” feature which is describe below has been introduced.

Each frame type (control, Management and Data frame) can be marked as “wrap-able”, “unwrap-able” or “partially wrap-able”. The storage of the payload when reaching the end of the buffer depends on this configuration.

The payload of a frame marked as “*wrap-able*” will be stored in the RX Buffer following the RPD and mays wrap if it reaches the end of the buffer.

The payload of a frame marked as “*unwrap-able*” will be stored in the RX Buffer following the RPD only if the entire payload fits in the remaining space of the Buffer. Otherwise, it will entirely stored starting at the *rxBufStartPtr* address.

The payload of a frame marked as *"partially unwrap-able"* will be stored in the RX Buffer following the RPD only if the first N words of the payload fits in the remaining space of the Buffer. Otherwise, it will entirely stored starting at the rxBufStartPtr address. The number of word is configured in partialUnwrapSize field of *Receive Controller 2 (rxCntrl2Reg)*.

The *Figure 13: Payload wrap-able cases* summarizes the different behaviors of the DMA Engine when writing the payload in the RX Buffer.

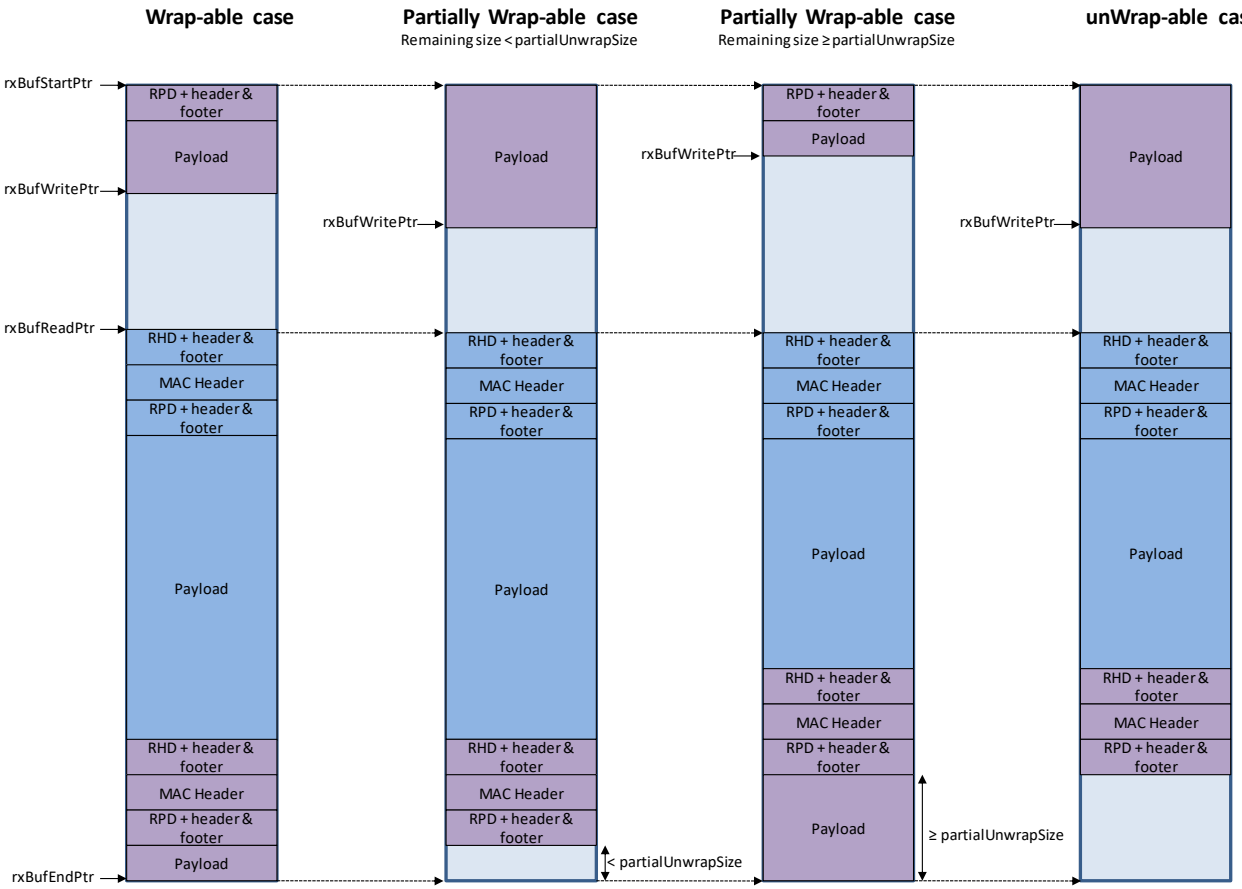


Figure 13: Payload wrap-able cases

Each time a RPD is inserted, the DMA Engine updates the Next Payload Descriptor Pointer of the previous one to build the chained list. Note that when written, the new RPD has its field Next Payload Descriptor Pointer written to 0 and the Data Start Pointer and Data End Pointer automatically updated.

2.3.3 MAC core receive procedure

The embedded Receive FIFO in the MAC HW buffers every frame as it is received from the PHY. When an A-MPDU is received, the HW de-aggregates it and writes the individual MPDUs into the Receive FIFO. If singleton MPDUs are received, they are written individually into the Receive FIFO. Each received frame is checked for correctness and decrypted if the frame is encrypted. After starting the write of an MPDU into the Receive FIFO, if it is discovered that the frame should be discarded, it cannot be removed from the Receive FIFO since the Receive DMA engine may have already started moving it to a memory buffer. Instead indication is passed to the Receive DMA engine via tag bits (in a Rx FIFO) to discard the partly or completely written frame.

It is not necessary to pass certain frames to the SW. Most control frames can be consumed in HW. Frames which have an error need not be passed to SW. Frames not meant for this device can also be dropped in HW. The MAC HW supports promiscuous mode settings which can selectively allow/disallow frames to be passed to SW. This default

behavior can be changed by programming the HW. Refer section *7.4.25*, *Receive Control Register (rxCntrlReg)*. Frames with decryption error are passed to the SW with the relevant indication in the *Receive DMA Header Descriptor*.

Note that the requirement to respond with a response frame is not affected by the promiscuous mode setting. The promiscuous mode setting only controls whether a particular frame should be passed to SW. Transmission of a response frame is controlled by the setting in *MAC Control 1 Register (macCntrl1Reg)*.

2.3.3.1 Frame filtering rules

The HW discards the following received frames by default:

- ✓ Frames which are received with some error like FCS, PHY, Protocol etc
- ✓ All unicast frames that are destined for other stations. Only those frames with the destination address as the local MAC address (or range) are delivered to the MAC SW.
- ✓ All multicast addressed frames.
- ✓ All broadcast frames that do not contain the BSSID of this BSS.
- ✓ RTS, CTS, ACK and CF-End Control frames.
- ✓ CF-ACK, CF-Poll, CF-ACK + CF-Poll, QoS CF-Poll, QoS CF-ACK + CF-Poll frames.
- ✓ BeamFormee frames such as NDPA, NDP and Beamforming Report Poll frames

The HW follows the steps to determine if a frame should be passed to SW:

1. If a frame is received with an error (Protocol/FCS), it is passed to SW with an indication in the Receive Header Descriptor if *acceptErrorFrames* is set, else it is discarded.
2. If a broadcast frame is received and *acceptBroadcast* bit is not set, it is discarded.
    a. If the *acceptBroadcast* bit is set, and the received broadcast frame has a BSSID of this BSS, then the individual frame types settings (like *acceptBeacon*, *acceptCFEnd*) are checked to determine if this type of broadcast frame should be accepted.
    b. If the *acceptBroadcast* bit is set, and the received broadcast frame does not have a BSSID of this BSS but *acceptOtherBSSID* is set, then the individual frame types settings (like *acceptBeacon*, *acceptCFEnd*) are checked to determine if this type of broadcast frame should be accepted.
    c. If the *acceptBroadcast* bit is set, and the received broadcast frame does not have a BSSID of this BSS and *acceptOtherBSSID* is not set, it is discarded.
3. A similar check is performed when a multicast frame is received based on the *acceptMulticast* bit.
4. If a unicast frame directed to this device is received and *acceptMyUnicast* is not set, it is discarded.
    a. If the *acceptMyUnicast* is set, then the individual frame types settings (like *acceptProbeReq*, *acceptBA*) are checked to determine if this type of unicast frame should be accepted. Note that the BSSID is not checked in this case.
5. If a unicast frame not directed to this device is received and *acceptUnicast* is not set, it is discarded.
    a. If the *acceptUnicast* bit is set, and the received unicast frame has a BSSID of this BSS, then the individual frame types settings (like *acceptProbeReq*, *acceptBA*) are checked to determine if this type of unicast frame should be accepted.
    b. If the *acceptUnicast* bit is set, and the received unicast frame does not have a BSSID of this BSS but *acceptOtherBSSID* is set, then the individual frame types settings (like *acceptProbeReq*, *acceptBA*) are checked to determine if this type of unicast frame should be accepted.
    c. If the *acceptUnicast* bit is set, and the received unicast frame does not have a BSSID of this BSS and *acceptOtherBSSID* is not set, it is discarded.
6. If a broadcast or multicast frame without BSSID field in the MAC Header, such as control frame, is received, a specific check is done on the Addr2. This check consists on verifying if the TA matches one of the MAC

Address stored in the Key Search memory. If yes, the frame is considered as coming from a known device and can be accepted. Otherwise, it is discarded.

After completion of reception of a valid frame which was completely written into the Receive FIFO without causing a Receive FIFO overflow, the core transmits a response frame if required. A valid frame in this context is a successfully received frame which is group-addressed, or directed to this device, and which is required to be passed to SW as determined by the setting of the *Receive Control Register (rxCntrlReg)*.

2.3.3.2 Frame Priority rules

BA and Trigger frame which have not been discarded have the highest priority and are stored in Rx Buffer 2.

All the other frames which have not been discarded are stored in Rx Buffer 1.

This feature can be disabled by setting *Receive Controller 2 (rxCntrl2Reg).disableRxBuffer2*. In this case, all the received frames (whatever their types) which have not been discarded are stored in Rx Buffer 1.

2.3.4 Receive DMA channel status updation

When the receive DMA engine finishes the complete movement[8] of an MPDU from the Receive FIFO into system memory, it updates the status of the received MPDU in the *Status Information* field of the Header Descriptor for the MPDU. The address of the first receive Payload Buffer in which the MPDU payload has been written is updated in the Header Descriptor field *firstBufferDPRx* for quick access from SW.

2.4 Encryption support

The core supports 64-bit and 128-bit RC4 key encryption depending on the *cipherLen* bits in the KeyStorageRAM. It supports also TKIP, CCMP 128-bit, CCMP 256-bit, GCMP 128-bit and GCMP 256-bit encryption algorithms. It optionally supports the WPI encryption/decryption needed for WAPI protocol.

2.4.1 KeyStorageRAM structure

The per-MAC address and group address encryption keys are stored in an embedded HW RAM, called the KeyStorageRAM. The first address locations of the KeyStorageRAM are reserved for storing 4 default key IDs for Virtual LANs[9] in sequence. The number of VAP is configurable by software using *Encryption RAM Configuration Register (encrRAMConfigReg).nVAP*. The remaining locations of the RAM holds address mapped (pairwise) encryption keys. The software can indicate to the keySearchEngine the location of the pairwise keys using the *Encryption RAM Configuration Register (encrRAMConfigReg).staKeyStartIndex* and *Encryption RAM Configuration Register (encrRAMConfigReg).staKeyEndIndex*. By default the *staKeyStartIndex* is set to nVAP*4 and *staKeyEndIndex* is set to the latest index available in the memory. However, by adjusting the pairwise keys range, the search procedure is speedup mainly if the number of STA is limited compare to the size of the keyRAM.

The number of index available in the key RAM is configurable using the macro **RW_KEY_INDEX_MAX** (see *8.5 Configuration*)

The structure of the KeyStorageRAM for 64 locations is illustrated below with 6 VAPs. For a KeyStorageRAM having 64 addresses, up to 40 Device addresses and corresponding keys can be stored.

[8] It knows this based on tag bits.

[9] VLANs are a mechanism to separate the broadcast domains, i.e. prevent “leakage” of group-addressed frames between broadcast domains.

keyIndex RAM	cType RAM [2:0]	vlanID RAM[3:0]	sppRAM[1:0]	useDef Key RAM[0]	cLen RAM [1:0]	macAddrRAM [47:0]	encrKeyRam [127:0]	intWPIKeyRam [10] [127:0]
0		Don't care				Don't care	Default Key ID 0 for VLAN 0	
1		Don't care				Don't care	Default Key ID 1 for VLAN 0	
2		Don't care				Don't care	Default Key ID 2 for VLAN 0	
3		Don't care				Don't care	Default Key ID 3 for VLAN 0	
4		Don't care				Don't care	Default Key ID 0 for VLAN 1	
5		Don't care				Don't care	Default Key ID 1 for VLAN 1	
. . .		Don't care				Don't care		
22		Don't care				Don't care	Default Key ID 2 for VLAN 5	
23		Don't care				Don't care	Default Key ID 3 for VLAN 5	
24						Device 1 MAC Address	Device 1 Secret Key	Device 1 WPI Integrity Key
25						Device 2 MAC Address	Device 2 Secret Key	Device 2 WPI Integrity Key
26						Device 3 MAC Address	Device 3 Secret Key	Device 3 WPI Integrity Key
. . .						. . .	. . .	. . .
62						Device 39 MAC Address	Device 39 Secret Key	Device 39 WPI Integrity Key
63						Device 40 MAC Address	Device 40 Secret Key	Device 40 WPI Integrity Key

Figure 14: KeyStorageRAM structure

2.4.2 Programming the KeyStorageRAM

Direct access is not given to the KeyStorageRAM from SW. The KeyStorageRAM is a single port memory that is arbitrated between the SW (for writes) and the MAC HW (for reads). A set of registers is provided to program the KeyStorageRAM from SW. The fields of the *KeyStorageRAM* are explained in the registers from section *7.4.40*, *Encryption Key Word 0 Register (encrKey0Reg)* to section *7.4.46*, *Encryption Control Register (encrCntrlReg)*.

To program a location of the KeyStorage RAM, software writes the encryption registers and sets the *newWrite* bit. Depending on the internal state of the encryption engines, the key programming may take a few microseconds. When the HW samples the *newWrite* bit, it copies the contents of the encryption fields into the KeyStorageRAM location as

[10] This field exists only if the MAC HW supports the WAPI (see 8.5).

programmed in the *indexRAM* field. When the write is completed, the core resets the *newWrite* bit. Software can now write a new set of values into the encryption register fields.

When the field *MAC Control 1 Register (macCntrl1Reg).keyStoRAMReset* is set from SW, the core initializes the KeyStorageRAM with zeros in all the fields for all the locations. When a device associates, its MAC addresses and corresponding key is programmed by the software into the KeyStorageRAM. When a device disassociates, the software should overwrite its MAC address field with all 0's. MAC addresses for which a Null key has to be programmed, can have any encryption key programmed with the *cTypeRAM* field set to "00".

Note tha the software shall wait for the completion of a read, write or search request before requesting a new one.

2.4.3 SW Search in KeyStorageRAM

The SW needs to search in the list MAC Address defined in the keyStorageRAM and gets the information associated to a MAC Address such as the keyIndex or the VLAN. A specific API for this purpose is available through the *Encryption Control Register (encrCntrlReg)* register. The Sw needs to first define the searched MAC Address in the *Encryption MAC Address Low Register (encrMACAddrLowReg)* and *Encryption MAC Address High Register (encrMACAddrHighReg)* registers. Then, by setting the *Encryption Control Register (encrCntrlReg).newSearch* bit, the processing is launched. Once completed, the Core resets the *Encryption Control Register (encrCntrlReg).newSearch* bit and makes all the information available in *Encryption Control Register (encrCntrlReg)* and key registers.

If the defined MAC Address is not found in the keyStorageRAM, the bit *Encryption Control Register (encrCntrlReg).searchError* is set.

2.4.4 Debug mode KeyStorageRAM read

The KeyStorageRAM does not require a read from SW. However, for debugging purposes, the RAM can be read back using the encryption registers. To read a location, SW performs the following steps:

1. Assert the *debugKSR* pin on the top level interface.
2. Write the location of the KeyStorageRAM from which the contents should be copied into the corresponding fields in the encryption registers.
3. Set the *newRead* bit.

When the HW samples the *newRead* bit, it copies the contents of the location of the KeyStorageRAM as programmed in the *indexRAM* field into the corresponding fields of the encryption registers. When the read is complete, the core resets the *newRead* bit. SW can now read the contents of the encryption registers and start the read cycle again.

2.4.5 Encrypted transmission

The *Policy Table entry* of each MPDU contains an index into the KeyStorageRAM at which the required encryption parameters can be found. If the *Protected Frame* bit in the *Frame Control* field of the MAC Header is set and the *dontEncrypt* fields in the MPDU Header Descriptor is reset, the core reads the required KeyStorageRAM location and gets the encryption parameters. Setting the *dontEncrypt* field in the frame to 0 allows the SW to bypass the HW encryption if it has already encrypted the frame. The decision to encrypt a frame is taken by the software, based on whether this frame is a unicast frame, whether it is a data frame, whether the STA to which this frame is directed supports encryption, etc.

The HW ignores the *vlanIDRAM*, *useDefKeyRAM* and *macAddrRAM* fields on transmission.

Note that the HW transmits the frame without encryption if the *cTypeRAM* field at the location pointed to by the index passed by SW is set to *Null Key*, i.e. the HW does not handle a Null Key event on transmission. If no encryption key is valid, SW must not pass a frame for transmission to HW.

2.4.6 Encrypted reception

The core checks the *Protected Frame* and the *Extended IV* bits in the *Frame Control* field in the MAC Header of the received MPDU. If the *Protected Frame* bit is set, the received MPDU should be decrypted and the KeyStorageRAM is searched to extract the parameters from it.

1. If the Address 1 field is a group address, the key search algorithm checks the parameters at the location corresponding to the default key ID in the IV of the frame[11].
    a. If the *cTypeRAM* field is set to Null Key, the frame is passed to SW with a Null Key decryption status, else
    b. The decryption parameters are taken from the key entry.
2. If the Address 1 field is not a group address, the HW searches for the Address 2 field. If the Address 2 field is found in the RAM, the key search algorithm checks the parameters at the location corresponding to the MAC address.
    a. If the *useDefKeyRAM* bit is set, the key search algorithm checks the parameters at the location corresponding to the default key ID in the IV of the frame pointed to by the *vlanIDRAM*. I.e. if the *vlanIDRAM* = 1, and default key ID = 2, then location 6 of the KeyStorageRAM is read from.
    b. If the *cTypeRAM* field is set to *Null*, the frame is passed to SW with a Null Key decryption status, else
    c. The decryption parameters are taken from the key entry.
3. If the Address 1 field is not a group address, and Address 2 field is not found in the RAM, the key search algorithm checks the parameters at the location corresponding to the default key ID in the IV of the frame.
    a. If the *cTypeRAM* field is set to *Null*, the frame is passed to SW with a Null Key decryption status, else
    b. The decryption parameters are taken from the key entry.

When the decryption parameters are taken from the key entry, HW checks if the received frame is an A-MSDU. If it is, then the *sppRAM* field is checked to decide further HW behavior. The A-MSDU is discarded if the *sppRAM* field so indicates. Note that every received encrypted MPDU is decrypted and written to the Receive FIFO unless the decryption is bypassed by SW by programming the *Receive Control Register (rxCntrlReg).dontDecrypt* field. At the end of the decryption, the status of decryption is indicated in the frame. Refer to section *3.2.3*, *Receive DMA Header Descriptor fields* for details on the *DecryptionStatus* indication.

Thus, in the case where *cTypeRAM* indicates Null Key, or *sppRAM* indicates that the A-MSDU should be discarded, HW will still pass the frame through the decryption engine, but the status of the decryption will be set to “Null Key” or “A-MSDU Discarded”.

2.5 Coexistence support

MAC HW includes the following provisions for Coexistence with BT devices:

- Ability to report real time WLAN Tx & Rx activity to a collocated BT device
- Ability to abort Tx when real time BT activity is reported

2.5.1 BT Coexistence basics

By default, the HW reports the on-going activities.
When a transmission is on-going (from *backoffDone* indication until *txEnd_p* reception from the PHY), the flag *coexWlanTx* is set.

[11] Note that the AP will not receive a group addressed frame from any STA in its BSS, hence any received group addressed frame must necessarily fail decryption. At a STA, only group addressed frames that originate in this BSS will be successfully decrypted

When a reception is on-going (from the reception of the first byte of *rxVector1*, until the *rxEndFortiming_p* indication, the flag *coexWlanRx* is set. In parallel with these two flags, additionals information are provided to the external PTA, such as the transmission/reception BW (*coexWlanBW* output), the current channel frequency and channel offset used (*coexWlanFreq* and *coexWlanChanOffset* outputs) and the PTI (*coexWlanPTI* and *coexWlanPTIToggle* outputs) (see below). Note that the current channel frequency and offset outputs are directly connected to the *Coex Control Register (coexCntlReg).coexFreqUsed* and *Coex Control Register (coexCntlReg).coexWlanChanOffset* field.

2.5.2 Transmission/Reception abort/delayed

In a WLAN – Bluetooth coexistence scenario, WLAN might be requested to abort transmission by means of *coexWlanTxAbort*, so as to prevent mutual interfering with a collocated BT device. This request is generated outside the MACHW.
When set and only when *Coex Control Register (coexCntlReg).coexEnable* bit is set, the *coexWlanTxAbort* is considered by the HW as a Medium Busy indication (similar to a Virtual CarrierSense). As a consequence, all the pending transmissions are postponed.

The *coexWlanTxAbort* also aborts the on-going transmission but this is done at the radio controller level. However, in case of Tx abort due to coexistence arbitration, the Singleton MPDU frames are retried but the retry numbers as well as the backoff CW are not incremented. In case of AMPDU, nothing special is done. In this specific case, the BACK is expected and the BAR is transmitted if the BA is not received.

It may also occur that the reception shall also be aborted due to a sharing of the PHY resources (RF for example) with collocated BT device. In this case, the *coexWlanRxAbort* is asserted and the rxReq is immediately de-asserted.

2.5.3 Packet Traffic Information

In order to allow a more efficient arbitration between WLAN and Bluetooth system, the arbitration could be done per packet based on their priority. For this purpose, the MAC HW provides a priority for each frame to the external PTA. This priority is named PTI (for packet traffic information) and is coded on 4 bits. This priority differs depending on the frame type (ACK /BA, Mgt, Control, VO QoS Data, VI QoS Data, BE QoS Data, BK QoS Data). The PTI value for each frame type is configured by the SW in the register *Coex PTI Register (coexPTIReg).* Additionally, the SW has the possibility to provide a specific PTI value for each SingletonMPDU or A-MPDU in the THD (see *3.1.1, Transmit DMA Header Descriptor*).
When the Core is neither transmitting nor receiving a frame, the *coexWlanPTI* output is set to 0. As soon as a transmission is starting, the *coexWlanPTI* output is set to the value defined in the THD (PTITx field of PHY Control Information 1) or to the value of *Coex PTI Register (coexPTIReg)* according to the packet type. During a reception, as soon as the packet type is kown, the *coexWlanPTI* is set based on the information conatined in *Coex PTI Register (coexPTIReg).* The same mechanism is used for the immediate responses. Note that the *coexWlanPTI* output is set as soon as the frameType or PTITx information is known.
As the PTI information can be used by an external PTA which does not run one macCoreClk, a compagnon signal name *coexWlanPTIToggle* can be used for the *coexWlanPTI* clock domain crossing. It toggles each time the *coexWlanPTI* value is updated.

2.5.3.1 Automatic PTI Adjustment

Additionally to the PTI value which is set based on the traffic type, it may be required to increase this value if the amount of abort is too important. For this purpose, the Automatic Adjustemnt feature can be used. When enabled by setting the *Coex Control Register (coexCntlReg).coexAutoPTIAdjEnable*, the PTI for this type of traffic is incremented by *Coex Control Register (coexCntlReg).coexAutoPTIAdjIncr* incremented if an abort is detected during the transmission or reception. If the transmission or reception is completed without being aborted, the PTI value for this type of traffic is reset to the value defined in *Coex PTI Register (coexPTIReg).*

2.5.4 Coexistence Interface SW Control

For specific case, such as scan or high priority traffic, the SW may decide to override the HW and controls the coexistence interface. This is done setting the *Coex Control Register (coexCntlReg).coexForceEnable* bit. In this case, all the coex interface ouput ports are directly controlled by the SW using the *Coex Control Register (coexCntlReg).coexForceWlanXXX* fields.

Additionally, the software can get the current status of the coex interface by reading the *Coex Status Register (coexStatReg)*.

The software can also get an interrupt (*General Interrupt Event Register (genIntEventReg).coexEvent*) when *coexWlanTxAbort* and/or *coexWlanRxAbort* inputs changes. The software can use the *Coex Interrupt Register (coexIntReg)* to select which events generate the *coexEvent* interrupt.

2.6 Beamforming support

2.6.1 Beamformee support

The support of beamformee feature is mainly managed by the Core. The MAC SW shall indicate in the association request the supports of beamformee. Once done, the SW is not involded anymore.

The Core supports the two beamforming calibration procedures (SU and MU) defined in the 802.11ac-2013 based on NPDA and NPD.

Regarding the reception of beamformed frame, the MACCore does not perform specific task. It reports as is the beamformed bit of the rxVector into the RHD.

Each time a beamformed frame is correctly received, the rwBeamformingReceivedFrameCount MIB is incremented.

Each time a beamforming report frame with SU FeedbackType response is transmitted, the rwSUBFRTransmittedCount MIB is incremented.

Each time a beamforming report frame with MU FeedbackType response is transmitted, the rwMUBFRTransmittedCount MIB is incremented.

To be used, this feature needs RW_BFMEE_EN and is enabled by the software using *Beamformee Control Register (bfmeeControlReg).bfmeeEnable* and *Beamformee Control Register (bfmeeControlReg).bfmeeMUSupport* (only required in case of Rx MU-MIMO support).

2.6.1.1 SU Beamforming calibration procedure

The SU beamforming frame exchange is shown of the *Figure 15: SU beamforming calibration frame exchange.*

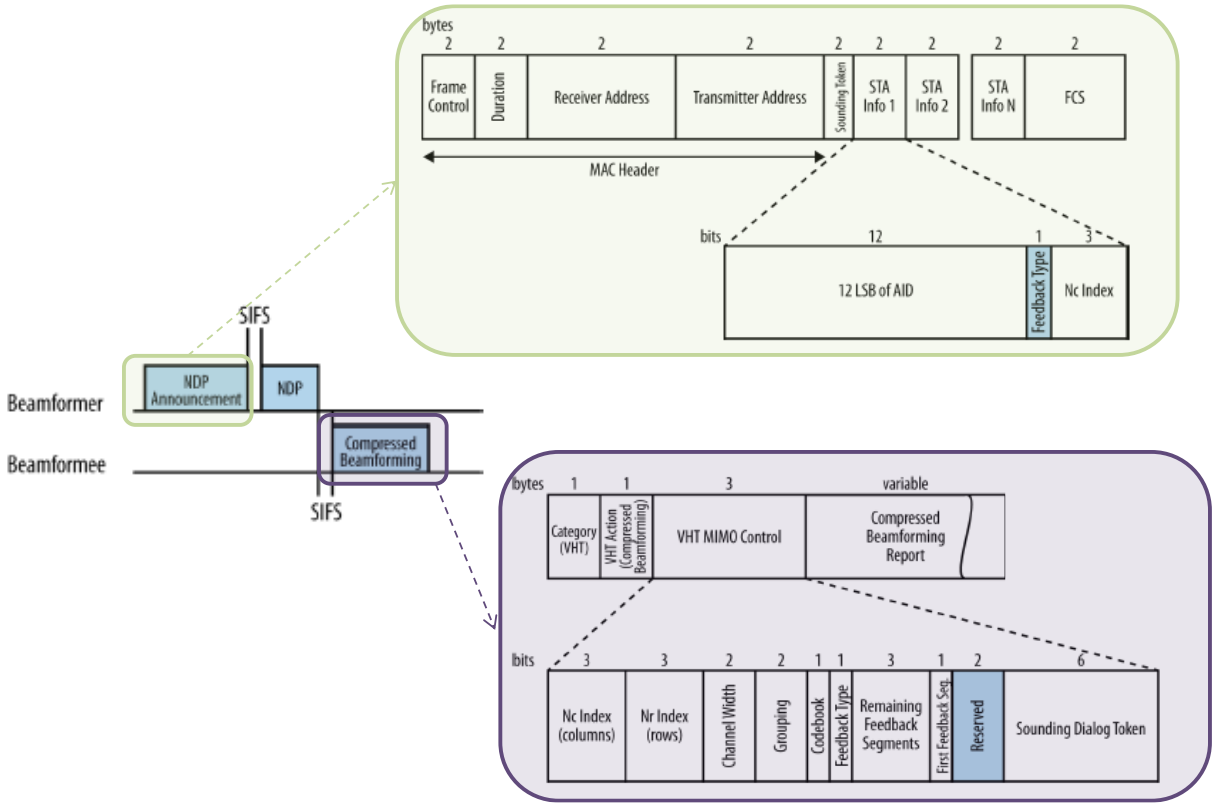


Figure 15: SU beamforming calibration frame exchange

The support of the SU beamforming calibration procedure is enabled by the SW using the *Beamformee Control Register (bfmeeControlReg).bfmeeEnable*. Before that, the SW provides to the Core the different parameters needed for the Beamforming Report frame generation. Two types of informations are provided by the SW:

1- Information required to control the external Beamforming report generation module such as
   a. The codebook information (*Beamformee Control Register (bfmeeControlReg).bfmeeCodebook*)
   b. The grouping information (*Beamformee Control Register (bfmeeControlReg).bfmeeGrouping*)
   c. The number of supported row (*Beamformee Control Register (bfmeeControlReg).bfmeeNr*)

2- Information required for the BFR TxVector generation.
   a. The MCS (*Beamformee Control Register (bfmeeControlReg).bfrMCS*)
   b. The Short GI (*Beamformee Control Register (bfmeeControlReg).bfrShortGI*)
   c. The FormatMod (*Beamformee Control Register (bfmeeControlReg).bfrFormatMod*)

3- Information required for the BFR MAC Header generation
   a. The transmiter address (ADDR2) of the BFR corresponds to the receiver address (ADDR1) of the received NDPA.

When enabled, upon reception of a correct NPDA frame addressed to the device, the Core exctracts and stores the Transmit Address, the Sounding Dialog Token, the FeedbackType and the Nc Index parameters. Then, it waits for the NDP reception. Once the NDP is completely received, it provides the following information to the external Beamforming report generation module:

- *bfrChBW* based on the rx BW of the NPD provided by PHY.
- *bfrGrouping* using the *Beamformee Control Register (bfmeeControlReg).bfmeeGrouping* value.

- *bfrCodebook* using the *Beamformee Control Register (bfmeeControlReg).bfmeeCodebook* value.
- *bfrFeedbackType* based on the FeedbackType exctracted from the NPDA.
- *bfrNc* using the *Beamformee Control Register (bfmeeControlReg).bfmeeNc* value.
- *bfrNr* based on the NSS fo the NDP.

Then, it de-asserts the *rxReq* and trigs the external Beamforming report generation module using *bfrStart.*

The external beamforming module is triggered only if the NDPA is correctly received (FCS OK), the RA is the Core MAC Address, the AID matches the AID defined in *(NA)*, Nc Index extracted from the NDPA is lower or equal to *Beamformee Control Register (bfmeeControlReg).bfmeeNc*, and a correct reception of NDP.

SIFS after the reception of the NDP, the BFR frame is generated and start to be transmitted. The TX Vector parameters using for the BFR transmission are partially defined in the *Beamformee Control Register (bfmeeControlReg)*. The transmission BW is the same than the reception BW of the NPD.

The formatting of the BFR is done in the txController which uses the information extracted from the NDPA to generate the MAC Header and the VHT MIMO Control fields.

The Compressed Beamforming Report field is provided in the right format by the external Beamforming report generation module. As soon as the VHT MIMO Control field is completed, the Core asserts the *bfrDataReady* to request the report. All the bytes of the report are provided byte per byte by the external module using *bfrData* and *bfrDataValid* inputs.

Note that the Core does not support segmented report. As a consequence,

- First Feedback Segment field of BFR frame is always set to 1.
- Remaining Feedback Segments of BFR frame is always set to 0.

When the *bfrDone* is received the *bfrStart* is de-asserted. In case of error during the transmission, the *bfrStart* is de-asserted.

Note that the informations extracted from the NDPA are valid only if a NPD frame is received after SIFS. In case of a NPD received without NDPA, the external beamforming module is not triggered.

2.6.1.2 MU Beamforming calibration procedure

The MU beamforming frame exchange is shown of the *Figure 16: MU beamforming calibration frame exchange.*

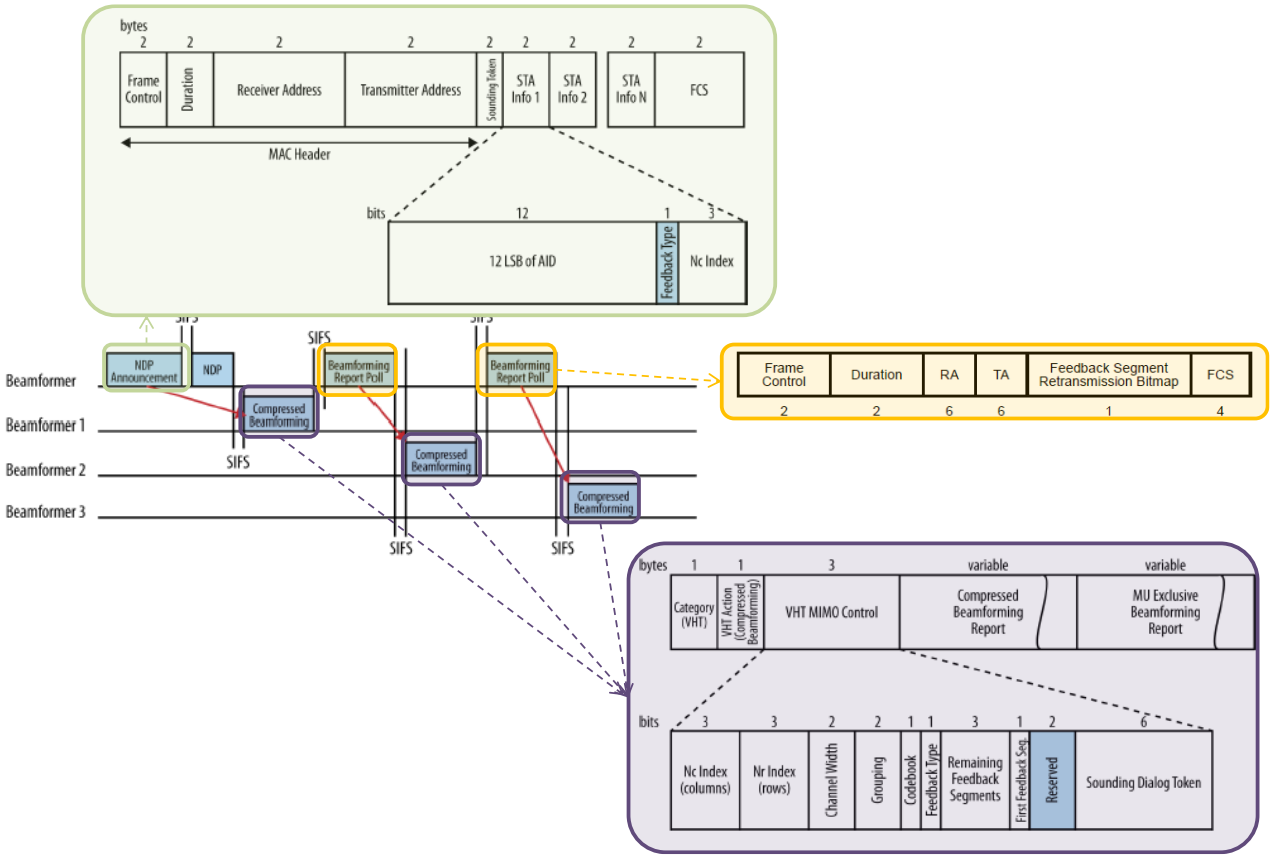


Figure 16: MU beamforming calibration frame exchange

The support of the MU beamforming calibration procedure is enabled by the SW by setting the *Beamformee Control Register (bfmeeControlReg).bfmeeEnable* and *Beamformee Control Register (bfmeeControlReg).bfmeeMUSupport*.

The configuration required for the SU beamforming calibration procedure is also required in MU. Additionally, the SW shall specified the AID of the device using *Beacon Control 2 Register (bcnCntrl2Reg)*(NA).*AID.*

When enabled, upon reception of a correct NPDA frame addressed to the device, the Core exctracts and stores the Transmit Address, the Sounding Dialog Token, the FeedbackType and the Nc Index parameters. Then, it waits for the NDP reception. Once the NDP is completely received, it provides the following information to the external Beamforming report generation module:

- *bfrChBW* based on the rx BW of the NPD provided by PHY.
- *bfrGrouping* using the *Beamformee Control Register (bfmeeControlReg).bfmeeGrouping* value.
- *bfrCodebook* using the *Beamformee Control Register (bfmeeControlReg).bfmeeCodebook* value.
- *bfrFeedbackType* based on the FeedbackType exctracted from the NPDA.
- *bfrNc* based on the Nc Index exctracted from the NPDA.
- *bfrNr* based on the NSS fo the NDP.

The processing done by the core on the NPD and NPD is similar than for the SU beamforming procedure except the processing of the STA Info field of the NDPA. In SU, the RA of the NDPA is the device MAC Address. In MU, the RA is a broadcast address which indicates more than one STA info fields. Thus, the core compares the 12 LSB of its AID with the AID defined in each STA Info fields. If one of them matches, it extracts the FeedbackType and Nc Index information and it also stores if its AID matches the first STA info or not. If none of them match, it consider that the NDPA is not addressed to its.

Upon reception of NDP, the Core performs the same checks as for SU and launch the external Beamforming Beamforming report generation module.

Then, two cases are possible:

1- If the Core AID matches the STA info 1. In this case, the Core transmits the BFR SIFS after the NDP and its transmission follow the same steps than for SU.

2- If the Core AID did not match the STA info 1. In this case, the Core waits for the completion of the SVD and compression performed by the external Beamforming Beamforming report generation module. When the *bfrDone* is received, the Core de-asserts the *bfrStart* and restart the PHY in Rx (assertion of *rxReq*). Then, upon reception of the BeamForming Report Poll frame addresses to the device, the device transmits the BFR.

Information required for the BFR MAC Header generation:

1- The transmiter address (ADDR2) of the BFR corresponds to the MAC Address defined in *macAddr* register.

2.7 MU-MIMO Reception as a STA support

From MAC HW point of view, the reception of a MU-MIMO frame does not really differ from a SU reception. Most of the processing is performed by the PHY. The transmission of the BA depends on the ACK Policy.

Note also that the PHY does not indicate if the received frame is a MU-MIMO or not. This is deducted by the MAC from the *groupID* information carried in the rxVector1. If the *groupdID* is 0 or 63, the received frame is a SU-MIMO, otherwise it is a MU-MIMO.

Each time a MU-MIMO frame addressed to the device is correctly received, the *rwMUReceivedFrameCount* MIB is incremented.

3 DMA descriptors

3.1 Transmit DMA

3.1.1 Transmit DMA Header Descriptor

A Transmit Header Descriptor contains the following fields:

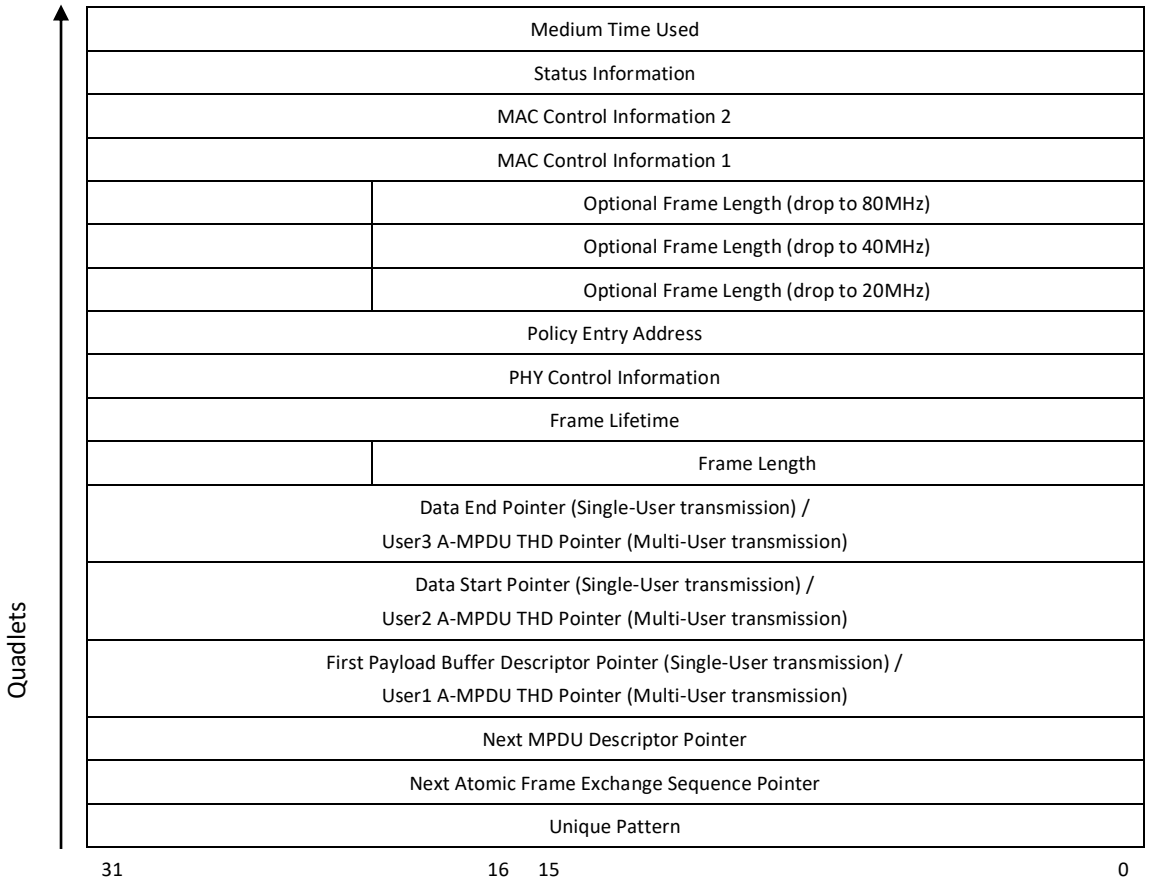


Figure 17: Transmit Header Descriptor

3.1.2 Transmit DMA Buffer Descriptor

A Transmit Buffer Descriptor contains the following fields:

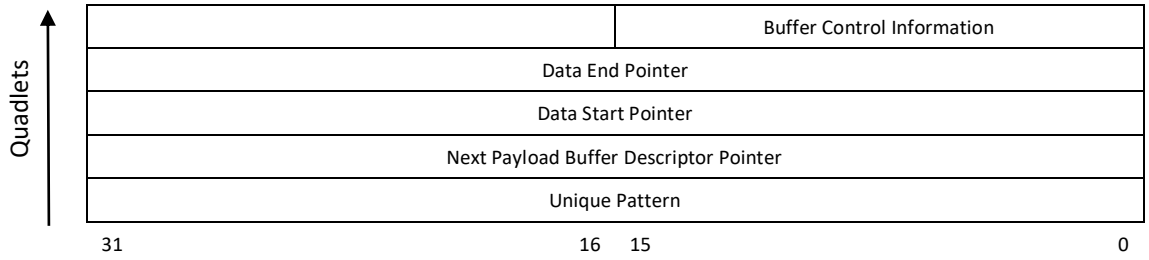


Figure 18: Transmit Buffer Descriptor

3.1.3 Transmit DMA Header Descriptor fields

Field In Descriptor	Sub-fields	Bit #	Remarks
uPatternTx		31:0	Unique Pattern for Transmit DMA If the HW does not find the pattern 0xCAFEBABE in this field, it raises the error interrupt *General Interrupt Event Register (genIntEventReg).txDMAError*. Refer to section *9.3*, *Terminal DMA errors* for details.

The *Next Atomic Frame Ex Seq Pointer* field is valid for all the Transmit Header Descriptors except for the Secondary Users Transmit Header Descriptor.

Field In Descriptor	Sub-fields	Bit #	Remarks
Next Atomic Frame Ex Seq Pointer	nextAFrmExSeqP	31:0	Next Atomic Frame Exchange Sequence Pointer This field points to the starting descriptor of the next atomic frame exchange sequence. It is a quadlet aligned field. It is used to find the start of the next atomic frame exchange sequence under different conditions: ✓ When a frame is discarded due to retry limit reached condition or lifetime expiry. ✓ When jumping over the explicit BAR frame that is attached by the SW behind every A-MPDU when A-MPDU was delivered with Implicit Block ACK and the BA frame is successfully received. If the HW finds an invalid address programmed, it raises the raises the error interrupt *General Interrupt Event Register (genIntEventReg).txDMAError*. Refer to section *9.3*, *Terminal DMA errors* for details.

Field In Descriptor	Sub-fields	Bit #	Remarks

Field In Descriptor	Sub-fields	Bit #	Remarks
Next MPDU Descriptor Pointer	nextMDPTx	31:0	Next MPDU Descriptor Pointer for Transmit DMA This field points to the Header Descriptor of the next MPDU that is a part of the atomic frame exchange sequence. It is a quadlet aligned field. It is used to point to the next fragment of a fragmented MSDU. It is used to point to the next constituent MPDU of an A-MPDU. If the HW finds an invalid address programmed, it raises the raises the error interrupt *General Interrupt Event Register (genIntEventReg).txDMAError*. Refer to section *9.3, Terminal DMA errors* for details.

The *First Buffer Descriptor Pointer* field is valid for the constituent MPDUs of an A-MPDU and for singleton MPDUs. It is reserved in the SU-MIMO A-MPDU Header Descriptor.

Field In Descriptor	Sub-fields	Bit #	Remarks
First Payload Buffer Descriptor Pointer (Single-User transmission)	firstPBufDPTx (SU-MIMO)	31:0	In case of SU-MIMO transmission First Payload Buffer Descriptor Pointer for Transmit DMA This field points to the first Buffer Descriptor of the MPDU when the MPDU is split across multiple buffers. It is a quadlet aligned field.

The *Data Start Pointer* field is valid for the constituent MPDUs of an A-MPDU and for singleton MPDUs. It is reserved in the SU-MIMO A-MPDU Header Descriptor.

Field In Descriptor	Sub-fields	Bit #	Remarks
dataStartPTx (Single-User transmission))	dataStartPTx (SU-MIMO) user2ATH	31:0	In case of SU-MIMO transmission Data Start Pointer for Transmit DMA This field is a byte aligned field and points to the address in the buffer where the data starts. The byte in the buffer at this address is consumed by the MAC HW. It is also possible to do not point where the data start and use the First Payload Buffer Descriptor for this purpose. In this case, the dataStartPTx shall be forced to 0.

The *Data End Pointer* field is valid for the constituent MPDUs of an A-MPDU and for singleton MPDUs. It is reserved in the SU-MIMO A-MPDU Header Descriptor.

Field In Descriptor	Sub-fields	Bit #	Remarks

dataEndPTx (Single-User transmission)	dataEndPTx (SU-MIMO)	31:0	In case of SU-MIMO transmission Data End Pointer for Transmit DMA This field is a byte aligned field and points to the address in the buffer where the data ends. The byte in the buffer at this address is consumed by the MAC HW. It is used to transmit only part of the contents of a buffer. E.g., in case of fragmentation of an MSDU residing in a single buffer, the *dataBufferPTx* points to the buffer containing the MSDU, and the *dataEndPTx* points to the end of the fragmented MPDU inside that buffer.

Field In Descriptor	Sub-fields	Bit #	Remarks
Reserved		31:20	
frameLengthTx		19:0	Frame Length for Transmit DMA This field indicates the total length of the frame on air. This field takes a value from 0 – 1048575. It carries different information in different contexts: In an A-MPDU context, an additional Transmit Header Descriptor is attached at the start of the A-MPDU. It contains the length of the entire A-MPDU. Each constituent MPDU of the A-MPDU has its own Transmit Header Descriptor which contains the length of the individual MPDU. In a non-A-MPDU context, each descriptor carries the length of the MSDU (if un-fragmented), or the length of the fragmented MPDU (if fragmented).

The *Frame Lifetime* field is valid for the SU-MIMO A-MPDU Header Descriptor and for singleton MPDUs. It is reserved in the MPDU Header Descriptor for the constituent MPDUs of an A-MPDU.

Field In Descriptor	Sub-fields	Bit #	Remarks

Field In Descriptor	Sub-fields	Bit #	Remarks
frameLifetime		31:0	Frame Lifetime This field gives the lower 32 bits [31:0] of the TSF at which this frame should be dropped by the MAC core. If the difference between the lifetime and the current TSF is bigger than 0x80000000, the lifetime is considered as expired. This allows handeling the TSF scroll around. For EDCA frames, this field represents the MSDU Lifetime parameter. This field is ignored by the core if it is all zeros, i.e. the frame lifetime check in hardware can be bypassed if software programs this field to 16'b0. This field contains the MSDU Lifetime of the last MPDU that is contained within an A-MPDU.

The *PHY Control Information* field is valid for the SU-MIMO A-MPDU Header Descriptor and for singleton MPDUs. It is reserved in the MPDU Header Descriptor for the constituent MPDUs of an A-MPDU. Note that only the *userPosition* field is relevant for a MU-MIMO A-MPDU Header Descriptor for a Secondary User.

Field In Descriptor	Sub-fields	Bit #	Remarks
PHY Control Information	partialAIDTx	30:22	partialAID for Transmission
PHY Control Information	groupIDTx	21:16	groupID for Transmission
PHY Control Information	muMIMO Tx	15	Multi-User Transmission Indicates whether the HW should transmit this frame in a MU-MIMO VHT frame. When set, it indicates that the current THD is a MU-MIMO A-MPDU Transmit Header Descriptor. When reset, it indicates that the current THD is a SU-MIMO A-MPDU Transmit Header Descriptor. This bit is reserved for STA
PHY Control Information	*Reserved*	14	Reserved
PHY Control Information	userPositionTx	13:12	User Position for Transmission Indicate the user position of the A-MPDU inside a MU-MIMO transmission. Note that the userPosition defined in this field is independent of the physical macPhy interface. The userPosition 0 can be defined in one of the secondaryPath whereas the primary path can get a userPosition higher than 0.

Field In Descriptor	Sub-fields	Bit #	Remarks
PHY Control Information	PTITx	11:8	Packet Traffic Information Gives the priority of the current frame for external Coexistence arbitration. When force to 0, the Core uses the priority defined the *Coex PTI Register (coexPTIReg)* according to the packet type.
PHY Control Information	*Reserved*	7	Reserved
PHY Control Information	continuousTx	6	Enable the continuous transmit mode. When set, the programmed transmission starts with an infinite length.
PHY Control Information	dozeNotAllowedTx	5	Indicates whether or not a VHT AP allows non-AP VHT STAs in TXOP power save mode to enter Doze state during the TXOP. When reset, it indicates that the VHT AP does not allow non-AP VHT STAs to enter doze mode during a TXOP. When set, it indicates that the VHT AP allows non-AP VHT STAs to enter doze mode during a TXOP.
PHY Control Information	dynBWTx	4	Dynamic Bandwidth for Transmission When set, it indicates that the BW Management is Dynamic. When reset, it indicates that the BW Management is Static This bit is valid only if *useBWSignalingTx* is set.
PHY Control Information	useBWSignalingTx	3	Use BW Signaling for Transmission Indictes whether the HW should use the BW signalling in the RTS/CTS protection
PHY Control Information	*Reserved*	2-0	*Reserved*

The *Policy Entry Address* field is valid for the SU-MIMO A-MPDU Header Descriptor and for singleton MPDUs. It is reserved in the MPDU Header Descriptor for the constituent MPDUs of an A-MPDU.

Field In Descriptor	Sub-fields	Bit #	Remarks
Policy Entry Address	polEntryAddress	31:0	Policy Entry Address This field is the address of the *Policy Table entry* in local memory where certain parameters which control the transmission can be found. If the HW finds an invalid address programmed, it raises the raises the error interrupt *General Interrupt Event Register (genIntEventReg).txDMAError*. Refer to section *9.3*, *Terminal DMA errors* for details.

The *Optional Frame Length* fields are valid for the SU-MIMO A-MPDU Header Descriptor only. It is reserved in the MPDU Header Descriptor for the constituent MPDUs of an A-MPDU and for singleton MPDUs.

Field In Descriptor	Sub-fields	Bit #	Remarks
frameLengthOpt20MHzTx		19:0	Optional Frame Length 20MHz for Transmit DMA This field indicates the total length of the frame on air in case the available BW is 20MHz. This field takes a value from 0 – 1048575.

Field In Descriptor	Sub-fields	Bit #	Remarks
frameLengthOpt40MHzTx		19:0	Optional Frame Length 40MHz for Transmit DMA This field indicates the total length of the frame on air in case the available BW is 40MHz. This field takes a value from 0 – 1048575.

Field In Descriptor	Sub-fields	Bit #	Remarks
frameLengthOpt80MHzTx		19:0	Optional Frame Length 40MHz for Transmit DMA This field indicates the total length of the frame on air in case the available BW is 40MHz. This field takes a value from 0 – 1048575.

The *MAC Control Information 1* field is valid for the SU-MIMO A-MPDU Header Descriptor and for singleton MPDUs. It is reserved in the MPDU Header Descriptor for the constituent MPDUs of an A-.

Field In Descriptor	Sub-fields	Bit #	Remarks
MAC Control Information 1	protFrmDur	31:16	Protection Frame Duration The value of this field is copied to the Duration field of any HW created protection frame (i.e. not prepared and chained by SW) when *dontTouchDur* = 1.
MAC Control Information 1	Reserved	15	Reserved
MAC Control Information 1	writeACK	14	Write ACK This bit is set if the SW expects the ACK frame for this frame to be passed to it when received.
MAC Control Information 1	lowRateRetry	13	Lower Rate on Retries This bit is set if the SW wants the HW to reduce the rate of the RTS frame or the frame when retransmitting. The lower RTS rate is selected from the programmed rates in the

Field In Descriptor	Sub-fields	Bit #	Remarks
			Rates Register (ratesReg) or the *HTMCS Register (HTMCSReg)* while the lower frame rate is selected from the *Policy Table fields MCSIndex1, 2, 3*. This bit is recommended[12] to be set only if the *dontTouchDur* = 0 which indicates that the HW should update the Duration field. The net effect of this restriction is that rate adaptation in HW should not be performed if the duration calculated in SW is used.
MAC Control Information 1	lstpProt	12	L-SIG TXOP Protection for protection frame Indicates whether the HW should use L-SIG TXOP protection policy when transmitting the protection frame created by HW.
MAC Control Information 1	lstp	11	L-SIG TXOP Protection Indicates whether the HW should use L-SIG TXOP protection policy when transmitting this frame. The L-SIG duration covers till the end of the expected acknowledgement frame.
MAC Control Information 1	expectedAck	10:9	Expected Acknowledgement Indicates what type of acknowledgement frame is expected after the transmission of this frame. The bits are encoded as follows: 2'b00: No acknowledgement. This is used when the frame does not expect an acknowledgement frame. It is set for group addressed frames and QoS Data frames with No-ACK and Block-ACK policy. 2'b01: Normal ACK. This is used when the frame expects an ACK frame in response. It is set for unicast non-QoS Data frames, Management frames, QoS Data frames with Normal ACK policy and BAR frames under Delayed BA setup or HT-Delayed BA setup with Normal ACK policy. It is also used in case of NPD or BFRPoll transmission which expects a BFR immediate response. 2'b10: BA: This is used when the frame expects an uncompressed BA frames in response. It is set for a BAR frame under Immediate BA setup. 2'b11: Compressed BA: This is used when the frame expects a compressed BA in response. It is set for an A-MPDU with Normal ACK policy and for a BAR frames under the HT-Immediate BA setup. The value in this field is used by the MAC HW to calculate the time on air of the frame exchange sequence.

[12] If this recomendation is not followed, there is a chance that adequate NAV protection will not be established when the HW lowers the rate for retries but the SW calculated Duration field is still used.

Field In Descriptor	Sub-fields	Bit #	Remarks
MAC Control Information 1	Reserved	8:0	Reserved

The *MAC Control Information 2* field is valid for the MPDU Header Descriptor for the constituent MPDUs of an A-MPDU and for singleton MPDUs. All fields except *aMPDU*, *whichDescriptor* and *interruptEnTx* are reserved for the SU-MIMO A-MPDU Header Descriptor.

Field In Descriptor	Sub-fields	Bit #	Remarks
MAC Control Information 2	dontGenerateMH	31	Don't Generate MAC Header This field informs HW that it should not generate the MAC Header by itself, and should transmit the MAC Header fields exactly as programmed by SW. Setting this bit bypasses the HW Header generation logic as explained in section *2.2.4.2.7, MAC Header generation logic in HW*.
MAC Control Information 2	dontEncrypt	30	Don't Encrypt This bit bypassed the encryption operation during transmission. If this is set, the MAC HW will not encrypt the frame even if the *Protected Frame* bit in the MAC Header is set. If this bit is reset, the frame will be encrypted only if the *Protected Frame* bit is set. This allows the SW to set this bit under certain debugging conditions only.
MAC Control Information 2	dontTouchFC	29	Don't Touch Frame Control This field informs HW that it should not update the Frame Control field, and should instead transmit the field exactly as programmed from SW.
MAC Control Information 2	dontTouchDur	28	Don't Touch Duration This field informs HW that it should not update the Duration field of this frame and any protection frame transmitted before the frame, and should instead transmit the field exactly as programmed from SW.
MAC Control Information 2	dontTouchQoS	27	Don't Touch QoS This field informs HW that it should not update the QoS field, and should instead transmit the field exactly as programmed from SW.
MAC Control Information 2	dontTouchHTC	26	Don't Touch HT Control This field informs HW that it should not update the HTC field, and should instead transmit the field exactly as programmed from SW.

Field In Descriptor	Sub-fields	Bit #	Remarks
MAC Control Information 2	dontTouchTSF	25	Don't Touch TSF This field informs HW that it should not update the TSF field (in case of Beacon, Probe Response and Timing Advertisement frames), and should instead transmit the field exactly as programmed from SW.
MAC Control Information 2	dontTouchDTIM	24	Don't Touch DTIM This field informs HW that it should not update the DTIM Count field (in case of Beacon and Probe Response frames), and should instead transmit the field exactly as programmed from SW.
MAC Control Information 2	dontTouchFCS	23	Don't Touch FCS This field informs HW that it should not update the FCS field, and should instead transmit the field exactly as programmed from SW.
MAC Control Information 2	underBASetup	22	Under BA Flow Indicates whether BA has been setup for this MPDU. This field is not used by HW. It is mirrored by HW into the *whichDescriptorSW* field in the Status Information field for SW to use when it reclaims this descriptor.
MAC Control Information 2	aMPDU	21	A-MPDU This field indicates whether this Transmit Header Descriptor belong to an A-MPDU.
MAC Control Information 2	whichDescriptor	20:19	Which Descriptor Indicates what kind of a descriptor this is. This field is used in conjunction with the *aMPDU* bit as described in detail in the table below.

aMPDU	whichDescriptor[1:0]	Indicates
0	00	3'b000: Only 1 THD possible, describing an unfragmented MSDU
0	01	3'b001: First THD describing the first MPDU of a fragmented MSDU
0	10	3'b010: Intermediate THD describing intermediate MPDUs of a fragmented MSDU
0	11	3'b011: Last THD describing the last MPDU of a fragmented MSDU
1	00	3'b100: "Extra" THD describing an A-MPDU
1	01	3'b101: First THD describing the first MPDU of an A-MPDU
1	10	3'b110: Intermediate THD describing intermediate MPDUs of an A-MPDU
1	11	3'b111: Last THD describing the last MPDU of an A-MPDU

Field In Descriptor	Sub-fields	Bit #	Remarks
MAC Control Information 2	nBlankMDelimiters	18:9	N Blank MPDU Delimiters Indicates the number of blank MPDU delimiters that are inserted by HW after the MPDU or MPDU padding if any. It is used to satisfy the Minimum MPDU Start Spacing requirement of the receiver. Note that the blank delimiters are also inserted after the last MPDU of the A-MPDU.
MAC Control Information 2	interruptEnTx	8	Interrupt Enable for Transmit DMA If this bit is set, HW will raise an interrupt under the following conditions: For a non-A-MPDU or for the A-MPDU Header Descriptor of an A-MPDU, the interrupt is raised after the *descriptorDoneHWTx* bit in the Header Descriptor of any frame has been set. For the constituent MPDUs of an A-MPDU, the interrupt is raised when the Header Descriptor is handled. This interrupt provides a context to SW to perform any activities for the queue, like descriptor book-keeping, chained MPDUs, starting aggregation, etc.
MAC Control Information 2	Reserved	7	Reserved
MAC Control Information 2	subtype	6:3	Frame Subtype Indicates the *Subtype* field of the *Frame Control* field of the MPDU.
MAC Control Information 2	type	2:1	Frame Type Indicates the *Type* field of the *Frame Control* field of the MPDU.
MAC Control Information 2	tsValid	0	Frame Type Subtype Valid Indicates if the Type and Subtype fields have been populated correctly. Note: The SW must populate this field when transmitting these frames through the Transmit DMA: BAR, RTS, CTS, CF-End and PS-Poll.

The *Status Information* field is valid for the A-MPDU Header Descriptor and for singleton MPDUs. It is reserved in the MPDU Header Descriptor for the constituent MPDUs of an A-MPDU. See also section *2.2.4.2.9*, *Transmit DMA channel status updation*.

Field In Descriptor	Sub-fields	Bit #	Remarks
Status Information	descriptorDoneHWTx	31	Descriptor Done by HW for Transmit DMA This bit is set by the HW to indicate that the

Field In Descriptor	Sub-fields	Bit #	Remarks
			descriptor has been handled by the HW. This bit may be set by SW in some conditions, to indicate that this descriptor (and maybe other descriptors) should be skipped by HW, and the MPDUs associated with the skipped descriptors should not be transmitted. Refer to section *2.2.3.4*, *Retransmitting unsuccessful MPDUs of an A-MPDU* for details about how this field is used by SW. The HW behavior related to this field is described in section *2.2.3.5*, *Transmit DMA channel HW procedure*.
Status Information	descriptorDoneSWTx	30	Descriptor Done by SW for Transmit DMA This bit is not set by HW. It is set and used by SW.
Status Information	whichDescriptorSW	29:26	Which Descriptor for SW Indicates what kind of a descriptor this is. This field is set by HW = {*underBASetup*, *aMPDU*, *whichDescriptor*[1:0]} when the *descriptorDoneHWTx* bit is set. It as described in detail in the table below.

The following table indicates the meaning of whichDescriptorSW.

underBASetup	aMPDU	whichDescriptor[1:0]	Indicates
0	0	00	4'b0000: Only 1 THD possible, describing an unfragmented MSDU under Imm-ACK or No-ACK policy
1	0	00	4'b1000: Only 1 THD possible, describing an unfragmented MSDU under BA policy
0	0	01	4'b0001: First THD describing the first MPDU of a fragmented MSDU under Imm-ACK or No-ACK policy
0	0	10	4'b0010: Intermediate THD describing intermediate MPDUs of a fragmented MSDU under Imm-ACK or No-ACK policy
0	0	11	4'b0011: Last THD describing the last MPDU of a fragmented MSDU under Imm-ACK or No-ACK policy
1	0	01	Reserved
1	0	10	Reserved
1	0	11	Reserved
0	1	00	4'b0100: "Extra" THD describing an A-MPDU under No-ACK policy
0	1	01	4'b0101: First THD describing the first MPDU of an A-MPDU under No-ACK policy

underBASetup	aMPDU	whichDescriptor[1:0]	Indicates
0	1	10	4'b0110: Intermediate THD describing intermediate MPDUs of an A-MPDU under No-ACK policy
0	1	11	4'b0111: Last THD describing the last MPDU of an A-MPDU under No-ACK policy
1	1	00	4'b1100: "Extra" THD describing an A-MPDU under BA policy
1	1	01	4'b1101: First THD describing the first MPDU of an A-MPDU under BA policy
1	1	10	4'b1110: Intermediate THD describing intermediate MPDUs of an A-MPDU under BA policy
1	1	11	4'b1111: Last THD describing the last MPDU of an A-MPDU under BA policy

Field In Descriptor	Sub-fields	Bit #	Remarks
Status Information	transmissionBW	25:24	Transmission BandWidth This bit is set by HW to indicate whether the frame was transmitted with 20MHz, 40MHz, 80MHz or 160MHz channel BW. 00 : 20MHz 01 : 40 MHz 10 : 80MHz 11 : 160MHz
Status Information	frmSuccessfulTx	23	Frame Successful for Transmit DMA This bit is set by the HW when any expected acknowledgement is received successfully.
Status Information	Reserved	22:19	Reserved
Status Information	baFrameReceived	18	BlockAck received indication This bit is set by SW when sending confirmation to Host, depending on frameSuccessfulTx bit value in AMPDU THD
Status Information	lifetimeExpired	17	Frame Unsuccessful – Frame Lifetime Expired This bit is set by the HW when a frame fails the lifetime check in HW.
Status Information	retryLimitReached	16	Frame Unsuccessful – Retry Limit Reached This bit is set by the HW when a non-A-MPDU reaches retry limit and is discarded. This bit cannot be set for an A-MPDU since MPDUs transmitted in an A-MPDU are not subject to retry counter and retry limit reached discarding[13].

[13] In any case, A-MPDU retries are performed from SW.

Field In Descriptor	Sub-fields	Bit #	Remarks
Status Information	numMPDURetries	15:8	Number of MPDU Retries Indicates the number of retries that the non-A-MPDU suffered before getting a successful response. This field does not include the internal collision count. This is used to provide inputs to the LAA, and for HW to keep track of how many retries have occurred previously[14]. This field is always zero for frames transmitted under No-ACK or Block-ACK policy, including an A-MPDU.
Status Information	numRTSRetries	7:0	Number of RTS Retries Indicates the number of retries that the RTS frame, that was transmitted before transmitting this frame, suffered before getting a successful response.

The *Medium Time Used* field is valid for the SU-MIMO A-MPDU Header Descriptor and for singleton MPDUs Header Descriptor.

Field In Descriptor	Sub-fields	Bit #	Remarks
Medium Time Used		31:0	Medium Time Used Accumulated transmission time of the AMPDU or Singleton MDPU including SIFS, ACK or BA, RTS/CTS, retries if any. The medium time used is provided in unit of 32us.

3.1.4 Transmit DMA Buffer Descriptor fields

Field In Descriptor	Sub-fields	Bit #	Remarks
uPatternTx		31:0	Unique Pattern for Transmit Buffer DMA If the HW does not find the pattern 0xCAFEFADE in this field, it raises the error interrupt *General Interrupt Event Register (genIntEventReg).txDMAError*. Refer to section *9.3, Terminal DMA errors* for details.

[14] This is possible only in a special case, where the TXOP ends before all the retransmissions have been performed. If the frame that was undergoing retry is removed from the HW queue and brought back later, the HW looks at this field to determine how many retries have already been performed.

Field In Descriptor	Sub-fields	Bit #	Remarks
Next Payload Buffer Descriptor Pointer	nextPBufDPTx	31:0	Next Payload Buffer Descriptor Pointer for Transmit DMA This field points to the next Buffer Descriptor of the MPDU when the MPDU is split across multiple buffers. It is a quadlet aligned field.

Field In Descriptor	Sub-fields	Bit #	Remarks
Buffer Control	bufferDoneHWTx	31	Buffer Done by HW for Transmit DMA This bit is set by the HW to indicate that the descriptor has been handled by the HW.
Buffer Control	bufferDoneSWTx	30	Buffer Done by SW for Transmit DMA This bit is not set by HW. It is set and used by SW.
Buffer Control	reserved	29:1	*Reserved for future use.*
Buffer Control	bufferInterruptEnTx	0	Interrupt Enable for Transmit Buffer If this bit is set, HW will raise an interrupt as soon as this buffer has been fully transmitted.

3.2 Receive DMA

3.2.1 Receive DMA Header Descriptor

A Receive Header Descriptor contains the following fields:

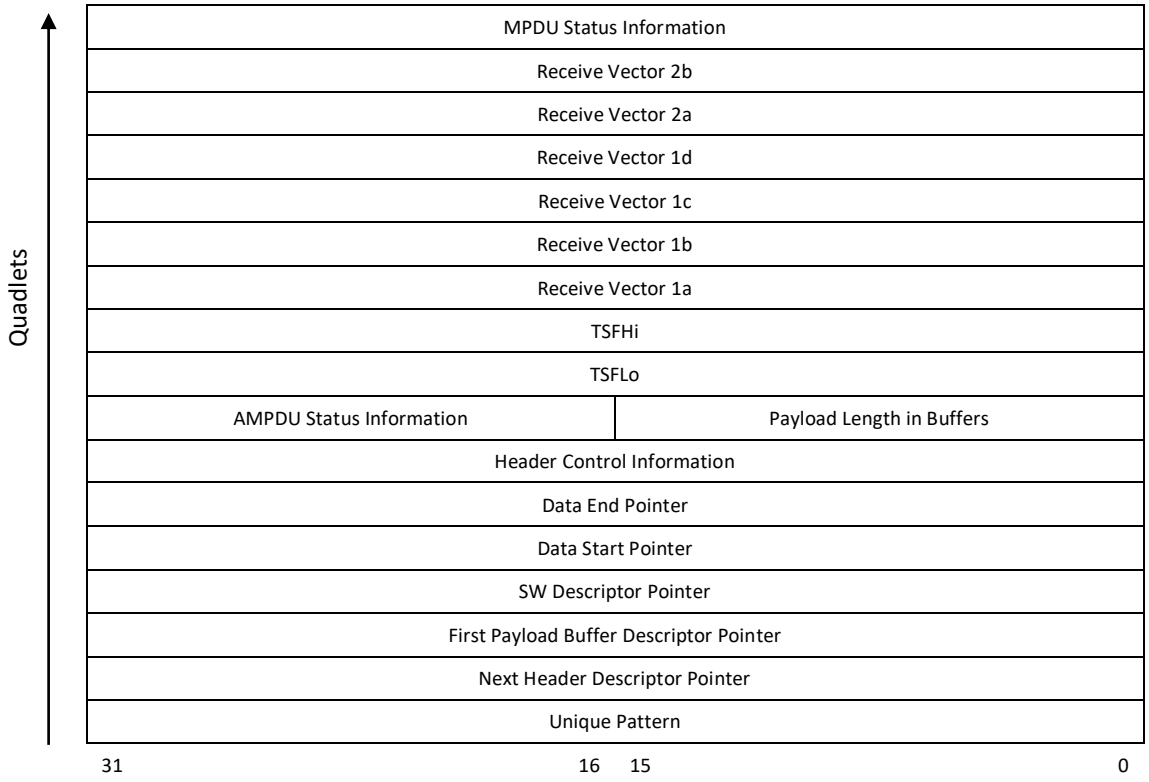


Figure 19: Receive Header Descriptor

3.2.2 Receive DMA Payload Descriptor

A Receive Payload Descriptor contains the following fields:

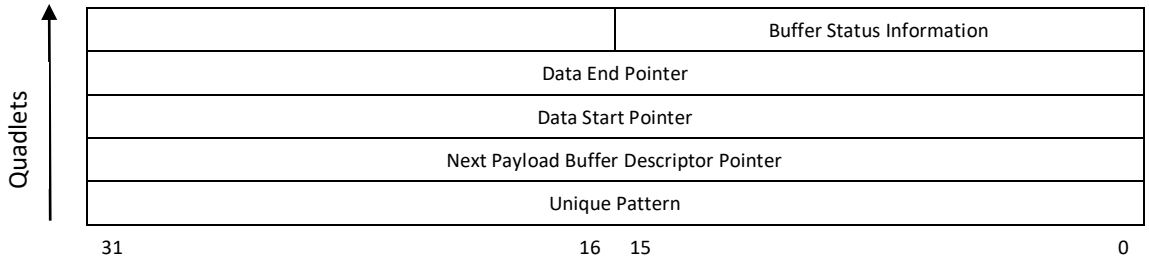


Figure 20: Receive Payload Descriptor

3.2.3 Receive DMA Header Descriptor fields

Field In Descriptor	Sub-fields	Bit #	Remarks
uPatternRx		31:0	Unique Pattern for Receive DMA If the HW does not find the pattern 0xBAADF00D in this field, it raises an error interrupt *General Interrupt Event Register (genIntEventReg).rxDMAError*. Refer to section *9.3*, *Terminal DMA errors* for details

Field In Descriptor	Sub-fields	Bit #	Remarks
Next Header Descriptor Pointer	nextHDPRx	31:0	Next Header Descriptor Pointer for Receive DMA This field points to the location of the next Header Descriptor. It is a quadlet aligned field. If the HW finds an invalid address programmed, it raises the raises the error interrupt *General Interrupt Event Register (genIntEventReg).txDMAError*. Refer to section *9.3*, *Terminal DMA errors* for details.

Field In Descriptor	Sub-fields	Bit #	Remarks
firstPBufDPRx		31:0	First Payload Buffer Descriptor Pointer for Receive DMA This field is null when the descriptor is prepared in SW. The HW updates this field to the address of the first Buffer Descriptor in which the payload of the frame has been moved.

Field In Descriptor	Sub-fields	Bit #	Remarks
swDescPRx		31:0	SW descriptor Pointer for Receive DMA This field points to the start of the Software descriptorr associated with this descriptor. It is a quadlet aligned field. It is not used by the HW. It is used by SW.

Field In Descriptor	Sub-fields	Bit #	Remarks
dataStartPRx		31:0	Data Start Pointer for Receive DMA This field is quadlet aligned and points to the address in the buffer where the first byte has been written in the memory.

Field In Descriptor	Sub-fields	Bit #	Remarks
dataEndPRx		32	Data End Pointer for Receive DMA This field points to the last valid byte in the memory.

Field In Descriptor	Sub-fields	Bit #	Remarks
Header Control Information	Reserved	31:1	Reserved
Header Control Information	interruptEnRx	0	Interrupt Enable for Receive DMA HW will raise an interrupt when it updates the status in this Header Descriptor. This interrupt provides a context to SW to perform any activities for the queue, like descriptor book-keeping, chained MPDUs, starting duplicate detection, etc.

Field In Descriptor	Sub-fields	Bit #	Remarks
AMPDU Status Information	AMPDUCnt	31:30	A-MPDU Counter This field provides the number of A-MPDU received modulo 4. It can be used by the SW to differenciate the MPDU received from different A-MPDU
AMPDU Status Information	MPDUCnt	29:24	MPDU Counter inside the A-MPDU This field indicates the position of the current MPDU inside the A-MPDU.
AMPDU Status Information	*Reserved*	23:16	
payloadLengthRx		15:0	Payload Length for Receive DMA This field indicates the length of the received MPDU, including MAC Header, Security Header & Footer and FCS

The Receive Vector 1 field of the constituent MPDUs of an A-MPDU carries the same information. Note that the size of Receive Vector 1 depends on the Rx FormatMod. If the Receive Vector 1 is shorter than 16bytes, the remaining bytes are padded with 0.

Field In Descriptor	Sub-fields	Bit #	Remarks
Receive Vector 1a		31:0	Receive Vector 1a Contains the bytes 3 – 0 of Receive Vector 1 as defined in *[3]*.

Field In Descriptor	Sub-fields	Bit #	Remarks

Field In Descriptor	Sub-fields	Bit #	Remarks
Receive Vector 1b		31:0	Receive Vector 1b Contains the bytes 7 – 4 of Receive Vector 1 as defined in *[3]*.

Field In Descriptor	Sub-fields	Bit #	Remarks
Receive Vector 1c		31:0	Receive Vector 1c Contains the bytes 11 – 8 of Receive Vector 1 as defined in *[3]*.

Field In Descriptor	Sub-fields	Bit #	Remarks
Receive Vector 1d		31:0	Receive Vector 1d Contains the bytes 14 – 12 of Receive Vector 1 as defined in *[3]*.

The Receive Vector 2 field of all but the last constituent MPDUs of an A-MPDU are reserved. Only the last MPDU of an A-MPDU carries valid information since the Receive Vector 2 information is received after the end of the PPDU (A-MPDU). Since the SW does not know that a particular MPDU is the last MPDU of the A-MPDU, the indication that the Receive Vector 2 field is valid is provided in the *MPDU Status Information* field.

Field In Descriptor	Sub-fields	Bit #	Remarks
Receive Vector 2a		31:0	Receive Vector 2 Contains the bytes 3 – 0 of Receive Vector 2 as defined in *[3]*.

Field In Descriptor	Sub-fields	Bit #	Remarks
Receive Vector 2b		31:0	Receive Vector 2 Contains the bytes 7 – 4 of Receive Vector 2 as defined in *[3]*.

Field In Descriptor	Sub-fields	Bit #	Remarks
TSF Lo		31:0	TSF Low Contains the lower 4 bytes of the timestamp (TSF) at which the frame ended on air.

Field In Descriptor	Sub-fields	Bit #	Remarks
TSF Hi		31:0	TSF High Contains the higher 4 bytes of the timestamp (TSF) at which the frame ended on air.

Field In Descriptor	Sub-fields	Bit#	Remarks

Field In Descriptor	Sub-fields	Bit#	Remarks
MPDU Status Information	subtype	31:28	Subtype Indicates the *Subtype* field of the *Frame Control* field of the MPDU.
MPDU Status Information	type	27:26	Type Indicates the *Type* field of the *Frame Control* field of the MPDU.
MPDU Status Information	keySRamIndexV	25	Key Storage RAM Index Valid This field indicates if the *keySRAMIndex* field is valid.
MPDU Status Information	keySRamIndex	24:15	Key Storage RAM Index It is the index of the HW Key Storage RAM where the TA of the received frame was found.
MPDU Status Information	descriptorDoneRH	14	Descriptor Done for Receive Header This bit is set by the HW to indicate that the Receive Header Descriptor has been handled by the HW.
MPDU Status Information	frmSuccessfulRx	13	Frame Successful for Receive DMA This bit is set by the HW when the frame has been successfully received. If it is not set, then one of the Frame Unsuccessful status bits should be set, and/or the *decrStatus* field should indicate decryption failure.
MPDU Status Information	currentAC	12:11	It is the current Access Categorie. This information is valide only in case of immediate response frame (ACK or BA)
MPDU Status Information	gaFrame	10	Group Addressed Frame This bit is set by the HW when this frame contains a broadcast or multicast address.
MPDU Status Information	addrMismatch	9	Address Mismatch This bit is set by the HW when this frame is not meant for this device but is still being passed to SW.
MPDU Status Information	fcsError	8	Frame Unsuccessful - FCS Error This bit is set by the HW when the frame has an FCS error.
MPDU Status Information	undefError	7	Frame Unsuccessful – Undefined Error This bit is set by the HW when the frame had some other error than FCS or Decryption error, like a protocol error, rxError.
MPDU Status Information	decryptionError	6	Indicate a decryption error (MIC Failure, ICV error, Null Key Found, ...)

Field In Descriptor	Sub-fields	Bit#	Remarks
MPDU Status Information	decryptionType	5:2	Decryption Type : Indicates the decryptio type. The bits are encoded as follows: 4'b0000: Frame was not encrypted 4'b0001: WEP 4'b0010: TKIP 4'b0011: CCMP-128 4'b0100: CCMP-256 4'b0101: GCMP-128 4'b0110: GCMP-256 4'b0111: WAPI 4'b1111: Null Key Found
MPDU Status Information	respFrame	1	Response Frame This bit is set by the HW if the frame is the immediate response of a frame transmitted by the device.
MPDU Status Information	rxVector2Valid	0	Receive Vector 2 Valid This bit is set by the HW if the Receive Vector 2 field of the MPDU contains valid data. It is set only for the last MPDU of the constituent MPDUs of an A-MPDU since only that MPDU carries the trailing RxVector received after the A-MPDU.

3.2.4 Receive DMA Payload Descriptor fields

Field In Descriptor	Sub-fields	Bit #	Remarks
uPatternRx		31:0	Unique Pattern for Receive DMA If the HW does not find the pattern 0xC0DEDBAD in this field, it raises an error interrupt *General Interrupt Event Register (genIntEventReg).rxDMAError*. Refer to section *9.3*, *Terminal DMA errors* for details

Field In Descriptor	Sub-fields	Bit#	Remarks
Next Payload Buffer Descriptor Pointer	nextPBufDPRx	31:0	Next Payload Buffer Descriptor Pointer for Receive DMA This field points to the next payload buffer in the list. If the HW finds an invalid address programmed, it raises the error interrupt *General Interrupt Event Register (genIntEventReg).txDMAError*. Refer to section *9.3*, *Terminal DMA errors* for details.

Field In Descriptor	Sub-fields	Bit #	Remarks
dataStartPRx		31:0	Data Start Pointer for Receive DMA This field is quadlet aligned and points to the address in the buffer where the first byte has been written in the memory.

Field In Descriptor	Sub-fields	Bit #	Remarks
dataEndPRx		32	Data End Pointer for Receive DMA This field points to the last valid byte in the memory.

Field In Descriptor	Sub-fields	Bit #	Remarks
bufferStatusRx	lastBuffer	0	Last Buffer The HW sets this bit when this is the last buffer of the MPDU.
bufferStatusRx	descriptorDoneRP	1	Descriptor Done for Receive Payload This bit is set by the HW to indicate that the Receive Payload Descriptor has been handled by the HW. It is used by the HW and no functionality in SW must be based on this bit.
bufferStatusRx	Reserved	15:2	Reserved

4 Policy table

4.1 Operation

The Policy Table is maintained in system memory and contains static parameters which control the transmission. The entries of the Policy Table may be maintained as a contiguous block of memory, or at separate locations. The HW is provided the *polEntryAddress* which is the 32-bit address in local memory where the entry for the current frame can be found. Each entry of the policy table is written by the SW in some system memory location and is fetched by the HW after the Header Descriptor is fetched. Multiple RA/TID queues can use the same Policy Table entry if all the parameters match. Each entry is organized as shown in *4.2 Policy Table entry*.

4.2 Policy Table entry

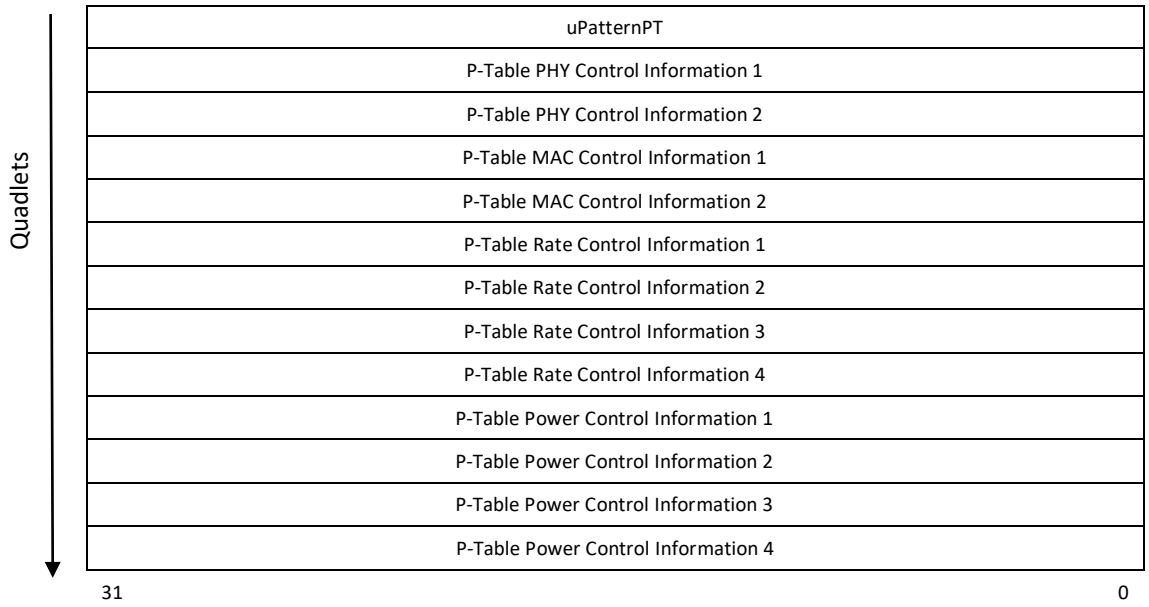


Figure 21: Policy Table entry

4.2.1 Policy Table fields

Field In Descriptor	Sub-fields	Bit #	Remarks
uPatternPT		31:0	Unique Pattern for Policy Table If the HW does not find the pattern 0xBADCAB1E in this field, it raises an error interrupt *General Interrupt Event Register (genIntEventReg).ptError* interrupt.

Field In Descriptor	Sub-fields	Bit #	Remarks
PHY Control Information 1	*Reserved*	31:28	Reserved

PHY Control Information 1	spatialReusePT	27:24	Spatial Reuse Information Indicates the Spatial Reuse parameter value to be used for transmitting the PPDU
PHY Control Information 1	*Reserved*	23:22	Reserved
PHY Control Information 1	midamblePT	21	Midamble Periodicity Indicates the midamble periodicity in number of OFDM symbols in the Data field. 1'b0: Periodicity of 10 symbols 1'b1: Periodicity of 20 symbols
PHY Control Information 1	dopplerPT	20	Doppler Indicates whether the doppler parameter value to be used for transmitting the PPDU
PHY Control Information 1	nTxProtPT	19:17	Number of Transmit Chains for Protection Frame Indicates the number of transmit chains to be used for transmitting the protection frame by the HW.
PHY Control Information 1	nTxPT	16:14	Number of Transmit Chains for PPDU Indicates the number of transmit chains to be used for transmitting the PPDU.
PHY Control Information 1	*Reserved*	13:9	Reserved
PHY Control Information 1	stbcPT	8:7	Space Time Block Coding Indicates the difference between the number of space time streams and the number of spatial streams indicated by the MCS. It is passed to the PHY in the Tx-Vector.
PHY Control Information 1	fecCodingPT	6	FEC Coding Indicates whether LDPC encoding is used for the transmission of the PPDU. It is not used by the MAC. It is passed to the PHY in the Tx-Vector.
PHY Control Information 1	numExtnSSPT	5:4	Number of Extension Spatial Streams Indicates the number of extension spatial streams that are sounded during the extension part of the HT training. It is passed to the PHY in the Tx-Vector.
PHY Control Information 1	bfFrmExPT	3	Beam Forming Frame Exchange It is used by the MAC to decide the kind of frame exchange that should precede the frame transmission to solicit a sounding frame to be used for I-BF. This bit is encoded as follows: 1'b0: QoS Null/ACK 1'b1: RTS/CTS Reserved for future use.

PHY Control Information 1	smoothingProtTxPT	2	Smoothing recommended for Protection 1'b1: Smoothing recommended as part of channel estimation 1'b0: Smoothing not recommended as part of channel estimation
PHY Control Information 1	smoothingTxPT	1	Smoothing recommended for PPDU 1'b1: Smoothing recommended as part of channel estimation 1'b0: Smoothing not recommended as part of channel estimation
PHY Control Information 1	soundingTxPT	0	Indicates whether this PPDU is Sounding 1'b0: NOT_SOUNDING 1'b1: SOUNDING

Field In Descriptor	Sub-fields	Bit #	Remarks
PHY Control Information 2	*Reserved*	31:29	*Reserved*
PHY Control Information 2	packetExtensionPT	28:26	Packet Extension Value Indicate the packet extension value to be used in HE PPDU Header.
PHY Control Information 2	bssColorPT	25:20	BSS Color Indicate the BSS color to be used in HE PPDU Header.
PHY Control Information 2	*Reserved*	19	*Reserved*
PHY Control Information 2	upLinkFlagPT	18	UP Link Flag Set to 1 if the HE PPDU is addressed to an AP. Set to 0 otherwise
PHY Control Information 2	beamChangePT	17	Beam Change Set to 1 to indicate that the pre-HE-STF portion of the PPDU is spatially mapped differently from HE-LTF1. Set to 0 to indicate that the pre-HE-STF portion of the PPDU is spatially mapped the same way as HE-LTF1 on each tone.
PHY Control Information 2	beamFormedPT	16	BeamFormed Set when the transmission is beamFormed
PHY Control Information 2	smmIndexPT	15:8	Spatial Map Matrix Index Selects the index of the SMM table in the PHY and indicates to the PHY to apply a particular kind of Spatial Multiplexing on the frame. It is not used by the MAC. It is passed to the PHY in the Tx-Vector.

PHY Control Information 2	antennaSetPT	7:0	Antenna Set Indicates which antennas of the available antennas are used for the transmission. It is not used by the MAC. It is passed to the PHY in the Tx-Vector.

Field In Descriptor	Sub-fields	Bit #	Remarks
MAC Control Information 1	*Reserved*	31:20	Reserved
MAC Control Information 1	keySRamIndexRA	19:10	Key Storage RAM Index for Receiver Address It is used to look up the RA from the HW Key Storage RAM to transmit in an RTS or CTS frame when such a frame is created by HW.
MAC Control Information 1	keySRamIndex	9:0	Key Storage RAM Index It is used by the encryption engine to extract the encryption key from the HW Key Storage RAM.

Field In Descriptor	Sub-fields	Bit #	Remarks
MAC Control Information 2	*Reserved*	31:28	Reserved
MAC Control Information 2	rtsThreshold	27:16	RTS Threshold The RTS Threshold is used to identify if a frame is long or short.
MAC Control Information 2	shortRetryLimit	15:8	Short Retry Limit Frames shorter than the RTS Threshold are subject to the Short Retry Limit.
MAC Control Information 2	longRetryLimit	7:0	Long Retry Limit Frames longer than the RTS Threshold are subject to the Long Retry Limit.

There are three fields named Rate Control 1-4 which allows the sotware to select the different parameters used for each retry. That provides a lot of flexibility and offers the possibility to map any rate adaptation algorithms.

Field In Descriptor	Sub-fields	Bit #	Remarks
Rate Control X	nRetryRCX	31:29	Number of trial which can be done using this Rate Control 1

Rate Control X	formatModProtTxRCX	28:26	Format and Modulation of Protection frame for Transmission This field indicates the format and the modulation of the protection frame created by HW. The encoding is as follows: 3'b000: NON-HT 3'b001: NON-HT-DUP-OFDM 3'b010: HT-MF 3'b011: HT-GF 3'b100: VHT 3'b101: HE-SU 3'b110: *Reserved* 3'b111: HE-EXT
Rate Control X	bwProtTxRCX	25:24	Band Width of Protection frame for Transmission. If *formatMod* = HT-MF or HT-GF, this field is encoded as follows: 2'b00: HT_CBW20 - 20 MHz 2'b01: HT_CBW40 - 40 MHz If *formatMod* = VHT or HE, this field is encoded as follows: 2'b00: VHT_CBW20 - 20 MHz 2'b01: VHT_CBW40 - 40 MHz 2'b10: VHT_CBW80 - 80 MHz 2'b11: VHT_CBW160 or VHT_CBW80p80 - 160 or 80+80 MHz For other values of *formatMod*, this field is encoded as follows: 2'b00: NON_HT_CBW20 for all other non-HT modulations 2'b01: NON_HT_CBW40 for non-HT duplicate 2'b10: NON_HT_CBW80 for non-HT duplicate 2'b11: NON_HT_CBW160 for non-HT duplicate

Rate Control X	mcsIndexProtTxRCX	23:17	MCS Index of Protection frame for Transmission This field indicates the rate at which the protection frame (RTS or CTS) created by HW is transmitted. The encoding is as follows: 1 Mbps: 7'd0 2 Mbps: 7'd1 5.5 Mbps: 7'd2 11 Mbps: 7'd3 6 Mbps: 7'd4 9 Mbps: 7'd5 12 Mbps: 7'd6 18 Mbps: 7'd7 24 Mbps: 7'd8 36 Mbps: 7'd9 48 Mbps: 7'd10 54 Mbps: 7'd11 HT rates: 7'd*MCS Index* VHT rates: {3'd*nSS*, 4'd*MCS index*} HE rates: {3'd*nSS*, 4'd*MCS index*} Note that *nSS* is the number of spatial stream minus 1
Rate Control X	navProtFrmExRCX	16:14	NAV Protection Frame Exchange This field indicates the procedure for frame exchange that is performed by the HW before transmitting the frame. The bits are encoded as follows: 3'b000: No protection 3'b001: Self-CTS 3'b010: RTS/CTS with intended receiver of this PPDU 3'b011: RTS/CTS with QAP 3'b100: STBC protection ✓ As a non-AP STA: HW performs RTS/Dual-CTS protection with AP ✓ As an AP: HW performs CTS protection with intended receiver of this PPDU SW must ensure that it does not chain a RTS frame in the linked list before the PPDU that is enabled for HW formed RTS. SW must ensure that it does not enable protection for a SW formed protection frame.
Rate Control X	formatModTxRCX	13:11	Format and Modulation of PPDU for Transmission This field indicates the format and the modulation of the PPDU. The encoding is the same as *formatModProtTxRCX*.

Rate Control X	giTypeTxRCX/ preTypeTxRCX	10:9	Guard Interval of PPDU and Preamble Type for Transmission In case of DSSS/CCK transmission, indicates the Preamble type of the PPDU: 2'b00: SHORT Preamble 2'b10: LONG Preamble In case of HT-MM, HT-GF or VHT, indicates whether a short Guard Interval (GI) is used in the transmission. 2'b00: Long GI 2'b01: Short GI In case of HE, indicate the GI Type of the transmission: 2'b00: 0.8 us 2'b01: 1.6 us 2'b10: 3.2 us It is reserved in all other conditions.
Rate Control X	bwTxRCX	8:7	Band Width of PPDU for Transmission The encoding is the same as *bwProtTxRCX*.
Rate Control X	mcsIndexTxRCX	6:0	MCS Index of PPDU for Transmission The encoding is the same as *mcsIndexProtTxRCX*.

Field In Descriptor	Sub-fields	Bit #	Remarks
Power Control Information *X*	*Reserved*	31:19	*Reserved*
Power Control Information *X*	dcmPT	18	Dual Carrier Modulation Set to 1 to indicate that dual carrier modulation is used for the HE-Data field. Set to 0 to indicate that dual carrier modulation is not used for the HE-Data field.
Power Control Information *X*	heLTFTypePT	17:16	HE-LTF Type Indicates the type of HE-LTF used during the transmission of the PPDU when the Rate Control *X* is used. 2'b00: 1x HE-LTF for 3.2 µs 2'b01: 2x HE-LTF for 6.4 µs 2'b10: 4x HE-LTF for 12.8 µs 2'b11: *Reserved*

Power Control Information *X*	txPwrLevelProtPTRC*X*	15:8	Transmit Power Level of Protection for RC*X* Indicates the power level used to transmit the protection when the Rate Control *X* is used. It is not used by the MAC. It is passed to the PHY in the Tx-Vector. Its 2's Complement and codded as follow : 8'h80 : -128 dBm 8'hFF : -1 dBm 8'h00 : 0 dBm 8'h01 : 1 dBm 8'h3F : 127dBm
Power Control Information *X*	txPwrLevelPTRC*X*	7:0	Transmit Power Level for RC*X* Indicates the power level used to transmit the PPDU when the Rate Control *X* is used. It is not used by the MAC. It is passed to the PHY in the Tx-Vector. Its 2's Complement and codded as follow : 8'h80 : -128 dBm 8'hFF : -1 dBm 8'h00 : 0 dBm 8'h01 : 1 dBm 8'h3F : 127dBm

5 MIB Table

5.1 Operation

A MIB table is maintained in a HW embedded RAM. Each counter is 32 bits wide. The MIB table is maintained in HW to prevent the per-frame updation of some counters by SW, thus reducing the SW operations and the MIPS. The MIB table memory is directly addressable on the Platform Interface bus from SW for reading. The MIB table memory is accessible via a window register (*MIB Table Write Register (mibTableWriteReg)*) for incrementing or overwriting a MIB entry value. MIB entry values are also updated by HW.

When the field *MAC Control 1 Register (macCntrl1Reg)*.*mibTableReset* is set from SW, the core initializes the MIB Table with zeros in all the fields for all the locations.

5.2 MIB Table organization

Index	Address Range	MIB Name	Purpose
Basic MIB set			
0	0x0800	dot11WEPExcludedCount	Counts the number of unencrypted frames that have been discarded.
1	0x0804	dot11FCSErrorCount	Counts the receive FCS errors.
2	0x0808	rwRxPHYErrorCount	Counts the number of PHY Errors reported during a receive transaction.
3	0x080C	rwRxFIFOOverflowCount	Counts the number of times the Receive FIFO has overflowed.
4	0x0810	rwTxUnderrunCount	Counts the number of times underrun has occurred on the Transmit side, either detected by MAC or PHY as described in section *9.2*, Non-terminal hardware errors.
5	0x0814	rwRxMPIFOverflowCount	Counts the number of times an overflow occurrs at the MPIF FIFO when receiving a frame from the PHY.
6 – 11	0x0818 - 0x082C	Reserved	Reserved for future use.
EDCA MIB set, maintained per TID, arranged as TID 0 to 7, with the lower address corresponding to TID 0			
12 - 19	0x0830 – 0x084C	rwQosUTransmittedMPDUCount	Counts the number of unicast MPDUs, not containing an A-MSDU, that were transmitted successfully without using BA policy. Note that statistics for frames transmitted under BA are maintained in SW.

Index	Address Range	MIB Name	Purpose
20 – 27	0x0850 – 0x086C	rwQoSGTransmittedMPDUCount	Counts the number of group-addressed MPDUs, not containing an A-MSDU, that were transmitted successfully.
28 – 35	0x0870 – 0x088C	dot11QosFailedCount	Counts the number of MSDUs that were discarded because of retry-limit-reached condition. Non QoS MSDUs transmitted under AC_BK increment the TID1 counter. Non QoS MSDUs transmitted under AC_BE increment the TID0 counter. Non QoS MSDUs transmitted under AC_VI increment the TID4 counter. Non QoS MSDUs transmitted under AC_VO increment the TID6 counter.
36 – 43	0x0890 – 0x08AC	dot11QosRetryCount	Counts the number of unfragmented MSDUs that were transmitted successfully after 1 or more retransmissions without using BA policy. Non QoS MSDUs transmitted under AC_BK increment the TID1 counter. Non QoS MSDUs transmitted under AC_BE increment the TID0 counter. Non QoS MSDUs transmitted under AC_VI increment the TID4 counter. Non QoS MSDUs transmitted under AC_VO increment the TID6 counter. Note that statistics for fragmented MSDUs and fragmented MMPDUs are maintained in SW. Also note that statistics for frames transmitted under BA are maintained in SW.
44 – 51	0x08B0 – 0x08CC	dot11QosRTSSuccessCount	Counts the number of successful RTS frame transmissions.
52 – 59	0x08D0 – 0x08EC	dot11QosRTSFailureCount	Counts the number of unsuccessful RTS frame transmissions

Index	Address Range	MIB Name	Purpose
60 – 67	0x08F0 – 0x090C	rwQosACKFailureCount	Counts the number of MPDUs, not containing an A-MSDU, that did not receive an ACK frame successfully in response. Note that statistics for frames transmitted under BA are maintained in SW.
68 – 75	0x0910 – 0x092C	rwQosUReceivedMPDUCount	Counts the number of unicast MPDUs, not containing an A-MSDU, received successfully, destined to this device.
76 – 83	0x0930 – 0x094C	rwQosGReceivedMPDUCount	Counts the number of group-addressed MPDUs, not containing an A-MSDU, received successfully.
84 – 91	0x0950 – 0x096C	rwQoSUReceivedOtherMPDU	Counts the number of unicast MPDUs, not containing an A-MSDU, received successfully, not destined to this device.
92 – 99	0x0970 – 0x098C	dot11QosRetriesReceivedCount	Counts the number of MPDUs that are received with the retry bit set.

Index	Address Range	MIB Name	Purpose
A-MSDU MIB set, maintained per TID, arranged as TID 0 to 7, with the lower address corresponding to TID 0			
100 – 107	0x0990 – 0x09AC	rwUTransmittedAMSDUCount	Counts the number of unicast A-MSDUs that were transmitted successfully without using BA policy. Note that statistics for frames transmitted under BA is maintained in SW.
108 - 115	0x09B0 – 0x09CC	rwGTransmittedAMSDUCount	Counts the number of group-addressed A-MSDUs that were transmitted successfully.
116 – 123	0x09D0 – 0x09EC	dot11FailedAMSDUCount	Counts the number of A-MSDUs that were discarded because of retry-limit-reached condition.
124 – 131	0x09F0 – 0x0A0C	dot11RetryAMSDUCount	Counts the number of A-MSDUs that were transmitted successfully after 1 or more retransmissions without using BA policy. Note that statistics for frames transmitted under BA are maintained in SW.

Index	Address Range	MIB Name	Purpose
132 – 139	0x0A10 – 0x0A2C	dot11TransmittedOctetsInAMSDU	Counts the number of bytes in the frame body of an A-MSDU that was transmitted successfully without using BA policy. Note that statistics for frames transmitted under BA is maintained in SW.
140 – 147	0x0A30 – 0x0A4C	dot11AMSDUAckFailureCount	Counts the number of A-MSDUs that did not receive an ACK frame successfully in response. Note that statistics for frames transmitted under BA are maintained in SW.
148 – 155	0x0A50 – 0x0A6C	rwUReceivedAMSDUCount	Counts the number of unicast A-MSDUs received successfully, destined to this device.
156 – 163	0x0A70 – 0x0A8C	rwGReceivedAMSDUCount	Counts the number of group-addressed A-MSDUs received successfully.
164 – 171	0x0A90 – 0x0AAC	rwUReceivedOtherAMSDU	Counts the number of unicast A-MSDUs received successfully, not destined to this device.
172 - 179	0x0AB0 – 0x0ACC	dot11ReceivedOctetsInAMSDUCount	Counts the number of bytes in the frame body of an A-MSDU that was received successfully.
Trigger Based MIB set			
180	0x0AD0	mibrwHETBBasicCount	Count the number of Basic frame sent in response to a basic trigger frame.
181	0x0AD4	mibrwHETBBFRCount	Count the number of Beamforming Report sent in response to a Beamforming Report Poll trigger frame.
182	0x0AD8	mibrwHETBMUBACount	Count the number of MU-BA sent in response to a MU-BAR trigger frame.
183	0x0ADC	mibrwHETBMUCTSCount	Count the number of MU-CTS sent in response to a MU-RTS trigger frame.
184	0x0AE0	mibrwHETBBSRCount	Count the number of Buffer Status Report sent in response to a Buffer Status Report Poll trigger frame.
185	0x0AE4	mibrwHETBGCRMUBACount	Count the number of GCR MU-BA sent in response to a GCR MU-BAR trigger frame.

Index	Address Range	MIB Name	Purpose
186	0x0AE8	mibrwHETBBQRCount	Count the number of Bandwidth Query Report sent in response to a Bandwidth Query Report Poll trigger frame.
187	0x0AEC	mibrwHETBNFRCount	Count the number of NDP Feedback Report sent in response to a NDP Feedback Report Poll trigger frame.
188-203	0x0AF0-0B2C	Reserved	Reserved for future use
A-MPDU MIB set			
204	0x0B30	dot11TransmittedAMPDUCount	Counts the number of A-MPDUs that were transmitted.
205	0x0B34	dot11TransmittedMPDUsInAMPDUCount	Counts the number of MPDUs that were transmitted in an A-MPDU.
206	0x0B38	dot11TransmittedOctetsInAMPDUCount	Counts the number of bytes in a transmitted A-MPDU.
207	0x0B3C	rwUAMPDUReceivedCount	Counts the number of unicast A-MPDUs received, destined to this device.
208	0x0B40	rwGAMPDUReceivedCount	Counts the number of group-addressed A-MPDUs received.
209	0x0B44	rwOtherAMPDUReceivedCount	Counts the number of unicast A-MPDUs received, not destined to this device.
210	0x0B48	dot11MPDUInReceivedAMPDUCount	Counts the number of MPDUs that were received in an A-MPDU.
211	0x0B4C	dot11ReceivedOctetsInAMPDUCount	Counts the number of bytes received in an A-MPDU.
212	0x0B50	dot11AMPDUDelimiterCRCErrorCount	Counts the number of CRC errors in MPDU Delimiter of an A-MPDU. An immediately repeated CRC error within an A-MPDU is not counted.
213	0x0B54	dot11ImplicitBARFailureCount	Counts the number of Implicit BAR frames that did not receive the BA frame successfully in response.
214	0x0B58	dot11ExplicitBARFailureCount	Counts the number of Explicit BAR frames that did not receive the BA frame successfully in response.
215 – 219	0x0B5C – 0x0B6C	Reserved	Reserved for future use

Index	Address Range	MIB Name	Purpose
20/40/80/160 MHz MIB set			
220	0x0B70	dot1120MHzFrameTransmittedCount	Counts the number of frames transmitted at 20 MHz BW.
221	0x0B74	dot1140MHzFrameTransmittedCount	Counts the number of frames transmitted at 40 MHz BW.
222	0x0B78	dot1180MHzFrameTransmittedCount	Counts the number of frames transmitted at 80 MHz BW.
223	0x0B7C	dot11160MHzFrameTransmittedCount	Counts the number of frames transmitted at 160 MHz BW.
224	0x0B80	dot1120MHzFrameReceivedCount	Counts the number of frames received at 20 MHz BW.
225	0x0B84	dot1140MHzFrameReceivedCount	Counts the number of frames received at 40 MHz BW.
226	0x0B88	dot1180MHzFrameReceivedCount	Counts the number of frames received at 80 MHz BW.
227	0x0B8C	dot11160MHzFrameReceivedCount	Counts the number of frames received at 160 MHz BW.
228	0x0B90	rw20MHzFailedTXOPs	Counts the number of unsuccessful attempts made to acquire a 20 MHz TXOP.
229	0x0B94	rw20MHzSuccessfulTXOPs	Counts the number of successful acquisitions of a 20 MHz TXOP.
230	0x0B98	rw40MHzFailedTXOPs	Counts the number of unsuccessful attempts made to acquire a 40 MHz TXOP.
231	0x0B9C	rw40MHzSuccessfulTXOPs	Counts the number of successful acquisitions of a 40 MHz TXOP.
232	0x0BA0	rw80MHzFailedTXOPs	Counts the number of unsuccessful attempts made to acquire a 80 MHz TXOP.
233	0x0BA4	rw80MHzSuccessfulTXOPs	Counts the number of successful acquisitions of a 80 MHz TXOP.
234	0x0BA8	rw160MHzFailedTXOPs	Counts the number of unsuccessful attempts made to acquire a 160 MHz TXOP.
235	0x0BAC	rw160MHzSuccessfulTXOPs	Counts the number of successful acquisitions of a 160 MHz TXOP.
236	0x0BB0	rwDynBWDropCount	Count the number of BW drop using dynamic BW management.
237	0x0BB4	rwStaBWFailedCount	Count the number of failure using static BW management.
238 – 239	0x0BB8 - 0x0BBC	Reserved	Reserved for future use.

Index	Address Range	MIB Name	Purpose
STBC MIB set			
240	0x0BC4	dot11STBCCTSSuccessCount	Counts the number of times the AP does not detect a collision PIFS after transmitting a STBC CTS frame.
241	0x0BC8	dot11STBCCTSFailureCount	Counts the number of times the AP detects a collision PIFS after transmitting a STBC CTS frame.
242	0x0BCC	dot11nonSTBCCTSSuccessCount	Counts the number of times the AP does not detect a collision PIFS after transmitting a non-STBC CTS frame.
243	0x0BD0	dot11nonSTBCCTSFailureCount	Counts the number of times the AP detects a collision PIFS after transmitting a non-STBC CTS frame.
244-247	0x0BD4-0x0BDC	Reserved	Reserved for future use.
Beamforming set			
248	0x0BE0	dot11BeamformingFrameCount	Counts the number of frame transmitted using beamforming
249	0x0BE4	rwBeamformingReceivedFrameCount	Counts the number of beamformed frames addressed to the device received
250	0x0BE8	rwSUBFRTransmittedCount	Counts the number of Beamforming Report frames transmitted with SU reports.
251	0x0BEC	rwMUBFRTransmittedCount	Counts the number of Beamforming Report frames transmitted with MU reports.
252	0x0BF0	rwBFRReceivedCount	Counts the number of Beamforming Report frames addressed to the device received.
253	0x0BF4	*Reserved*	*Reserved*
MU-MIMO RX set			
254	0x0BF8	rwMUReceivedFrameCount	Counts the number of MU-MIMO frames addressed to the device received

Figure 22: MIB table

6 MPDU formats

6.1 Transmit MPDU template

All the frames that are created by SW and passed to the HW are formatted as they will be sent over the air. In this case, the MAC Header depends on the frame type, subtype and certain fields are present or not. In this case, the MAC Header generation procedure defined in the Section *2.2.4.2.7, MAC Header generation logic in HW* is not used.

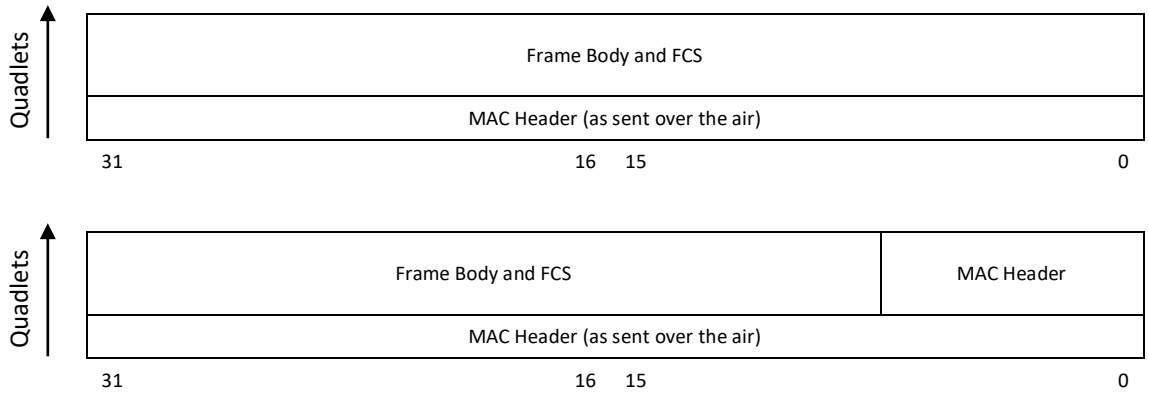


Figure 23: Transmit MPDU template

6.2 Receive MPDU template

All frames that are received from the baseband are passed to the SW as is. As a consequence, the MAC Header size depends on the frame type and subtype. The Receive Vector field is as received from the PHY.

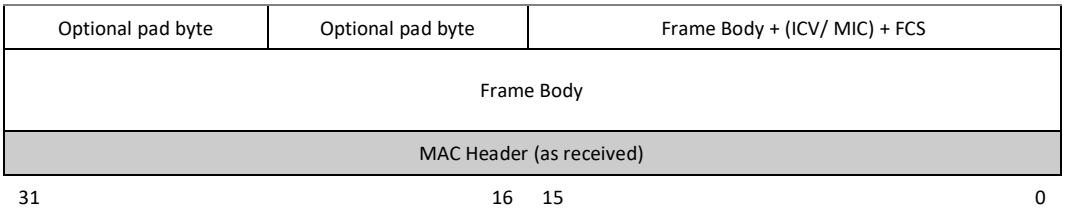


Figure 24: Receive MPDU template

7 Register description

7.1 Convention

7.1.1 Read/Write Registers

Read/Write registers are registers with a single address defined. The fields of these registers may be defined with one or more of the attributes as given in the table below.

Access Tag	Name	Meaning
r	Read	field may be read by software
w	Write	field may be written by software
u	Update	field may be updated by hardware

Table 11: Read/Write Registers

7.1.2 Set and Clear Registers

A Set and Clear register has two addresses defined: “Set Address” and “Clear Address”. For write operations, these two addresses show different behavior. A “one” written to a bit position in the “Set Address” causes the corresponding bit position in the register to be set, while a “zero” leaves the corresponding bit position in the register unaffected. On the other hand, a “one” written to a bit position in the “Clear Address” causes the corresponding bit position in the register to be cleared, while a “zero” leaves the corresponding bit position in the register unaffected. The fields of these registers may be defined with one or more of the attributes as given in the table below.

Access Tag	Name	Meaning
r	Read	field may be read by software
s	Set	field may be set by software
c	Clear	field may be cleared by software
u	Update	field may be updated by hardware

Table 12: Set and Clear Registers

7.1.3 Reserved Registers

Addresses within the address space that are reserved return all zeroes on a read while writes to such registers are ignored.

7.1.4 Reserved Fields

All reads to reserved fields within a register return all zeroes. All writes to reserved fields within a register are ignored.

7.2 Address ranges

Register Range	Description
0x0000 – 0x017C 0x8000 – 0x817C	Basic MAC functionality: control, interrupts, timers, status, security.

Register Range	Description
0x0180 – 0x01FC 0x8180 – 0x81FC	DMA registers
0x0200 – 0x027C 0x8200 – 0x827C	QoS: EDCA
0x0280 – 0x02FC 0x8280 – 0x82FC	Spectrum Management Extensions
0x0300 – 0x037C 0x8300 – 0x837C	High Throughput, Very High Troughput and High Efficiency
0x0400 – 0x04FC 0x8400 – 0x84FC	Reserved for other protocol extensions
0x0500 – 0x05FC 0x8500 – 0x85FC	Debug registers
0x0800 – 0x0BFC 0x8800 – 0x8BFC	MIB Table counters

Table 13: Address range for the MAC HW

7.3 Register set at a glance

Register Offset	Register Name	Description	Clock
Basic registers			
0x0000	Signature	Contains the Signature string.	macCoreClk
0x0004	Version 1	Contains the Version String for this hardware build.	macCoreClk
0x0008	Version 2		
0x000C	Bitmap Count	This indicates the current Bitmap number.	macCoreClk
0x0010	MAC Address Low	This device's MAC address is loaded here.	macCoreClk
0x0014	MAC Address High		
0x0018	MAC Address Low Mask	This device's MAC address mask is loaded here.	macCoreClk
0x001C	MAC Address High Mask		
0x0020	BSS ID Low	The BSSID of this BSS/IBSS is loaded here.	macCoreClk
0x0024	BSS ID High		
0x0028	BSS ID Low Mask	The mask pattern for the BSSID of this BSS/IBSS is loaded here.	macCoreClk
0x002C	BSSID High Mask		
0x0030	AID	Association ID	macCoreClk
0x0034	Reserved		
0x0038	State Control	This register controls the core's state transitions.	macCoreClk
0x003C	Scan Control	Contains settings for controlling SCAN state.	macCoreClk

Register Offset	Register Name	Description	Clock
0X8040	Next TBTT Register	Indicates the remaining time until the next TBTT	macPIClk
0x0044	Doze Control 1	Contains settings for controlling DOZE state.	macCoreClk
0x8048	Doze Control 2		
0x004C	MAC Control 1	Contains various settings for controlling the operation of the core.	macCoreClk
0x8050	MAC Control 2	Contains various settings for controlling the operation of the core.	macPIClk
0x0054	MAC Error Recovery Control	Contains setting for detection and recovery from HW errors.	macCoreClk
0x0058	MAC Error Status Set	HW error status set register	macCoreClk
0x005C	MAC Error Status Clear	HW error status clear register.	macCoreClk
0x0060	Receive Control	Controls the HW receive operation.	macCoreClk
0x0064	Beacon Control 1	Information related to beacon transmission is programmed here.	macCoreClk
0x0068	Beacon Control 2		macCoreClk
0x806C	General Interrupt Event Set	General interrupt status set register.	macPIClk
0x8070	General Interrupt Event Clear	General interrupt status clear register.	macPIClk
0x8074	General Interrupt Unmask	This register is used to unmask the general interrupts.	macPIClk
0x8078	Transmit / Receive Interrupt Event Set	Transmit / Receive interrupt status set register.	macPIClk
0x807C	Transmit / Receive Interrupt Event Clear	Transmit / Receive interrupt status clear register.	macPIClk
0x8080	Transmit / Receive Interrupt Unmask	This register is used to unmask the transmit / receive interrupts.	macPIClk
0x8084	Timers Interrupt Event Set	Timers interrupt status set register.	macPIClk
0x8088	Timers Interrupt Event Clear	Timers interrupt status clear register.	macPIClk
0x808C	Timers Interrupt Event Unmask	This register is used to unmask the Timers interrupts.	macPIClk
0x0090	DTIM	Contains the DTIM Count.	macCoreClk
0x0094	*Reserved*		
0x0098	Retry Limits	Contains the value of the short and long retry limit.	macCoreClk
0x009C	BB service	Register used for programming the Service parameter	macCoreClk
0x00A0	Max Power Level	Contains the power level for response frame	macCoreClk

Register Offset	Register Name	Description	Clock
0x80A4	TSF Timer Low	Contains the TSF bits.	macPIClk
0x80A8	TSF Timer High		
0x00AC	Encryption Key 0	Window registers used to program the encryption parameters into the KeyStorage RAM from software.	macCoreClk
0x00B0	Encryption Key 1		
0x00B4	Encryption Key 2		
0x00B8	Encryption Key 3		
0x00BC	Encryption MAC Address Low		
0x00C0	Encryption MAC Address High		
0x00C4	Encryption Control		
0x00C8	Encryption WPI Integrity Key 0		
0x00CC	Encryption WPI Integrity Key 1		
0x00D0	Encryption WPI Integrity Key 2		
0x00D4	Encryption WPI Integrity Key 3		
0x00DC	Rates	Indicates the BSS Basic Rate Set.	macCoreClk
0x00E0	Overlapping Legacy BSS Condition	Contains settings to determine if there is an OLBC.	macCoreClk
0x00E4	Timings 1	Contains protocol timing information.	macCoreClk
0x00E8	Timings 2		
0x00EC	Timings 3		
0x00F0	Timings 4		
0x00F4	Timings 5		
0x00F8	Timings 6		
0x00FC	Timings 7		
0x0100	Timings 8		
0x0104	Timings 9		
0x0108	Reserved		
0x010C	Reserved		
0x0110	Transmit Trigger Timer	Contains timers for transmit interrupt moderation.	macCoreClk
0x0114	Receive Trigger Timer	Contains timers for receive interrupt moderation.	macCoreClk

Register Offset	Register Name	Description	Clock
0x0118	MIB Table Write	Window register used to write to the MIB Table from software.	macCoreClk
0x011C	Monotonic Counter 1	General purpose timer provided to SW.	macCoreClk
0x0120	Monotonic Counter 2 Low	General purpose timer provided to SW.	macCoreClk
0x0124	Monotonic Counter 2 High		
0x0128-0x14C	Absolute Timer 0-9	General purpose counter provided to SW.	macCoreClk
0x0150	Max Rx Length	Configure the maximum PSDU length received. Should be 4095 in abg and 65535 in HT	macCoreClk
TimeOnAir registers			
0x8160	Time On Air Parameters 1	This register contains parameters used for computing time taken for packet on air.	macPIClk
0x8164	Time On Air Parameters 2	This register contains parameters used for computing time taken for packet on air.	macPIClk
0x8168	Time On Air Value	This register contains time taken for packet on air.	macPIClk
DMA registers			
0x8180	DMA Control Set	Contains control set bits for the DMA channels.	macPIClk
0x8184	DMA Control Clear	Contains control clear bits for the DMA channels.	macPIClk
0x8188	DMA Status 1	Indicates HW DMA status.	macPIClk
0x818C	DMA Status 2		
0x8190	DMA status 3		
0x8194	DMA status 4		
0x8198	Beacon Head Pointer	Head Pointer of the Beacon DMA channel is programmed here.	macPIClk
0x819C	AC0 Head Pointer	Head Pointer of the AC_BK DMA channel is programmed here.	macPIClk
0x81A0	AC1 Head Pointer	Head Pointer of the AC_BE DMA channel is programmed here.	macPIClk
0x81A4	AC2 Head Pointer	Head Pointer of the AC_VI DMA channel is programmed here.	macPIClk
0x81A8	AC3 Head Pointer	Head Pointer of the AC_VO DMA channel is programmed here.	macPIClk
0x81AC	Transmit Structure Sizes	Indicates size of DMA structures	macPIClk
0x81B0	Reserved		
0x81B4	Reserved		
0x81B8	Reserved		

Register Offset	Register Name	Description	Clock
0x81BC	Reserved		
0x81C0	DMA Threshold	Contains FIFO threshold value for DMA triggering	macPIClk
0x81C4	Receive Header Trigger Frame Pointer Register	Contains the address of the Rx Header descriptor of the latest Trigger Frame received	macPIClk
0x81C8	Receive Buffer 1 Start Pointer Register	Contains the starting address of the DMA Rx Buffer 1	macPIClk
0x81CC	Receive Buffer 1 End Pointer Register	Contains the end address of the DMA Rx Buffer 1	macPIClk
0x81D0	Receive Buffer 1 Read Pointer Register	Contains the read address of the DMA Rx Buffer 1	macPIClk
0x81D4	Receive Buffer 1 Write Pointer Register	Contains the write address of the DMA Rx Buffer 1	macPIClk
0x81D8	Receive Buffer 2 Start Pointer Register	Contains the starting address of the DMA Rx Buffer 2	macPIClk
0x81DC	Receive Buffer 2 End Pointer Register	Contains the end address of the DMA Rx Buffer 2	macPIClk
0x81E0	Receive Buffer 2 Read Pointer Register	Contains the read address of the DMA Rx Buffer 2	macPIClk
0x81E4	Receive Buffer 2 Write Pointer Register	Contains the write address of the DMA Rx Buffer 2	macPIClk
0x81E8	Receive Buffer Configuration Register	Contains the receive buffer blank spaces configuration	macPIClk
QoS registers			
0x0200	EDCA AC0	Contains the AC0 parameters.	macCoreClk
0x0204	EDCA AC1	Contains the AC1 parameters.	
0x0208	EDCA AC2	Contains the AC2 parameters.	
0x020C	EDCA AC3	Contains the AC3 parameters.	
0x8210	EDCA AC Has Data Set	Indicates whether an AC queue in SW has data	macPIClk
0x8214	EDCA AC Has Data Clear		macPIClk
0x0218	Reserved		
0x021C	Reserved		
0x0220	EDCA CCA Busy Time	Indicates the CCA busy time.	macCoreClk
0x0224	EDCA Control	Contains various settings for controlling the operation of the core in EDCA.	macCoreClk
0x8228	MOT 1	Provides the value of the TXOP for EDCA.	macPIClk
0x822C	MOT 2		
Spectrum Management Extensions registers			
0x0280	Quiet Element 1a Register	Contains parameters received in Quiet IEs.	macCoreClk

Register Offset	Register Name	Description	Clock
0x0284	Quiet Element 1b Register		macCoreClk
0x0290	EDCA CCASec20 Busy Time	Indicates the CCA on Secondary 20MHz busy time.	macCoreClk
0x0294	EDCA CCASec40 Busy Time	Indicates the CCA on Secondary 40MHz busy time.	macCoreClk
0x0298	EDCA CCASec80 Busy Time	Indicates the CCA on Secondary 80MHz busy time.	macCoreClk
HT, VHT & HE registers			
0x0300	STBC Control	Controls STBC protection functionality.	macCoreClk
0x0304	Start Transmission 1	Used to trigger the HW to transmit a control frame at the start of a TXOP.	macCoreClk
0x0308	Start Transmission 2	Used to trigger the HW to transmit a control frame at the start of a TXOP.	macCoreClk
0x030C	Start Transmission 3	Used to trigger the HW to transmit a control frame at the start of a TXOP.	macCoreClk
0x0310	Transmission Bandwidth Control register	Controls 40/80/160 MHz HW operation	macCoreClk
0x0314	MCS	Indicates the BSS Basic MCS Set.	macCoreClk
0x031C	VHTMCS register	Indicates the BSS VHTBasic MCS Set.	macCoreClk
0x0320	LSTP register	Used to control LSTP	macCoreClk
0x0324	HE Configuration Register	Used to control the HW behavior related to HE operation	macCoreClk
0x0328	Spatial Reuse Configuration Register	Contains parameters for Spatial Reuse	macCoreClk
0x032C	SRG BSS Color Bitmap Low Register	Contains LSB of SRG BSS Color Bitmap	macCoreClk
0x0330	SRG BSS Color Bitmap High Register	Contains MSB of SRG BSS Color Bitmap	macCoreClk
0x8330	Transmit Bandwidth Drop Information register	Indicates the bandwidth in case of bandwith drop.	macPIClk
0x0334	SRG Partial BSSID Bitmap Low Register	Contains LSB of SRG Partial BSSID Bitmap	macCoreClk
0x0338	SRG Partial BSSID Bitmap High Register	Contains MSB of SRG Partial BSSID Bitmap	macCoreClk
0x0350	Beamformee Control register	Used to control the Beamforming as a Beamformee feature	macCoreClk
0x8354	Transmit Trigger Based Information Register	Provides information to the SW during HE TB transmission	macPIClk

Register Offset	Register Name	Description	Clock
0x8358	Receive HE Trigger Common Information Register	Provides Common Information to the SW upon reception of Trigger frame to ease the HE_TB generation	macPIClk
0x835C	Receive HE Trigger User Information Register	Provides User Information to the SW upon reception of Trigger frame to ease the HE_TB generation	macPIClk
0x8364	Secondary Users Transmit Interrupt Event Register	Secondary Users Transmit interrupts status set register.	macPIClk
0x8368	Secondary Users Transmit Interrupt Event Register	Secondary Users Transmit interrupts status clear register.	macPIClk
0x836C	Secondary Users Transmit Interrupt UnMask Register	This register is used to unmask the Secondary Users Transmit interrupts.	macPIClk
BT Coexistence registers			
0x0400	Coex Control register	Controls the Coex operation	macCoreClk
0x0404	Coex PTI register	Configure the PTI information	macCoreClk
Debug registers			
0x0500	Debug HW State Machine 1	Used to observe the MAC HW state machines.	macCoreClk
0x0504	Debug HW State Machine 2		macCoreClk
0x0508	Reserved		
0x050C	Debug Port Value	Return the current value of the debugPort	macCoreClk
0x0510	Debug Port Select	Used to multiplex different sets of signals on the debug pins.	macCoreClk
0x0514	Debug NAV	Current value of the NAV and CW for debug.	macCoreClk
0x0518	Debug Contention Window		macCoreClk
0x051C	Debug QoS SSRC	Current value of the QoS Station Short Retry Counter for debug.	macCoreClk
0x0520	Debug QoS SLRC	Current value of the QoS Station Long Retry Counter for debug.	macCoreClk
0x8524	Debug Beacon Status Pointer	Current value of the Beacon status pointer for debug.	macPIClk
0x8528	Debug AC_BK Status Pointer	Current value of the AC_BK status pointer for debug.	macPIClk
0x852C	Debug AC_BE Status Pointer	Current value of the AC_BE status pointer for debug.	macPIClk
0x8530	Debug AC_VI Status Pointer	Current value of the AC_VI status pointer for debug.	macPIClk
0x8534	Debug AC_VO Status Pointer	Current value of the AC_VO status pointer for debug.	macPIClk

Register Offset	Register Name	Description	Clock
0x0538	Reserved		
0x053C	Reserved		
0x8540	Debug Transmit DMA Current Pointer	Current value of the Transmit DMA current pointer for debug.	macPIClk
0x8544	Debug Receive Payload Status Pointer	Current value of the Receive Payload DMA status pointer for debug.	macPIClk
0x8548	Debug Receive Header Status Pointer	Current value of the Receive Header DMA status pointer for debug.	macPIClk
0x854C	Debug Receive Payload Current Pointer	Current value of the Receive Payload DMA current pointer for debug.	macPIClk
0x8550	Debug DMA State Machine	Used to observe the DMA HW state machines.	macPIClk
0x0558	Debug UORA	Used to debug the UP OFDMA Random Access	macCoreClk
0x055C	Debug PHY	Used to debug the PHY	macCoreClk
0x8560	Software Profiling	Software profiling register. Allow the SW to write on the SW debug port	macPIClk
0x8564	Software Profiling Set	Software profiling Set register. Allow the SW to set bits on the SW debug port	macPIClk
0x8568	Software Profiling Clear	Software profiling register. Allow the SW to clear bits on the SW debug port	macPIClk
0x8570	Debug Secondary User 1 Transmit DMA Current Pointer Register	The Secondary User1 Transmit DMA Current Pointer is exposed for debug purposes.	macPIClk
0x8574	Debug Secondary User 2 Transmit DMA Current Pointer Register	The Secondary User2 Transmit DMA Current Pointer is exposed for debug purposes.	macPIClk
0x8578	Debug Secondary User 3 Transmit DMA Current Pointer Register	The Secondary User3 Transmit DMA Current Pointer is exposed for debug purposes.	macPIClk
0x857C	Debug Secondary User 1 Status Pointer Register	The Secondary User1 Status Pointer is exposed for debug purposes.	macPIClk
0x8580	Debug Secondary User 2 Status Pointer Register	The Secondary User2 Status Pointer is exposed for debug purposes.	macPIClk
0x8584	Debug Secondary User 3 Status Pointer Register	The Secondary User3 Status Pointer is exposed for debug purposes.	macPIClk

Table 14: Register summary

7.4 Basic register definitions

7.4.1 Signature Register (signatureReg)

Offset Address: 0x0000

Field name	rwu	Bit #	Reset	Description

Field name	rwu	Bit #	Reset	Description
Signature	r	31:0	`MACHW_SIGNATURE	This register contains the synthesis-time configurable signature string `SIGNATURE. The string is used by the software to determine the endianness of the core based on a read of this pre-defined string. Example: Assume that the `SIGNATURE is defined as 32'h0a0b0c0d. If the data read from this register is 32'h0a0b0c0d, then the processor access is little endian else if the data is 32'h0d0c0b0a, then the processor access is big endian.

7.4.2 Version 1 Register (version1Reg)

Offset Address: 0x0004

This register contains the values of the Verilog `defines used at the time of synthesis of the core.

Note: Reset values are assigned as per features supported by current latest release of RTL Code.

Field name	rwu	Bit #	Reset	Description
qos	r	0	1'b1	Quality of Service This bit indicates if the core supports quality of service features. Note the MAC HW always supports QoS, and this bit is put here for completeness.
edca	r	1	1'b1	Enhanced Distributed Channel Access This bit indicates if the core supports EDCA features. Note the MAC HW always supports EDCA, and this bit is put here for completeness.
Reserved	r	2	1'b0	Reserved
sme	r	3	`MACHW_SME	Spectrum Management Extension This bit indicates if the core supports spectrum management extensions.
security	r	4	`MACHW_SECURITY	Security This bit indicates if the core supports security algorithms.
wep	r	5	`MACHW_WEP	WEP This bit indicates if the core supports WEP encryption in hardware.
tkip	r	6	`MACHW_TKIP	TKIP This bit indicates if the core supports TKIP encryption in hardware.
ccmp	r	7	`MACHW_CCMP	CCMP This bit indicates if the core supports CCMP encryption in hardware

Field name	rwu	Bit #	Reset	Description
rce	r	8	` MACHW_RCE	Regulatory Classes Extensions This bit indicates if the core supports regulatory classes extensions.
Reserved	r	9	1'b0	Reserved
ht	r	10	1'b1	High Throughput This bit indicates if the core supports enhancements for high throughput.
vht	r	11	1'b1	Very High Throughput This bit indicates if the core supports enhancements for very high throughput.
tpc	r	12	1'b1	Transmit Power Control This bit indicates if the core supports enhancement for transmit power control
wapi	r	13	`RW_WAPI_EN	WAPI This bit indicates if the core supports WAPI encryption in hardware
coex	r	14	`RW_WLAN_COEX_EN	Coexistence This bit indicates if the core supports BT Coexistence Interface
he	r	15	1' b1	High Efficiency This bit indicates if the core supports enhancements for very high throughput.
Reserved	r	16	1' b0	Reserved
bfmee	r	17	`RW_BFMEE_EN	This bit indicates if the core supports the beamforming as a beamformee
bfmer	r	18	`RW_BFMER_EN	This bit indicates if the core supports the beamforming as a beamformer
muMIMOTx	r	19	`RW_MUMIMO_TX_EN	This bit indicates if the core supports the MU-MIMO transmission as an AP
Reserved	r	31:20	12'b0	*Reserved*

7.4.3 Version 2 Register (version2Reg)

Offset Address: 0x0008

This register contains the values of the Verilog `defines used at the time of synthesis of the core.

Note: Reset values are assigned as per features supported by current latest release of RTL Code.

Field name	rwu	Bit #	Reset	Description
UM version	r	6:0	`MACHW_UM_VERSION	These bits are used to define the FS compliancy of the current hardware as xx.yyyyy.
I/E Release	r	7	`MACHW_IE_RELEASE	This bit is set if it is an external (client) release, and is reset if it is an internal (test team) release.

Field name	rwu	Bit #	Reset	Description
Release number	r	13:8	`MACHW_RELEASE_NUMBER	These bits are used to define the release number under this phase (same as the *intermediate release* field of the tag in SVN). Note: (`RELEASE_NUMBER) = same as the release number field of the tag in SVN.
Phase number	r	16:14	`MACHW_PHASE_NUMBER	These bits are used to define the phase number of the release (same as the *minor version* field of the tag in SVN).
Reserved	r	31:17	15'b0	Reserved

7.4.4 Bitmap Count Register (bitmapCntReg)

Offset Address: 0x000C

Field name	rwu	Bit #	Reset	Description
bitmapCnt	r	15:0	`MACHW_BITMAP_COUNT	Bitmap Count This field is used to identify a bitmap created for FPGA debugging. It is a unique number for each bitmap created, and is incremented for each bitmap.
Reserved	r	31:16	16'b0	Reserved

7.4.5 MAC Address Low Register (macAddrLowReg)

Offset Address: 0x0010

This register contains the 32 LSBs of this devices' unique 48-bit MAC Address. This register should be written to only when the core is in the IDLE state.

Field name	rwu	Bit #	Reset	Description
macAddr [31:0]	rw	31:0	32'b0	MAC Address SW programs the 32 LSBs of the MAC Address of this device into this register.

7.4.6 MAC Address High Register (macAddrHiReg)

Offset Address: 0x0014

This register contains the 16 MSBs of this devices' unique 48-bit MAC Address. This register should be written to only when the core is in the IDLE state.

Field name	rwu	Bit #	Reset	Description
macAddr [47:32]	rw	15:0	16'b0	MAC Address SW programs the 16 MSBs of the MAC Address of this device into this register.
Reserved	r	31:16	16'b0	Reserved

7.4.7 MAC Address Low Mask Register (macAddrLowMaskReg)

Offset Address: 0x0018

This register contains the 32 LSBs of a 48-bit masking pattern that allows the MAC to behave as if has a range of MAC addresses, rather than a single address. Any received MPDU matching the address in the range of addresses is accepted and responded with the suitable acknowledgement frame. This register should be written to only when the core is in the IDLE state.

Field name	rwu	Bit #	Reset	Description
macAddrMask [31:0]	rw	31:0	32'b0	MAC Address Mask SW programs the 32 LSBs of the MAC Address Mask into this register. If a mask bit is set, the corresponding value of the bit position in the macAddr field is not compared to the receiver address.

7.4.8 MAC Address High Mask Register (macAddrHiMaskReg)

Offset Address: 0x001C

This register contains the 16 MSBs of a 48-bit masking pattern that allows the MAC to behave as if has a range of MAC addresses, rather than a single address. Any received MPDU matching the address in the range of addresses is accepted and responded with the suitable acknowledgement frame. This register should be written to only when the core is in the IDLE state.

Field name	rwu	Bit #	Reset	Description
macAddrMask [47:32]	rw	15:0	16'b0	MAC Address Mask SW programs the 16 MSBs of the MAC Address Mask into this register. If a mask bit is set, the corresponding value of the bit position in the macAddr field is not compared to the receiver address.
Reserved	r	31:16	16'b0	Reserved

7.4.9 BSSID Low Register (bssIDLowReg)

Offset Address: 0x0020

This register contains the 32 LSBs of this devices' unique 48-bit BSSID. This register should be written to only when the core is in the IDLE state.

Field name	rwu	Bit #	Reset	Description
bssID [31:0]	rw	31:0	32'b0	BSSID Software programs the 32 LSBs of the BSSID of the BSS or IBSS to which this STA belongs. As an AP, this field contains the 32 LSBs of the AP's MAC address.

7.4.10 BSSID High Register (bssIDHiReg)

Offset Address: 0x0024

This register contains the 16 MSBs of this devices' 48-bit BSSID. This register should be written to only when the core is in the IDLE state.

Field name	rwu	Bit #	Reset	Description
bssID [47:32]	rw	15:0	16'b0	BSSID

Field name	rwu	Bit #	Reset	Description
				Software programs the 16 MSBs of the BSSID of the BSS or IBSS to which this STA belongs. As an AP, this field contains the 16 MSBs of the AP's MAC address.
Reserved	r	31:16	16'b0	Reserved

7.4.11 BSSID Low Mask Register (bssIDLowMaskReg)

Offset Address: 0x0028

This register contains the 32 LSBs of a 48-bit masking pattern that allows the MAC to behave as if has a range of BSSIDs, rather than a single BSSID. This register should be written to only when the core is in the IDLE state.

Field name	rwu	Bit #	Reset	Description
bssIDMask [31:0]	rw	31:0	32'b0	BSSID Mask SW programs the 32 LSBs of the BSSID Mask into this register. If a mask bit is set, the corresponding value of the bit position in the *bssID* field is masked when comparing.

7.4.12 BSSID High Mask Register (bssIDHiMaskReg)

Offset Address: 0x002C

This register contains the 16 MSBs of a 48-bit masking pattern that allows the MAC to behave as if has a range of BSSIDs, rather than a single BSSID. This register should be written to only when the core is in the IDLE state.

Field name	rwu	Bit #	Reset	Description
bssIDMask [47:32]	rw	15:0	16'b0	BSSID Mask SW programs the 16 MSBs of the BSSID Mask into this register. If a mask bit is set, the corresponding value of the bit position in the *bssID* field is masked when comparing.
Reserved	r	31:16	16'b0	Reserved

7.4.13 Reserved (NA)

Offset Address: 0x0030

Reserved for future use.

Field name	rwu	Bit #	Reset	Description
Reserved	r	31:0	32'b0	Reserved

7.4.14 BSS Color Register (bssColorReg)

Offset Address: 0x0034

This register contains the BSS Color. This register should be written to only when the core is in the IDLE state.

Field name	rwu	Bit #	Reset	Description
bssColor	rw	5:0	6'b0	BSS Color

Field name	rwu	Bit #	Reset	Description
				SW programs the BSS Color into this register.
Reserved	r	7:6	2'b0	Reserved
bssColorEn	rw	8	1'b0	BSS Color Enable Indicate that the BSS Color shall be used.
partialBssColorEn	rw	9	1'b0	Partial BSS Color Enable Indicate that the Partial BSS Color shall be used.
Reserved	r	31:9	23'b0	Reserved

7.4.15 State Control Register (stateCntrlReg)

Offset Address: 0x0038

The MAC core can be in one of the main states as given below. The states are IDLE, ACTIVE and DOZE, as explained in section *2.1*, *MAC core states*.

Software can program a state other than IDLE into this field ONLY if the *stateCntrlReg.currentState* field is IDLE. The state encoding of this field is identical to that of the *currentState* field. The core automatically updates this field to IDLE whenever it updates the *stateControRegl.currentState* field to IDLE.

Example: If core is in ACTIVE state, and if software wants to program the core in DOZE then software programs core initially to IDLE and then to DOZE.

NOTE: When the core automatically changes its current state to IDLE from some other state then it updates this field to IDLE. When the core updates the *currentState* field to IDLE from DOZE, it updates this field too to IDLE.

Field name	rwu	Bit #	Reset	Description
currentState [3:0]	ru	3:0	4'b0	Current State Indicates the current state of the core. The state encoding is as follows: 0000 – IDLE state. This is the default state. 0001 – Reserved 0010 – DOZE state 0011 – ACTIVE state 0100 to 1111 – reserved
nextState [3:0]	rwu	7:4	4'b0	Next State Written by software to change the state of the core. If the *stateCntlrReg.currentState* is other than IDLE, software shall only write an IDLE into this field.
Reserved	r	31:8	24'b0	Reserved

7.4.16 Scan Control Register (scanCntrlReg)

Offset Address: 0x003C

This register contains fields that control the SCAN operation of the HW. This register should be written to only when the core is in the IDLE state.

Field name	rwu	Bit #	Reset	Description
probeDelay	rw	15:0	16'b0	Probe Delay This field contains the probe delay in microseconds.
Reserved	r	19:16	4'b0	Reserved
primaryChPosition	rw	22:20	3'b0	Primary Channel Position This field provides the position of the primary channel within the full bandwith encoded as follow : 3'd0: Primary is on the lowest 20MHz sub-band ... 3'd3: Primary is on the highest 20MHz sub-band in case of 80MHz channel ... 3'd7: Primary is on the highest 20MHz sub-band in case of 160MHz channel
Reserved	r	31:23	9'b0	Reserved

7.4.17 Next TBTT Register (nextTBTTreg)

Offset Address: 0x8040

This register contains fields that indicate the next TBTT.

Field name	rwu	Bit #	Reset	Description
nextTBTT	r	15:0	32'b0	Next TBTT This register indicates the remaining time (in 32us) until the next TBTT
Reserved	r	31:16	16'b0	Reserved

7.4.18 Doze Control 1 Register (dozeCntrl1Reg)

Offset Address: 0x0044

This register contains fields that control the DOZE operation of the HW. This register should be written to only when the core is in the IDLE state.

Field name	rwu	Bit #	Reset	Description
listenInterval	rw	15:0	16'b0	Listen Interval If stateCntrl.nextState = DOZE, the core will wake up every listenInterval beacon intervals. If the listenInterval field is programmed to 0, the core will not transition out of DOZE (to IDLE) on its own unless explicitly programmed by software or unless the wakeUpDTIM is set.
wakeupDTIM	rw	16	1'b0	Wakeup DTIM interval When set, the core wakes up just before the TBTT at which the DTIM Count becomes zero.

Field name	rwu	Bit #	Reset	Description
Reserved	rw	31:17	15'b0	*Reserved*

7.4.19 Doze Control 2 Register (dozeCntrl2Reg)

Offset Address: 0x8048

This register should be written to only when the core is in the IDLE state.

Field name	rwu	Bit #	Reset	Description
wakeUpSW	rw	0	1'b1	Wake Up SW This bit is set if the SW wants the HW to wake it up when the HW moves out of the Doze state. (Refer *2.1.3.2.1.2*, *Auto wake-up mode - legacy*).
Reserved	r	30:1	30'b0	Reserved
wakeUpFromDoze	rw	31	1'b0	Wake Up from Doze. This bit is set if the SW wants to wake up the HW from the Doze state.

7.4.20 MAC Control 1 Register (macCntrl1Reg)

Offset Address: 0x004C

This register controls the operation of the core. Except for *pwrMgt*, none of the other bits should be modified by software in any state other than IDLE (i.e. *State Control Register (stateCntrlReg).currentState* = *State Control Register (stateCntrlReg).nextState* = IDLE).

Field name	rwu	Bit #	Reset	Description
bssType	rw	0	1'b1	BSS Type If set, it indicates that this device is in an infrastructure BSS, either as a STA or an AP. If reset, it indicates that the core is part of an IBSS.
ap	rw	1	1'b0	Access Point If set indicates that this device is an Access Point.
pwrMgt	rw	2	1'b0	Power Management The contents of this bit are copied to the *Pwr Mgt* bit in the *Frame Control* field of any frame transmitted by the core.
Reserved	r	4-3	2'b0	Reserved
lpClk32786Hz	rw	5	1'b0	LowPower Clock 32768Hz When set, it indicates that the LowPower clock frequency is 32.768kHz When reset, it indicates that the LowPower clock frequency is 32kHz

Field name	rwu	Bit #	Reset	Description
enableLPClkSwitch	rw	6	1'b0	Enable the switch to Low Power clock This bit is set by the SW to enable the switch of the macLPClk to LowPower Clock.
activeClkGating	rw	7	1'b1	Active Clock Gating This bit is set by SW to turn on active clock gating in HW. When reset, the HW does not perform active clock gating, i.e. does not turn off clocks to save power in ACTIVE state, as described in section *2.1.3.2.2*, *Design based power save*.
disableACKResp	rw	8	1'b0	Disable ACK Response This bit is set by SW to disable the automatic ACK generation logic in HW.
disableCTSResp	rw	9	1'b0	Disable CTS Response This bit is set by SW to disable the automatic CTS generation logic in HW. Note that this field does not control whether a valid RTS frame meant for this device is passed to SW. That must be enabled in the *Receive Control Register (rxCntrlReg)*.
disableBAResp	rw	10	1'b0	Disable BA Response This bit is set by SW to disable the automatic BA generation logic in HW. Note that this field does not control whether a valid BAR frame meant for this device is passed to SW. That must be enabled in the *Receive Control Register (rxCntrlReg)*.
rateControlledMPIF	rw	11	1'b1	Rate Controlled MAC-PHY InterFace This bit is set by SW to indicate to the HW that the attached PHY does not use reverse synchronization across the MAC-PHY transmit interface, and performs rate control at the interface. It is used to determine when a transmit underrun has occurred, as described in section *9.2*, *Non-terminal hardware errors*.
mibTableReset	rwu	12	1'b0	MIB Table reset When software sets this bit, the core will clear the contents of the MIB Table. A string of 0s will be written to every location of this memory. Once the reset is complete, the core automatically resets this bit. This bit must be set by SW during initialization.

Field name	rwu	Bit #	Reset	Description
keyStoRAMReset	rwu	13	1'b0	Key Storage RAM Reset When software sets this bit, the core will clear the contents of the Key Storage RAM. A string of 0s will be written to every location of this memory. Once the reset is complete, the core automatically resets this bit. This bit must be set by SW during initialization.
abgnMode	rw	16:14	3'b011	802.11a/b/g/n/ac Mode This field indicates the current operating mode of the MAC HW. The bits are encoded as follows: 3'b000: 802.11b 3'b001: 802.11a 3'b010: 802.11g 3'b011: 802.11n @ 2.4 GHz 3'b100: 802.11n @ 5 GHz 3'b110: 802.11ac @5 GHz 3'111: Reserved
Reserved	r	23:17	7'b0	Reserved
tsfUpdatedBySW	rwu	24	1'b0	TSF Updated By SW This bit is set by SW after the contents of the *TSF Timer Low Register (tsfLoReg)* and *TSF Timer High Register (tsfHiReg)* have been written to by SW in debug mode. The bit is cleared by HW after the contents of the TSF have been processed. Note: It is recommended that SW perform the first write to *tsfLoReg*, followed immediately by a write to *tsfHiReg*.
tsfMgtDisable	rw	25	1'b0	TSF Management Disable When SW sets this bit, the TSF Management done by the HW is disabled. In this case, the SW shall maintain the TSF by itself when configured in STA mode.
rxRIFSEn	rw	26	1'b0	Rx RIFS Enable This bit is set by the SW to indicate that the reception of frame after RIFS is possible.
Reserved	r	31:27	5'b0	Reserved

7.4.21 MAC Control 2 Register (macCntrl2Reg)

Offset Address: 0x8050

This register controls the operation of the core. Except for *softReset*, none of the other bits should be modified by software in any state other than IDLE (i.e. *State Control Register (stateCntrlReg).currentState* = *State Control Register (stateCntrlReg).nextState* = IDLE).

Field name	rwu	Bit #	Reset	Description

Field name	rwu	Bit #	Reset	Description
softReset	rw	0	1'b0	Soft Reset When set by software, hardware will bring all registers and state machines to their hard reset state. Once set, it will be automatically reset by hardware once the reset of all registers and state machines are complete.
Reserved	r	31:1	31'b0	Reserved

7.4.22 MAC Error Recovery Control Register (macErrRecCntrlReg)

Offset Address: 0x0054

This register contains settings for detection of HW errors in HW and recovery from HW errors. This register should be written to only when the core is in the IDLE state.

Field name	rwu	Bit #	Reset	Description
useErrRec	r	0	1'b0	Use Error Recovery When set, this bit enables intelligent run-time error detection by the MAC hardware. Reserved for future use.
useErrDet	rw	1	1'b0	Use Error Detection When set, this bit enables intelligent run-time error detection by the MAC hardware and provides feedback to SW. It does not enable error recovery in HW.
hwFSMReset	rwu	2	1'b0	Hardware Finite State Machine Reset When set by software will bring all state machines and variables in MAC Hardware to their hard reset state. The registers exposed to SW and memories inside the MAC HW are not reset. Once set, it will be automatically reset by hardware once the reset of all state machines is complete.
rxFIFOReset	rwu	3	1'b0	Receive FIFO Reset When software sets this bit, the core will reset the Receive FIFO and Receive Tag FIFO read and write pointers. Once the reset is complete, the core automatically resets this bit.
txFIFOReset	rwu	4	1'b0	Transmit FIFO Reset When software sets this bit, the core will reset the Transmit FIFO and Transmit Tag FIFO read and write pointers. Once the reset is complete, the core automatically resets this bit.
macPHYIFFIFOReset	rwu	5	1'b0	MAC-PHY InterFace FIFO Reset When software sets this bit, the core will reset the

Field name	rwu	Bit #	Reset	Description
				MAC-PHY Interface FIFOs read and write pointers. Once the reset is complete, the core automatically resets this bit.
Reserved	r	6	1'b0	Reserved
baPSBitmapReset	rwu	7	1'b0	Block ACK Partial Bitmap Reset When software sets this bit, the core will clear the contents of the Block ACK Partial State Bitmap memory. A string of 0s will be written to every location of this memory. Once the reset is complete, the core automatically resets this bit.
Reserved	r	15:8	8'b0	Reserved
rxFlowCntrlEn	rw	16	1'b0	RX Flow Control Enable When enabled, the MAC HW does not move to DEAD state in case of lack of rx descriptor. In this case, it does not acknowledge the received frames until new RX descriptor are linked.
Reserved	R	31:17	15'b0	Reserved

7.4.23 MAC Error Status Set Register (macErrStatusSetReg)

Offset Address: 0x0058

This register indicates the actual HW error that has occurred. Refer to section *9.4, Terminal hardware errors* for more information.

Field name	rwu	Bit #	Reset	Description
errInTxLevel1	rwu	0	1'b0	Error In Transmit – Level 1 This bit is set by the core when a Level 1 transmission error occurs.
errInRxLevel1	rwu	1	1'b0	Error In Receive – Level 1 This bit is set by the core when a Level 1 receive error occurs.
errInTxRxLevel2	rwu	2	1'b0	Error in Transmit/Receive – Level 2 This bit is set by the core when a Level 2 error occurs in Transmit or receive.
errInHWLevel3	rwu	3	1'b0	Error in HW – Level 3 This bit is set by the core when a Level 3 error in the MAC HW occurs.
Reserved	r	31:4	28'b0	Reserved

7.4.24 MAC Error Status Clear Register (macErrStatusClearReg)

Offset Address: 0x005C

Field name	rwu	Bit #	Reset	Description

Field name	rwu	Bit #	Reset	Description
errInTxLevel1	rw	0	1'b0	Error In Transmit – Level 1 This bit is written to by SW to clear the *MAC Error Status Set Register (macErrStatusSetReg).errInTxLevel1*.
errInRxLevel1	rw	1	1'b0	Error In Receive – Level 1 This bit is written to by SW to clear the *MAC Error Status Set Register (macErrStatusSetReg).errInRxLevel1*.
errInTxRxLevel2	rw	2	1'b0	Error in Transmit/Receive – Level 2 This bit is written to by SW to clear the *MAC Error Status Set Register (macErrStatusSetReg).errInTxRxLevel2*.
errInHWLevel3	rw	3	1'b0	Error in HW – Level 3 This bit is written to by SW to clear the *MAC Error Status Set Register (macErrStatusSetReg).errInHWLevel3*.
Reserved	r	31:4	28'b0	Reserved

7.4.25 Receive Control Register (rxCntrlReg)

Offset Address: 0x0060

The SW can control the reception process of the HW by setting the bits in this register. Multiple bits can be set together and controls whether a received frame is delivered to SW. Refer section *2.3.3*, *MAC core receive procedure*.

Except for *multicastRxDis* and *bcMcRxDis*, none of the other bits should be modified by software in any state other than IDLE (i.e. *State Control Register (stateCntrlReg).currentState* = *State Control Register (stateCntrlReg).nextState* = IDLE).

Field name	rwu	Bit #	Reset	Description
excUnencrypted	rw	0	1'b0	Exclude Unencrypted Receive fragments When this field is set, the core will not receive MPDUs that have the *Protected Frame* subfield of the *Frame Control* field equal to zero. When this field is reset, the core accepts MPDUs that have the *Protected Frame* subfield of the *Frame Control* field set to zero. Management frames are excluded from this check.
dontDecrypt	rw	1	1'b0	Don't Decrypt When this bit is set, the MAC HW does not decrypt any received encrypted MPDU and passes it as received to SW.
acceptMulticast	rw	2	1'b0	Accept Multicast If this bit is not set, multicast frames are not passed to SW.

Field name	rwu	Bit #	Reset	Description
acceptBroadcast	rw	3	1'b1	Accept Broadcast If this bit is not set, broadcast frames are not passed to SW.
acceptOtherBSSID	rw	4	1'b0	Accept Other BSSID If this bit is not set, broadcast and multicast frames that contain a BSSID not matching this BSS's BSSID are not passed to SW.
acceptErrorFrames	rw	5	1'b0	Accept Error Frames When set, the core passes error frames to SW with the relevant error status indicated in the *Receive DMA Header Descriptor*. Note that only the Protocol and FCS errors are reported.
acceptUnicast	rw	6	1'b0	Accept Unicast If this bit is not set, unicast frames not directed to this device are not passed to SW. Note that the relevant acknowledgement in this case is only transmitted for frames that are directed to the range of addresses that is determined from the *MAC Address Low Register (macAddrLowReg)* , the *MAC Address High Register (macAddrHiReg)* and their masking patterns.
acceptMyUnicast	rw	7	1'b1	Accept My Unicast If this bit is not set, unicast frames directed to this device are not passed to SW.
acceptProbeReq	rw	8	1'b1	Accept Probe Request If this bit is not set, Probe Request subtype frames are not passed to SW. Note that the BSSID check is overridden by this bit. When set, all the Probe Req are passed to the SW whatever the BSSID.
acceptProbeResp	rw	9	1'b1	Accept Probe Response If this bit is not set, Probe Response subtype frames are not passed to SW.
acceptBeacon	rw	10	1'b1	Accept Beacon If this bit is not set, Beacon subtype frames are not passed to SW.
acceptDecryptErrorFrames	rw	11	1'b0	Accept Decryption Error Frames When set, the core passes decryption error frames to SW with the relevant error status indicated in the *Receive DMA Header Descriptor*.

Field name	rwu	Bit #	Reset	Description
acceptNoExpectedBA	rw	12	1'b0	Accept Not Expected BA When set, the core passes the BA which are not received after SIFS to the SW. When reset, only the BA received in response to a transmission are passed (it depends on acceptBA filter)
acceptAllBeacon	rw	13	1'b0	Accept All Beacon If this bit is set, all the Beacon frames correctly received are passed. Note that the BSSID check is overridden by this bit. When set, all the Beacon are passed to the SW whatever the BSSID.
acceptBfmeeFrames	rw	14	1'b0	Accept All Beamformee Frames If this bit is set, all the NDPA, NPD and Beamforming Report Poll frames correctly received are passed.
acceptOtherMgmtFrames	rw	15	1'b1	Accept Other Management Frames If this bit is not set, other Management subtype frames which are not explicitly covered (like *acceptProbeRes*, *acceptProbeResp*) are not passed to SW.
acceptBAR	rw	16	1'b1	Accept Block Ack Request If this bit is not set, BAR subtype frames are not passed to SW. Note that this setting is also used for a BAR frame carried in a Control Wrapper frame.
acceptBA	rw	17	1'b1	Accept Block Ack If this bit is not set, BA subtype frames are not passed to SW. Note that this setting is also used for a BA frame carried in a Control Wrapper frame.
acceptPSPoll	rw	18	1'b1	Accept PS-Poll If this bit is not set, PS-Poll subtype frames are not passed to SW. Note that this setting is also used for a PS-Poll frame carried in a Control Wrapper frame.
acceptRTS	rw	19	1'b0	Accept RTS If this bit is not set, RTS subtype frames are not passed to SW. Note that this setting is also used for an RTS frame carried in a Control Wrapper frame.

Field name	rwu	Bit #	Reset	Description
acceptCTS	rw	20	1'b0	Accept CTS If this bit is not set, CTS subtype frames are not passed to SW. Note that this setting is also used for a CTS frame carried in a Control Wrapper frame.
acceptACK	rw	21	1'b0	Accept ACK If this bit is not set, ACK subtype frames are not passed to SW. Note that this setting is also used for an ACK frame carried in a Control Wrapper frame.
acceptCFEnd	rw	22	1'b0	Accept CF-End If this bit is not set, CF-End or CF-End + CF-ACK subtype frames are not passed to SW. Note that this setting is also used for a CF-End frame carried in a Control Wrapper frame.
acceptOtherCntrlFrames	rw	23	1'b0	Accept Other Control Frames If this bit is not set, other Control subtype frames which are not explicitly covered (like *acceptACK*, *acceptCTS*) are not passed to SW.
acceptData	rw	24	1'b1	Accept Data If this bit is not set, Data, Data + CF-ACK, Data + CF-Poll, Data + CF-ACK + CF-Poll subtype frames are not passed to SW.
acceptCFWOData	rw	25	1'b0	Accept CF WithOut Data If this bit is not set, CF-ACK, CF-Poll, CF-ACK + CF-Poll subtype frames are not passed to SW.
acceptQData	rw	26	1'b1	Accept QoS Data If this bit is not set, QoS Data, QoS Data + CF-ACK, QoS Data + CF-Poll, QoS Data + CF-ACK + CF-Poll subtype frames are not passed to SW.
acceptQCFWOData	rw	27	1'b0	Accept QoS CF WithOut Data If this bit is not set, QoS CF-Poll, QoS CF-ACK + CF-Poll subtype frames are not passed to SW.
acceptQoSNull	rw	28	1'b1	Accept QoS Null If this bit is not set, QoS Null subtype frames are not passed to SW.
acceptOtherDataFrames	rw	29	1'b0	Accept Other Data Frames If this bit is not set, other Data subtype frames which are not explicitly covered (like *acceptQData*, *acceptData*) are not passed to SW.
acceptUnknownType	rw	30	1'b0	Accept Unknown Type field If this bit is not set, any frame containing unknown Type is not passed to SW.

Field name	rwu	Bit #	Reset	Description
enDuplicateDetection	r	31	1'b0	Enable Duplicate Detection *Reserved for future use.*

7.4.26 Beacon Control 1 Register (bcnCntrl1Reg)

Offset Address: 0x0064

This register is programmed with fields that control Beacon transmission. This register should be written to only when the core is in the IDLE state.

Field name	rwu	Bit #	Reset	Description
beaconInt	rw	15:0	16'b0	Beacon Interval This field contains the beacon interval in TUs.
impTBTTPeriod	rw	22:16	`IMP_TBTT_TIME	Impending TBTT Period This register is in terms of 128μs or 1μs. The core will raise the impending TBTT interrupt *impTBTTPeriod* μs before TBTT occurs.
impTBTTIn128Us	rw	23	1'b0	Impending TBTT Interrupt Period in 128μs When this bit is set, the *impTBTTPeriod* is in terms of 128μs, else it is in terms of 1μs.
noBcnTxTime	rw	31:24	`NO_BCN_TX_TIME	No Beacon Transmit Time This register is in units of 16 μs. This is the duration before TBTT during which the core will not transmit a beacon since the transmission of the beacon so close to the next TBTT will make the beacon cross the TBTT. The duration programmed in this register should be the time the beacon will take on air.

7.4.27 Beacon Control 2 Register (bcnCntrl2Reg)

Offset Address: 0x0068

This register is programmed with fields that control Beacon transmission and reception. This register should be written to only when the core is in the IDLE state.

Field name	rwu	Bit #	Reset	Description
bcnUpdateOffset	rw	7:0	8'b0	Beacon Update Offset This field is used to locate the *DTIM Count* field in the Beacon. It indicates to the hardware the offset, from the first byte of the Beacon frame, at which the *DTIM Count* field is located. *DTIM Count* field is updated by the HW with the latest values when transmitting a Beacon or Probe Response.

Field name	rwu	Bit #	Reset	Description
timOffset	rw	15:8	8'b0	TIM Offset This field is used to locate the TIM field in the received Beacon frame. It indicates to the HW the offset, from the first byte of the Beacon frame, at which the first byte of the *TIM* field is located. Refer section *2.1.3.2.1.2, Auto wake-up mode - legacy*. Reserved for future use.
aid	rw	31:16	16'b0	Association ID Contains the AID of this device. It is used in case of NPD Annoucement reception.

7.4.28 General Interrupt Event Register (genIntEventReg)

Offset Address: 0x806C (*genIntEventSetReg*) and 0x8070 (*genIntEventClearReg*)

This register reflects the state of the various interrupt sources. The interrupt bits are set by HW on an asserting edge of the corresponding interrupt signal provided the corresponding bit in the *genIntUnmaskReg* register is set. Each bit is also set by HW when SW performs a write with a value "1" to the corresponding bit in the *genIntEventSetReg* address, for debug purposes. Each bit is reset by HW when SW performs a write with a value "1" to the corresponding bit in the *genIntEventClearReg* address.

When any bit in this register is set, the *interruptGen*_n line is asserted. When all the bits in this register are cleared, the *interruptGen*_n line is deasserted.

Field name	rscu	Bit #	Reset	Description
impPriTBTT	rscu	0	1'b0	Impending Primary Target Beacon Transmission Time The core sets this bit *Beacon Control 1 Register (bcnCntrl1Reg).impTBTTPeriod* microsecond before the Primary TBTT. This is used by SW to chain the Primary Beacon frame for delivery at the Primary TBTT. This interrupt is suppressed for the Primary DTIM Beacon since the *impPriDTIM* interrupt will be generated.
impSecTBTT	rscu	1	1'b0	Impending Secondary Target Beacon Transmission Time The core sets this bit *Beacon Control 1 Register (bcnCntrl1Reg).impTBTTPeriod* microsecond before the Secondary TBTT. This is used by SW to chain the Secondary Beacon frame for delivery at the Secondary TBTT. This interrupt is suppressed for the Secondary DTIM Beacon since the *impSecDTIM* interrupt will be generated.
idleInterrupt	rscu	2	1'b0	Idle Interrupt The core sets this bit whenever the *State Control Register (stateCntrlReg).currentState* transitions to IDLE state.

Field name	rscu	Bit #	Reset	Description
absGenTimer	ru	3	1'b0	Absolute General Timer Interrupt The HW sets this bit everytime one of the absolute timers expires. The SW cannot set or clear this field. It shall use the *Timers Interrupt Event Register (timersIntEventReg)*.
Reserved	r	4	1'b0	*Reserved*
coexEvent	rscu	5	1'b0	Coex Interface Event
macPHYIFOverFlow	rscu	6	1'b0	MAC PHY InterFace Over Flow This interrupt is raised when the hardware detects that an overflow has occurred at the MPIF FIFO when receiving a PPDU from the PHY.
rxDMAEmpty	rscu	7	1'b0	Receive DMA Empty The core sets this bit when it finds that the *nextHDPRx* in the current Header Descriptor is null, indicating no more Header Descriptors are queued. The core sets this bit when it finds that the *nextPBufDPRx* in the current Buffer Descriptor is null, indicating no more Buffer Descriptors are queued.
rxFIFOOverFlow	rscu	8	1'b0	Receive FIFO Over Flow The core sets this bit whenever an Over Flow of Receive FIFO occurs.
olbcOFDM	rscu	9	1'b0	OFDM OLBC Indicates that *Overlapping Legacy BSS Condition Register (olbcReg).ofdmCount* number of OFDM frames not belonging to this BSS were received in *olbcReg.olbcTimer* µs.
olbcDSSS	rscu	10	1'b0	DSSS OLBC Indicates that the *Overlapping Legacy BSS Condition Register (olbcReg).dsssCount* of DSSS/CCK frames not belonging to this BSS were received in *olbcReg.olbcTimer* µs.
timSet	rscu	11	1'b0	TIM Set Indicates that the TIM bit for this device is set in the last received Beacon. The HW parses the received Beacon to determine the status of the TIM bit. Reserved for future use.
ptError	rscu	12	1'b0	Policy Table Error The core sets this bit when it finds that the *uPatternPT* read from the Policy Table did not contain the expected pattern.

Field name	rscu	Bit #	Reset	Description
ac0TxDMADead	rscu	13	1'b0	AC0 Transmit DMA Dead This interrupt is raised when the AC_BK DMA channel moves to the DEAD state. Refer to section *9.3*, *Terminal DMA errors* for details.
ac1TxDMADead	rscu	14	1'b0	AC1 Transmit DMA Dead This interrupt is raised when the AC_BE DMA channel moves to the DEAD state. Refer to section *9.3*, *Terminal DMA errors* for details.
ac2TxDMADead	rscu	15	1'b0	AC2 Transmit DMA Dead This interrupt is raised when the AC_VI DMA channel moves to the DEAD state. Refer to section *9.3*, *Terminal DMA errors* for details.
ac3TxDMADead	rscu	16	1'b0	AC3 Transmit DMA Dead This interrupt is raised when the AC_VO DMA channel moves to the DEAD state. Refer to section *9.3*, *Terminal DMA errors* for details.
bcnTxDMADead	rscu	17	1'b0	Beacon Transmit DMA Dead This interrupt is raised when the Beacon DMA channel moves to the DEAD state. Refer to section *9.3*, *Terminal DMA errors* for details.
impPriDTIM	rscu	18	1'b0	Impending Primary DTIM The core sets this bit *Beacon Control 1 Register (bcnCntrl1Reg).impTBTTPeriod* microsecond before the Primary TBTT at which the DTIM count will be zero. This is used by SW to chain buffered broadcast and multicast frames for delivery after the Primary Beacon.
impSecDTIM	rscu	19	1'b0	Impending Secondary DTIM The core sets this bit *Beacon Control 1 Register (bcnCntrl1Reg).impTBTTPeriod* microsecond before the Secondary TBTT at which the DTIM count will be zero. This is used by SW to chain buffered broadcast and multicast frames for delivery after the Secondary Beacon.
hwErr	rscu	20	1'b0	Hardware Error This interrupt is raised when the hardware detects an unrecoverable error and needs to initiate software error recovery procedure. Refer to section *9.4*, *Terminal hardware errors*.
macPHYIFUnderRun	rscu	21	1'b0	MAC PHY InterFace Under Run This interrupt is raised when the hardware detects that an underrun has occurred when transferring a frame to the PHY.

Field name	rscu	Bit #	Reset	Description
phyErr	rscu	22	1'b0	PHY Error This interrupt is raised when a PHY Error is reported by the PHY.
phyRxStart	rscu	23	1'b0	PHY Receive Start indication This interrupt is raised to indicate the occurrence of the PHY-RXSTART.indication. This interrupt can be used in Active Scan, so that SW can switch over from the *MinChannelTime* to *MaxChannelTime*.
rxHeaderDMADead	rscu	24	1'b0	Receive Payload DMA Dead This interrupt is raised when the Receive Header DMA channel moves to the DEAD state. Refer to section *9.3*, *Terminal DMA errors* for details.
rxPayloadDMADead	rscu	25	1'b0	Receive Payload DMA Dead This interrupt is raised when the Receive Payload DMA channel moves to the DEAD state. Refer to section *9.3*, *Terminal DMA errors* for details.
tbTxDMADead	rscu	26	1'b0	Trigger Based Transmit DMA Dead This interrupt is raised when the Trigger Based DMA channel moves to the DEAD state. Refer to section *9.3*, *Terminal DMA errors* for details.
Reserved	r	30:27	4'b0	Reserved
Reserved	r	31	1'b0	Always reserved for the *masterGenIntEn* bit.

7.4.29 General Interrupt UnMask Register (genIntUnMaskReg)

Offset Address: 0x8074

The bits in this register have the same format as the *genIntEventSetReg* register, with the addition of *masterGenIntEn* (bit 31). A one bit in the *genIntUnmaskReg* register enables the corresponding bit in the *genIntEventSetReg* register to generate a processor interrupt. A zero bit in the *genIntUnMask* register disables the corresponding *genIntEventSetReg* register bit from getting set and hence generating an interrupt.

If *masterGenIntEn* is 0, all interrupts are disabled regardless of the values of all other bits in the *genIntUnmaskReg* register. The value of *masterGenIntEn* has no effect on the value returned by reading the *genIntEventSetReg* or *genIntEventClearReg*. Even if *masterGenIntEn* is 0, reading *genIntEventSetReg* or the *genIntEventClearReg* will return the status of the interrupt bits.

Field name	rwu	Bit #	Reset	Description
Unmask bits for Interrupts defined in bits 30:0 in *genIntEventSetReg* register	rw	30:0	31'b0	See *General Interrupt Event Register (genIntEventReg)* definition.
masterGenIntEn	rw	31	1'b0	If set, external interrupts will be generated in accordance with the rest of the *genIntUnMaskReg* register bits. If clear, no external interrupts will be generated regardless of the *genIntUnMaskReg* register bit settings.

7.4.30 Transmit / Receive Interrupt Event Register (txRxIntEventReg)

Offset Address: 0x8078 (*txRxIntEventSetReg*) and 0x807C (*txRxIntEventClearReg*)

This register reflects the state of the various interrupt sources. The interrupt bits are set by HW on an asserting edge of the corresponding interrupt signal provided the corresponding bit in the *txRxIntUnmaskReg* register is set. Each bit is also set by HW when SW performs a write with a value "1" to the corresponding bit in the *txRxIntEventSetReg* address, for debug purposes. Each bit is reset by HW when SW performs a write with a value "1" to the corresponding bit in the *txRxIntEventClearReg* address.

The following *protocol triggers* can be generated before the start of the TXOP on air: *ac0/1/2/3ProtTrigger* or the start of a Trigger Based transmission (*tbProtTrigger)*. Refer section *2.2.4.2.1*, *Generating the early protocol trigger* for more details.

A *transmission trigger* (e.g. *ac0TxTrigger*) is generated when the *interruptEnTx* bit is set in any *Transmit DMA Header Descriptor* that is handled by the HW.

A *transmission buffer trigger* (e.g. *ac0TxBufTrigger*) is generated when the *interruptEnTx* bit is set in any *Transmit DMA Header Descriptor* that is handled by the HW.

Multiple interrupt lines are used to group similar interrupts together to allow the use of a vectored interrupt controller.

The *intProtTrigger_n* line is asserted when the following bits in this register are set: *ac0ProtTrigger*, *ac1ProtTrigger*, *ac2ProtTrigger*, *ac3ProtTrigger*, *tbProtTrigger*, *rdProtTrigger*. When those bits are cleared, the *intProtTrigger_n* line is deasserted.

The *intTxTrigger_n* line is asserted when the following bits in this register are set: *ac0TxTrigger*, *ac1TxTrigger*, *ac2TxTrigger*, *ac3TxTrigger*, *bcnTxTrigger*, *tbTxTrigger*, *tbTxCancelled or secUserTxTrigger*. When those bits are cleared, the *intTxTrigger_n* line is deasserted.

Note that the *secUserTxTrigger* interrupt is cleared in *Secondary Users Transmit Interrupt Event Register (secUsersTxIntEventReg)* register.

The *intRxTrigger_n* line is asserted when the following bits in this register are set: *rxBuffer1Trigger*, *rxBuffer2Trigger* When those bits are cleared, the *intRxTrigger_n* line is deasserted.

The *intTxRxMisc_n* line is asserted when the following bit in this register is set: *txopComplete*. When this bit is cleared, the *intTxRxMisc_n* line is deasserted.

The *intTxRxTimer_n* line is asserted when the following bits in this register are set: *timerTxTrigger*, *timerRxTrigger*. When those bits are cleared, the *intTxRxTimer_n* line is deasserted.

Field name	rscu	Bit #	Reset	Description
ac0ProtTrigger	rscu	0	1'b0	AC0 Protocol Trigger This interrupt signals the imminent start of a TXOP that is acquired for AC_BK.
ac1ProtTrigger	rscu	1	1'b0	AC1 Protocol Trigger This interrupt signals the imminent start of a TXOP that is acquired for AC_BE.
ac2ProtTrigger	rscu	2	1'b0	AC2 Protocol Trigger This interrupt signals the imminent start of a TXOP that is acquired for AC_VI.

Field name	rscu	Bit #	Reset	Description
ac3ProtTrigger	rscu	3	1'b0	AC3 Protocol Trigger This interrupt signals the imminent start of a TXOP that is acquired for AC_VO.
tbProtTrigger	r	4	1'b0	TB Protocol Trigger This interrupt signals the imminent start of a Triger Based transmission and signals to SW to chain frames on TB Channel.
rdProtTrigger	r	5	1'b0	Reverse Direction Protocol Trigger This interrupt signals the imminent start of a TXOP that is acquired for RD and signals to SW to chain frames for the RDG. Reserved for future use.
ac0TxTrigger	rscu	6	1'b0	AC0 Transmission Trigger This interrupt indicates that a frame from the AC_BK channel has been transmitted. This interrupt is enabled from the THD of the frame.
ac1TxTrigger	rscu	7	1'b0	AC1 Transmission Trigger This interrupt indicates that a frame from the AC_BE channel has been transmitted. This interrupt is enabled from the THD of the frame.
ac2TxTrigger	rscu	8	1'b0	AC2 Transmission Trigger This interrupt indicates that a frame from the AC_VI channel has been transmitted. This interrupt is enabled from the THD of the frame.
ac3TxTrigger	rscu	9	1'b0	AC3 Transmission Trigger This interrupt indicates that a frame from the AC_VO channel has been transmitted. This interrupt is enabled from the THD of the frame.
bcnTxTrigger	rscu	10	1'b0	Beacon Transmission Trigger This interrupt indicates that a frame from the Beacon channel has been transmitted. This interrupt is enabled from the THD of the frame.
tbTxTrigger	rscu	11	1'b0	TB Transmission Trigger This interrupt indicates that a frame from the Trigger Based channel has been transmitted. This interrupt is enabled from the THD of the frame.
tbTxCancelled	rscu	12	1'b0	TB Transmission Cancelled This interrupt indicates that transmission of Trigger Based channel has been cancelled by the HW.
txopComplete	rscu	13	1'b0	TXOP Complete This interrupt indicates that the TXOP on air has completed for ACs with non zero TXOPLimit.

Field name	rscu	Bit #	Reset	Description
timerTxTrigger	rscu	14	1'b0	Timer Transmission Trigger This interrupt is generated based on two timers that are programmed in the *Transmit Trigger Timer Register (txTriggerTimerReg)*.
secUserTxTrigger	r	15	1'b0	Secondary User Transmission Trigger This interrupt indicates that a transmit interrupt from one of the secondary users is pending in the *Secondary Users Transmit Interrupt Event Register (secUsersTxIntEventReg)*.
rxBuffer1Trigger	rscu	16	1'b0	Receive Buffer1 Trigger This interrupt is generated to inform the SW that frames have been received and moved into the Buffer 1 in system memory.
timerRxTrigger	rscu	17	1'b0	Timer Receive Trigger This interrupt is generated based on two timers that are programmed in the *Receive Trigger Timer Register (rxTriggerTimerReg)*.
rxBuffer2Trigger	rscu	18	1'b0	Receive Buffer 2 Trigger This interrupt is generated to inform the SW that frames have been received and moved into the Buffer 2 in system memory. "
counterRxTrigger	rscu	19	1'b0	Counter Receive Trigger This interrupt is generated based on the counter that is programmed in the *Receive Trigger Timer Register (rxTriggerTimerReg)*.
ac0BWDropTrigger	rscu	20	1'b0	AC0 Bandwith Drop Trigger This interrupt indicates that a bandwidth drop has been decided during a transmission on the AC_BK channel.
ac1BWDropTrigger	rscu	21	1'b0	AC1 Bandwith Drop Trigger This interrupt indicates that a bandwidth drop has been decided during a transmission on the AC_BE channel.
ac2BWDropTrigger	rscu	22	1'b0	AC2 Bandwith Drop Trigger This interrupt indicates that a bandwidth drop has been decided during a transmission on the AC_VI channel.
ac3BWDropTrigger	rscu	23	1'b0	AC3 Bandwith Drop Trigger This interrupt indicates that a bandwidth drop has been decided during a transmission on the AC_VO channel.
ac0TxBufTrigger	rscu	24	1'b0	AC0 Transmission Buffer Trigger This interrupt indicates that a Payload Buffer from the AC_BK channel has been transmitted. This interrupt is enabled from the TPD of the frame.

Field name	rscu	Bit #	Reset	Description
ac1TxBufTrigger	rscu	25	1'b0	AC1 Transmission Buffer Trigger This interrupt indicates that a Payload Buffer from the AC_BE channel has been transmitted. This interrupt is enabled from the TPD of the frame.
ac2TxBufTrigger	rscu	26	1'b0	AC2 Transmission Buffer Trigger This interrupt indicates that a Payload Buffer from the AC_VI channel has been transmitted. This interrupt is enabled from the TPD of the frame.
ac3TxBufTrigger	rscu	27	1'b0	AC3 Transmission Buffer Trigger This interrupt indicates that a Payload Buffer from the AC_VO channel has been transmitted. This interrupt is enabled from the TPD of the frame.
bcnTxBufTrigger	rscu	28	1'b0	Beacon Transmission Buffer Trigger This interrupt indicates that a frame from the Beacon channel has been transmitted.
tbTxBufTrigger	rscu	29	1'b0	TB Transmission Buffer Trigger This interrupt indicates that a frame from the TB channel has been transmitted.
Reserved	r	30	1'b0	*Reserved*
Reserved	r	31	1'b0	Always reserved for the *masterTxRxIntEn* bit.

7.4.31 Transmit / Receive Interrupt UnMask Register (txRxIntUnMaskReg)

Offset Address: 0x8080

The bits in this register have the same format as the *txRxIntEventSetReg* register, with the addition of *masterTxRxIntEn* (bit 31). A one bit in the *txRxIntUnmaskReg* register enables the corresponding bit in the *genIntEventSetReg* register to generate a processor interrupt. A zero bit in the *txRxIntUnMask* register disables the corresponding *txRxIntEventSetReg* register bit from getting set and hence generating an interrupt.

If *masterTxRxIntEn* is 0, all interrupts are disabled regardless of the values of all other bits in the *txRxIntUnmaskReg* register. The value of *masterTxRxIntEn* has no effect on the value returned by reading the *txRxIntEventSetReg* or *txRxIntEventClearReg*. Even if *masterTxRxIntEn* is 0, reading *txRxIntEventSetReg* or the *txRxIntEventClearReg* will return the status of the interrupt bits.

Field name	rwu	Bit #	Reset	Description
Unmask bits for Interrupts defined in bits 30:0 in *txRxIntEventSetReg* register	rw	30:0	31'b0	See *Transmit / Receive Interrupt Event Register (txRxIntEventReg)* definition.
masterTxRxIntEn	rw	31	1'b0	If set, external interrupts will be generated in accordance with the rest of the *txRxIntUnMaskReg* register bits. If clear, no external interrupts will be generated regardless of the *txRxIntUnMaskReg* register bit settings.

7.4.32 Timers Interrupt Event Register (timersIntEventReg)

Offset Address: 0x8084 (*timersIntEventSetReg*) and 0x8088 (*timersIntEventClearReg*)

This register reflects the state of the absoluter timers interrupt sources. The interrupt bits are set by HW on an asserting edge of the corresponding interrupt signal provided the corresponding bit in the *timersIntUnmaskReg* register is set. Each bit is also set by HW when SW performs a write with a value "1" to the corresponding bit in the *timersIntEventSetReg* address, for debug purposes. Each bit is reset by HW when SW performs a write with a value "1" to the corresponding bit in the *timersIntEventClearReg* address.

Field name	rscu	Bit #	Reset	Description
absTimers	rscu	9:0	10'b0	Absolute Timers Each time the value programmed in *Absolute Timer 1 Register (absTimer1Reg)*.*absTimerXValue* matches matches .*monotonicCounter2*, the corresponding bit of asbTimers is set and an interrupt is generated. absTimers[0] is set when monotonicCounter2 reaches absTimer0Value absTimers[1] is set when monotonicCounter2 reaches absTimer1Value … absTimers[9] is set when monotonicCounter2 reaches absTimer9Value
Reserved	r	31:20	12'b0	Reserved

7.4.33 Timers Interrupt UnMask Register (timersIntUnMaskReg)

Offset Address: 0x808C

The bits in this register have the same format as the *timersIntEventSetReg* register. A one bit in the *timersIntUnMaskReg* register enables the corresponding bit in the *timerIntEventSetReg* register to generate a processor interrupt. A zero bit in the *timersIntUnMaskReg* register disables the corresponding *timersIntEventSetReg* register bit from getting set and hence generating an interrupt.

Field name	rwu	Bit #	Reset	Description
Unmask bits for Interrupts defined in bits 30:0 in *timersIntEventSetReg* register	rw	30:0	31'b0	See *Timers Interrupt Event Register (timersIntEventReg)* definition.
masterTxRxIntEn	rw	31	1'b0	If set, external interrupts will be generated in accordance with the rest of the *timersIntUnMaskReg* register bits. If clear, no external interrupts will be generated regardless of the *timersIntUnMaskReg* register bit settings.

7.4.34 DTIM 1 Register (dtim1Reg)

Offset Address: 0x0090

This register contains the parameters necessary to support power save mode. This register should be written to only when the core is in the IDLE state.

Field name	rwu	Bit #	Reset	Description
dtimPeriod	rwu	7:0	8'b0	DTIM Period The DTIM Period field indicates the number of Beacon intervals between successive DTIMs. As an AP, software updates this field in IDLE state. As a STA, HW updates this field from any received Beacon or Probe Response frame which contains the required BSSID. The SSID of the received Beacon or Problem Response frame is not checked in HW. Hence parameters which are extracted by the MAC HW may be taken from a Beacon or Probe Response that does not contain the correct SSID, but contains the correct BSSID.
Reserved	rw	30:8	23'b0	Reserved
dtimUpdatedBySW	rw	31	1'b0	DTIM Updated by SW This bit is set by SW after the contents of this register have been written to by SW. The bit is cleared by HW after the contents of this register have been processed by HW.

7.4.35 Retry Limits Register (retryLimitsReg)

Offset Address: 0x0098

Field name	rwu	Bit #	Reset	Description
dot11ShortRetryLimit	rw	7:0	8'd7	Dot11 Short Retry Limit Contains the value of the MIB attribute dot11ShortRetryLimit.
dot11LongRetryLimit	rw	15:8	8'd4	Dot11 Long Retry Limit Contains the value of the MIB attribute dot11LongRetryLimit.
Reserved	r	31:16	16'b0	Reserved

7.4.36 Baseband Service Register (bbService)

Offset Address: 0x009C

Field name	rwu	Bit #	Reset	Description

Field name	rwu	Bit #	Reset	Description
bbServiceA	rw	15:0	16'b0	This field contains the service field that is to be sent to the baseband. The 16 bits from this field are transmitted to the BB in the 16 bits of the Service field of the Tx-Vector when the modulation is OFDM.
bbServiceB	rw	23:16	8'b0	This field contains the service field that is to be sent to the baseband. The 8 bits from this field are transmitted to the BB in the lower 8 bits of the Service field of the Tx Vector when the modulation is DSSS/CCK. Only the "Locked Clock" bit (bbServiceB[2]) should be programmed from software. The "Length Extension" bit (bbServiceB[7]) should be updated by the Modem before transmitting the Service field on air.
maxPHYNtx	rw	28:26	3'd1	Maximum number of PHY Transmit Chains This field indicated how many Transmit Chains are available in the attached PHY. It is used to fill the corresponding TxVector parameter for HW formed control response frames.
Reserved	r	31:29	5'h0	Reserved

7.4.37 Maximum Power Level Register (maxPowerLevelReg)

Offset Address: 0x00A0

Field name	rwu	Bit #	Reset	Description
ofdmMaxPwrLevel	rw	7:0	8'h0F	OFDM Maximum Power Level This field contains the maximum Tx Power for the regulatory domain in which the MAC is operating. This power level will be used for the hardware OFDM generated frames and during the computation of Tx Power in case of Trigger Based generation. The coding of this field is 2's Complement : 8'h80 : -128 dBm 8'hFF : -1 dBm 8'h00 : 0 dBm 8'h01 : 1 dBm 8'h3F : 127dBm
dsssMaxPwrLevel	rw	15:8	8'h0F	DSSS Maximum Power Level This field contains the maximum Tx Power for the regulatory domain in which the MAC is operating. This power level will be used for the hardware DSSS/CCK generated frames. This field is encoded as ofdmMaxPwrLevel
ofdmMinPwrLevel	rw	23:16	8'h00	OFDM Minimum Power Level This field contains the minimum Tx Power for the regulatory domain in which the MAC is operating.

Field name	rwu	Bit #	Reset	Description
				This power level is used during the computation of Tx Power in case of Trigger Based generation. This field is encoded as ofdmMaxPwrLevel
Reserved	-	31:24	8'b0	*Reserved*

7.4.38 TSF Timer Low Register (tsfLoReg)

Offset Address: 0x80A4

This register is updated from the TSF in a received Beacon as a STA. As an AP, it starts from 0. It is a free running counter and increments every microsecond (µs).

Field name	rwu	Bit #	Reset	Description
tsfTimer[31:0]	rwu	31:0	32'b0	TSF Timer This field contains the 32 LSBs of the Timing Synchronization Function timer in microseconds. SW can write into this register only for debug purposes.

7.4.39 TSF Timer High Register (tsfHiReg)

Offset Address: 0x80A8

This register is updated from the TSF in a received Beacon as a STA. As an AP, it starts from 0. It is a free running counter and increments every microsecond (µs).

Field name	rwu	Bit #	Reset	Description
tsfTimer[63:32]	rwu	31:0	32'b0	TSF Timer This register contains the 32 MSBs of the Timing Synchronization Function timer in microseconds. SW can write into this register only for debug purposes.

7.4.40 Encryption Key Word 0 Register (encrKey0Reg)

Offset Address: 0x00AC

Reset Value: 32'b0

Field name	rwu	Bit #	Reset	Description
encrKeyRAM[31:0]	rwu	31:0	32'b0	Encryption Key for RAM Contains lower 32-bits of the encryption key to be programmed into the Key Storage RAM.

7.4.41 Encryption Key Word 1 Register (encrKey1Reg)

Offset Address: 0x00B0

Field name	rwu	Bit #	Reset	Description

Field name	rwu	Bit #	Reset	Description
encrKeyRAM[63:32]	rwu	31:0	32'b0	Encryption Key for RAM It contains bits 39 to 32 of the 40-bit WEP key to be programmed in the Key Storage RAM. The rest of the bits are reserved. It contains bits 63 to 32 of the 104-bit WEP key or the TKIP/CCMP key to be programmed in the key storage RAM.

7.4.42 Encryption Key Word 2 Register (encrKey2Reg)

Offset Address: 0x00B4

Field name	rwu	Bit #	Reset	Description
encrKeyRAM[95:64]	rwu	31:0	32'b0	Encryption Key for RAM It contains bits 95 to 64 of the 104-bit WEP key or the TKIP/CCMP key to be programmed in the Key Storage RAM.

7.4.43 Encryption Key Word 3 Register (encrKey3reg)

Offset Address: 0x00B8

Reset Value: 32'b0

Field name	rwu	Bit #	Reset	Description
encrKeyRAM[127:96]	rwu	31:0	32'b0	Encryption Key for RAM It contains bits 103 to 96 of the 104-bit WEP key to be programmed in the Key Storage RAM. The rest of the bits are reserved. It contains bits 127 to 96 of the TKIP/CCMP key to be programmed in the Key Storage RAM.

7.4.44 Encryption MAC Address Low Register (encrMACAddrLowReg)

Offset Address: 0x00BC

Field name	rwu	Bit #	Reset	Description
macAddrRAM[31:0]	rwu	31:0	32'hFFFFFFFF	MAC Address for RAM This field contains bits 31 to 0 of the 48-bit MAC address corresponding to the secret key programmed in the encryption registers. It is reserved when the *keyIndexRAM* points to a default key location.

7.4.45 Encryption MAC Address High Register (encrMACAddrHighReg)

Offset Address: 0x00C0

Field name	rwu	Bit #	Reset	Description

Field name	rwu	Bit #	Reset	Description
macAddrRAM[47:32]	rwu	15:0	16'hFFFF	MAC Address for RAM This field contains bits 47 to 32 of the 48-bit MAC address corresponding to the secret key programmed in the encryption registers. It is reserved when the *keyIndexRAM* points to a defauly key location.
Reserved	r	31:16	16'b0	Reserved

7.4.46 Encryption Control Register (encrCntrlReg)

Offset Address: 0x00C4

Field name	rwu	Bit #	Reset	Description
Reserved	r	0	1'b0	Reserved
useDefKeyRAM	rwu	1	1'b0	Use Default Key for RAM This field indicates that the Default Key for this MAC address must be used instead of the decryption key programmed at this RAM location.
sppRAM	rwu	3:2	2'b0	SPP for RAM Indicates the action to be taken on transmission and reception for A-MSDU frames. The encoding of this field is as follows: 01 – PP A-MSDU 10 – SPP A-MSDU 00, 11 – Discard received A-MSDU
vlanIDRAM	rwu	7:4	4'b0	Virtual LAN ID for RAM This field indicates the VLAN to which this particular address belongs to and is used to determine the Default Key to be used for decryption.
cTypeRAM	rwu	10:8	2'b0	Cipher Type for RAM This field contains the cipher type of the key being programmed. The encoding of this field is as follows: 000 – Null key 001 – WEP 010 – TKIP 011 – CCMP 100 – WPI 101 – GCMP
Reserved	R	11:10	2'b0	Reserved

Field name	rwu	Bit #	Reset	Description
cLenRAM	rwu	13:12	2'b0	Cipher Length for RAM This field contains the cipher length of the WEP key being programmed. The encoding of this field is as follows: 0 - 64-bit encryption 1 - 128-bit encryption It shall be forced to 1 when the cTypeRAM indicates TKIP. This field contains the cipher length of the CCMP/GCMP key being programmed. The encoding of this field is as follows: 0 - 128-bit encryption 2 - 256-bit encryption
Reserved	r	15:10	6'b0	Reserved
keyIndexRAM	rwu	25:16	10'b0	Key Index RAM This field is programmed with the index of the Key Storage RAM into which the contents of the encryption registers are programmed. In case of NewSearch, this field returns the index which matches the defined macAddrRAM.
Reserved	r	27:26	2'b0	Reserved
searchError	r	28	1'b0	Search Error When a SW search is requested but the defined macAddrRAM does not match any of the mac Address contained in keyRAM, this bit is set. It is valid only if newSearch is low.
newSearch	rwu	29	1'b0	New Search The software uses this field to search in the Key Storage RAM the index which matches the defined macAddrRAM. Once completed, this bit is cleared by the Core.
newWrite	rwu	30	1'b0	New Write Once the software has completed programming the encryption registers, it should set this bit to indicate to the core that the contents of the encryption registers should be copied into the Key Storage RAM at the location indicated by the *keyIndexRAM* field. Once the core completes the write into the Key Storage RAM, it resets this bit. Software shall write into any of the encryption registers only if this bit is 0 and the *newRead* bit is 0.

Field name	rwu	Bit #	Reset	Description
newRead	rwu	31	1'b0	New Read The SW can read back any location of the Key Storage RAM for debugging purposes. To do this it should assert the *debugKSR* signal on the interface and write the *keyRamIndex* field and set this bit. The HW reads the indicated RAM location and copies the contents from the RAM into the corresponding encryption registers fields and resets this bit. Software shall write into any of the encryption registers only if this bit is 0 and the *newWrite* bit is 0.

7.4.47 Encryption Extended or WPI integrity Key Word 0 Register (encrWPIIntKey0Reg)

Offset Address: 0x00C8

Reset Value: 32'b0

Field name	rwu	Bit #	Reset	Description
encrIntKeyRAM[31:0]	rwu	31:0	32'b0	Encryption Extended or WPI integrity Key for RAM Contains lower 32-bits of the integrity key needed for WPI reads or to be programmed into the Key Storage RAM. Contains the bits 159-128 of the Encryption Key required for CCMP/GCMP-256bits.

7.4.48 Encryption Extended or WPI integrity Key Word 1 Register (encrWPIIntKey1Reg)

Offset Address: 0x00CC

Field name	rwu	Bit #	Reset	Description
encrIntKeyRAM[63:32]	rwu	31:0	32'b0	Encryption Extended or WPI integrity Key for RAM Contains bits 63 to 32 of the integrity key needed for WPI to be programmed into the Key Storage RAM. Contains the bits 191-160 of the Encryption Key required for CCMP/GCMP-256bits.

7.4.49 Encryption Extended or WPI integrity Key Word 2 Register (encrWPIIntKey2Reg)

Offset Address: 0x00D0

Field name	rwu	Bit #	Reset	Description

Field name	rwu	Bit #	Reset	Description
encrIntKeyRAM[95:64]	rwu	31:0	32'b0	Encryption Extended or WPI integrity Key for RAM Contains bits 95 to 64 of the integrity key needed for WPI to be programmed into the Key Storage RAM. Contains the bits 223-192 of the Encryption Key required for CCMP/GCMP-256bits.

7.4.50 Encryption Extended or WPI integrity Key Word 3 Register (encrWPIIntKey3Reg)

Offset Address: 0x00D4

Reset Value: 32'b0

Field name	rwu	Bit #	Reset	Description
encrIntKeyRAM[127:96]	rwu	31:0	32'b0	Encryption Extended or WPI integrity Key for RAM Contains bits 127 to 96 of the integrity key needed for WPI to be programmed into the Key Storage RAM. Contains the bits 255-224 of the Encryption Key required for CCMP/GCMP-256bits.

7.4.51 Encryption RAM Configuration Register (encrRAMConfigReg)

Offset Address: 0x00D8

Field name	rwu	Bit #	Reset	Description
staKeyStartIndex [7:0]	rw	7:0	8'd24	STA Key Start Index This field configures the index of the first STA in the Key Storage RAM. **Note**: staKeyStartAddr shall be greater or equal to nVAP*4.
staKeyEndIndex[7:0]	rw	15:8	8'd`RW_KEY_INDEX_MAX	STA Key End Index This field configures the index of the latest STA in the Key Storage RAM. **Note**: staKeyEndAddr shall be greater or equal to staKeyStartAddr. **Note**: staKeyEndAddr shall be lower or equal to staKeyMaxIndex.
nVAP[3:0]	rw	23:16	4'd6	Number of VAP This field configures the number of VAP.
staKeyMaxIndex[7:0]	r	31:24	8'd`RW_KEY_INDEX_MAX	STA Key Max Index This field contains the index max which can be used in the key Storage RAM. It corresponds to the depth of the Key Storage RAM.

7.4.52 Rates Register (ratesReg)

Offset Address: 0x00DC

Contains the various common rates for the BSS. HW uses this register to determine the transmission rate of the packets that are sent by HW. This register should be written to only when the core is in the IDLE state.

Field name	rwu	Bit #	Reset	Description
bssBasicRateSet	rw	11:0	12'h150	BSS Basic Rate Set This field is used to determine the transmission rates of some of the packets that are sent by hardware. A one-hot encoding scheme is followed for this field. Each bit position decodes to a particular PHY rate; the combined setting of the bits decide which rates belong to the BSS Basic Rate Set. The encoding is as follows: basicRateSet[0] indicates 1Mbps basicRateSet[1] indicates 2Mbps basicRateSet[2] indicates 5.5Mbps basicRateSet[3] indicates 11Mbps basicRateSet[4] indicates 6Mbps basicRateSet[5] indicates 9Mbps basicRateSet[6] indicates 12Mbps… basicRateSet[11] indicates 54Mbps Multiple values can be set at a time. If there is no BSS Basic Rate for DSS/CCK or OFDM available, SW must program the BSS Mandatory Rates for that modulation.
Reserved	r	31:12	20'b0	Reserved

7.4.53 Overlapping Legacy BSS Condition Register (olbcReg)

Offset Address: 0x00E0

This register is programmed with settings to detect legacy overlapping BSSes that cannot understand the modulations being used in this BSS and hence protection mechanisms should be used.

Field name	rwu	Bit #	Reset	Description
olbcTimer	rw	15:0	16'b0	OLBC Timer This register is programmed with a time in µs for checking if an OLBC condition exists. The internal OLBC timer continuously loads and counts down from this value.
ofdmCount	rw	23:16	8'b0	OFDM Count This register is programmed with the number of OFDM frames, not belonging to this BSS, which should be received in the time programmed in the field *olbcTimer*. If such a condition occurs, the HW raises the *General Interrupt Event Register (genIntEventReg).olbcOFDM* interrupt to SW. The count restarts when the OLBC timer expires.
dsssCount	rw	31:24	8'b0	DSSS/CCK Count This register is programmed with the number of

Field name	rwu	Bit #	Reset	Description
				DSS/CCK frames, not belonging to this BSS, which should be received in the time programmed in the field *olbcTimer*. If such a condition occurs, the HW raises the *General Interrupt Event Register (genIntEventReg).olbcDSSS* interrupt to SW. The count restarts when the OLBC timer expires.

7.4.54 Timings 1 Register (timings1Reg)

Offset Address: 0x00E4

This register should be written to only when the core is in the IDLE state.

Field name	rwu	Bit #	Reset	Description
macCoreClkFreq	rw	7:0	`MAC_FREQ	MAC Core Clock Frequency This is frequency of the MAC core clock in MHz and must be an integer value.
txRFDelayInMACClk	rw	17:8	`TX_RF_DLY x `MAC_FREQ	Transmit RF Delay in MAC Clocks This indicates the Transmit RF Delay in terms of MAC Core Clock cycles.
txChainDelayInMACClk	rw	27:18	`TX_CHAIN_DLY x `MAC_FREQ	Transmit Chain Delay in MAC Clocks This indicates the PHY Transmit Chain Delay in terms of MAC Core Clock cycles.
Reserved	r	31:28	4'b0	Reserved

7.4.55 Timings 2 Register (timings2Reg)

Offset Address: 0x00E8

This register should be written to only when the core is in the IDLE state.

Field name	rwu	Bit #	Reset	Description
slotTime	rw	7:0	`SHORT_SLOT	This field is programmed with the aSlotTime parameter. This field is in terms of 1µs.
slotTimeInMACClk	rw	23:8	`SHORT_SLOT x MAC_FREQ	This register should be programmed with the value of *macCoreClkFreq* * *slotTime*.
Reserved	r	31:24	8'b0	Reserved

7.4.56 Timings 3 Register (timings3Reg)

Offset Address: 0x00EC

This register should be written to only when the core is in the IDLE state.

Field name	rwu	Bit #	Reset	Description
macProcDelayInMACClk	rw	9:0	`MAC_PROC_DELAY x `MAC_FREQ	MAC Processing Delay in MAC clocks This register should be programmed with the value of *macCoreClkFreq* * the MAC Processing Delay.

Field name	rwu	Bit #	Reset	Description
Reserved	rw	19:10	`10'b0	Reserved
rxRFDelayInMACClk	rw	29:20	`RX_RF_DLY A x `MAC_FREQ	Receive RF Delay in MAC Clock This indicates the Receive RF delay in terms of MAC Core Clock cycles.
Reserved	r	31:30	2'b0	Reserved

7.4.57 Timings 4 Register (timings4Reg)

Offset Address: 0x00F0

This register should be written to only when the core is in the IDLE state.

Field name	rwu	Bit #	Reset	Description
wt2BBClkRatio	rw	1:0	2'b`WEP_2_BB _CLK_RATIO	Define the clock ratio between wtClk and macCryptClk 2'd0 : Reserved 2'd1 : wtClk = macCryptClk 2'd2 : wtClk = 2 x macCryptClk 2'd3 : wtClk = 3 x macCryptClk
Reserved	r	11:2	10'b0	Reserved
radioChirpTime	rw	21:12	`RADIO_CHIRP _TIME	Radio Chirp Time MAC uses this value to wait for the RF PLL to lock before starting the Active Scan process. This field is in terms of 32µs.
radioWakeUpTime	rw	31:22	`RADIOWAKEU P_TIME	Radio Wakeup Time MAC uses this value to switch on the BB earlier then data is expected on air when coming out of Power Save. This field is in terms of 32µs.

7.4.58 Timings 5 Register (timings5Reg)

Offset Address: 0x00F4

This register should be written to only when the core is in the IDLE state.

Field name	rwu	Bit #	Reset	Description
sifsB	rw	7:0	`SIFS_B	SIFS Duration for 802.11b This field should be programmed with the SIFS value for DSSS and CCK frames. This field is in terms of 1µs.
sifsBInMACClk	rw	23:8	`SIFS_B x `MAC_FREQ	This register should be programmed with the value of *macCoreClkFreq* * *sifsB*.
Reserved	r	31:24	8'b0	Reserved

7.4.59 Timings 6 Register (timings6Reg)

Offset Address: 0x00F8

This register should be written to only when the core is in the IDLE state.

Field name	rwu	Bit #	Reset	Description
sifsA	rw	7:0	`SIFS_A	SIFS Duration for 802.11a This field should be programmed with the SIFS value for OFDM and MIMO-OFDM frames. This field is in terms of 1µs.
sifsAInMACClk	rw	23:8	`SIFS_A x `MAC_FREQ	This register should be programmed with the value of *macCoreClkFreq* * *sifsA*.
Reserved	r	31:24	8'b0	Reserved

7.4.60 Timings 7 Register (timings7Reg)

Offset Address: 0x00FC

This register should be written to only when the core is in the IDLE state.

Field name	rwu	Bit #	Reset	Description
Reserved	rw	7:0	8'b0	Reserved
rxCCADelay	rw	11:8	`RX_CCA_DELAY	Receive Chain CCA Delay Indicates the PHY CCA Delay. This field is in terms of 1µs.
Reserved	r	31:12	20'b0	Reserved

7.4.61 Timings 8 Register (timings8Reg)

Offset Address: 0x0100

This register should be written to only when the core is in the IDLE state.

Field name	rwu	Bit #	Reset	Description
rxStartDelayOFDM	rw	7:0	8'd33	Receive Start Delay for OFDM This is the value of the aPHY-RX-START-Delay parameter for OFDM frames. This field is in terms of 1µs.
rxStartDelayLong	rw	15:8	8'd198	Receive Start Delay with Long preamble This is the value of the aPHY-RX-START-Delay parameter for DSSS/CCK frames with long preamble. This field is in terms of 1µs.
rxStartDelayShort	rw	23:16	8'd102	Receive Start Delay with Short preamble This is the value of the aPHY-RX-START-Delay parameter for DSSS/CCK frames with short preamble. This field is in terms of 1µs.

Field name	rwu	Bit #	Reset	Description
rxStartDelayMIMO	rw	31:24	8'd33	Receive Start Delay for MIMO This is the value of the aPHY-RX-START-Delay parameter for MIMO OFDM frames. This field is in terms of 1μs.

7.4.62 Timings 9 Register (timings9Reg)

Offset Address: 0x0104

This register should be written to only when the core is in the IDLE state.

Field name	rwu	Bit #	Reset	Description
txDMAProcDlyInMACClk	rw	9:0	`TX_DMA_PROC_DLY x `MAC_FREQ	Transmit DMA Processing Delay in MAC Clocks This indicates the Transmit DMA processing delay in terms of MAC Core Clock cycles.
Reserved	rw	19:10	10'b0	Reserved
rifsTOInMACClk	r	31:20	`RIFSTO x `MAC_FREQ	This register should be programmed with the value of RIFS TimeOut in terms of macCoreClk clock cycles.

7.4.63 Receive Controller 2 (rxCntrl2Reg)

Offset Address: 0x010C

The SW can control the reception process of the HW by setting the bits in this register. Multiple bits can be set together and controls whether how the received frame are stored in the RX Buffer.

Field name	rwu	Bit #	Reset	Description
ctrlFrmWrapMode	rw	1:0	2'd1	Control Frame Wrap Mode This register indicates how the payload of the received control frame is stored by the DMA. The encoding is : 2d'0 : The payload is stored as is and may frame if the end of the RX Buffer is reached. (wrap-able mode) 2d'1 : The full payload is stored in one block in the memory. If the payload does not fit in the end of the RX Buffer, the entire payload is stored starting at the beginning of the RX Buffer (unwrap-able mode) 2d'2 : The first part of the payload is stored in one block in the memory. If the N first words of the payload does not fit in the end of the RX Buffer, the payload is stored starting at the beginning of the RX Buffer (Partially wrap-able mode). Int his case, the N first words of the payload is given by the partialUnWrapSize register.

Field name	rwu	Bit #	Reset	Description
				2d'3 : *Reserved*
mgtFrmWrapMode	rw	3:2	2'd1	Management Frame Wrap Mode This register indicates how the payload of the received management frame is stored by the DMA. Same encoding than ctrlFrmWrapMode
dataFrmWrapMode	rw	5:4	2'd2	Data Frame Wrap Mode This register indicates how the payload of the received data frame is stored by the DMA. Same encoding than ctrlFrmWrapMode
Reserved	r	7:6	2'b0	Reserved
partialUnwrapSize	rw	15:8	7'd50	Partial Unwrap Size Indicate the number of word (32bits) at the beginning of the payload which cannot be wrapped in case of partial wrap-able mode.
ctrlMHStoredwithPld	rw	16	1'b1	Control Frame Mac Header Stored With Payload If set, the MAC Header of the Control frame is stored along with the Payload in the RPD. If reset, the MAC Header is stored in the RHD and the reste of the payload sotred in the RPD
mgtMHStoredwithPld	rw	17	1'b1	Management Frame Mac Header Stored With Payload If set, the MAC Header of the Control frame is stored along with the Payload in the RPD. If reset, the MAC Header is stored in the RHD and the reste of the payload sotred in the RPD
dataMHStoredwithPld	rw	18	1'b0	Data Frame Mac Header Stored With Payload If set, the MAC Header of the Control frame is stored along with the Payload in the RPD. If reset, the MAC Header is stored in the RHD and the reste of the payload sotred in the RPD
disableRxBuffer2	rw	19	1'b0	Disable Rx Buffer 2 If set, all the received traffic is stored in Rx Buffer 1. If reset, the BA and Trigger Frames are stored in Rx Buffer 2 and the other frame formats in Rx Buffer 1.
Reserved	r	31:20	12'b0	*Reserved*

7.4.64 Transmit Trigger Timer Register (txTriggerTimerReg)

Offset Address: 0x0110

This register contains the timers as described in *[4]* for interrupt moderation during transmission.

Field name	rwu	Bit #	Reset	Description
txAbsoluteTimeout	rw	7:0	8'd156	Transmission Absolute Timeout This value is loaded in the Transmission Absolute Timer when the first MPDU is successfully transmitted. When the timer expires, an interrupt is generated in the *Transmit / Receive Interrupt Event Register (txRxIntEventReg)*.*timerTxTrigger*. Subsequently, the timer reloads its value from this register when another MPDU is successfully transmitted. The timer is cancelled when the Transmission Packet Timer expires. It is set to a large delay, several MPDU times in duration. This allows for many frames to be transmitted before an interrupt is raised. This mechanism works well for a heavily loaded network. It is programmed in units of 32µs.
txPacketTimeout	rw	15:8	8'd15	Transmission Packet Timeout This value is loaded in the Transmission Packet Timer when a MPDU is successfully transmitted. If another MPDU is successfully transmitted before the timer expires, it is reloaded. When the timer expires, an interrupt is generated in the *Transmit / Receive Interrupt Event Register (txRxIntEventReg)*.*timerTxTrigger*. The timer is cancelled when the Transmission Absolute Timer expires. This is an inactivity timer, and is set to a small delay, few frames times in duration. This mechanism works well for a lightly loaded system. It is programmed in units of 32µs.
Reserved	rw	31:16	16'b0	Reserved

7.4.65 Receive Trigger Timer Register (rxTriggerTimerReg)

Offset Address: 0x0114

This register contains the timers as described in *[4]* for interrupt moderation during reception.

Field name	rwu	Bit #	Reset	Description
rxAbsoluteTimeout	rw	7:0	8'd156	Reception Absolute Timeout This value is loaded in the Reception Absolute Timer when the first MPDU is successfully passed to SW. When the timer expires, an interrupt is generated in the *Transmit / Receive Interrupt Event Register (txRxIntEventReg)*.*timerRxTrigger*. Subsequently, the timer reloads its value from this

Field name	rwu	Bit #	Reset	Description
				register when another MPDU is successfully passed to SW. The timer is cancelled when the Reception Packet Timer expires. It is set to a large delay, several MPDU times in duration. This allows for many frames to be received before an interrupt is raised. This mechanism works well for a heavily loaded network. It is programmed in units of 32µs.
rxPacketTimeout	rw	15:8	8'd15	Reception Packet Timeout This value is loaded in the Reception Packet Timer when a MPDU is successfully passed to SW. If another MPDU is successfully passed to SW before the timer expires, it is reloaded. When the timer expires, an interrupt is generated in the *Transmit / Receive Interrupt Event Register (txRxIntEventReg)*.*timerRxTrigger*. The timer is cancelled when the Reception Absolute Timer expires. This is an inactivity timer, and is set to a small delay, few frames times in duration. This mechanism works well for a lightly loaded system. It is programmed in units of 32µs.
rxPayloadUsedCount	rw	23:16	8'd15	Received Payload Used Count The SW programs this value in order to get an interrupt based on the number of RX Payload Descriptors consumed. Each time a RX Payload Descriptor is consumed, an internal counter is incremented. When it reaches the rxPayloadUsedCount value, an interrupt is generated in the *Transmit / Receive Interrupt Event Register (txRxIntEventReg)*.*timerRxTrigger*. The counter is reset when the interrupt is generated.
Reserved	rw	31:24	8'b0	Reserved

7.4.66 MIB Table Write Register (mibTableWriteReg)

Offset Address: 0x0118

This register is used to write to the *MIB Table*

Field name	rwu	Bit #	Reset	Description

Field name	rwu	Bit #	Reset	Description
mibTableIndex	rw	9:0	10'b0	MIB Table Index Contains the index of the MIB entry in the MIB Table that needs to be updated.
Reserved	r	13:10	4'b0	Reserved
mibIncrementMode	rw	14	1'b1	MIB Increment Mode Indicates whether a MIB entry should be incremented or overwritten, as follows: When set, the contents of the MIB entry pointed to by the *mibTableIndex* are incremented by the *mibValue*. When reset, the contents of the MIB entry pointed to by the *mibTableIndex* are overwritten by the *mibValue*. Note that SW is recommended to overwrite the contents of a MIB entry only when the core is in the IDLE state.
mibWrite	rwu	15	1'b0	MIB Write SW sets this bit to indicate to the core that a MIB update should be performed. Once the core completes the write into the MIB Table, it resets this bit. Software shall write into this register only if this bit is 0.
mibValue	rw	31:16	16'b0	MIB Value The contents of this register are incremented or overwritten to the contents of the MIB entry pointed to the by *mibTableIndex*, depending on the setting of *mibIncrementMode*.

7.4.67 Monotonic Counter 1 Register (monotonicCounter1Reg)

Offset Address: 0x011C

A general-purpose counter is provided for software timing purposes.

Field name	Rwu	Bit #	Reset	Description
monotonicCounter1	ru	31:0	32'b0	Monotonic Counter 1 This counter increments by 1 every *macCoreClk* cycle. It is reset by the HW when the MAC Core clock domain is hard reset or the *MAC Control 2 Register (macCntrl2Reg).softReset* bit is set by SW.

7.4.68 Monotonic Counter 2 Low Register (monotonicCounter2LoReg)

Offset Address: 0x0120

A general-purpose counter is provided for software timing purposes.

Field name	rwu	Bit #	Reset	Description
monotonicCounter2 [31:0]	rwu	31:0	32'b0	Monotonic Counter 2 This field contains the 32 LSBs of the *monotonicCounter2* in microseconds. The counter increments by 1 every µs. It is reset by the HW when the MAC Core clock domain is hard reset or the *MAC Control 2 Register (macCntrl2Reg).softReset* bit is set by SW.

7.4.69 Monotonic Counter 2 High Register (monotonicCounter2HiReg)

Offset Address: 0x0124

A general-purpose counter is provided for software timing purposes.

Field name	rwu	Bit #	Reset	Description
monotonicCounter2 [47:32]	rwu	15:0	16'b0	Monotonic Counter 2 This field contains the 16 MSBs of the *monotonicCounter2* in microseconds. The counter increments by 1 every µs. It is reset by the HW when the MAC Core clock domain is hard reset or the *MAC Control 2 Register (macCntrl2Reg).softReset* bit is set by SW.
Reserved	R	30:16	15'b0	Reserved
monotonicCounter2SW Update	rw	31	1'b0	Monotonic Counter 2 SW Update By setting this field, the SW stops the Monotonic Counter 2 and can write a new value in monotonicCounter2. By resetting this field, the Monotonic Counter 2 counter is running. Note that if the SW changes the monotonicCounter2 without setting this field, the Monotonic Counter 2 will not be updated.

7.4.70 Absolute Timer 0 Register (absTimer0Reg)

Offset Address: 0x0128

A general-purpose timer is provided for software timing purposes.

Field name	rwu	Bit #	Reset	Description

Field name	rwu	Bit #	Reset	Description
absTimer0Value	rw	31:0	32'b0	Absolute Timer 0 Value The HW raises the *General Interrupt Event Register (genIntEventReg).absGenTimers* interrupt (if enabled) everytime the value programmed in *absTimer0Value* matches *.monotonicCounter2[31:0]*. In this case, the Timers Interrupt Event Register (timersIntEventReg).absTimers[0] bit is set

7.4.71 Absolute Timer 1 Register (absTimer1Reg)

Offset Address: 0x012C

A general-purpose timer is provided for software timing purposes.

Field name	rwu	Bit #	Reset	Description
absTimer1Value	rw	31:0	32'b0	Absolute Timer 1 Value The HW raises the *General Interrupt Event Register (genIntEventReg).absGenTimers* interrupt (if enabled) everytime the value programmed in *absTimer1Value* matches*Monotonic Counter 2 Low Register (monotonicCounter2LoReg).monotonicCounter2[31:0]*. In this case, the *Timers Interrupt Event Register (timersIntEventReg).absTimers[1]* bit is set

7.4.72 Absolute Timer 2 Register (absTimer2Reg)

Offset Address: 0x0130

A general-purpose timer is provided for software timing purposes.

Field name	Rwu	Bit #	Reset	Description
absTimer2Value	rw	31:0	32'b0	Absolute Timer 2 Value The HW raises the *General Interrupt Event Register (genIntEventReg).absTimer2* interrupt (if enabled) everytime the value programmed in *absTimer2Value* matches *.monotonicCounter2[31:0]*. In this case, the *Timers Interrupt Event Register (timersIntEventReg).absTimers[2]* bit is set

7.4.73 Absolute Timer 3 Register (absTimer3Reg)

Offset Address: 0x0134

A general-purpose timer is provided for software timing purposes.

Field name	Rwu	Bit #	Reset	Description
absTimer3Value	rw	31:0	32'b0	Absolute Timer 3 Value The HW raises the *General Interrupt Event Register (genIntEventReg).absTimer3* interrupt (if enabled) everytime the value programmed in *absTimer2Value* matches *.monotonicCounter2[31:0]*. In this case, the *Timers Interrupt Event Register (timersIntEventReg).absTimers[3]* bit is set

7.4.74 Absolute Timer 4 Register (absTimer4Reg)

Offset Address: 0x0138

A general-purpose timer is provided for software timing purposes.

Field name	Rwu	Bit #	Reset	Description
absTimer4Value	rw	31:0	32'b0	Absolute Timer 4 Value The HW raises the *General Interrupt Event Register (genIntEventReg).absTimer4* interrupt (if enabled) everytime the value programmed in *absTimer2Value* matches *.monotonicCounter2[31:0]*. In this case, the *Timers Interrupt Event Register (timersIntEventReg).absTimers[4]* bit is set

7.4.75 Absolute Timer 5 Register (absTimer5Reg)

Offset Address: 0x013C

A general-purpose timer is provided for software timing purposes.

Field name	Rwu	Bit #	Reset	Description
absTimer5Value	rw	31:0	32'b0	Absolute Timer 5 Value The HW raises the *General Interrupt Event Register (genIntEventReg).absTimer5* interrupt (if enabled) everytime the value programmed in *absTimer2Value* matches *.monotonicCounter2[31:0]*. In this case, the *Timers Interrupt Event Register (timersIntEventReg).absTimers[5]* bit is set

7.4.76 Absolute Timer 6 Register (absTimer6Reg)

Offset Address: 0x0140

A general-purpose timer is provided for software timing purposes.

Field name	Rwu	Bit #	Reset	Description

Field name	Rwu	Bit #	Reset	Description
absTimer6Value	rw	31:0	32'b0	Absolute Timer 6 Value The HW raises the *General Interrupt Event Register (genIntEventReg).absTimer6* interrupt (if enabled) everytime the value programmed in *absTimer2Value* matches *.monotonicCounter2[31:0]*. In this case, the *Timers Interrupt Event Register (timersIntEventReg).absTimers[6]* bit is set

7.4.77 Absolute Timer 7 Register (absTimer7Reg)

Offset Address: 0x0144

A general-purpose timer is provided for software timing purposes.

Field name	Rwu	Bit #	Reset	Description
absTimer7Value	rw	31:0	32'b0	Absolute Timer 7 Value The HW raises the *General Interrupt Event Register (genIntEventReg).absTimer7* interrupt (if enabled) everytime the value programmed in *absTimer2Value* matches *.monotonicCounter2[31:0]*. In this case, the *Timers Interrupt Event Register (timersIntEventReg).absTimers[7]* bit is set

7.4.78 Absolute Timer 8 Register (absTimer8Reg)

Offset Address: 0x0148

A general-purpose timer is provided for software timing purposes.

Field name	Rwu	Bit #	Reset	Description
absTimer8Value	rw	31:0	32'b0	Absolute Timer 8 Value The HW raises the *General Interrupt Event Register (genIntEventReg).absTimer8* interrupt (if enabled) everytime the value programmed in *absTimer2Value* matches *.monotonicCounter2[31:0]*. In this case, the *Timers Interrupt Event Register (timersIntEventReg).absTimers[8]* bit is set

7.4.79 Absolute Timer 9 Register (absTimer9Reg)

Offset Address: 0x014C

A general-purpose timer is provided for software timing purposes.

Field name	Rwu	Bit #	Reset	Description

Field name	Rwu	Bit #	Reset	Description
absTimer9Value	rw	31:0	32'b0	Absolute Timer 9 Value The HW raises the *General Interrupt Event Register (genIntEventReg).absTimer9* interrupt (if enabled) everytime the value programmed in *absTimer2Value* matches *.monotonicCounter2[31:0]*. In this case, the *Timers Interrupt Event Register (timersIntEventReg).absTimers[9]* bit is set

7.4.80 Max Rx Length Register (maxRxLengthReg)

Offset Address: 0x0150

This register configures the MAC to limit the length of received frames.

Field name	Rwu	Bit #	Reset	Description
maxRxLength [19:0]	rw	19:0	20'h0FFFF	Max Rx Length This field contains the maximum legacy length accepted in RX. All the received frames with a length bigger than maxRxLength will be discarded and the reception will be aborted as soon as the rxVector1 will be received.
Reserved	r	31:20	12'b0	Reserved

7.5 TimeOnAir register definitions

7.5.1 Time On Air Parameters Register (timeOnAirParamReg)

Offset Address: 0x8160

This register is used by the SW to calculate the time on air for any frame, with/without the RTS/CTS/ACK. This is a HW based TXTIME calculator to make it easy for SW to calculate the air time for any frame exchange.

Field name	rwu	Bit #	Reset	Description
ppduLength	rw	19:0	20'b0	PPDU Length This field gives the length of the PPDU for computing time on air.
ppduPreType	rw	23:20	4'b0	PPDU Preamble Type This field indicates what type of preamble is transmitted for the PPDU. The encoding is as follows: 4'b0000: Non-HT-Short 4'b0001: Non-HT-Long 4'b0010: HT-MF 4'b0011: HT-GF 4'b0100: VHT 4'b0101: HE_SU

Field name	rwu	Bit #	Reset	Description
				4'b0110: HE_MU 4'b0111: HE_EXT_SU 4'b1000: HE_TB
ppduGI	rw	25:24	2'd0	PPDU Guard Interval In case of HT-MM, HT-GF and VHT indicates whether a short Guard Interval (GI) is used in the transmission 2'b00: Long GI 2'b01: Short GI In case of HE, indicate the GI Type of the transmission: 2'b00: 0.8 us 2'b01: 1.6 us 2'b10: 3.2 us It is reserved in all other conditions.
ppduBW	rw	27:26	2'b0	PPDU Bandwidth This field indicates the transmission bandwidth of that the PPDU. 00 : 20MHz 01 : 40 MHz 10 : 80MHz 11 : 160MHz
ppduNumExtnSS	rw	29:28	2'b0	PPDU Number of Extension Spatial Streams Indicates the number of extension spatial streams that are sounded during the extension part of the HT training of PPDU.
ppduSTBC	rw	31:30	2'b0	PPDU Space Time Block Coding Indicates the difference between the number of space time streams and the number of spatial streams indicated by the MCS for the PPDU.

7.5.2 Time On Air Parameters 2 Register (timeOnAirParam2Reg)

Offset Address: 0x8164

Field name	rwu	Bit #	Reset	Description
ppduMCSIndex	rw	6:0	7'b0	PPDU MCS Index This field indicates the rate at which this PPDU is transmitted. The encoding is as follows: 1 Mbps: 7'd0 2 Mbps: 7'd1 5.5 Mbps: 7'd2 11 Mbps: 7'd3

Field name	rwu	Bit #	Reset	Description
				6 Mbps: 7'd4 9 Mbps: 7'd5 12 Mbps: 7'd6 18 Mbps: 7'd7 24 Mbps: 7'd8 36 Mbps: 7'd9 48 Mbps: 7'd10 54 Mbps: 7'd11 HT rates: 7'd*MCS Index* VHT rates: {3'd*nSS*, 4'd*MCS index*} HE rates: {3'd*nSS*, 4'd*MCS index*} Note that *nSS* is the number of spatial stream minus 1
Reserved	r	7	1'b0	Reserved
ppduDCM	rw	8	1'b0	PPDU Dual Carrier Modulation Indicates the use of Dual Carrier Modulation for the HE-Data field.
ppduHELTFType	rw	10:9	2'b0	PPDU HE-LTF Type Indicates the type of HE-LTF encoded as follow : 2'b00: 1x HE-LTF for 3.2 µs 2'b01: 2x HE-LTF for 6.4 µs 2'b10: 4x HE-LTF for 12.8 µs 2'b11: Reserved
ppduSigBCompMode	rw	11	1'b0	PPDU SIG B Compression Mode Indicates the SIG B Compression Mode
ppduDCMSigB	rw	12	1'b0	PPDU Dual Carrier Modulation on SIG B Indicates the use of DCM on SIG B
ppduMCSSigB	rw	15:13	3'b0	PPDU MCS of SIG B Indicates the modulation and coding scheme used for HE-SIGB field.
ppduPE	rw	18:16	4'b0	PPDU Packet Extension The duration of the PE field in 4us unit. Possible values are 0 us, 4 us, 8 us, 12 us and 16 us. 3'b000: PE0 for 0 us 3'b001: PE1 for 4 us 3'b010: PE2 for 8 us 3'b011: PE3 for 12 us 3'b100: PE4 for 16 us
Reserved	r	19	1'b0	Reserved
ppduNUser	rw	26:20	7'b0	Number of User in PPDU
ppduNumHeLtf	rw	30:28	3'b0	PPDU Number of HE LTF Indicates the Number of HE LTF of the PPDU

7.5.3 Time On Air Parameters 3 Register (timeOnAirParam3Reg)

Offset Address: 0x8164

Field name	rwu	Bit #	Reset	Description
ppduDoppler	rw	0	1'b0	PPDU Doppler This field indicates a high Doppler mode of transmission. Set to 0 to indicate a high Doppler mode of transmission. Set to 1 otherwise.
ppduMma	rw	1	1'b0	PPDU Midamble Indicates the Midamble periodicity 0 : 10 1 : 20
ppduRuType	rw	6:4	3'b0	PPDU Ru Type Indicates the RU type of the transmission
Reserved	r	31:7	25'b0	*Reserved*

7.5.4 Time On Air Value Register (timeOnAirValue)

Offset Address: 0x816C

This register returns the calculated TXTIME for the parameters programmed.

Field name	rwu	Bit #	Reset	Description
timeOnAir	ru	15:0	16'h0	Time On Air This field gives the time taken for frame transmission in microseconds (µs) computed from parameters programmed in *timeOnAirParamReg* register.
Reserved	r	29:16	14'h0	Reserved
timeOnAirValid	ru	30	1'b0	Time On Air Valid This field is set by the HW when the air time computation is completed. It is cleared when software sets *computeDuration* bit.
computeDuration	rwu	31	1'b0	Compute Duration This field when set by software triggers hardware to perform a Time on Air computation for the frame parameters programmed in . This field is cleared by hardware when the computation is complete, indicating that the value in *timeOnAir* is valid.

7.6 DMA register definitions

7.6.1 DMA Control Register (dmaCntrlReg)

Offset Address: 0x8180 (dmaCntrlSetReg) and 0x8184 (dmaCntrlClearReg)

This register reflects the state of the various transmit DMA control bits. The bits are set by software by writing a one to the corresponding bit in the *dmaCntrlSetReg*. Writing a one to the corresponding bit in the *dmaCntrlClearReg* address clears them. Reading the *dmaCntrlSetReg* or *dmaCntrlClearReg* register returns the current state of the *dmaCntrlSetReg* register.

Refer to section *2.2.3*, *Transmit DMA channel control and procedure* and section ***Error! Reference source not found.***, ***Error! Reference source not found.***.

Field name	rscu	Bit #	Reset	Description
txBcnNewTail	rscu	0	1'b0	Transmit Beacon DMA New Tail Indicates that a fresh descriptor has been appended at the end of the existing Transmit Beacon DMA linked list.
txAC0NewTail	rscu	1	1'b0	Transmit AC 0 DMA New Tail Indicates that a fresh descriptor has been appended at the end of the existing Transmit AC_BK DMA linked list.
txAC1NewTail	rscu	2	1'b0	Transmit AC 1 DMA New Tail Indicates that a fresh descriptor has been appended at the end of the existing Transmit AC_BE DMA linked list.
txAC2NewTail	rscu	3	1'b0	Transmit AC 2 DMA New Tail Indicates that a fresh descriptor has been appended at the end of the existing Transmit AC_VI DMA linked list.
txAC3NewTail	rscu	4	1'b0	Transmit AC 3 DMA New Tail Indicates that a fresh descriptor has been appended at the end of the existing Transmit AC_VO DMA linked list.
Reserved	r	7:5	3'b0	Reserved
txBcnNewHead	rscu	8	1'b0	Transmit Beacon DMA New Head Indicates that the *Transmit Beacon Head Pointer Register (txBcnHeadPtrReg)* has been updated.
txAC0NewHead	rscu	9	1'b0	Transmit AC 0 DMA New Head Indicates that the *Transmit AC0 Head Pointer Register (txAC0HeadPtrReg)* has been updated.
txAC1NewHead	rscu	10	1'b0	Transmit AC 1 DMA New Head Indicates that the *Transmit AC1 Head Pointer Register (txAC1HeadPtrReg)* has been updated.
txAC2NewHead	rscu	11	1'b0	Transmit AC 2 DMA New Head Indicates that the *Transmit AC2 Head Pointer Register (txAC2HeadPtrReg)* has been updated.

Field name	rscu	Bit #	Reset	Description
txAC3NewHead	rscu	12	1'b0	Transmit AC 3 DMA New Head Indicates that the *Transmit AC3 Head Pointer Register (txAC3HeadPtrReg)* has been updated.
txTBNewHead	rscu	13	1'b0	Transmit TB DMA New Head Indicates that the *Transmit TB Head Pointer Register (txtbHeadPtrReg)* has been updated.
Reserved	r	14	1'b0	Reserved
haltBcnAfterTXOP	rsc	15	1'b0	Halt BCN queue This bit is set by SW to indicate that the Bcn DMA channel should move to the HALTED state when a non-zero TXOP ends or after a single MSDU is transmitted.
haltAC0AfterTXOP	rsc	16	1'b0	Halt AC0 After TXOP This bit is set by SW to indicate that the AC_BK DMA channel should move to the HALTED state when a non-zero TXOP ends or after a single MSDU is transmitted.
haltAC1AfterTXOP	rsc	17	1'b0	Halt AC1 After TXOP This bit is set by SW to indicate that the AC_BE DMA channel should move to the HALTED state when a non-zero TXOP ends or after a single MSDU is transmitted.
haltAC2AfterTXOP	rsc	18	1'b0	Halt AC2 After TXOP This bit is set by SW to indicate that the AC_VI DMA channel should move to the HALTED state when a non-zero TXOP ends or after a single MSDU is transmitted.
haltAC3AfterTXOP	rsc	19	1'b0	Halt AC3 After TXOP This bit is set by SW to indicate that the AC_VO DMA channel should move to the HALTED state when a non-zero TXOP ends or after a single MSDU is transmitted.
Reserved	r	31:20	12'b0	Reserved

7.6.2 DMA Status 1 Register (dmaStatus1Reg)

Offset Address: 0x8188

This register reflects the status of the various DMA channels. The DMA states are explained in section *2.2.2*, *Transmit DMA channel states*.

Field name	rwu	Bit #	Reset	Description

Field name	rwu	Bit #	Reset	Description
txBcnState	ru	1:0	2'b0	Transmit Beacon State Indicates the state of the Beacon DMA logical channel. The bits are encoded as follows: 2'b00: HALTED 2'b01: PASSIVE 2'b10: ACTIVE 2'b11: DEAD
Reserved	r	3:2	2'b0	Reserved
txAC0State	ru	5:4	2'b0	Transmit AC0 State Indicates the state of the AC_BK DMA logical channel. The bits are encoded as follows: 2'b00: HALTED 2'b01: PASSIVE 2'b10: ACTIVE 2'b11: DEAD
Reserved	r	7:6	2'b0	Reserved
txAC1State	ru	9:8	2'b0	Transmit AC1 State Indicates the state of the AC_BE DMA logical channel. The bits are encoded as follows: 2'b00: HALTED 2'b01: PASSIVE 2'b10: ACTIVE 2'b11: DEAD
Reserved	r	11:10	2'b0	Reserved
txAC2State	ru	13:12	2'b0	Transmit AC2 State Indicates the state of the AC_VI DMA logical channel. The bits are encoded as follows: 2'b00: HALTED 2'b01: PASSIVE 2'b10: ACTIVE 2'b11: DEAD
Reserved	r	15:14	2'b0	Reserved
txAC3State	ru	17:16	2'b0	Transmit AC2 State Indicates the state of the AC_VO DMA logical channel. The bits are encoded as follows: 2'b00: HALTED 2'b01: PASSIVE 2'b10: ACTIVE 2'b11: DEAD
Reserved	r	19:18	2'b0	Reserved

Field name	rwu	Bit #	Reset	Description
txAC3State	ru	21:20	2'b0	Transmit TB State Indicates the state of the TB DMA logical channel. The bits are encoded as follows: 2'b00: HALTED 2'b01: PASSIVE 2'b10: ACTIVE 2'b11: DEAD
Reserved	r	31:22	10'b0	Reserved

7.6.3 DMA Status 2 Register (dmaStatus2Reg)

Offset Address: 0x818C

This register indicates the reason why a Transmit DMA channel moved into the DEAD state. When the DMA channel is moved out of DEAD state, the bits corresponding to that channel are reset by HW. Refer section *9.3*, *Terminal DMA errors related to data movement*.

Field name	rwu	Bit #	Reset	Description
txLenMismatch	ru	0	1'b0	Transmit Length Mismatch Indicates that the current Transmit DMA channel had a length mismatch error.
txUPatternErr	ru	1	1'b0	Transmit Unique Pattern Error Indicates that the current Transmit DMA channel did not find the expected pattern in the uPatternTx field when reading the Header Descriptor.
txNextPointerErr	ru	2	1'b0	Transmit Next Pointer Error Indicates that the current Transmit DMA channel found an invalid Next Atomic Frame Exchange Sequence Pointer or Next MPDU Descriptor Pointer.
txPTAddressErr	ru	3	1'b0	Transmit Policy Table Address Error Indicates that the current Transmit DMA channel did not find a valid PT address when reading the Header Descriptor.
txBusErr	ru	4	1'b0	Transmit Bus Error Indicates that the current Transmit DMA channel encountered a bus error when reading/writing to memory.
txNewHeadErr	ru	5	1'b0	Transmit New Head Error Indicates that the current Transmit DMA channel's newHead bit was set but the queue HeadPointer did not have a valid address.
Reserved	r	31:6	26'b0	Reserved

7.6.4 DMA Status 3 Register (dmaStatus3Reg)

Offset Address: 0x8190

This register indicates the reason of Rx DMA error. When the DMA channel is in error state, the bits corresponding to that channel are reset by HW. Refer section *9.3*, *Terminal DMA errors related to data movement*.

Field name	rwu	Bit #	Reset	Description
Reserved	r	3:0	4'b0	Reserved
rxHdrBusErr	ru	4	1'b0	Receive Header Bus Error Indicates that the Receive Header DMA channel encountered a bus error when reading/writing to memory.
rxPayBusErr	ru	5	1'b0	Receive Payload Bus Error Indicates that the Receive Payload DMA channel encountered a bus error when reading/writing to memory.
Reserved	r	31:6	26'b0	Reserved

7.6.5 DMA Status 4 Register (dmaStatus4Reg)

Offset Address: 0x8194

This register indicates the reason why a DMA channel moved into the HALTED state.

Field name	rwu	Bit #	Reset	Description
txBcnStartup	ru	0	1'b1	Transmit Beacon Startup Indicates that the Transmit Beacon DMA has never been moved out the HALTED state.
txAC0Startup	ru	1	1'b1	Transmit AC0 New Head Error Indicates that the Transmit AC_BK DMA has never been moved out the HALTED state.
txAC1Startup	ru	2	1'b1	Transmit AC1 New Head Error Indicates that the Transmit AC_BE DMA has never been moved out the HALTED state.
txAC2Startup	ru	3	1'b1	Transmit AC2 New Head Error Indicates that the Transmit AC_VI DMA has never been moved out the HALTED state.
txAC3Startup	ru	4	1'b1	Transmit AC3 New Head Error Indicates that the Transmit AC_VO DMA has never been moved out the HALTED state.
txBcnEndQ	ru	5	1'b0	Transmit Beacon End of Queue Indicates that the Transmit Beacon DMA has reached the end of the linked list.
txAC0EndQ	ru	6	1'b0	Transmit AC0 End of Queue Indicates that the Transmit AC_BK DMA has reached the end of the linked list.

Field name	rwu	Bit #	Reset	Description
txAC1EndQ	ru	7	1'b0	Transmit AC1 End of Queue Indicates that the Transmit AC_BE DMA has reached the end of the linked list.
txAC2EndQ	ru	8	1'b0	Transmit AC2 End of Queue Indicates that the Transmit AC_VI DMA has reached the end of the linked list.
txAC3EndQ	ru	9	1'b0	Transmit AC3 End of Queue Indicates that the Transmit AC_VO DMA has reached the end of the linked list.
txBcnHaltAfterTXOP	ru	10	1'b0	Transmit BCN Halt After TXOP Indicates that the BCN DMA has stopped because the TXOP has ended and the *DMA Control Register (dmaCntrlReg).haltACxAfterTXOP* bit is set.
txAC0HaltAfterTXOP	ru	11	1'b0	Transmit AC0 Halt After TXOP Indicates that the AC_BK DMA has stopped because the TXOP has ended and the *DMA Control Register (dmaCntrlReg).haltACxAfterTXOP* bit is set.
txAC1HaltAfterTXOP	ru	12	1'b0	Transmit AC1 Halt After TXOP Indicates that the AC_BE DMA has stopped because the TXOP has ended and the *DMA Control Register (dmaCntrlReg).haltACxAfterTXOP* bit is set.
txAC2HaltAfterTXOP	ru	13	1'b0	Transmit AC2 Halt After TXOP Indicates that the AC_VI DMA has stopped because the TXOP has ended and the *DMA Control Register (dmaCntrlReg).haltACxAfterTXOP* bit is set.
txAC3HaltAfterTXOP	ru	14	1'b0	Transmit AC3 Halt After TXOP Indicates that the AC_VO DMA has stopped because the TXOP has ended and the *DMA Control Register (dmaCntrlReg).haltACxAfterTXOP* bit is set.
Reserved	r	31:15	17'b0	Reserved

7.6.6 Transmit Beacon Head Pointer Register (txBcnHeadPtrReg)

Offset Address: 0x8198

The address of the first descriptor of the Beacon DMA linked list is programmed in this register.

Field name	rwu	Bit #	Reset	Description

Field name	rwu	Bit #	Reset	Description
txBcnHeadPtr	rw	31:0	32'b0	Transmit Beacon list Head Pointer The quadlet-aligned address of the first descriptor of the Beacon DMA linked list is programmed by SW in this field. If the HW finds an invalid address programmed, it raises the raises the error interrupt *General Interrupt Event Register (genIntEventReg).txDMAError*. Refer to section *9.3*, *Terminal DMA errors* for details.

7.6.7 Transmit AC0 Head Pointer Register (txAC0HeadPtrReg)

Offset Address: 0x819C

The address of the first descriptor of the AC0 DMA linked list is programmed in this register.

Field name	rwu	Bit #	Reset	Description
txAC0HeadPtr	rw	31:0	32'b0	Transmit AC 0 list Head Pointer The quadlet-aligned address of the first descriptor of the transmit AC_BK DMA linked list is programmed by SW in this field. If the HW finds an invalid address programmed, it raises the raises the error interrupt *General Interrupt Event Register (genIntEventReg).txDMAError*. Refer to section *9.3*, *Terminal DMA errors* for details.

7.6.8 Transmit AC1 Head Pointer Register (txAC1HeadPtrReg)

Offset Address: 0x81A0

The address of the first descriptor of the AC1 DMA linked list is programmed in this register.

Field name	rwu	Bit #	Reset	Description
txAC1HeadPtr	rw	31:0	32'b0	Transmit AC 1 list Head Pointer The quadlet-aligned address of the first descriptor of the transmit AC_BE DMA linked list is programmed by SW in this field. If the HW finds an invalid address programmed, it raises the raises the error interrupt *General Interrupt Event Register (genIntEventReg).txDMAError*. Refer to section *9.3*, *Terminal DMA errors* for details.

7.6.9 Transmit AC2 Head Pointer Register (txAC2HeadPtrReg)

Offset Address: 0x81A4

The address of the first descriptor of the AC2 DMA linked list is programmed in this register.

Field name	rwu	Bit #	Reset	Description

Field name	rwu	Bit #	Reset	Description
txAC2HeadPtr	rw	31:0	32'b0	Transmit AC 2 list Head Pointer The quadlet-aligned address of the first descriptor of the transmit AC_VI DMA linked list is programmed by SW in this field. If the HW finds an invalid address programmed, it raises the raises the error interrupt *General Interrupt Event Register (genIntEventReg).txDMAError*. Refer to section *9.3*, *Terminal DMA errors* for details.

7.6.10 Transmit AC3 Head Pointer Register (txAC3HeadPtrReg)

Offset Address: 0x81A8

The address of the first descriptor of the AC3 DMA linked list is programmed in this register.

Field name	rwu	Bit #	Reset	Description
txAC3HeadPtr	rw	31:0	32'b0	Transmit AC 3 list Head Pointer The quadlet-aligned address of the first descriptor of the transmit AC_VO DMA linked list is programmed by SW in this field. If the HW finds an invalid address programmed, it raises the raises the error interrupt *General Interrupt Event Register (genIntEventReg).txDMAError*. Refer to section *9.3*, *Terminal DMA errors* for details.

7.6.11 Transmit TB Head Pointer Register (txtbHeadPtrReg)

Offset Address: 0x81A8

The address of the first descriptor of the TB DMA linked list is programmed in this register.

Field name	rwu	Bit #	Reset	Description
txTBHeadPtr	rw	31:0	32'b0	Transmit TB Head Pointer The quadlet-aligned address of the first descriptor of the transmit TB DMA linked list is programmed by SW in this field. If the HW finds an invalid address programmed, it raises the raises the error interrupt General Interrupt Event Register (genIntEventReg).txDMAError.

7.6.12 Transmit Structure Sizes Register (txStructSizesReg)

Offset Address: 0x81B0

This register indicates the size of various structures associated with the DMA engine that needs to be fetched by the MAC Core. The size allows SW to maintain private information in each structure, if desired, beyond the size read by the MAC Core.

Field name	rwu	Bit #	Reset	Description

Field name	rwu	Bit #	Reset	Description
ptEntrySize	rw	5:0	`PT_SIZE	Policy Table Entry Size This field indicates the size in quadlets of each Policy Table entry that needs to be fetched by the MAC Core, starting at the base address as indicated by the *Transmit DMA Header Descriptor*.
dmaTHDSize	rw	11:6	`THD_SIZE	DMA Transmit Header Descriptor Size This field indicates the size in quadlets of each DMA Transmit Header Descriptor that needs to be fetched by the MAC Core, starting at the base address as indicated by the *Transmit DMA Header Descriptor*or by a Queue Head Pointer register.
dmaTBDSize	rw	17:12	`TBD_SIZE	DMA Transmit Buffer Descriptor Size This field indicates the size in quadlets of each DMA Transmit Buffer Descriptor that needs to be fetched by the MAC Core, starting at the base address as indicated by the *Transmit DMA Header Descriptor*.
dmaRHDSize	rw	23: 18	`RHD_SIZE	DMA Receive Header Descriptor Size This field indicates the size in quadlets of each DMA Receive Header Descriptor that needs to be fetched by the MAC Core, starting at the base address as indicated by the *Receive DMA Header Descriptor* or by a Queue Head Pointer register.
dmaRBDSize	rw	29:24	`RBD_SIZE	DMA Receive Buffer Descriptor Size This field indicates the size in quadlets of each DMA Receive Buffer Descriptor that needs to be fetched by the MAC Core, starting at the base address as indicated by the *Receive DMA Header Descriptor*.
Reserved	r	31:30	2'b0	Reserved

7.6.13 Reserved (NA)

Offset Address: 0x81B4

Reserved for future use.

Field name	rwu	Bit #	Reset	Description
Reserved	r	31:0	32'b0	Reserved

7.6.14 Reserved (NA)

Offset Address: 0x81B8

Reserved for future use.

Field name	rwu	Bit #	Reset	Description
Reserved	r	31:0	32'b0	Reserved

7.6.15 Reserved (NA)

Offset Address: 0x81BC

Reserved for future use.

Field name	rwu	Bit #	Reset	Description
Reserved	r	31:0	32'b0	Reserved

7.6.16 DMA Threshold Register (dmaThresholdReg)

Offset Address: 0x81C0

This register contains the threshold value of the MAC Transmit and Receive FIFOs at which the MDA engine is triggered.

Field name	rwu	Bit #	Reset	Description
txFIFOThreshold	rw	7:0	8'd16	Transmit FIFO Threshold Indicates the number of quadlets of free space that should be available in the Transmit FIFO to start a read transaction to fetch a part of data from memory. This threshold is used to make the data movement operation efficient, and prevent small inefficient transactions. Note that the threshold is ignored if the number of quadlets remaining to be fetched for the current buffer is smaller than the threshold.
Reserved	r	15:8	8'b0	Reserved
rxFIFOThreshold	rw	23:16	8'd16	Receive FIFO Threshold Indicates the number of quadlets of data that should be available in the Receive FIFO to start a write transaction to move a part of the frame into memory. This threshold is used to make the data movement operation efficient, and prevent small inefficient transactions.
Reserved	r	31:24	8'b0	Reserved

7.6.17 Receive Header Trigger Frame Pointer Register (rxHeaderTFPtrReg)

Offset Address: 0x81C4

The address of the Rx Header descriptor of the latest Trigger Frame received is provided by the HW to the SW in this register.

Field name	rwu	Bit #	Reset	Description
Reserved	r	1:0	2'b0	Reserved
rxHeaderTFPtrReg	r	31:2	30'b0	Receive Header Trigger Frame Pointer The quadlet-aligned address of the Receive Header Descriptor of the latest Trigger Frame received.

7.6.18 Receive Buffer 1 Start Pointer Register (rxBuf1StartPtrReg)

Offset Address: 0x81C8

The starting address of the DMA Rx Buffer 1 is programmed in this register.

Field name	rwu	Bit #	Reset	Description
rxBuf1StartPtr	rw	30:0	31'b0	DMA Receive buffer 1 Start address Pointer The quadlet-aligned start address of the DMA Receive buffer 1.
Reserved	r	31	1'b0	Reserved

7.6.19 Receive Buffer 1 End Pointer Register (rxBuf1EndPtrReg)

Offset Address: 0x81CC

The end address of the DMA Rx Buffer 1 is programmed in this register.

Field name	rwu	Bit #	Reset	Description
rxBuf1EndPtr	rw	30:0	31'b0	DMA Receive buffer 1 End address Pointer The quadlet-aligned end address of the DMA Receive buffer 1.
Reserved	r	31	1'b0	Reserved

7.6.20 Receive Buffer 1 Read Pointer Register (rxBuf1RdPtrReg)

Offset Address: 0x81D0

The read pointer of the DMA Rx Buffer 1 is programmed in this register.

Field name	rwu	Bit #	Reset	Description
rxBuf1RdPtr	rw	31:0	32'b0	DMA Receive buffer 1 Read Pointer The quadlet-aligned read pointer of the DMA Receive buffer 1. Note: The MSB (bit 31) is used to indicate a wrap. It toggles each time the rxBuf1RdPtr wraps.

7.6.21 Receive Buffer 1 Write Pointer Register (rxBuf1WrPtrReg)

Offset Address: 0x81D4

The write pointer of the DMA Rx Buffer 1 is programmed in this register.

Field name	rwu	Bit #	Reset	Description
rxBuf1WrPtr	rw	31:0	32'b0	DMA Receive buffer 1 Write Pointer The quadlet-aligned read pointer of the DMA Receive buffer 1. Note: The MSB (bit 31) is used to indicate a wrap. It toggles each time the rxBuf1WrPtr wraps.

7.6.22 Receive Buffer 2 Start Pointer Register (rxBuf2StartPtrReg)

Offset Address: 0x81D8

The starting address of the DMA Rx Buffer 2 is programmed in this register.

Field name	rwu	Bit #	Reset	Description
rxBuf2StartPtr	rw	30:0	31'b0	DMA Receive buffer 2 Start address Pointer The quadlet-aligned start address of the DMA Receive buffer 2.
Reserved	r	31	1'b0	Reserved

7.6.23 Receive Buffer 2 End Pointer Register (rxBuf2EndPtrReg)

Offset Address: 0x81DC

The end address of the DMA Rx Buffer 2 is programmed in this register.

Field name	rwu	Bit #	Reset	Description
rxBuf2EndPtr	rw	30:0	31'b0	DMA Receive buffer 2 End address Pointer The quadlet-aligned end address of the DMA Receive buffer 2.
Reserved	r	31	1'b0	Reserved

7.6.24 Receive Buffer 2 Read Pointer Register (rxBuf2RdPtrReg)

Offset Address: 0x81E0

The read pointer of the DMA Rx Buffer 2 is programmed in this register.

Field name	rwu	Bit #	Reset	Description
rxBuf2RdPtr	rw	31:0	32'b0	DMA Receive buffer 2 Read Pointer The quadlet-aligned read pointer of the DMA Receive buffer 2. Note: The MSB (bit 31) is used to indicate a wrap. It toggles each time the rxBuf2RdPtr wraps.

7.6.25 Receive Buffer 2 Write Pointer Register (rxBuf2WrPtrReg)

Offset Address: 0x81E4

The write pointer of the DMA Rx Buffer 2 is programmed in this register.

Field name	rwu	Bit #	Reset	Description
rxBuf2WrPtr	rw	31:0	32'b0	DMA Receive buffer 2 Write Pointer The quadlet-aligned read pointer of the DMA Receive buffer 2. Note: The MSB (bit 31) is used to indicate a wrap. It toggles each time the rxBuf2WrPtr wraps.

7.6.26 Receive Buffer Configuration Register (rxBufConfigReg)

Offset Address: 0x81E8

The receive buffer blank space are programmed in this register.

Field name	rwu	Bit #	Reset	Description
rxBufRHDHeader	rw	7:0	8'b0	DMA RX buffer Receive Header Descriptor blank space Header The number of blank words before the Receive Header Descriptor.
rxBufRHDFooter	rw	15:8	8'b0	DMA RX buffer Receive Header Descriptor blank space Footer The number of blank words after the Receive Header Descriptor.
rxBufRPDHeader	rw	23:16	8'b0	DMA RX buffer Receive Payload Descriptor blank space Header The number of blank words before the Receive Payload Descriptor.
rxBufRPDFooter	rw	31:24	8'b0	DMA RX buffer Receive Payload Descriptor blank space Footer The number of blank words after the Receive Payload Descriptor.

7.7 QoS register definitions

7.7.1 EDCA AC0 Register (edcaAC0Reg)

Offset Address: 0x0200

This register is programmed with the parameters that govern AC_BK transmission.

Field name	rwu	Bit #	Reset	Description
aifsn0	rw	3:0	4'd7	AIFSN for AC 0 Indicates the AIFS in units of slots after SIFS that HW should wait for before starting backoff, for Access Category "AC_BK" frames.
cwMin0	rw	7:4	4'd4	Minimum CW for AC 0 Indicates the Minimum Contention Window as an encoded value of $2^{cwMin0} - 1$, for Access Category "AC_BK" frames.
cwMax0	rw	11:8	4'd10	Maximum CW for AC 0 Indicates the Maximum Contention Window as an encoded value of $2^{cwMax0} - 1$, for Access Category "AC_BK" frames.
txOpLimit0	rw	27:12	16'd0	Transmit Opportunity Limit for AC 0 Indicates the Transmit Opportunity Limit in units of 32μs, for Access Category "AC_BK" frames.

Field name	rwu	Bit #	Reset	Description
Reserved	r	31:28	4'b0	Reserved

7.7.2 EDCA AC1 Register (edcaAC1Reg)

Offset Address: 0x0204

This register is programmed with the parameters that govern AC_BE transmission.

Field name	rwu	Bit #	Reset	Description
aifsn1	rw	3:0	4'd3	AIFSN for AC 1 Indicates the AIFS in units of slots after SIFS that h/w should wait for before starting backoff, for Access Category "AC_BE" frames.
cwMin1	rw	7:4	4'd4	Minimum CW for AC 1 Indicates the Minimum Contention Window as an encoded value of $2^{cwMin1} - 1$, for Access Category "AC_BE" frames.
cwMax1	rw	11:8	4'd10	Maximum CW for AC 1 Indicates the Maximum Contention Window as an encoded value of $2^{cwMax1} - 1$, for Access Category "AC_BE" frames.
txOpLimit1	rw	27:12	16'b0	Transmit Opportunity Limit for AC 1 Indicatesthe Transmit Opportunity Limit in units of 32μs, for Access Category "AC_BE" frames.
Reserved	r	31:28	4'b0	Reserved

7.7.3 EDCA AC2 Register (edcaAC2Reg)

Offset Address: 0x0208

This register is programmed with the parameters that govern AC_VI transmission.

Field name	rwu	Bit #	Reset	Description
aifsn2	rw	3:0	4'd2	AIFSN for AC 2 Indicates the AIFS in units of slots after SIFS that h/w should wait for before starting backoff, for Access Category "AC_VI" frames.
cwMin2	rw	7:4	4'd3	Minimum CW for AC 2 Indicatesthe Minimum Contention Window as an encoded value of $2^{cwMin2} - 1$, for Access Category "AC_VI" frames.
cwMax2	rw	11:8	4'd4	Maximum CW for AC 2 Indicates the Maximum Contention Window as an encoded value of $2^{cwMax2} - 1$, for Access Category "AC_VI" frames.
txOpLimit2	rw	27:12	16'h5E	Transmit Opportunity Limit for AC 2 Indicates the Transmit Opportunity Limit in units of 32μs, for Access Category "AC_VI" frames.

Field name	rwu	Bit #	Reset	Description
Reserved	r	31:28	4'b0	Reserved

7.7.4 EDCA AC3 Register (edcaAC3Reg)

Offset Address: 0x020C

This register is programmed with the parameters that govern AC_VO transmission.

Field name	rwu	Bit #	Reset	Description
aifsn3	rw	3:0	4'd2	AIFSN for AC 3 Indicates the AIFS in units of slots after SIFS that h/w should wait for before starting backoff, for Access Category "AC_VO" frames.
cwMin3	rw	7:4	4'd2	Minimum CW for AC 3 Indicates the Minimum Contention Window as an encoded value of $2^{cwMin3} - 1$, for Access Category "AC_VO" frames.
cwMax3	rw	11:8	4'd3	Maximum CW for AC 3 Indicates the Maximum Contention Window as an encoded value of $2^{cwMax3} - 1$, for Access Category "AC_VO" frames.
txOpLimit3	rw	27:12	16'h2F	Transmit Opportunity Limit for AC 3 Indicates the Transmit Opportunity Limit in units of 32μs, for Access Category "AC_VO" frames.
Reserved	r	31:28	4'b0	Reserved

7.7.5 Reserved (NA)

Offset Address: 0x0218

Reserved for future use.

Field name	rwu	Bit #	Reset	Description
Reserved	r	31:0	32'b0	Reserved

7.7.6 Reserved (NA)

Offset Address: 0x021C

Reserved for future use.

Field name	rwu	Bit #	Reset	Description
Reserved	r	31:0	32'b0	Reserved

7.7.7 EDCA CCA Busy Time (edcaCCABusyReg)

Offset Address: 0x0220

The CCA utilization of the BSS is collected in this register.

Field name	rwu	Bit #	Reset	Description

Field name	rwu	Bit #	Reset	Description
ccaBusyDur	rwu	31:0	32'b0	CCA Busy Duration This field gives the duration (in units of μs) for which the QAP has sensed the medium busy, as indicated by either the physical or virtual carrier sense mechanism. This register should be read by the software after every dot11ChannelUtilizationBeaconIntervals at TBTT. Upon read, SW must immediately clear the register by writing 0 to it to prevent inaccuracies in channel busy measurement.

7.7.8 EDCA Control Register (edcaCntrlReg)

Offset Address: 0x0224

The SW uses this register to control miscellaneous EDCA settings. Refer section *2.2.4.2.10*, *Terminating a TXOP* for more details.

Field name	rwu	Bit #	Reset	Description
sendCFEndNow	rwu	0	1'b0	Send CF-End Now SW writes to this bit to indicate to HW to transmit a CF-End frame as soon as it can. It is used under the following conditions: ✓ If the HW is currently transmitting frames in a TXOP and the DMA linked list subsequently becomes empty and the NAV at other devices has been set for a time greater than the time for the CF-End procedure to complete. ✓ If the DMA linked list is already empty and the NAV at other devices has been set for a time greater than the time for the CF-End procedure to complete. This bit is reset as soon as the CF-End is transmitted or the time for the CF-End frame is greater than the NAV at other devices.
sendCFEnd	rw	1	1'b0	Send CF-End SW sets this bit to indicate to HW to transmit a CF-End whenever the DMA linked list becomes empty and the NAV at other devices has been set for a time greater than the time for the CF-End procedure to complete. If this bit is not set, SW should transmit a CF-End frame or set the *sendCFEndNow* bit to enable the HW to send a CF-End.
Reserved	r	3:2	2'b0	Reserved
remTXOPInDurField	rw	4	1'b1	Remaining TXOP Indicated in Duration Field When set, the duration field of the transmitted frame includes only the duration of SIFS and the immediate response frame for both null and not null TXOPLimit.

Field name	rwu	Bit #	Reset	Description
				When reset, the duration field contains the TXOP remaining duration in case of not null TXOPLimit or the duration of SIFS and the immediate response frame in case of null TXOPLimit.
keepTXOPOpen	rw	5	1'b0	Keep TXOP Open SW sets this bit if it wants to keep the TXOP even if there is not new frame linked for the selected AC (DMA channel is in HALTED). This can be used to provide additional time for the software for linking a new frame. In this case, the MACHW will stay in ACTIVE_TX stats until the end of the TXOP.
Reserved	r	31:6	26'b0	Reserved

7.7.9 Medium Occupancy Timer 1 Register (mot1Reg)

Offset Address: 0x8228

If the *TXOPLimit* for AC_BK and AC_BE are non-zero, then the MOT value indicates the remaining TXOP value to SW.

- ✓ When a *protocol trigger* is raised to SW, the HW loads this field with the *TXOPLimit* value for that AC, capped by the start of the next scheduled Quiet interval, if any.
- ✓ Once the TXOP starts on air, this register indicates the estimated TXOP that will remain after the current frame exchange.
- ✓ Note that the HW makes an estimation of the time that the current frame exchange will take on air. In some cases, the actual TXOP that remains after the current frame exchange might be more or less than estimated. This is corrected by the HW at the end of the atomic frame exchange sequence.

If the *TXOPLimit* for AC_BK and AC_BE are zero, then the MOT value is 0.

Field name	rwu	Bit #	Reset	Description
ac0MOT	ru	15:0	16'b0	AC0 Medium Occupancy Timer This field indicates the remaining TXOP for EDCA AC_BK in units of 32μs and is used by the SW to calculate the number of frames that should be MPDU aggregated.
ac1MOT	ru	31:16	16'b0	AC1 Medium Occupancy Timer This field indicates the remaining TXOP for EDCA AC_BE in units of 32μs and is used by the SW to calculate the number of frames that should be MPDU aggregated.

7.7.10 Medium Occupancy Timer 2 Register (mot2Reg)

Offset Address: 0x822C

If the *TXOPLimit* for AC_VI and AC_VO are non-zero, then the MOT value indicates the remaining TXOP value to SW.

- ✓ When a *protocol trigger* is raised to SW, the HW loads this field with the *TXOPLimit* value for that AC, capped by the start of the next scheduled Quiet interval, if any.
- ✓ Once the TXOP starts on air, this register indicates the estimated TXOP that will remain after the current frame exchange.

✓ Note that the HW makes an estimation of the time that the current frame exchange will take on air. In some cases, the actual TXOP that remains after the current frame exchange might be more or less than estimated. This is corrected by the HW at the end of the atomic frame exchange sequence.

If the *TXOPLimit* for AC_BK and AC_BE are zero, then the MOT value is 0.

Field name	rwu	Bit #	Reset	Description
ac2MOT	ru	15:0	16'h5E	AC2 Medium Occupancy Timer This field indicates the remaining TXOP for EDCA AC_VI in units of 32µs and is used by the SW to calculate the number of frames that should be MPDU aggregated.
ac3MOT	ru	31:16	16'h2F	AC3 Medium Occupancy Timer This field indicates the remaining TXOP for EDCA AC_VO in units of 32µs and is used by the SW to calculate the number of frames that should be MPDU aggregated.

7.8 Spectrum management extensions register definitions

7.8.1 Quiet Element 1a Register (quietElement1aReg)

Offset Address: 0x0280

This register contains parameters received in the first Quiet IE. As a STA, HW updates this field from any received Beacon or Probe Response frame belonging to its BSS. The SSID of the received Beacon or Problem Response frame is not checked in HW. Hence parameters which are extracted by the MAC HW may be taken from a Beacon or Probe Response that does not contain the correct SSID, but contains the correct BSSID.

Field name	rwu	Bit #	Reset	Description
quietCount1	rwu	7:0	8'b0	Quiet Count 1 Indicates the number of TBTTs until the Beacon Interval in which the Quiet interval for Quiet IE 1 starts. HW programs a value of 0 when this Quiet IE is not scheduled.
quietPeriod1	rwu	15:8	8'b0	Quiet Period 1 Indicates the number of Beacon Intervals between successive Quiet intervals for Quiet IE 1. A value of 0 indicates that no periodic quiet interval is defined.
quietDuration1	rwu	31:16	16'b0	Quiet Duration 1 Indicates the duration of the Quiet interval for Quiet IE 1, in units of TUs.

7.8.2 Quiet Element 1b Register (quietElement1bReg)

Offset Address: 0x0284

This register contains parameters received in the first Quiet IE. As a STA, HW updates this field from any received Beacon or Probe Response frame belonging to its BSS. The SSID of the received Beacon or Problem Response frame is

not checked in HW. Hence parameters which are extracted by the MAC HW may be taken from a Beacon or Probe Response that does not contain the correct SSID, but contains the correct BSSID.

Field name	rwu	Bit #	Reset	Description
quietOffset1	rwu	15:0	16'b0	Quiet Offset 1 Indicates the offset from the start of the TBTT at which the Quiet interval starts for Quiet IE 1, in units of TUs.
Reserved	r	31:16	16'b0	Reserved

7.8.3 Additional CCA Busy Time on Secondary 20MHz (addCCABusySec20Reg)

Offset Address: 0x0290

The CCA utilization of the BSS is collected in this register.

Field name	rwu	Bit #	Reset	Description
ccaBusyDurSec20	rwu	31:0	32'b0	CCA Busy Duration on secondary 20MHz This field gives the duration (in units of μs) for which the QAP has sensed the medium busy on the secondary 20MHz, as indicated by either the physical or virtual carrier sense mechanism. This register should be read by the software after every dot11ChannelUtilizationBeaconIntervals at TBTT. Upon read, SW must immediately clear the register by writing 0 to it to prevent inaccuracies in channel busy measurement.

7.8.4 Additional CCA Busy Time on Secondary 40MHz (addCCABusySec40Reg)

Offset Address: 0x0294

The CCA utilization of the BSS is collected in this register.

Field name	rwu	Bit #	Reset	Description
ccaBusyDurSec40	rwu	31:0	32'b0	CCA Busy Duration on secondary 40MHz This field gives the duration (in units of μs) for which the QAP has sensed the medium busy on the secondary 40MHz, as indicated by either the physical or virtual carrier sense mechanism. This register should be read by the software after every dot11ChannelUtilizationBeaconIntervals at TBTT. Upon read, SW must immediately clear the register by writing 0 to it to prevent inaccuracies in channel busy measurement.

7.8.5 Additional CCA Busy Time on Secondary 80MHz (addCCABusySec80Reg)

Offset Address: 0x0298

The CCA utilization of the BSS is collected in this register.

Field name	rwu	Bit #	Reset	Description
ccaBusyDurSec80	rwu	31:0	32'b0	CCA Busy Duration on secondary 80MHz This field gives the duration (in units of μs) for which the QAP has sensed the medium busy on the secondary 80MHz, as indicated by either the physical or virtual carrier sense mechanism. This register should be read by the software after every dot11ChannelUtilizationBeaconIntervals at TBTT. Upon read, SW must immediately clear the register by writing 0 to it to prevent inaccuracies in channel busy measurement.

7.9 High and Very High throughput register definitions

7.9.1 STBC Control Register (stbcCntrlReg)

Offset Address: 0x0300

This register should be written to only when the core is in the IDLE state.

Field name	rwu	Bit #	Reset	Description
cfEndSTBCDur	rw	15:0	16'b0	CF-End STBC Duration The SW programs the time on air for the transmission of 1 CF-End frame transmitted at the lowest basic rate, one CF-End frame transmitted at the Basic STBC MCS and 2 SIFS durations. The HW uses this value to check if there is time remaining at the end of the TXOP to curtail the TXOP in case of Dual-CTS protection.
ctsSTBCDur	rw	23:16	8'b0	CTS STBC Duration The SW programs the time on air for the transmission of one CTS frame at the Basic STBC MCS.
dualCTSProt	rw	24	1'b0	Dual-CTS Protection Indicates that dual-CTS protection is required.
basicSTBCMCS	rw	31:25	7'b0	Basic STBC MCS Indicates the Basic STBC MCS.

7.9.2 Transmission Bandwidth Control Register (txBWCntrlReg)

Offset Address: 0x0310

This register is used to control the HW behavior related to 40/80/160 MHz operation. Refer section *2.2.4.2.3, Determining 20/40/80/160* MHz TXOP.

Field name	rwu	Bit #	Reset	Description

Field name	rwu	Bit #	Reset	Description
defaultBWTXOPV	rw	0	1'b0	Default BW TXOP Valid Indicates if the *defaultBWTXOP* setting is valid.
defaultBWTXOP	rw	2:1	2'b0	Default Bandwidth TXOP This bit is valid only if the *defaultBWTXOPV* bit is set and overrides any other setting of 20/40/80/160 MHz acquisition indicated via the first chained PPDU. HW tries to acquire a 20/40/80/160 MHz TXOP when it wins contention. If a 40/80/160 MHz TXOP cannot be acquired, then subsequent behavior depends on the *dropToLowerBW* field.
dropToLowerBW	rw	3	1'b1	Drop To Lower Bandwidth PPDU If the HW fails to acquired a TXOP at the requested bandwidth, the behaviour of the MAC HW depends on this bit: When set, the HW will transmit the 40/80/160 MHz PPDU using the highest available bandwidth if the frame exchange sequence at the lower rate will fit into the remaining TXOP. When reset, the HW will release the TXOP without attempting to transmit the frame.
numTryBWAcquisition	rw	6:4	3'b1	Number of Tries for Requested Bandwidth Acquisition If the SW indicates that a 40/80/160 MHz TXOP should be acquired (either through the register field *defaultBWTXOP* or through a PPDU's DMA descriptor field *bandWidthTx*), and the HW fails to do so for the number of time programmed in this field, the HW switches over to a lower bandwidth. 40/80/160 MHz PPDUs are transmitted at highest available bandwidth using the same MCS and other PHY parameters, if the frame exchange sequence fits into the remaining TXOP at the lower rate. If this field is set to 0, this feature is disabled, i.e. the HW will keep on trying to acquire a 40/80/160 MHz TXOP until it succeeds. If this field is set to 1, this bandwidth drop is immediate if the requested bandwidth is not possible, i.e. the HW will acquire the highest available bandwidth lower or equal to the requested one. This field is meaningless if the SW has not indicated that a 40/80/160 MHz TXOP should be acquired.

Field name	rwu	Bit #	Reset	Description
dynBWEn	rw	7	1'b0	Dynamic Bandwidth Enable When set, the HW will support the dynamic bandwidth selection. When reset, the HW does not support the dynamic bandwidth selection.
aPPDUMaxTime	rw	15:8	8'd10	aPPDU Maximum Time Indicates the value of the *aPPDUMaxTime* parameter in units of 1 ms. This value is used by HW when it changes the BW of a PPDU from 160/80/40 MHz to a lower BW to check if the resultant time on air does not exceed this value.
maxSupportedBW	rw	17:16	2'd1	Max Supported Bandwidth Indicates the bandwidth supported by the system. The field is encoded as follows: 00 : 20MHz 01 : 40 MHz 10 : 80MHz 11 : 160MHz
Reserved	r	23:18	6'b0	Reserved
currentBW	r	25:24	2'b0	Current Transmission Bandwidth Indicates the bandwidth used for the on-going transmissions. The field is encoded as *maxSupportedBW*
Reserved	r	31:26	6'b0	Reserved

7.9.3 HTMCS Register (HTMCSReg)

Offset Address: 0x0314

Contains the various common HT MCS for the BSS. HW uses this register to determine the transmission rate of the packets that are sent by HW. This register should be written to only when the core is in the IDLE state.

Field name	rwu	Bit #	Reset	Description
bssBasicHTMCSSetEM	rw	15:0	16'hFFFF	BSS Basic MCS Set for Equal Modulation This field is used to determine the transmission MCS of some of the packets that are sent by hardware. A one-hot encoding scheme is followed for this field. Each bit position decodes to a particular PHY MCS; the combined setting of the bits decide which rates belong to the BSS Basic MCS Set. The encoding is as follows: bssBasicHTMCSSetEM [0] indicates MCS 0 bssBasicHTMCSSetEM [1] indicates MCS 1 . .

Field name	rwu	Bit #	Reset	Description
				. bssBasicHTMCSSetEM [15] indicates MCS 15 Multiple values can be set at a time. If there is no BSS Basic MCS available, SW must program the BSS Mandatory MCS.
bssBasicHTMCSSetUM	rw	2vht1 :16	6'b0	BSS Basic MCS Set for Unequal Modulation This field is used to determine the transmission MCS of some of the packets that are sent by hardware. A one-hot encoding scheme is followed for this field. Each bit position decodes to a particular PHY MCS; the combined setting of the bits decide which rates belong to the BSS Basic MCS Set. The encoding is as follows: bssBasicHTMCSSetU [0] indicates MCS 33 bssBasicHTMCSSetU [1] indicates MCS 34 . . . bssBasicHTMCSSetU [5] indicates MCS 38 Multiple values can be set at a time. If there is no BSS Basic MCS available, SW must program the BSS Mandatory MCS.

7.9.4 VHTMCS Register (VHTMCSReg)

Offset Address: 0x031C

Contains the various common VHT MCS for the BSS. HW uses this register to determine the transmission rate of the packets that are sent by HW. This register should be written to only when the core is in the IDLE state.

Field name	rwu	Bit #	Reset	Description
bssBasicVHTMCSSet	rw	15:0	16'hFFFF	BSS Basic VHT MCS Set This field is used to determine the transmission MCS of some of the VHT packets that are sent by hardware. The encoding is the same as the VHT Supported MCS defined in the standard The encoding is as follows: bssBasicVHTMCSSet [1:0] indicates nSS1 bssBasicVHTMCSSet [3:2] indicates nSS2 . . . bssBasicHTMCSSetEM [15:14] indicates nSS8 For each nSS,

Field name	rwu	Bit #	Reset	Description
				2'b00 : indicates support of MCS 0-7 2'b01 : indicates support of MCS 0-8 2'b10 : indicates support of MCS 0-9 2'b11 : indicates that this nSS is not supported Multiple values can be set at a time.
Reserved	r	31:16	16'b0	

7.9.5 L-SIG TXOP Register (lstpReg)

Offset Address: 0x0320

This register is used to control the HW behavior related to L-SIG TXOP operation.

Field name	rwu	Bit #	Reset	Description
supportLSTP	rw	0	1'b0	Support of L-SIG TXOP This field indicates if the system supports the L-SIG TXOP. In this case, the hardware detection of the L-SIG TXOP is enabled.

7.9.6 Transmit Bandwidth Drop Information Register (txBWDropInfoReg)

Offset Address: 0x8330

This register provides information to the SW in case of Transmission Bandwidth drop to allow it the reduction of the AMPDU size.

Field name	rwu	Bit #	Reset	Description
txBWAfterDrop	r	1:0	2'h0	Transmit Bandwidth After Drop Indicate the BW used for the on-going transmission 00 : 20MHz 01 : 40MHz 10 : 80MHz 11 : 160MHz
reserved	r	31:2	30'b0	*Reserve*

7.9.7 HE Configuration Register (HEConfigReg)

Offset Address: 0x0324

This register is used to control the HW behavior related to HE operation.

Field name	rwu	Bit #	Reset	Description
defaultPEDur	rw	2:0	3'h0	Provide the PE default duration used for trigger based transmission in TRS
reserved	r	3	1'b0	*Reserved*

Field name	rwu	Bit #	Reset	Description
disableTBResp	rw	4	1'b0	Disable Trigger Based Response This bit is set by SW to disable the automatic Trigger Based response generation logic in HW. Note that this field does not control whether a valid Trigger frame meant for this device is passed to SW.
disableTBCS	rw	5	1'b0	Disable Carrier Sense on Trigger Based Response This bit is set by SW to disable the Carrier Sense on Trigger Based response generation.
reserved	r	7:6	2'b0	*Reserved*
acceptTriggerHWFrames	rw	8	1'b0	Accept Trigger Frame handled by HW If this bit is not set, the Trigger frames Beamforming Report Poll (BRP), MU-BAR, MU-RTS, GCR MU-BAR and Bandwidth Query Report Poll (BQRP) are not passed to SW.
acceptTriggerSWFrames	rw	9	1'b0	Accept Trigger Frame handled by SW If this bit is not set, the Trigger frames Basic Trigger, Buffer Status Report Poll (BSRP) and NDP Feedback Report Poll are not passed to SW.
acceptAllTriggerFrames	rw	10	1'b0	Accept All Trigger Frame If this bit is set, all the Trigger frames addressed or not to the device are passed to SW.
reserved	r	15:11	6'b0	*Reserved*
enableSRP	rw	16	1'b0	Enable SRP Spatial Reuse This field enables the SRP Spatial Reuse feature
enableOBSSPD	rw	17	1'b0	Enable OBSS PD Spatial Reuse This field enables the OBSS-PD feature
reserved	r	19:18	2'b0	*Reserved*
dopplerSupport	rw	20	1'b0	Doppler Support This field indicates the support of Doppler feature
dcmSupport	rw	21	1'b0	DCM Support This field indicates the support of DCM feature
reserved	r	31:22	10'b0	*Reserved*

7.9.8 Spatial Reuse Configuration Register (SPConfig1Reg)

Offset Address: 0x0328

This register contains parameters for Spatial Reuse

Field name	rwu	Bit #	Reset	Description
SRPDisallowed	rw	2:0	3'h0	SRP Disallowed Indicates whether or not SRP-based SR transmissions are allowed
nonSRGOBSSPDSRDisallowed	rw	8	1'b0	Non-SRG OBSS PD SR Disallowed Indicates whether non-SRG OBSS PD SR

Field name	rwu	Bit #	Reset	Description
				transmissions are allowed or not
nonSRGOffsetPresent	rw	8	1'b0	Non-SRG Offset Present Indicates whether or not the Non-SRG OBSS PD Max Offset field is present
srgInformationPresent	rw	9	1'b0	SRG Information Present Indicates whether or not the SRG OBSS PD Min Offset, SRG OBSS PD Max Offset, SRG BSS Color Bitmap and SRG Partial BSSID Bitmap fields are present
reserved	r	15:10	6'b0	Reserved
nonSRGOBSSPDMaxOffset	rw	15:8	8'b0	Non-SRG OBSS PD Max Offset The Non-SRG OBSS PD Max Offset field contains an unsigned integer that is added to -82 dBm to generate the value of the Non-SRG OBSS PD Max parameter
srgOBSSPDMinOffset	rw	23:16	8'b0	SRG OBSS PD Min Offset The SRG OBSS PD Min Offset field contains an unsigned integer that is added to -82 dBm to generate the value of the SRG OBSS PD Min parameter.
srgOBSSPDMaxOffset	rw	31:24	8'b0	SRG OBSS PD Max Offset The SRG OBSS PD Max Offset field contains an unsigned integer that is added to -82 dBm to generate the value of the SRG OBSS PD Max parameter.

7.9.9 SRG BSS Color Bitmap Low Register (SRGBSSColorBitmapLowReg)

Offset Address: 0x032C

This register contains LSB of SRG BSS Color Bitmap

Field name	rwu	Bit #	Reset	Description
SRGBSSColorBitmapLow	rw	31:0	32'h0	SRG BSS Color Bitmap

7.9.10 SRG BSS Color Bitmap High Register (SRGBSSColorBitmapHighReg)

Offset Address: 0x0330

This register contains MSB of SRG BSS Color Bitmap

Field name	rwu	Bit #	Reset	Description
SRGBSSColorBitmapHigh	rw	31:0	32'h0	SRG BSS Color Bitmap

7.9.11 SRG Partial BSSID Bitmap Low Register (SRGPartialBSSIDBitmapLowReg)

Offset Address: 0x0334

This register contains LSB of Partial BSSID Bitmap

Field name	rwu	Bit #	Reset	Description

Field name	rwu	Bit #	Reset	Description
SRGPartialBSSIDBitmapLow	rw	31:0	32'h0	Partial BSSID Bitmap

7.9.12 SRG Partial BSSID Bitmap High Register (SRGPartialBSSIDBitmapHighReg)

Offset Address: 0x0338

This register contains MSB of Partial BSSID Bitmap

Field name	rwu	Bit #	Reset	Description
SRGPartialBSSIDBitmapHigh	rw	31:0	32'h0	Partial BSSID Bitmap

7.9.13 Beamformee Control Register (bfmeeControlReg)

Offset Address: 0x0350

This register allows the SW to control the Beamforming as a Beamformee feature. It is available only if RW_BFMEE_EN is defined. Refer section *8.5 Configuration*.

Field name	rwu	Bit #	Reset	Description
bfmeeEnable	rw	0	1'h0	Beamformee Enable When set by the SW, this field enables the support of the beamforming calibration procedure as a beamformee.
bfmeeMUSupport	rw	1	1'h0	Beamformee MU Support When set by the SW, this field indicates that the Core supports both SU and MU beamforming calibration procedure. When reset, this field indicates that the Core supports only the SU beamforming calibration procedure.
bfmeeCodebook	rw	2	1'h1	Beamformee Codebook Indicates the size of codebook entries: If Feedback Type is SU: Set to 0 for 2 bits for ψ, 4 bits for ϕ Set to 1 for 4 bits for ψ, 6 bits for ϕ If Feedback Type is MU: Set to 0 for 5 bits for ψ, 7 bits for ϕ Set to 1 for 7 bits for ψ, 9 bits for ϕ
bfmeeGrouping	rw	4:3	2'd0	Beamformee Grouping Indicates the subcarrier grouping, Ng, used for the compressed beamforming feedback matrix: Set to 0 for Ng = 1 (No grouping)

Field name	rwu	Bit #	Reset	Description
				Set to 1 for Ng = 2 Set to 2 for Ng = 4 The value 3 is reserved
bfmeeNr	rw	7:5	3'd1	Beamformee Number of rows Indicates the number of rows, Nr, in the compressed beamforming feedback matrix minus 1: Set to 0 for Nr = 1 Set to 1 for Nr = 2 … Set to 7 for Nr = 8
bfmeeNc	rw	10:8	3'd1	Beamformee Number of columns Indicates the number of columns, Nc, in the compressed beamforming feedback matrix minus 1: Set to 0 for Nc = 1 Set to 1 for Nc = 2 … Set to 7 for Nc = 8
Reserved	r	15:8	30'b0	*Reserved*
bfrMCS	rw	23:16	8'h84	Beamforming Report MCS This field indicates the rate at which the Beamforming Report created by HW is transmitted. The encoding is as follows: 1 Mbps: {1'b0, 7'd0} 2 Mbps: {1'b0, 7'd1} 5.5 Mbps: {1'b0, 7'd2} 11 Mbps: {1'b0, 7'd3} 6 Mbps: {1'b0, 7'd4} 9 Mbps: {1'b0, 7'd5} 12 Mbps: {1'b0, 7'd6} 18 Mbps: {1'b0, 7'd7} 24 Mbps: {1'b0, 7'd8} 36 Mbps: {1'b0, 7'd9} 48 Mbps: {1'b0, 7'd10} 54 Mbps: {1'b0, 7'd11} HT rates: {1'b1, 7'd*MCS Index*} VHT rates: {1'b1 ,3'd*nSS*, 4'd*MCS index*}
bfrShortGI	rw	24	1'b0	Beamforming Report Short GI Indicates whether a short Guard Interval (GI) is used in the transmission of the Beamforming Report.
bfrFormatMod	rw	27:25	3'h4	Beamforming Report Format Modulation

Field name	rwu	Bit #	Reset	Description
				This field indicates the format and the modulation of the beamforming report frame created by HW. The encoding is as follows: 3'b000: NON-HT 3'b001: NON-HT-DUP-OFDM 3'b010: HT-MF 3'b011: HT-GF 3'b100: VHT
Reserved	r	31:28	4'b0	*Reserved*

7.9.14 Transmit Trigger Based Information Register (txHETBInfoReg)

Offset Address: 0x8354

This register provides information to the SW during HE TB transmission

Field name	rwu	Bit #	Reset	Description
txHETBRemDur	r	5:0	6'h0	HE TB Preparation Remaining Duration This field indicates the remaining duration for completing the HE_TB programming and for setting the newHead in TB DMA channel. If this downcounter expired and the newHead has not been set, an underrun may occur and phyErr may be generated by the PHY.
reserved	r	7:6	2'b0	*Reserved*
txHETBMaxDur	rw	13:8	6'h0	HE TB Preparation Max Duration This field indicates the maximum duration allowed to the SW for completing the HE_TB programming and for setting the newHead in TB DMA channel.
reserved	r	15:14	2'b0	*Reserved*
raRUEnable	rw	16	1'b0	Random Access RU Enable This field enables the transmission in Random Access RU if eligible RA-RU is found.
raRUType	rw	17	1'b0	Random Access RU Type This field indicates which type of Random Access RU can be used. 1'b0 : The device is associated and will search for Associated RU with AID = 0 1'b1 : The device is not associated and will search for Non-Associated RU with AID = 2045
reserved	r	19:18	2'b0	*Reserved*
eOCWMin[2:0]	rw	22:20	3'd3	Exposent of OFDMA Contention Window Minimum This field indicates the minimum exposent of

Field name	rwu	Bit #	Reset	Description
				OFDMA Contention Window.
reserved	r	23	1'b0	*Reserved*
eOCW[2:0]	rwu	26:24	3'd3	Exposent of OFDMA Contention Window This field indicates the current exposent of OFDMA Contention Window.
reserved	r	27	1'b0	*Reserved*
maxMCSInHETB[3:0]	rw	31:28	4'd7	Max MCS in HE TB This field indicates the highest MCS supported within a Trigger Based transmission.

7.9.15 Receive HE Trigger Common Information Register (rxHETrigCommonInfoReg)

Offset Address: 0x8358

This register provides Common Information to the SW upon reception of Trigger frame to ease the HE_TB generation

Field name	rwu	Bit #	Reset	Description
ulTriggerType	R	2:0	3'h0	UpLink Trigger Type This field indicates the Type of the trigger frame received.
ulLength	r	15:4	12'b0	UpLink Length This field indicates the UL Length extracted from the Common Info subfield of the Trigger frame.
ulBW	r	17:16	2'b0	UpLink Bandwidth This field indicates the BW extracted from the Common Info subfield of the Trigger frame.
ulGILTFType	r	19:18	2'b0	UpLink GI And LTF Type This field indicates the GI And LTF Type extracted from the Common Info subfield of the Trigger frame.
ulMULTFMode	r	20	1'b0	UpLink MU-MIMO LTF Mode This field indicates the MU-MIMO LTF Mode extracted from the Common Info subfield of the Trigger frame.
ulNLTFAndMidamble	r	23:21	3'b0	UpLink Number Of HE-LTF Symbols And Midamble Periodicity This field indicates the Number Of HE-LTF Symbols And Midamble Periodicity extracted from the Common Info subfield of the Trigger frame.
ulSTBC	r	24	1'h0	UpLink STBC This field indicates the STBC extracted from the Common Info subfield of the Trigger frame.
ulLDPCExtrSymb	r	25	1'b0	UpLink LDPC Extra Symbol Segment This field indicates the LDPC Extra Symbol Segment extracted from the Common Info subfield of the Trigger frame.
ulPreFecPadding	r	27:26	2'b0	UpLink Pre-FEC Padding Factor This field indicates the Pre-FEC Padding Factor extracted from the Common Info subfield of the Trigger frame.

Field name	rwu	Bit #	Reset	Description
ulPEDisambiguity	r	28	1'b0	UpLink PE Disambiguity This field indicates the PE Disambiguity extracted from the Common Info subfield of the Trigger frame.
ulDoppler	r	29	1'b0	UpLink Doppler This field indicates the Doppler extracted from the Common Info subfield of the Trigger frame.
Reserved	r	31:30	2'b0	Reserved

7.9.16 Receive HE Trigger User Information Register (rxHETrigUserInfoReg)

Offset Address: 0x835C

This register provides User Information to the SW upon reception of Trigger frame to ease the HE_TB generation

Field name	rwu	Bit #	Reset	Description
ulRUSize	r	2:0	3'h0	UpLink RU Size This field indicates the ruSize derivated from the RU Allocation extracted from the User Info subfield of the Trigger frame.
ulFECCoding	r	3	1'h0	UpLink Fec Coding This field indicates the FEC Coding extracted from the User Info subfield of the Trigger frame.
ulMCS	r	7:4	4'h0	UpLink MCS This field indicates the MCS extracted from the User Info subfield of the Trigger frame.
ulDCM	r	8	1'h0	UpLink DCM This field indicates the DCM extracted from the User Info subfield of the Trigger frame.
ulNSS[2:0]	r	11:9	3'h0	UpLink NSS This field indicates the Number of SS extracted from the User Info subfield of the Trigger frame.
ulRUType	r	12	1'h0	UpLink RU Type This field indicates if the RU is a RA-RU and a user-Specific RU. 1'b0 : User Specific RU 1'b1 : RA-RU
reserved	r	23:13	11'h0	*Reserved*
tdUserInfoSubfield	r	31:24	8'h0	Trigger Dependent User Info subfield This field indicates the Trigger Dependent User Info subfield extracted from the User Info subfield of the Trigger frame.

7.9.17 Secondary Users Transmit Interrupt Event Register (secUsersTxIntEventReg)

Offset Address: 0x8364 (*secUsersTxIntEventSetReg*) and 0x8368 (*secUsersTxIntEventClearReg*)

This register reflects the state of the various interrupt sources coming from secondary users transmit paths.

The interrupt bits are set by HW on an asserting edge of the corresponding interrupt signal provided the corresponding bit in the *secUsersTxIntUnmaskReg* register is set. Each bit is also set by HW when SW performs a write with a value "1" to the corresponding bit in the *secUsersTxIntEventSetReg* address, for debug purposes. Each bit is reset by HW when SW performs a write with a value "1" to the corresponding bit in the *secUsersTxIntEventClearReg* address.

When an interrupt is generated, the *secUserTxTrigger* in *Transmit / Receive Interrupt Event Register (txRxIntEventReg)* is set. (Refer to *Transmit / Receive Interrupt Event Register (txRxIntEventReg)* for more information).

A *transmission trigger* (e.g. *secUXacXTxTrigger*) is generated when the *interruptEnTx* bit is set in any *Transmit DMA Header Descriptor* that is handled by the HW.

A *transmission buffer trigger* (e.g. *secUXacXTxBufTrigger*) is generated when the *interruptEnTx* bit is set in any *Transmit DMA Header Descriptor* that is handled by the HW on the Secondary User Transmit path.

It is available only if **RW_MUMIMO_TX_EN** is defined.

Field name	rscu	Bit #	Reset	Description
secU1ac0TxTrigger	rscu	0	1'b0	Secondary User 1 AC0 Transmission Trigger This interrupt indicates that a frame from the AC_BK channel on the secondary User 1 path has been transmitted. This interrupt is enabled from the THD of the frame.
secU1ac1TxTrigger	rscu	1	1'b0	Secondary User 1 AC1 Transmission Trigger This interrupt indicates that a frame from the AC_BE channel on the secondary User 1 path has been transmitted. This interrupt is enabled from the THD of the frame.
secU1ac2TxTrigger	rscu	2	1'b0	Secondary User 1 AC2 Transmission Trigger This interrupt indicates that a frame from the AC_VI channel on the secondary User 1 path has been transmitted. This interrupt is enabled from the THD of the frame.
secU1ac3TxTrigger	rscu	3	1'b0	Secondary User 1 AC3 Transmission Trigger This interrupt indicates that a frame from the AC_VO channel on the secondary User 1 path has been transmitted. This interrupt is enabled from the THD of the frame.
secU1ac0TxBufTrigger	r	4	1'b0	Secondary User 1 AC0 Transmission Buffer Trigger This interrupt indicates that a Payload Buffer from the AC_BK channel on the secondary User 1 path has been transmitted. This interrupt is enabled from the TPD of the frame.
secU1ac1TxBufTrigger	r	5	1'b0	Secondary User 1 AC1 Transmission Buffer Trigger This interrupt indicates that a Payload Buffer from the AC_BE channel on the secondary User 1 path has been transmitted. This interrupt is enabled from the TPD of the frame.

Field name	rscu	Bit #	Reset	Description
secU1ac2TxBufTrigger	rscu	6	1'b0	Secondary User 1 AC2 Transmission Buffer Trigger This interrupt indicates that a Payload Buffer from the AC_VI channel on the secondary User 1 path has been transmitted. This interrupt is enabled from the TPD of the frame.
secU1ac3TxBufTrigger	rscu	7	1'b0	Secondary User 1 AC3 Transmission Buffer Trigger This interrupt indicates that a Payload Buffer from the AC_VO channel on the secondary User 1 path has been transmitted. This interrupt is enabled from the TPD of the frame.
secU2ac0TxTrigger	rscu	8	1'b0	Secondary User 2 AC0 Transmission Trigger This interrupt indicates that a frame from the AC_BK channel on the secondary User 2 path has been transmitted. This interrupt is enabled from the THD of the frame.
secU2ac1TxTrigger	rscu	9	1'b0	Secondary User 2 AC1 Transmission Trigger This interrupt indicates that a frame from the AC_BE channel on the secondary User 2 path has been transmitted. This interrupt is enabled from the THD of the frame.
secU2ac2TxTrigger	rscu	10	1'b0	Secondary User 2 AC2 Transmission Trigger This interrupt indicates that a frame from the AC_VI channel on the secondary User 2 path has been transmitted. This interrupt is enabled from the THD of the frame.
secU2ac3TxTrigger	r	11	1'b0	Secondary User 2 AC3 Transmission Trigger This interrupt indicates that a frame from the AC_VO channel on the secondary User 2 path has been transmitted. This interrupt is enabled from the THD of the frame.
secU2ac0TxBufTrigger	r	12	1'b0	Secondary User 2 AC0 Transmission Buffer Trigger This interrupt indicates that a Payload Buffer from the AC_BK channel on the secondary User 2 path has been transmitted. This interrupt is enabled from the TPD of the frame.
secU2ac1TxBufTrigger	rscu	13	1'b0	Secondary User 2 AC1 Transmission Buffer Trigger This interrupt indicates that a Payload Buffer from the AC_BE channel on the secondary User 2 path has been transmitted. This interrupt is enabled from the TPD of the frame.
secU2ac2TxBufTrigger	rscu	14	1'b0	Secondary User 2 AC2 Transmission Buffer Trigger This interrupt indicates that a Payload Buffer from the AC_VI channel on the secondary User 2 path has been transmitted. This interrupt is enabled from the TPD of the frame.

Field name	rscu	Bit #	Reset	Description
secU2ac3TxBufTrigger	Rscu	15	1'b0	Secondary User 2 AC3 Transmission Buffer Trigger This interrupt indicates that a Payload Buffer from the AC_VO channel on the secondary User 2 path has been transmitted. This interrupt is enabled from the TPD of the frame.
secU3ac0TxTrigger	rscu	16	1'b0	Secondary User 3 AC0 Transmission Trigger This interrupt indicates that a frame from the AC_BK channel on the secondary User 3 path has been transmitted. This interrupt is enabled from the THD of the frame.
secU3ac1TxTrigger	rscu	17	1'b0	Secondary User 3 AC1 Transmission Trigger This interrupt indicates that a frame from the AC_BE channel on the secondary User 3 path has been transmitted. This interrupt is enabled from the THD of the frame.
secU3ac2TxTrigger	rscu	18	1'b0	Secondary User 3 AC2 Transmission Trigger This interrupt indicates that a frame from the AC_VI channel on the secondary User 3 path has been transmitted. This interrupt is enabled from the THD of the frame.
secU3ac3TxTrigger	rscu	19	1'b0	Secondary User 3 AC3 Transmission Trigger This interrupt indicates that a frame from the AC_VO channel on the secondary User 3 path has been transmitted. This interrupt is enabled from the THD of the frame.
secU3ac0TxBufTrigger	rscu	20	1'b0	Secondary User 3 AC0 Transmission Buffer Trigger This interrupt indicates that a Payload Buffer from the AC_BK channel on the secondary User 3 path has been transmitted. This interrupt is enabled from the TPD of the frame.
secU3ac1TxBufTrigger	rscu	21	1'b0	Secondary User 3 AC1 Transmission Buffer Trigger This interrupt indicates that a Payload Buffer from the AC_BE channel on the secondary User 3 path has been transmitted. This interrupt is enabled from the TPD of the frame.
secU3ac2TxBufTrigger	rscu	22	1'b0	Secondary User 3 AC2 Transmission Buffer Trigger This interrupt indicates that a Payload Buffer from the AC_VI channel on the secondary User 3 path has been transmitted. This interrupt is enabled from the TPD of the frame.
secU3ac3TxBufTrigger	rscu	23	1'b0	Secondary User 3 AC3 Transmission Buffer Trigger This interrupt indicates that a Payload Buffer from the AC_VO channel on the secondary User 3 path has been transmitted. This interrupt is enabled from the TPD of the frame.

Field name	rscu	Bit #	Reset	Description
secU1PTError	rscu	24	1'b0	Secondary User 1 Policy Table Error The core sets this bit when it finds that the *uPatternPT* read from the Policy Table on the secondary User 1 path did not contain the expected pattern.
secU1TxDMADead	rscu	25	1'b0	Secondary User 1 Transmit DMA Dead This interrupt is raised when the secondary User 1 path DMA channel moves to the DEAD state. Refer to section *9.3*, *Terminal DMA errors* for details.
secU2PTError	rscu	26	1'b0	Secondary User 2 Policy Table Error The core sets this bit when it finds that the *uPatternPT* read from the Policy Table on the secondary User 2 path did not contain the expected pattern.
secU2TxDMADead	rscu	27	1'b0	Secondary User 2 Transmit DMA Dead This interrupt is raised when the secondary User 2 path DMA channel moves to the DEAD state. Refer to section *9.3*, *Terminal DMA errors* for details.
secU3PTError	rscu	28	1'b0	Secondary User 3 Policy Table Error The core sets this bit when it finds that the *uPatternPT* read from the Policy Table on the secondary User 3 path did not contain the expected pattern.
secU3TxDMADead	rscu	29	1'b0	Secondary User 3 Transmit DMA Dead This interrupt is raised when the secondary User 3 path DMA channel moves to the DEAD state. Refer to section *9.3*, *Terminal DMA errors* for details.
Reserved	r	31:30	2'b0	*Reserved*

7.9.18 Secondary Users Transmit Interrupt UnMask Register (secUsersTxIntUnMaskReg)

Offset Address: 0x836C

The bits in this register have the same format as the *secUsersTxIntEventSetReg* register, with the addition of *masterSecUsersTxIntEn* (bit 31). A one bit in the *secUsersTxIntUnmaskReg* register enables the corresponding bit in the *genIntEventSetReg* register to generate a processor interrupt. A zero bit in the *txRxIntUnMask* register disables the corresponding *txRxIntEventSetReg* register bit from getting set and hence generating an interrupt.

If *masterSecUsersTxIntEn* is 0, all interrupts are disabled regardless of the values of all other bits in the *secUsersTxIntUnmaskReg* register. The value of *masterSecUsersTxIntEn* has no effect on the value returned by reading the *secUsersTxIntEventSetReg* or *secUsersTxIntEventClearReg*. Even if *masterSecUsersTxIntEn* is 0, reading *secUsersTxIntEventSetReg* or the *secUsersTxIntEventClearReg* will return the status of the interrupt bits.

It is available only if **RW_MUMIMO_TX_EN** is defined.

Field name	rwu	Bit #	Reset	Description

Field name	rwu	Bit #	Reset	Description
Unmask bits for Interrupts defined in bits 30:0 in *secUsersTxIntEventSetReg* register	rw	30:0	31'b0	See *Secondary Users Transmit Interrupt Event Register (secUsersTxIntEventReg)* definition.
masterSecUsersTxIntEn	rw	31	1'b0	If set, external interrupts will be generated in accordance with the rest of the *secUsersTxIntUnMaskReg* register bits. If clear, no external interrupts will be generated regardless of the *secUsersTxIntUnMaskReg* register bit settings.

7.10 Coex register definitions

7.10.1 Coex Control Register (coexCntlReg)

Offset Address: 0x400

This register is used to control the HW behavior related to Coexistence operation. It is available only if RW_WLAN_COEX_EN is defined. Refer section *2.5 Coexistence support*.

Field name	rwu	Bit #	Reset	Description
coexEnable	rw	0	1'b0	Coex Enable When set, the informations used for coexistence are provided to an external module. Otherwise, all the coexistence interface output signals are force to 0
coexPHYTxAbortEnable	rw	1	1'b0	Coex Phy Tx Abort Enable. When sets, the MAC handles the Tx abort of the PHY by de-asserting the txReq. This is valid only in case of ***coexEnable*** and ***coexWlanTxAbort*** input set.
coexPHYRxAbortEnable	rw	2	1'b0	Coex Phy Rx Abort Enable. When sets, the MAC handles the Rx abort of the PHY by de-asserting the rxReq. This is valid only in case of ***coexEnable*** and ***coexWlanRxAbort*** input set.
coexPostponeTxEnable	rw	3	1'b1	Coex Postpone Tx Enable When sets, the MACHW postpone all the backoff mechanism and as the consequence all the transmission when ***coexWlanTxAbort*** input is set.
coexForceEnable	rw	4	1'b0	Coex Force Enable When set, the output signals of the coexistence interface are directly controlled by SW using ***coexForceWlanXXX*** fields

Field name	rwu	Bit #	Reset	Description
coexAutoPTIAdjEnable	rw	5	1'b1	Coex Automatic PTI Adjustment Enable When sets, the MAC HW dynamically adjusts the PTI based on the abort rate. Each time an on-going transmission/reception is aborted, the corresponding PTI information is incremented. On the other side, when a transmission /reception is completed without being aborted, the PTI for this category is reset to its register value. See *2.5.3.1 Automatic PTI Adjustment*.
coexAutoPTIAdjIncr	rw	7:6	2'b1	Coex Automatic PTI Adjustmnent Increment This field indicates the increment step done on the PTI value each time an abort is detected. This field is valid only when ***coexAutoPTIAdjEnable*** is enabled.
Reserved	r	11:8	4'b0	*Reserved*
coexWlanChanOffset	rw	12	1'b0	Coex Wlan Channel offset Indicate where the primary channel is in the 40MHz band. When set, the primary channel is on the upper 20MHz of the 40MHz channel. When reset, the primary channel is on the lower 20MHz of the 40MHz channel.
Reserved	r	15:13	3'b0	*Reserved*.
coexWlanChanFreq	rw	22:16	7'b0	Coex Wlan Channel Frequency Indicates the current center frequency (in MHz) of the primary channel used by the WLAN system starting from 2400MHz.
Reserved	r	23	1'b0	*Reserved*.
coexForceWlanTx	rw	24	1'b0	Coex Force Wlan Tx output Force the value of coexWlanTx output.
coexForceWlanRx	rw	25	1'b0	Coex Force Wlan Rx output Force the value of coexWlanR x output.
coexForceWlanChanBw	rw	26	1'b0	Coex Force Wlan Channel Bandwidth output Force the value of coexWlanChanBw output.
coexForceWlanPTIToggle	rw	27	1'b0	Coex Force Wlan PTI toggle output Force the value of coexWlanPTIToggle output.
coexForceWlanPTI	rw	31:28	4'b0	Coex Force Wlan PTI output Force the value of coexWlanPTI output.

7.10.2 Coex PTI Register (coexPTIReg)

Offset Address: 0x404

This register provides the PTI value for each frame types. It is available only if RW_WLAN_COEX_EN is defined. Refer section *2.5 Coexistence support*.

Field name	rwu	Bit #	Reset	Description
coexPTIAck	rw	3:0	4'd7	Coex Packet traffic information for Ack/BA frames
coexPTICntl	rw	7:4	4'd3	Coex Packet traffic information for Control frames
coexPTIMgt	rw	11:8	4'd6	Coex Packet traffic information for Management frames
coexPTIVOData	rw	15:12	4'd6	Coex Packet traffic information for Data frames sent on the AC VO Channel
coexPTIVIData	rw	19:16	4'd4	Coex Packet traffic information for Data frames sent on the AC VI Channel
coexPTIBEData	rw	23:20	4'd2	Coex Packet traffic information for Data frames sent on the AC BE Channel
coexPTIBKData	rw	27:24	4'd0	Coex Packet traffic information for Data frames sent on the AC BK Channel
coexPTIBcnData	rw	31:28	4'd0	Coex Packet traffic information for Data frames sent on the Beacon Channel

7.10.3 Coex Status Register (coexStatReg)

Offset Address: 0x408

This register is used by the SW to get the Coexistence Interface current status. . It is available only if RW_WLAN_COEX_EN is defined. Refer section *2.5 Coexistence support*.

Field name	rwu	Bit #	Reset	Description
coexWlanTxAbortState	r	0	1'b0	Coex Wlan Tx Abort State Current value of coexWlanTxAbort input
coexWlanRxAbortState	r	1	1'b0	Coex Wlan Rx Abort State Current value of coexWlanRxAbort input
coexWlanTxState	r	2	1'b0	Coex Wlan Tx State Current value of coexWlanTx output.
coexWlanRxState	r	3	1'b0	Coex Wlan Rx State Current value of coexWlanRx output.
coexWlanChanBwState	r	4	1'b0	Coex Wlan Channel Bandwidth State Current value of coexWlanChanBw output.
coexWlanPTIToggleState	r	5	1'b0	Coex Wlan PTI Toggle State Current value of coexWlanPTIToggle output.
Reserved	r	7:6	2'b0	*Reserved*
coexWlanPTIState	r	11:8	4'b0	Coex Wlan PTI State Current value of coexWlanPTI output.

Field name	rwu	Bit #	Reset	Description
Reserved	r	31:12	20'b0	*Reserved*

7.10.4 Coex Interrupt Register (coexIntReg)

Offset Address: 0x40c

This register is used by the SW to control how the coex interrupt is generated. . It is available only if RW_WLAN_COEX_EN is defined. Refer section *2.5 Coexistence support*.

Field name	rwu	Bit #	Reset	Description
coexWlanTxAbortRiseEn	rw	0	1'b0	Coex Wlan Tx Abort Rising Edge Enable When set, the Coex Interrupt is generated on the rising edge of *coexWlanTxAbort* input
coexWlanTxAbortFallEn	rw	1	1'b0	Coex Wlan Tx Abort Falling Edge Enable When set, the Coex Interrupt is generated on the falling edge of *coexWlanTxAbort* input
coexWlanRxAbortRiseEn	rw	2	1'b0	Coex Wlan Rx Abort Rising Edge Enable When set, the Coex Interrupt is generated on the rising edge of *coexWlanRxAbort* input
coexWlanRxAbortFallEn	rw	3	1'b0	Coex Wlan Rx Abort Falling Edge Enable When set, the Coex Interrupt is generated on the falling edge of *coexWlanRxAbort* input
Reserved	rw	31:4	28'b0	*Reserved*

7.11 Debug register definitions

7.11.1 Debug HW State Machine 1 Register (debugHWSM1Reg)

Offset Address: 0x0500

This register holds the latched value of the state of the three important HW state machines when a HW error is detected, i.e. when *General Interrupt Event Register (genIntEventReg).hwErr* interrupt is raised. The value is latched to make it easier for SW to read the value at a later time. This register is reset by hardware when software sets *MAC Error Recovery Control Register (macErrRecCntrlReg).hwFSMReset*.

Note that this field may be overwritten before SW has had a chance to read the register if another error is detected.

Field name	rwu	Bit #	Reset	Description
rxControlerLs	ru	5:0	6'b0	Receive Controller Latched State Indicates the latched state of the core's Receive Controller state machine in hex.
Reserved	r	6	1'b0	Reserved.
txControlLs	ru	15:7	9'h0	Transmit Controller Latched State Indicates the latched state of the core's Transmit Controller state machine in hex.

Field name	rwu	Bit #	Reset	Description
Reserved	r	23:16	8'b0	Reserved
macControlLs	ru	30:24	7'b0	MAC Controller Latched State Indicates the latched state of the core's MAC Controller state machine in hex.
Reserved	r	31	1'b0	Reserved.

7.11.2 Debug HW State Machine 2 Register (debugHWSM2Reg)

Offset Address: 0x0504

This register indicates the current state of the three important HW state machines at any instant of time.

Field name	rwu	Bit #	Reset	Description
rxControlCs	ru	5:0	6'b0	Receive Controller Current State Indicates the current state of the core's Receive controller state machine in hex.
Reserved	r	6	1'b0	Reserved.
txControlCs	ru	15:7	9'h0	Transmit Controller Current State Indicates the current state of the core's Transmit controller state machine in hex.
Reserved	r	23:16	8'b0	Reserved
macControlCs	ru	30:24	7'b0	MAC Controller Current State Current state of the core's MAC Controller state machine in hex.
Reserved	r	31	1'b0	Reserved.

7.11.3 Reserved (NA)

Offset Address: 0x0508

Reserved for future use.

Field name	rwu	Bit #	Reset	Description
Reserved	r	31:0	32'b0	Reserved

7.11.4 Debug Port Value Register (debugPortValueReg)

Offset Address: 0x050C

Field name	rwu	Bit #	Reset	Description
debugPortRead	ru	31:0	Unknown	This register reflects the value of the *debugPort* [31:0] signals, the 32 top level debug signals.

7.11.5 Debug Port Select Register (debugPortSelReg)

Offset Address: 0x0510

This register is used as a dial-in mux to select different sets of signals and bring them out to the top level debug pins for debug purposes.

Field name	rwu	Bit #	Reset	Description
debugPortSel1	rw	7:0	8'b0	Debug Port Selection 1 The value in this field is used to select a group of debug signals which will reflect on *debugPort* [15:0] of the 32 top level debug signals available.
debugPortSel2	rw	15:8	8'b0	Debug Port Selection 2 The value in this field is used to select a group of debug signals which will reflect on *debugPort* [31:16] of the 32 top level debug signals available.
Reserved	r	31:16	16'b0	Reserved

7.11.6 Debug Basic NAV Register (debugBasicNAVReg)

Offset Address: 0x0514

The HW Basic NAV counter is exposed for debug purposes.

Field name	rwu	Bit #	Reset	Description
basicNAVCounter	ruw	25:0	26'b0	Basic Network Allocation Vector Counter. Gives the current value of the Basic NAV counter for debug purposes. SW can write into the register for debug purposes.
Reserved	ru	31:26	6'b0	Reserved

7.11.7 Debug Contention Window Register (debugCWReg)

Offset Address: 0x0518

The HW CW values are exposed for debug purposes.

Field name	rwu	Bit #	Reset	Description
currentCW0	ru	3:0	4'd4	Current Contention Window for AC0 Gives the current value 'n' of the contention window (CW) in use by the back-off algorithm for Access Category "AC_BK", where CW = 2^n - 1.
currentCW1	ru	7:4	4'd4	Current Contention Window for AC1 Gives the current value 'n' of the contention window (CW) in use by the back-off algorithm for Access Category "AC_BE", where CW = 2^n - 1.
currentCW2	ru	11:8	4'd3	Current Contention Window for AC2 Gives the current value 'n' of the contention window (CW) in use by the back-off algorithm for Access Category "AC_VI", where CW = 2^n - 1.
currentCW3	ru	15:12	4'd2	Current Contention Window for AC3 Gives the current value 'n' of the contention window (CW) in use by the back-off algorithm for Access Category "AC_VO", where CW = 2^n - 1.

Field name	rwu	Bit #	Reset	Description
activeAC	r	18:16	3'b0	Current Access Category Gives the current AC used
Reserved	r	23:19	13'b0	Reserved
BackoffOffset	rw	25:24	2'b0	Backoff Offset Allow to add a programmable number of slots in the backoff procedure. This feature is for debug purpose only.
Reserved	r	31:19	13'b0	Reserved

7.11.8 Debug QoS Station Short Retry Count Register (debugQSRCReg)

Offset Address: 0x051C

The HW QoS Short Retry Counters are exposed for debug purposes.

Field name	rwu	Bit #	Reset	Description
ac0QSRC	ru	7:0	8'b0	AC 0 QoS station Short Retry Count Gives the current value of the QSRC in use by the back-off algorithm for Access Category "AC_BK".
ac1QSRC	ru	15:8	8'b0	AC 1 QoS station Short Retry Count Gives the current value of the QSRC in use by the back-off algorithm for Access Category "AC_BE".
ac2QSRC	ru	23:16	8'b0	AC 2 QoS station Short Retry Count Gives the current value of the QSRC in use by the back-off algorithm for Access Category "AC_VI".
ac3QSRC	ru	31:24	8'b0	AC 3 QoS station Short Retry Count Gives the current value of the QSRC in use by the back-off algorithm for Access Category "AC_VO".

7.11.9 Debug QoS Station Long Retry Count Register (debugQLRCReg)

Offset Address: 0x0520

The HW QoS Long Retry Counters are exposed for debug purposes.

Field name	rwu	Bit #	Reset	Description
ac0QLRC	ru	7:0	8'b0	AC 0 QoS station Long Retry Count Gives the current value of the QLRC in use by the back-off algorithm for Access Category "AC_BK".
ac1QLRC	ru	15:8	8'b0	AC 1 QoS station Long Retry Count Gives the current value of the QLRC in use by the back-off algorithm for Access Category "AC_BE".
ac2QLRC	ru	23:16	8'b0	AC 2 QoS station Long Retry Count Gives the current value of the QLRC in use by the back-off algorithm for Access Category "AC_VI".

Field name	rwu	Bit #	Reset	Description
ac3QLRC	ru	31:24	8'b0	AC 3 QoS station Long Retry Count Gives the current value of the QLRC in use by the back-off algorithm for Access Category "AC_VO".

7.11.10 Debug Beacon Status Pointer Register (debugBcnSPtrReg)

Offset Address: 0x8524

This register reflects the current Pointer on BCN channel

Field name	rwu	Bit #	Reset	Description
bcnStatusPointer	ru	31:0	32'b0	Beacon Status Pointer The Beacon DMA channel Status Pointer is exposed for debug purposes.

7.11.11 Debug AC0 Status Pointer Register (debugAC0SPtrReg)

Offset Address: 0x8528

This register reflects the current Pointer on AC0 channel

Field name	rwu	Bit #	Reset	Description
ac0StatusPointer	ru	31:0	32'b0	AC0 Status Pointer The AC_BK DMA channel Status Pointer is exposed for debug purposes.

7.11.12 Debug AC1 Status Pointer Register (debugAC1SPtrReg)

Offset Address: 0x852C

This register reflects the current Pointer on AC1 channel

Field name	rwu	Bit #	Reset	Description
ac1StatusPointer	ru	31:0	32'b0	AC1 Status Pointer The AC_BE DMA channel Status Pointer is exposed for debug purposes.

7.11.13 Debug AC2 Status Pointer Register (debugAC2SPtrReg)

Offset Address: 0x8530

This register reflects the current Pointer on AC2 channel

Field name	rwu	Bit #	Reset	Description

Field name	rwu	Bit #	Reset	Description
ac2StatusPointer	ru	31:0	32'b0	AC2 Status Pointer The AC_VI DMA channel Status Pointer is exposed for debug purposes.

7.11.14 Debug AC3 Status Pointer Register (debugAC3SPtrReg)

Offset Address: 0x8534

This register reflects the current Pointer on AC3 channel

Field name	rwu	Bit #	Reset	Description
ac3StatusPointer	ru	31:0	32'b0	AC3 Status Pointer The AC_VO DMA channel Status Pointer is exposed for debug purposes.

7.11.15 Debug TB Status Pointer Register (debugTBSPtrReg)

Offset Address: 0x0538

This register reflects the current Pointer on TB channel

Field name	rwu	Bit #	Reset	Description
tbStatusPointer	ru	31:0	32'b0	TB Status Pointer The TB DMA channel Status Pointer is exposed for debug purposes

7.11.16 Reserved (NA)

Offset Address: 0x053C

Reserved for future use.

Field name	rwu	Bit #	Reset	Description
Reserved	ru	31:0	32'b0	Reserved

7.11.17 Debug Transmit DMA Current Pointer Register (debugTxCPtrReg)

Offset Address: 0x8540

Field name	rwu	Bit #	Reset	Description
txCurrentPointer	ru	31:0	32'b0	Transmit Current Pointer The Transmit DMA Current Pointer is exposed for debug purposes.

7.11.18 Reserved (NA)

Offset Address: 0x8544

Reserved for future use.

Field name	rwu	Bit #	Reset	Description
Reserved	ru	31:0	32'b0	Reserved

7.11.19 Reserved (NA)

Offset Address: 0x8548

Reserved for future use.

Field name	rwu	Bit #	Reset	Description
Reserved	ru	31:0	32'b0	Reserved

7.11.20 Reserved (NA)

Offset Address: 0x854C

Reserved for future use.

Field name	rwu	Bit #	Reset	Description
Reserved	ru	31:0	32'b0	Reserved

7.11.21 Debug Intra BSS NAV Register (debugIntraNAVReg)

Offset Address: 0x8550

The HW Intra NAV counter is exposed for debug purposes.

Field name	rwu	Bit #	Reset	Description
intraNavCounter	ruw	25:0	26'b0	Intra BSS Network Allocation Vector Counter. Gives the current value of the Intra BSS NAV counter for debug purposes. SW can write into the register for debug purposes.
Reserved	ru	31:26	6'b0	Reserved

7.11.22 Debug UORA Register (debugUORAReg)

Offset Address: 0x0558

UORA for debug purposes.

Field name	rwu	Bit #	Reset	Description
ocwLFSR	rwu	7	7'b1	OFDMA Contention Window LFSR Gives the current value of the OCW LFSR for debug purposes. Allow software to set the current value of OCW LFSR for debug purposes or during initialization to seed the LFSR. SW can write into the register for debug purposes.

Field name	rwu	Bit #	Reset	Description
Reserved	ru	31:7	25'b0	Reserved

7.11.23 Debug PHY Register (debugPHYReg)

Offset Address: 0x055C

Field name	rwu	Bit #	Reset	Description
rxReqForceDeassertion	rw	0	1'b0	rxReq Force Deassertion Force the de-assertion of the rxReq after each rxEnd_p
rxEndForTimingErrRec	rw	1	1'b0	rxEndForTiming Error Recovery When set, guaranty the reception of rxEndForTiming_p before rxEnd_p at the MAC-PHY interface whatever the PHY behavior is.
Reserved	ru	31:2	30'b0	Reserved

7.11.24 Software Profiling Register (swProfilingReg)

Offset Address: 0x8560 (*swProfilingReg*), 0x8564 (*swSetProfilingReg*) and 0x8568 (*swClearProfilingReg*)

This register allows the software profiling by providing its content to the debug port. The software can use different methods to access this register which provides flexibility during the profiling. At the addess 0x8560, the software can read the content of the *swProfilingReg* and it can write it. Each bit is also set by HW when SW performs a write with a value "1" to the corresponding bit in the *swSetProfilingReg* address. Each bit is reset by HW when SW performs a write with a value "1" to the corresponding bit in the *swClearProfilingReg* address.

Field name	rwu	Bit #	Reset	Description
swProf	rwscu	31:0	32'b0	Software Profiling value The software can write, read, set or clear each bit of this field and get the swProf Value on the debug port.

8 Integration guide

8.1 Interface signal definition

Signal	I/O	Description
Interrupt and Reset Signals		
intGen_n	O	Active low general interrupt signal. This line is triggered when any bit in the *General Interrupt Event Register (genIntEventReg)* is set, and cleared when none of the bits in the register are set.
intProtTrigger_n	O	Active low Protocol Trigger interrupt signal. This line is triggered when some of the bits in the *Transmit / Receive Interrupt Event Register (txRxIntEventReg)* are set, and cleared when those bits in the register are reset.
intTxTrigger_n	O	Active low Transmission Trigger interrupt signal. This line is triggered when some of the bits in the *Transmit / Receive Interrupt Event Register (txRxIntEventReg)* are set, and cleared when those bits in the register are reset.
intRxTrigger_n	O	Active low Receive Trigger interrupt signal. This line is triggered when some of the bits in the *Transmit / Receive Interrupt Event Register (txRxIntEventReg)* are set, and cleared when those bits in the register are reset.
intTxRxMisc_n	O	Active low Miscellaneous Transmit/Receive interrupt signal. This line is triggered when some of the bits in the *Transmit / Receive Interrupt Event Register (txRxIntEventReg)* are set, and cleared when those bits in the register are reset.
intTxRxTimer_n	O	Active low Transmit/Receive Timer interrupt signal. This line is triggered when some of the bits in the *Transmit / Receive Interrupt Event Register (txRxIntEventReg)* are set, and cleared when those bits in the register are reset.
macPIClkHardRst_n	I	Active low hard reset signal synchronized to the *macPIClk*. This reset line is used for the FFs in the following clock domains: *macPIClk*, *macPI1Clk*, *macPI2Clk*.
macPISlaveClkHardRst_n	I	Active low hard reset signal synchronized to the *macPISlaveClk*. This reset line is used for the FFs in the following clock domains: *macPISlaveClk*.
macCoreClkHardRst_n	I	Active low hard reset signal synchronized to the *macCoreClk.*This reset line is used for the FFs in the following clock domains: *macCoreClk*, *macCoreTxClk*, *macCoreRxClk, macCryptClk*.
macWTClkHardRst_n	I	Active low hard reset signal synchronized to the *macWTClk.* This reset line is used for the FFs in the *macWTClk* domain.
mpIFClkHardRst_n	I	Active low hard reset signal synchronized to the *mpIFClk.* This reset line is used for the FFs in the *mpIFClk* domain.
Clock Signals		
macPriClkEn	O	Enable MAC Primary Clocks This signal indicates to the platform clock controller that 3 primary MAC clocks (*macPIrClk*, *macCoreClk*, *mpIFClk*) should be turned on.

Signal	I/O	Description
platformWakeUp	O	Enable Platform Clock This signal indicates to the platform clock controller that the platform clocks should be turned on.
macPIClk	I	Primary MAC Platform Interface Clock This clock times all AHB Master port transfers as well as the DMA Engine. All signal timings are related to the rising edge of *macPIClk*. This clock is turned on when the *macPriClkEn* or *platformWakeUp* is asserted or when programmed from SW.
macPISlaveClk	I	Primary MAC Platform Interface Slave Clock This clock times all AHB Slave port. All signal timings are related to the rising edge of *macPISlaveClk*.
macPITxClk	I	Secondary MAC Platform Interface Tx Clock.
macPITxClkEn	O	Enable signal for the *macPITxClk*.
macPIRxClk	I	Secondary MAC Platform Interface Rx Clock.
macPIRxClkEn	O	Enable signal for the *macPIRxClk*.
macCoreClk	I	Primary MAC Core clock This is the primary MAC Core clock and drives the register block and some other MAC HW blocks. This clock is turned on when the *macPriClkEn* is asserted or when programmed from SW.
macCoreTxClk	I	Secondary MAC Core Tx clock This is one of the clocks to the MAC HW blocks.
macCoreTxClkEn	I	Enable signal for the *macCoreTxClk*.
macCoreRxClk	I	Secondary MAC Core Rx clock. This is one of the clocks to the MAC HW blocks.
macCoreRxClkEn	O	Enable signal for the *macCoreRxClk*.
macCryptClk	I	Secondary MAC Core Crypto clock. This is one of the clocks to the MAC HW blocks.
macCryptClkEn	O	Enable signal for the *macCryptClk*.
macLPClk	I	MAC Low Power clock.
macLPClkSwitch	I	Switch MAC Lower Clock. When this signal is asserted by the MAC HW, the *macLPClk* is switched from a high frequency clock to a 32 kHz LowPower clock.
macWTClk	I	The WEP/TKIP clock is a faster clock input for WEP/TKIP blocks. This clock should be synchronized to the *macCoreClk* and may be twice or thrice the *macCoreClk*.
macWTClkEn	O	Enable signal for the *macWTClk*.
mpIFClk	I	MAC PHY Interface Clock The data is generated and sampled by the PHY on this clock. It is given to the MAC-PHY IF block in the MAC.
mpIFClk En	O	Enable signal for the *mpIFClk*.

Signal	I/O	Description
Debug Interface		
internalError	O	Internal Error This signal is generated by the MAC HW based on any internal error condition as required and can be used to trigger a Logic Analyzer or to gate the MAC and PHY clocks for better observability of the internal states.
debugPort [31:0]	O	Debug Port A set of 32 signals used for debug. Various set of internal signals are muxed onto these lines based on the setting in the Debug Port Select Register (debugPortSelReg) for observing on a Logic Analyzer.
debugKSR	I	Debug Key Storage RAM. This signal is asserted to allow read-back of the embedded Key Storage RAM. On a production system, the pin is expected to be tied to LOW, thus preventing malicious access to the security keys.
ARM AHB/AHBLite Slave Interface		
hSAddr [15:0]	I	Lower 16 bits of the 32-bit system address bus.
hSTrans [1:0]	I	Transfer type of the current transfer, which can be NONSEQUENTIAL, SEQUENTIAL, IDLE or BUSY.
hSWrite	I	Transfer direction. When HIGH this signal indicates a write transfer and when LOW a read transfer.
hSSize [2:0]	I	Transfer size. Indicates the size of the transfer, which is typically byte (8-bit), halfword (16-bit) or word (32-bit). The protocol allows for larger transfer sizes up to a maximum of 1024 bits.
hSBurst [2:0]	I	Burst type. Indicates if the transfer forms part of a burst. Four, eight and sixteen beat bursts are supported and the burst may be either incrementing or wrapping.
hSProt [3:0]	I	Protection control. Not used.
hSWData [31:0]	I	Write data bus. The write data bus is used to transfer data from the master to the bus slaves during write operations.
hSSel	I	Slave select. This signal is asserted from the decoder to indicate that the current transfer is meant for a particular slave.
hSReadyIn	I	Transfer Done. When HIGH the *hSReadyIn* signal indicates that a transfer has finished on the bus. This signal signals the status of other slaves on the AHB.
hSRData [31:0]	O	Read data bus. The read data bus is used to transfer data from bus slaves to the bus master during read operations.
hSReadyOut	O	Transfer done. When HIGH the *hSReadyIn* signal indicates that a transfer has finished on the bus. This signal may be driven LOW to extend a transfer. Note: Slaves on the bus require hReady as both an input and an output signal.
hSResp [1:0]	O	Transfer response. The transfer response provides additional information on the status of a transfer. Two different responses are provided, OKAY and ERROR. RETRY and SPLIT are NA in an AHBLite system.

Signal	I/O	Description
hMaster [2:0]	I	NA in an AHBLite system.
hSplitX [5:0]	O	NA in an AHBLite system.
hMastLock	I	Locked sequence. Indicates that the current master is performing a locked sequence of transfers. This signal has the same timing as the *hMaster* signal.
ARM AHB/AHBLite Master Interface		
hMAddr [31:0]	O	The 32-bit system address bus.
hMTrans [1:0]	O	Transfer type of the current transfer, which can be NONSEQUENTIAL, SEQUENTIAL, IDLE or BUSY.
hMWrite	O	Transfer direction. When HIGH this signal indicates a write transfer and when LOW a read transfer.
hMSize [2:0]	O	Transfer size. Indicates the size of the transfer, which is typically byte (8-bit), halfword (16-bit) or word (32-bit). The protocol allows for larger transfer sizes up to a maximum of 1024 bits.
hMBurst [2:0]	O	Burst type. Indicates if the transfer forms part of a burst. Four, eight and sixteen beat bursts are supported and the burst may be either incrementing or wrapping.
hMProt [3:0]	O	Protection control. Fixed to non-cache data.
hMWData [31:0]	O	Write data bus. The write data bus is used to transfer data from the master to the bus slaves during write operations.
hMRData [31:0]	I	Read data bus. The read data bus is used to transfer data from bus slaves to the bus master during read operations.
hMReady	I	Transfer done. When HIGH the *hMReady* signal indicates that a transfer has finished on the bus. This signal may be driven LOW to extend a transfer.
hMResp [1:0]	I	Transfer response. The transfer response provides additional information on the status of a transfer. Two different responses are provided, OKAY and ERROR. RETRY and SPLIT are NA in an AHBLite system.
hBusReq	O	NA in an AHBLite system
hLock	O	Locked transfers. When HIGH this signal indicates that the master requires locked access to the bus and no other master should be granted the bus until this signal is LOW.
hGrant	I	NA in an AHBLite system
MAC-PHY Interface for RivieraWaves's 802.11 Baseband		
txReq	O	Transmit Request This signal is asserted by the MAC at the start of a transmission cycle, when it starts transmitting the TxVector to the PHY. It is de-asserted by the MAC one clock after the *txEnd_p* signal is generated by the PHY, thus ending the transmission cycle. If the *txReq* is de-asserted while the PHY is in a transmit cycle, it serves to abort the current transmission. The MAC waits for *txEnd_p* to be asserted before asserting *txReq* or *rxReq* again.

Signal	I/O	Description
txData[7:0] / mimoCmd[7:0]	O	Transmit Data / MIMO Command During transmission: This data bus carries the TxVector and the data to be transmitted. The TxVector is changed every *mpIFClk* clock by the MAC and *phyRdy* is not required. During data transmission, the MAC puts a new data byte on the bus if the *phyRdy* signal is asserted and it has data available to transmit. The MAC indicates that it has put valid data on the *txData* bus by asserting *macDataValid*. During idle (no transmission or reception cycle is ongoing): This data bus carries the MIMO Command. The MIMO Command is valid only if the *mimoCmdValid* signal is asserted.
macDataValid	O	MAC Data Valid This signal is asserted when the MAC puts a data byte on the *txData* lines.
mimoCmdValid	O	MIMO Command Valid This signal is used to indicate that *mimoCmd* bus (multiplexed TxData bus) contains a valid MIMO Command.
phyRdy	I	PHY Ready During Transmission: This signal is asserted when the PHY is ready to receive data from the MAC on the *txData* lines during a transmission cycle. The MAC puts new data on the bus every *mpIFClk* tick, if *phyRdy* signal is asserted and it has data available to transmit. During Reception: This signal is asserted when the PHY puts valid data on the *rxData* lines during a receive cycle. The MAC must sample this data when it detects *phyRdy* asserted. The *rxData* may change every clock. The number of *mpIFClks* the *phyRdy* is asserted is equal to the number of *rxData* bytes available at PHY.
txEnd_p	I	Transmit End Pulse The modem asserts this signal to indicate end of transmit processing, and helps the MAC synchronize with the PHY and for transmit timing reference. It is the last signal generated by the PHY in a transmit cycle under all circumstances. In case of a normal transmission without error, this signal is generated while the modem outputs the last sample of the last data symbol. In case of an abnormal transmission (transmit abort from MAC, or transmit error detected by PHY), this signal is generated by the PHY when it has completed any transmit processing or internal state cleanup. The PHY detects an error in transmission when: ✓ If it finds an incorrect TxVector parameter ✓ If the *phyRdy* signal has been asserted but the MAC has not passed it adequate number of data bytes, thus leading to an underrun during the transmission.

Signal	I/O	Description
rxReq	O	Receive Request MAC asserts this signal to put the PHY in receive state. The MAC de-asserts this signal after the PHY signals *rxEnd*_p, indicating the last byte of the current packet on the *rxData* bus. If the MAC de-asserts *rxReq* while the PHY is actively receiving a PPDU, it is treated as Rx Abort. The MAC waits for the *rxEnd_p* signal from the PHY as an acknowledgement that the PHY has finished processing the abort condition.
rxData[7:0]	I	Receive Data This data bus carries the RxVector and the PSDU received by the PHY. The PHY indicates that it has put valid data on the rxData bus by asserting *phyRdy*.
CCAPrimary20	I	Clear Channel Assessment for the primary 20MHz This signal is asserted when the PHY senses that the medium is busy in the primary 20 MHz band. This signal is also asserted when the PHY senses that the medium is busy when operating in a single 20 MHz band.
CCASecondary20	I	Clear Channel Assessment for the secondary 20MHz This signal is asserted when the PHY senses that the medium is busy in the secondary 20 MHz band. This signal is used in 40MHz, 80MHz, 80+80MHz and 160MHz mode. In 20MHz, this signal is forced low.
CCASecondary40	I	Clear Channel Assessment for the secondary 40MHz This signal is asserted when the PHY senses that the medium is busy in the secondary 40 MHz band. This signal is used in 80MHz, 80+80MHz and 160MHz mode. In 20MHz and 40MHz, this signal is forced low.
rxEndForTiming_p	I	Receive End for Timing Pulse This signal is asserted when the modem has received the last sample of the last data symbol from the RF. It is asserted for one *mpIFClk* period and is used by the MAC for receive timing reference.
rxErr_p	I	Receive Error Pulse If an error (FormatViolation, UnsupportedRate, CarrierLost) occurs during reception, the PHY indicates this to MAC by asserting the *rxErr*_p signal. The MAC discards any data bytes given by the PHY until *rxEnd_p* is asserted. The MAC de-asserts *rxReq* after the PHY has signalled *rxEnd*_p.

Signal	I/O	Description
rxEnd_p	I	Receive End Pulse The PHY asserts this signal to indicate end of receive processing, and helps the MAC synchronize with the PHY. It is the last signal generated by the PHY in a receive cycle under all circumstances. In case of a normal reception without error, this signal is generated while putting the last data byte on the *rxData* bus. In case of an abnormal reception (receive abort from MAC, or error during receive detected by PHY), this signal is generated by the PHY when it has completed any receive processing or internal state cleanup. Note that when the MAC aborts the current reception this signal is asserted after the *rxReq* is de-asserted. Note that when receive error is detected by the PHY, this signal is asserted after *rxErr_p* has been asserted.
keepRFOn	O	Keep RF On During RX: The MAC asserts this signal to indicate that the PHY should remain in RX state even after reception of the current packet. The PHY starts its search for packet next packet after the current packet is completely received on the air. This is to enable the PHY to receive packets arriving within RIFS.
phyErr_p	I	PHY Error Pulse The PHY may assert this signal due to various reasons, like incorrect data passed through Tx Vector, transmit underrun detected by the PHY, incorrect command passed through the *mimoCmd* lines. This signal may also be asserted in scenarios where the PHY state machines are hung.
rifsRxDetected	I	RIFS Reception Detected The PHY asserts this signal when it detects a RIFS separated PPDU from the last received PPDU. This signal is not directly used in the MAC HW, but is used for statistics and debugging purposes.
Bluetooth Coexistence Interface (Optional see *2.5,Coexistence support*)		
coexWlanTx	O	Coexistence WLAN Tx Indicates a on-going transmission of the WLAN system
coexWlanRx	O	Coexistence WLAN Rx Indicates a on-going reception of the WLAN system
coexWlanTxAbort	I	Coexistence WLAN Tx Abort Request the WLAN to immediately abort the transmission and prevent the MAC to start new transmission
coexWlanRxAbort	I	Coexistence WLAN Rx Abort Request the WLAN to immediately abort the reception
coexWlanPTI[3:0]	O	Coexistence WLAN Packet Traffic Information Indicate the priority of the on-going transmission or reception.
coexWlanPTIToggle	O	Coexistence WLAN Packet Traffic Information Toggle indication This signal toggles each time the coexWlanPTI value is updated. This signal shall be used by the external PTA for resynchronization in case it is not on the same clock domain than macCoreClk.

Signal	I/O	Description
coexWlanBW	O	Coexistence WLAN bandwidth Indicate the bandwidth (0:20MHz or 1:40MHz) of the on-going transmission or reception.
coexWlanChanFreq[6:0]	O	Coexistence WLAN channel Indicate the current center frequency in MHz of the primary channel starting from 2400MHz.
coexWlanChanOffset	O	Coexistence WLAN channel offset Indicate where the primary channel is in the 40MHz band. When set, the primary channel is on the upper 20MHz of the 40MHz channel. When reset, the primary channel is on the lower 20MHz of the 40MHz channel. If the bandwidth is 20MHz, it is forced to 0.
BeamForming Report Interface (Optional see *2.6.1 Beamformee support*)		
bfrData[7:0]	I	BeamForming Report Byte
bfrDataValid	I	BeamForming Report Byte Valid Indication from the beamforming HW accelerator that the bfrData is valid and shall be captured.
bfrDataReady	O	BeamForming Report Byte Ready Indicate to the beamforming HW accelerator that the TX Controller is ready to accept new byte.
bfrStart	O	Beamforming Report Start Indicates to the beamforming HW accelerator that the SVD and compression shall be launched.
bfrDone	I	Beamforming Report Done Indication from the beamforming HW accelerator that the SVD and compression processing is completed.
bfrChBW[1:0]	O	Beamforming Report Channel Bandwidth Indicates to the beamforming HW accelerator the bandwidth of the report to be generated.
bfrGrouping[1:0]	O	Beamforming Report Grouping Indicates to the beamforming HW accelerator the grouping used for the compression.
bfrCodebook	O	Beamforming Report Codebook Indicates to the beamforming HW accelerator the codebook used for the compression.
bfrFeedbackType	O	Beamforming Report FeedbackType Indicates to the beamforming HW accelerator the FeedbackType (SU or MU) used for the compression and report generation.
bfrNc[2:0]	O	Beamforming Report Number of columns Indicates to the beamforming HW accelerator the number of columns information extracted from the NDPA.

Signal	I/O	Description
bfrNr[2:0]	O	Beamforming Report Number of rows Indicates to the beamforming HW accelerator the number of rows information extracted from the NDP (NSS of the NPD frame).

Signal	I/O	Description
Top-level Embedded Memories Interface		
TX FIFO		
txFIFOWriteData [37:0]	O	Transmit FIFO and Transmit Tag FIFO concatenated Write Data bus.
txFIFOReadData [37:0]	I	Transmit FIFO and Transmit Tag FIFO concatenated Read Data bus.
txFIFOWriteAddr [7:0]	O	Transmit FIFO and Transmit Tag FIFO Write Address bus.
txFIFOReadAddr [7:0]	O	Transmit FIFO and Transmit Tag FIFO Read Address bus.
txFIFOReadEn	O	Transmit FIFO and Transmit Tag FIFO Read Enable. This signal is asserted for read operations to the Transmit FIFO and the Transmit Tag FIFO.
txFIFOWriteEn	O	Transmit FIFO and Transmit Tag FIFO Write Enable. This signal is asserted for write operations to the Transmit FIFO and the Transmit Tag FIFO.
RX FIFO		
rxFIFOWriteData [35:0]	O	Receive FIFO and Receive Tag FIFO concatenated Write Data bus.
rxFIFOReadData [35:0]	I	Receive FIFO and Receive Tag FIFO concatenated Read Data bus.
rxFIFOWriteAddr [7:0]	O	Receive FIFO and Receive Tag FIFO Write Address bus.
rxFIFOReadAddr [7:0]	O	Receive FIFO and Receive Tag FIFO Read Address bus.
rxFIFOReadEn	O	Receive FIFO and Receive Tag FIFO Read Enable. This signal is asserted for read operations to the Receive FIFO and the Receive Tag FIFO.
rxFIFOWriteEn	O	Receive FIFO and Receive Tag FIFO Write Enable. This signal is asserted for write operations to the Receive FIFO and the Receive Tag FIFO.
Key Storage Memory		
keyStorageWriteData [187:0] or [315:0] [15]	O	Key Storage RAM Write Data Key Storage RAM write data bus.
keyStorageReadData [187:0] or [315:0] [16]	I	Key Storage RAM Read Data Key Storage RAM read data bus.

[15] Depend if WAPI or CCMP/GCMP 256-bit is supported or not.

[16] Depend if WAPI or CCMP/GCMP 256-bit is supported or not.

Signal	I/O	Description
keyStorageAddr [`RW_KEY_INDEX_WIDTH-1:0]	O	Key Storage RAM Address Key Storage RAM address bus.
keyStorageEn	O	Key Storage RAM Enable This signal is asserted for both read and write operations to the Key Storage RAM.
keyStorageWriteEn	O	Key Storage RAM Write Enable This signal is asserted for write operations to the Key Storage RAM.
SBOX memory		
sbox_mem_rd_data[15:0]	I	SBox Read Data
sbox_mem_wr_data [15:0]	O	SBox Write Data
sbox_mem_wr_addr [6:0]	O	SBox Read Address
sbox_mem_rd_addr [6:0]	O	SBox Write Address
sbox_mem_wr_en [1:0]	O	SBox Write Byte Enable
sbox_mem_rd_en	O	SBox Read Enable
MAC-PHY RX FIFO		
mpIFTxFIFOWriteData [7:0]	O	MAC-PHY InterFace Transmit FIFO Write Data bus.
mpIFTxFIFOReadData [7:0]	I	MAC-PHY InterFace Transmit FIFO Read Data bus.
mpIFTxFIFOWriteAddr [6:0]	O	MAC-PHY InterFace Transmit FIFO Write Address bus.
mpIFTxFIFOReadAddr [6:0]	O	MAC-PHY InterFace Transmit FIFO Read Address bus.
mpIFTxFIFOReadEn	O	MAC-PHY InterFace Transmit FIFO Read Enable. This signal is asserted for read operations to the MAC-PHY interface Transmit FIFO.
mpIFTxFIFOWriteEn	O	MAC-PHY InterFace Transmit FIFO Write Enable. This signal is asserted for write operations to the MAC-PHY interface Transmit FIFO.
MAC-PHY RX FIFO		
mpIFRxFIFOWriteData [7:0]	O	MAC-PHY InterFace Receive FIFO Write Data bus.
mpIFRxFIFOReadData [7:0]	I	MAC-PHY InterFace Receive FIFO Read Data bus.
mpIFRxFIFOWriteAddr [6:0]	O	MAC-PHY InterFace Receive FIFO Write Address bus.
mpIFRxFIFOReadAddr [6:0]	O	MAC-PHY InterFace Receive FIFO Read Address bus.
mpIFRxFIFOReadEn	O	MAC-PHY InterFace Receive FIFO Read Enable. This signal is asserted for read operations to the MAC-PHY interface Receive FIFO.
mpIFRxFIFOWriteEn	O	MAC-PHY InterFace Receive FIFO Write Enable. This signal is asserted for write operations to the MAC-PHY interface Receive FIFO.

Signal	I/O	Description
MIB Memory		
mibTableWriteData [31:0]	O	MIB Table Write Data bus.
mibTableReadData [31:0]	I	MIB Table Read Data bus.
mibTableAddr [7:0]	O	MIB Table Address bus.
mibTableEn	O	MIB Table Enable. This signal is asserted for both read and write operations to the MIB Table.
mibTableWriteEn	O	MIB Table Write Enable. This signal is asserted for write operations to the MIB Table.

Table 15: I/O Signal Description

Note: I = Input port, O = Output port, I/O = Bi-directional port.

8.2 Clock domains

8.2.1 Primary clock domains

Primary clock domains (except *macLPClk*) are controlled globally by a single controlling signal. They are mandatorily turned off/on in tandem by the MAC HW core, and are turned OFF when in the DOZE state (refer section *2.1*, *MAC core states*) and are turned ON when in any other state. Note that the *macLPClk* is never turned off.

Sl No.	Clock Name	Clock Frequency	Source	Remark
1.	macPIClk	70 – 300 MHz	External	MAC Platform Interface Clock - Primary This clock times all AHB Master port transfers as well as the DMA Engine. It is allowed to be asynchronous to the *macCoreClk*. This clock domain supports dynamic frequency scaling. The MAC Core supports the following clock relations: *macPIClk = macCoreClk (MUST be aligned in this case)* *macPIClk > macCoreClk*

Sl No.	Clock Name	Clock Frequency	Source	Remark
2.	macPISlaveClk	70 – 300 MHz	External	MAC Platform Interface Slave Clock They should be: 1. derived from the same crystal from which *macPIClk* is derived 2. of the same frequency as *macPIClk* 3. synchronized and phase matched with *macPIClk*. There are no synchronizers on paths between *macPIClk* and *macPISlaveClk* domains. This clock times all AHB Slave port transfers. It is allowed to be asynchronous to the *macCoreClk*. This clock domain supports dynamic frequency scaling. The MAC Core supports the following clock relations: *macPISlaveClk = macCoreClk (MUST be aligned in this case)* *macPISlaveClk > macCoreClk*
3.	macCoreClk	20 – 255 MHz	External	MAC Core Clock - Primary This is the primary MAC core clock and drives the register block and some other MAC HW blocks. It is allowed to be asynchronous to the *macPIClk*. And *macPISlaveClk*.
4.	macLPClk	Same as *macCoreClk* OR 32 kHz LowPower Clock	External	MAC Low Power Clock. This clock line supplies the same frequency as the *macCoreClk* in ACTIVE mode and a low frequency in DOZE mode. It is supplied to the MAC HW blocks that continue to be active in the power save mode. In ACTIVE mode, it should be: 1. derived from the same crystal from which *macCoreClk* is derived 2. of the same frequency as *macCoreClk* 3. synchronized and phase matched with *macCoreClk*. There are no synchronizers on paths between *macCoreClk* and *macLPClk* domains. In DOZE mode, it should be a LowPower clock running at either 32kHz or 32.768kHz. The register *MAC Control 1 Register (macCntrl1Reg).lpClk32786Hz* shall be used to indicate to the MAC the frequency of the LowPower clock.

Sl No.	Clock Name	Clock Frequency	Source	Remark
5.	mpIFClk	30 – 180 MHz	External	MAC-PHY IF Clock The interface signals and data is transmitted and received to/from the PHY on this clock. It is allowed to be asynchronous to the *macCoreClk*.

Table 16: Primary clock domains

8.2.2 Secondary clock domains

Secondary clock domains are individually controlled. They are mandatorily turned off by the MAC HW core when in the DOZE state (refer section *2.1*, *MAC core states*) and are separately turned ON/OFF when required in any other state.

Sl No.	Clock Name	Clock Frequency	Source	Remark
1.	macWTClk	Same, twice or thrice *macCoreClk*	External	WEP/TKIP Clock It is a faster clock input for WEP/TKIP blocks. This clock may be same, twice or thrice the frequency of the *macCoreClk*. It should be: 1. derived from the same crystal from which *macCoreClk* is derived 2. synchronized and phase matched with macCoreClk. There are no synchronizers on paths between *macCoreClk* and *macWTClk* domains.
2.	macPITxClk	Same as *macPIClk*	External	Secondary MAC Platform Interface clocks for active clock gating. These clocks drive different MAC HW blocks, like the Transmit and receive DMA engines. They should be: 1. derived from the same crystal from which *macPIClk* is derived 2. of the same frequency as *macPIClk* 3. synchronized and phase matched with *macPIClk*. There are no synchronizers on paths between *macPIClk* and *macPI1/2Clk* domains. This clock domain supports dynamic frequency scaling. The MAC Core supports the following clock relations: *macPI1/2Clk = macCoreClk* *macPI1/2Clk > macCoreClk*
3.	macPIRxClk			

Sl No.	Clock Name	Clock Frequency	Source	Remark
4.	macCoreTxClk	Same as *macCoreClk*	External	Secondary MAC Core clocks for active clock gating. These clocks drive different MAC HW blocks. They should be: 1. derived from the same crystal from which *macCoreClk* is derived 2. of the same frequency as *macCoreClk* 3. synchronized and phase matched with macCoreClk. There are no synchronizers on paths between *macCoreClk* and *macCoreTx*/Rx*Clk &macCryptClk* domains.
5.	macCoreRxClk			
6.	macCryptClk			

Table 17: Secondary clock domains

8.2.3 Clock domains and MAC HW functional blocks

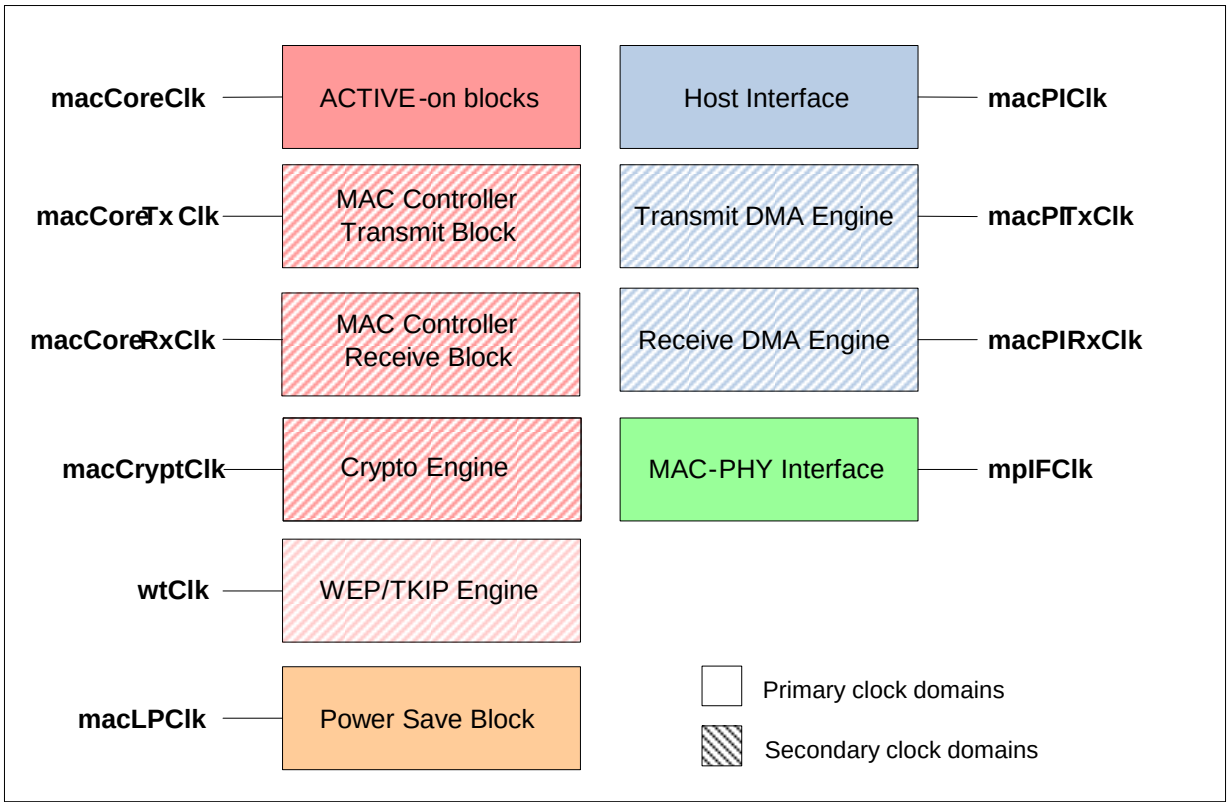


Figure 25: Clock inputs to MAC HW modules

8.3 Memory requirements

Memory Name	Config	Size	Number of instances	Depth	Read Port Width	Write Port Width	Read Port Freq	Write Port Freq
TX FIFO	Two port	304 bytes	Number of MU_MIMO Tx User	64	38 bit	38 bit	macCoreTxClk	macPITxClk
RX FIFO	Two port	288 bytes	1	64	36 bit	36 bit	macPIRxClk	macCoreRxClk
KEY Storage RAM	Single port	≈ 1.5 KB	1	2**KEY_INDEX_WIDTH	188/316[17] bit	188/316[18] bit	macCoreClk	macCoreClk
RC4 PRNG	Two port	256 bytes	1	128	16 bit	16 bit	macWTClk	macWTClk
MAC-PHY IF TX FIFO	Two port	128 bytes	Number of MU_MIMO Tx User	128	8 bit	8 bit	macCoreTxClk	mpIFClk
MAC-PHY IF RX FIFO	Two port	128 bytes	1	128	8 bit	8 bit	mpIFClk	macCoreRxClk
MIB table	One port	1024 bytes	1	256	32	32	macCoreClk	macCoreClk

Table 18: Memory Requirements

Refer to Section *8.2, Clock domains* for a detailed explanation on each clock.

[17] Depend if WAPI or CCMP/GCMP 256-bit is supported or not.

[18] Depend if WAPI or CCMP/GCMP 256-bit is supported or not.

8.4 Reset strategy

The MAC core has two types of resets: Multiple external hard resets and multiple internally generated soft reset. The external hard reset is asynchronous, and is a negative edge trigged reset. The soft reset is generated internally by a register write by the software. This reset is synchronous to all the clock domains.

SL No.	Reset Name	Type
1.	macPIClkHardRst_n	External Active Low Hard reset which should be applied for at-least TBD clock cycle duration of *macPIClk*.
2.	macCoreClkHardRst_n	External Active Low Hard reset which should be applied for at-least TBD clock cycle duration of *macCoreClk*.
3.	macWTClkHardRst_n	External Active Low Hard reset which should be applied for at-least TBD clock cycle duration of *macWTClk*.
4.	mpIFClkHardRst_n	External Active Low Hard reset which should be applied for at-least TBD clock cycle duration of *mpIFClk*.
5.	softRstMACPIClk	Active high soft reset, generated internally, synchronized to the *macPIClk*.
6.	softRstMACCoreClk	Active high soft reset, generated internally, synchronized to the *macCoreClk*.
7.	softRstWTClk	Active high soft reset, generated internally, synchronized to the *macWTClk*.
8.	softRstMPIFClk	Active high soft reset, generated internally, synchronized to the *mpIFClk*.

Table 19: Reset Strategy

8.5 Configuration

The MAC HW can be configured depending on the targeted application and the clock frequency.

The following defines are used to configure the RTL as expected.

Must be defined:

MAC_FREQ <value> : Configure the macCore Clock frequency. This is used to configure all the internal timers. The value is given in MHz

Depending on macCoreClk frequency, the WEP/TKIP clock frequency shall be defined.This clock frequency must be at least 60MHz, and an equal or a multiple of macCoreClk.

If the WTClk is equal to macCoreClk, **WEP_2_BB_CLK_RATIO** must be defined to 1.

If the WTClk is a multiple (n) of macCoreClk, **WEP_2_BB_CLK_RATIO** must be defined to (n).

The number of index available in the keySearchRAM is configurable using the define **RW_KEY_INDEX_MAX**

Depending on the number of spatial stream supported, the following defines can be defined.

RW_MAC_2SS_SUPPORT enables the support of 2 SS

RW_MAC_3SS_SUPPORT enables the support of 3 SS

RW_MAC_4SS_SUPPORT enables the support of 4 SS

To enable the support of WAPI, the following define can be set.

RW_WAPI_EN: Enable the support of WAPI in HW.

To enable the support of GCMP, the following define can be set.

RW_GCMP_EN: Enable the support of WAPI in HW.

To configure the Key Ram data width,

RW_KEY_EXTENDED: Enable the 316-bit Key RAM datawidth. Otherwise, it is only 188 bits.

To enable the support of Bluetooth coexistence (see *2.5,Coexistence support*), the following define can be set.

RW_WLAN_COEX_EN: Enable the blueooth coexistence mechanism.

To enable the support of Beamforming as a beamformee (see *2.6.1 Beamformee support*), the following define can be set.

RW_BFMEE _EN: Enable the beamformee feature.

The size of PS Bitmap memory of BA Controller can be configured using **RW_PSBITMAP_SIZE** (defined to 4 by default).

8.6 Debug and test features

Several internal state machines are observable by software through the *Debug HW State Machine 2 Register (debugHWSM2Reg)*. The Debug 2 register has the present state values of the 3 important state machines in the MAC Controller: macCntlrCs, txControllerCs, and rxControllerCs. This register when read at the CLI can be used for diagnostics and monitoring of the internal operations of the core. The state explanation can be found out from the hardware code for the corresponding module.

In addition to this, various internal variables are exposed for debugging purposes (observability). These include the NAV and contention window values, the retry counters, the DMA status and current pointers.

Most frames are “touched” by the HW when transmitting. This can be turned off by setting various bits in the *MAC Control Information 2* field of the *Transmit DMA Header Descriptor*. This is explained in section *2.2.4.2.8*, *Updating MAC Header fields in HW when transmitting*. This allows SW to transmit any value in MAC Header fields without any modification from HW, for debug purposes. In addition, SW can bypass the MAC Header generation logic in HW by setting the *dontGenerateMH* field in the *MAC Control Information 2* field of the *Transmit DMA Header Descriptor*. This allows SW to transmit any MAC Header without any modification from HW, for debug purposes.

During reception, the SW can receive a frame “as-is” from air without any modification by HW. HW can also be put in various promiscuous modes, in order to pass to SW any frame from air. This can be turned on by setting various bits in the *Receive Control Register (rxCntrlReg)*.

Various frame transfer statistics are collected by the MAC HW. These statistics are extensively used for debugging.

There is a provision to tap sets of MAC hardware internal signals (see *[6]*) on Test Points (for LA debugging). The signals are selected using *Debug Port Select Register (debugPortSelReg)*. The value of the signals can be read on the *Debug Port Value Register (debugPortValueReg)*. Additionally, the software can set some flags by writing into *Software Profiling Register (swProfilingReg)* and these flags will be available on the debug Port. That eases the software profiling.

Any location of the Key Storage RAM can be read back from SW as long as the *debugKSR* input is set. Otherwise, the keys are not read back.

9 Error detection and recovery mechanisms

9.1 Protocol Errors

Various protocol errors during transmission and reception are detected by the MAC core, gracefully handled and indicated in the Header Descriptor status fields and updated in statistic counters:

1. Transmission errors like Retry Limit Reached, Lifetime Reached
2. Reception errors like PHY error, FCS error, decryption error.

9.2 Non-terminal hardware errors

The following non-terminal hardware errors are detected by the MAC core and gracefully handled, with indications to SW:

1. Out of Receive DMA channel descriptors: If the Receive DMA engine finds that the *nextHDPRx* in the current Header Descriptor is null, indicating no more Header Descriptors are queued or the *nextPBufDPRx* in the current Buffer Descriptor is null, indicating no more Buffer Descriptors are queued, the *General Interrupt Event Register (genIntEventReg).rxDMAEmpty* interrupt is raised to SW.
2. Data overflow during reception:
    a. Receive FIFO overflow: If the Receive FIFO overflows, the *General Interrupt Event Register (genIntEventReg).rxFIFOOverflow* interrupt is raised to SW. The *rwRdFIFOOverflowCount* counter in the MIB Table is incremented. Any required acknowledgement frame is not transmitted for a MPDU which overflows the Receive FIFO, if the MPDU should be passed to SW.
    b. Receive MPIF FIFO overflow: If the Receive MPIF FIFO overflows, the *General Interrupt Event Register (genIntEventReg).macPHYIFOverFlow* interrupt is raised to SW. The *rwRxMPIFOverflowCount* counter in the MIB Table is incremented. The PPDU which caused the overflow is discarded.
3. Data underrun during transmission: If an underrun is detected, the MAC stops its processing and waits for the PHY to indicate that the processing of the PPDU is complete. The *General Interrupt Event Register (genIntEventReg).macPHYIFUnderRun* interrupt is raised to SW. The *rwTxUnderrunCount* counter in the MIB Table is incremented. Subsequently, if NAV coverage exists, the MAC core retransmits the current PPDU without treating it as a protocol transmission error. A data underrun during transmission may be detected by the MAC or the PHY, as follows:
    a. If the PHY performs rate control at the MAC-PHY interface, i.e. pulls data no faster than the data rate on air, then the MAC detects an underrun if it fails to supply data on the MAC PHY IF when requested by the PHY. This mode is indicated to the MAC by setting the *MAC Control 1 Register (macCntrl1Reg).rateControlledMPIF*.
    b. If the PHY does not performs rate control at the MAC-PHY interface, i.e. pulls data faster than the data rate on air, then the MAC cannot detect an underrun. In this case, the *phyErr_p* signal during a transmission cycle indicated underrun.

9.3 Terminal DMA errors related to data movement

DMA errors related to data movement during transmission and reception are detected by the MAC core. The HW cannot handle these errors and requires SW intervention:

1. Transmit DMA error for any channel.
    a. Length mismatch, i.e. there is less data in the MPDU buffers than the *frameLengthTx* indicated.
    b. The *uPatternTx* fields (of THD or TPD) do not contain the expected pattern.
    c. If the *Next Atomic Frame Exchange Sequence Pointer* or the *Next MPDU Descriptor Pointer* is not valid (i.e. the 2 LSBs of these address fields are not equal to 2'b00).

d. If the Policy Table Address in the DMA Descriptor is not valid (i.e. the address field is null or the 2 LSBs are not equal to 2'b00).
e. There is a bus error while performing any DMA operations.
f. *newHead* bit for a channel is set and the *queueHeadPointer* for that channel does not have a valid address (i.e. the address field is null or the 2 LSBs are not equal to 2'b00).

2. Receive DMA error for any channel:
   a. The remaining space in Rx Buffer does not allow writing the descriptor.
   b. The remaining space in Rx Buffer does not allow writing the payload.
   c. There is a bus error while performing any DMA operations.

In these conditions, the relevant HW DMA channel moves to DEAD state and raises the relevant interrupt: *General Interrupt Event Register (genIntEventReg).ac0/1/2/3TxDMADead* or *General Interrupt Event Register (genIntEventReg).bcnTxDMADead* or *General Interrupt Event Register (genIntEventReg).rxHeaderDMADead* or *General Interrupt Event Register (genIntEventReg).rxPayloadDMADead*.

The exact cause of the error is indicated in *DMA Status 2 Register (dmaStatus2Reg)*.

SW must undertake error recovery when these notifications are given to it by the HW.

9.3.1 Error recovery

Refer to section *9.4.1*, *Error recovery* for details.

9.4 Terminal hardware errors

The HW incorporates sophisticated error detection logic which constantly monitors the internal state machines of the HW as well as the MAC-PHY IF and can detect and indicate these to SW if enabled. This feature is turned on by setting the *MAC Error Recovery Control Register (macErrRecCntrlReg).useErrDet* bit. When an error is detected, the *General Interrupt Event Register (genIntEventReg).hwErr* interrupt is raised to SW. The specific error is indicated in the *MAC Error Status Set Register (macErrStatusSetReg)*. The HW cannot handle these errors and requires SW intervention.

Level 1: Level 1 errors are related to the PHY, and occur when a transmission or reception take longer than expected.

Level 2: Level 2 errors are related to the MAC, and occur when a transmission or reception take longer than expected.

Level 3: Level 3 errors are related to the MAC, and occur when internal processing takes longer than expected.

Note that the HW does NOT stop under these conditions, since there is a possibility of spurious error detection by the HW.

When SW is given a HW Error interrupt, it may choose to immediately determine if an error has occurred, or it may choose to ignore the notification.

Immediately determining if an error has occurred:

1. SW can examine the *Debug HW State Machine 1 Register (debugHWSM1Reg)* for probable HW state machine error patterns. If the register's state machine pattern in this register is a pre-determined error pattern, then the error event might be genuine.
2. SW can monitor the *Debug HW State Machine 2 Register (debugHWSM2Reg)* for a period of time and check if the value is changing on subsequent random reads. If the value is not changing on subsequent reads, then the error might be genuine. In this condition the SW is recommended NOT to read the register using a periodic timer to successfully detect changing states of the HW.

Ignoring the notification:

The SW might choose to ignore the HW Error notification until a certain number of notifications have been received from the HW or a timer expires. In both these cases, SW should perform the above steps to determine if an error has occurred.

If it can be determined with a considerable probability that an error has occurred in HW, the SW should undertake error recovery.

9.4.1 Error recovery

To perform error recovery the following steps must be followed:

1. Put the MAC HW into IDLE state.
2. Wait for a period of time for the HW to return to IDLE state. If the period of time expires without the HW moving to IDLE state, perform Step 3.
3. Reset the HW internal state machines using *MAC Error Recovery Control Register (macErrRecCntrlReg).hwFSMReset*. No registers are reset. None of the embedded memories are reset.
    a. *State Control Register (stateCntrlReg)*
    b. *DMA register definitions*
    c. *MAC Error Status Set Register (macErrStatusSetReg)*
    d. *General Interrupt Event Register (genIntEventReg)*
    e. *Transmit / Receive Interrupt Event Register (txRxIntEventReg)*
    f. *EDCA CCA Busy Time (edcaCCABusyReg)*
    g. *Medium Occupancy Timer 1 Register (mot1Reg*)
    h. *Medium Occupancy Timer 2 Register (mot2Reg)*
4. Once FSM reset is complete, reset the Transmit and Receive FIFOs using *MAC Error Recovery Control Register (macErrRecCntrlReg).txFIFOReset* and *MAC Error Recovery Control Register (macErrRecCntrlReg).rxFIFOReset*.
5. Reset the MAC-PHY Interface FIFO using *MAC Error Recovery Control Register (macErrRecCntrlReg).macPHYIFFIFOReset*.
6. Reset the Partial State Bitmap using *MAC Error Recovery Control Register (macErrRecCntrlReg).baPSBitmapReset*.
7. Resume normal operation by moving the core out of IDLE
8. Reprogram the various DMA registers.

9.5 Reporting and logging errors

It is imperative that SW provide adequate notification on a debug/prototyping system when non-protocol related errors are reported by HW. Such notification can be reported on the output console or logged to facilitate debug. This should be done irrespective of error recovery mechanisms in SW.

References

[1] Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications, IEEE Std 802.11™-2016

[2] Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: Enhancements for Higher Efficiency, IEEE P802.11ax™ - draft 4.0

[3] RW-WLAN-nX MAC PHY Interface Functional Specification (*RW-WLAN-nX-MAC-PHY-IF-FS*)

[4] Interrupt Moderation Using Intel Gigabit Ethernet Controllers (ap450 - Intel - Interrupt Moderation.pdf)

[5] RW-WLAN-nX System Multirate Support App Note (*RW-WLAN-nX-S-MS-AN*)

[6] RW-WLAN-nX MAC HW Diagnostic Port App Note (*RW-WLAN-nX-MAC-HW-DP-AN*)