**HWA Driver**

**Software Design**

**Version 1.0**

**Business Unit: Radar**

**Project Name: mmWave SDK**

***TI Confidential***

Table of Contents

[1. Purpose 4](#_Toc40787996)

[2. Functional Overview 4](#_Toc40787997)

[2.1 HWA 1.0 4](#_Toc40787998)

[2.2 HWA 2.0 4](#_Toc40787999)

[3. Assumptions 5](#_Toc40788000)

[4. Definitions, Abbreviations, Acronyms 5](#_Toc40788001)

[5. References 6](#_Toc40788002)

[6. Design 7](#_Toc40788003)

[6.1 System Architecture 7](#_Toc40788004)

[6.2 Platforms supported 7](#_Toc40788005)

[**6.2.1** HWA 1.0 7](#_Toc40788006)

[**6.2.2** HWA 2.0 7](#_Toc40788007)

[6.3 External Interfaces (API) 8](#_Toc40788008)

[**6.3.1** HWA 1.0 8](#_Toc40788009)

[**6.3.1.1** Initialization 8](#_Toc40788010)

[**6.3.1.2** Open 8](#_Toc40788011)

[**6.3.1.3** Close 8](#_Toc40788012)

[**6.3.1.4** Config Common Registers 8](#_Toc40788013)

[**6.3.1.5** Config Paramsets 9](#_Toc40788014)

[**6.3.1.6** Config RAM 9](#_Toc40788015)

[**6.3.1.7** Enable Paramset done interrupt 9](#_Toc40788016)

[**6.3.1.8** Enable Done interrupt 9](#_Toc40788017)

[**6.3.2** HWA 2.0 9](#_Toc40788018)

[**6.3.2.1** **Initialization** 9](#_Toc40788019)

[**6.3.2.2** Open 10](#_Toc40788020)

[**6.3.2.3** Close 10](#_Toc40788021)

[**6.3.2.4** Config Common Registers 10](#_Toc40788022)

[**6.3.2.5** Config Paramsets 12](#_Toc40788023)

[**6.3.2.6** Config/read/clear RAM 12](#_Toc40788024)

[**6.3.2.7** Read data from common register 12](#_Toc40788025)

[**6.3.2.8** Read/clear clip status register 12](#_Toc40788026)

[**6.3.2.9** Context switch 13](#_Toc40788027)

[**6.3.2.10** Paramset done polling 13](#_Toc40788028)

[**6.3.2.11** Enable Paramset done interrupt 13](#_Toc40788029)

[**6.3.2.12** Enable Done interrupt 13](#_Toc40788030)

[**6.3.2.13** Software reset 14](#_Toc40788031)

[6.4 Error Handling 14](#_Toc40788032)

[6.5 Driver Resources 14](#_Toc40788033)

[**6.5.1** HWA 1.0 14](#_Toc40788034)

[**6.5.2** HWA 2.0 14](#_Toc40788035)

[6.6 Application programming sequence 14](#_Toc40788036)

[7. Standards, Conventions and Procedures 15](#_Toc40788037)

[7.1 Documentation Standards 15](#_Toc40788038)

[7.2 Coding Standards 15](#_Toc40788039)

[7.3 Software development tools 15](#_Toc40788040)

[7.4 Safety Standards 15](#_Toc40788041)

Table of Figures

Figure 1: HWA Driver Design 2

**Revision Control**

|  |  |  |  |
| --- | --- | --- | --- |
| **Author Name** | **Description** | **Version** | **Date** |
| Madhvi Jain | Initial version | 0.5 | 9/20/2016 |
| Xiaozhou Huang | Add hwa 2.0 support | 1.0 | 5/13/2020 |

# Purpose

This document describes the design and implementation considerations for Hardware Accelerator driver software and its features. It is intended for internal (development team mainly) audience.

# Functional Overview

The HWA driver exposes the following hardware functionalities of the HWA IP.

## HWA 1.0

Functionalities present in AR14xx SOC:

1. Enable/Disable the HWA
2. Configure PARAM sets
3. Configure the common registers
4. Enable/Disable interrupt to the R4F and external DMA
5. Read back of status and debug registers
6. Various trigger modes – DFE, DMA, Software, Immediate
7. Configuration of Window RAM and Internal RAM

In addition, following software functionality is exposed:

1. Open/Close Driver APIs
2. Provision for calling application provided callback on receiving interrupts from HWA
3. Provision for obtaining the DMA configuration for DMA based trigger of paramset

## HWA 2.0

Functionalities present in TPR12 SOC

1. Enable/Disable, Reset the HWA
2. Configure PARAM sets
3. Configure the common registers
4. Enable/Disable pamaset done interrupt 1 and interrupt 2 to the R4F, DSP cores and external DMA
5. Enable/disable two hwa done interrupts for two threads
6. Read/clear clip status register
7. Software reset for all 12 DC estimation accumulators, interference stats module, and the param set counter in complex multiplier mode 8, and mode 10.
8. Read the 2D statistics results from data RAM, and histogram, histogram threshold results from corresponding data RAM
9. Various trigger modes – DMA, Software, Immediate
10. Configuration of windows RAM, vector multiply RAM, shuffle RAM and frequency de-rotation RAM
11. Enable/disable context switch.
12. read results from common registers: including CFAR peak detection, DC estimation, interference statistics from common registers
13. Trigger context switch – software and DMA

In addition, following software functionality is exposed:

1. Open/Close Driver APIs
2. Provision for setting the paramset source address
3. Provision for calling application provided callback on receiving interrupts from HWA including two paramset done interrupts, and two thread done interrupts
4. Provision of getting data RAM address API.
5. Provision of polling functions for single and multiple paramsets done.
6. Provision for obtaining the DMA configuration for DMA based trigger of paramset
7. Provision of clear histogram RAM API

# Assumptions

1. The driver doesn’t provide any resource management for PARAM sets and DMA source trigger channels. It is the application responsibility to handle the resource management and making sure configured param sets are not accidentally over-written.
2. Following features are yet to be implemented:

In HWA 1.0

* 1. Error interrupts from HWA
  2. Power down of memories via HWACCREG14
  3. LFSR Load

In HWA 2.0

1. Error interrupts from HWA
2. MEM\_INIT\_START for all available memory
3. Read FSM\_STATE
4. HWA single step enable, and software trigger single enable.
5. Read MEM\_ACCESS\_ERR\_STATUS
6. Interrupt Enable register and interrupt enable clear register
7. SAFETY related registers
8. FAULT related registers.
9. LFSR load

# Definitions, Abbreviations, Acronyms

|  |  |
| --- | --- |
| **Term** | **Definition** |
| **HWA** | **Hardware Accelerator** |
| **PARAM** | **Parameter** |
| **DMA** | **Direct Memory Access** |

# References

1. Functional Requirement:

HWA 1.0

<https://jira.itg.ti.com/browse/MMWSDK-7>

1. HWA 1.0 IP Spec: <https://sps08.itg.ti.com/sites/autoradar/Shared%20Documents/AR1xxx%20PG1.0%20Architecture/Radar_HW_accelerator_v0p75.doc>
2. HWA User Guide: TBA
3. AR1xxx Architecture Spec: <https://sps08.itg.ti.com/sites/autoradar/Shared%20Documents/AR1xxx%20PG1.0%20Architecture/AR1642_AutoRadar_ASD_V1.1.pdf>
4. HWA 1.0 Register Map:

<http://www-open.india.ti.com/~wdccm/autoradar/verif/docs/magillem_db/ar12xx/html/MSS_HTML/DSS_HW_ACC.html>

<http://www-open.india.ti.com/~wdccm/autoradar/verif/docs/magillem_db/ar12xx/html/MSS_HTML/DSS_HW_ACC_PARAM.html>

<http://www-open.india.ti.com/~wdccm/autoradar/verif/docs/magillem_db/ar12xx/html/MSS_HTML/DSS_HW_ACC_WIN.html>

1. HWA 2.0 User Guide [https://sps16.itg.ti.com/sites/autoradar/\_layouts/15/WopiFrame.aspx?sourcedoc={288CEC32-B3CE-4AFA-9B69-EC13C35E0BAD}&file=Radar\_Hardware\_Accelerator\_2.0\_User\_Guide\_Part1\_v0\_93.docx&action=default](https://sps16.itg.ti.com/sites/autoradar/_layouts/15/WopiFrame.aspx?sourcedoc=%7b288CEC32-B3CE-4AFA-9B69-EC13C35E0BAD%7d&file=Radar_Hardware_Accelerator_2.0_User_Guide_Part1_v0_93.docx&action=default%20)

And

[https://sps16.itg.ti.com/sites/autoradar/\_layouts/15/WopiFrame.aspx?sourcedoc={CF14239B-040C-489B-A7C5-AE3A4DFA12DC}&file=Radar\_Hardware\_Accelerator\_2.0\_User\_Guide\_Part2\_v0\_93.docx&action=default](https://sps16.itg.ti.com/sites/autoradar/_layouts/15/WopiFrame.aspx?sourcedoc=%7bCF14239B-040C-489B-A7C5-AE3A4DFA12DC%7d&file=Radar_Hardware_Accelerator_2.0_User_Guide_Part2_v0_93.docx&action=default%20)

1. HWA 2.0 Register Map:

Common register:

<https://www-open.india.ti.com/~wdccm/tpr12/verif/docs/reg_db/html/DSS_CORE/DSS_HWA_CFG.html>

paramset:

[https://sps16.itg.ti.com/sites/autoradar/\_layouts/15/WopiFrame.aspx?sourcedoc={31BA5509-A30A-45C1-B5E7-2FF0EF102F4C}&file=HWA2.0\_Parameter\_Set\_Registers\_Layout\_v0p93.xlsx&action=default](https://sps16.itg.ti.com/sites/autoradar/_layouts/15/WopiFrame.aspx?sourcedoc=%7b31BA5509-A30A-45C1-B5E7-2FF0EF102F4C%7d&file=HWA2.0_Parameter_Set_Registers_Layout_v0p93.xlsx&action=default)

# Design

## System Architecture

The HWA Driver is a low level driver which encapsulates the hardware accelerator and allows applications to process the radar data being generated by the Analog front end

OSAL

TI RTOS

HWA hardware IP

Device/CPU configuration

HWA Driver

API

**HWA\_HWAttrs** structure

**SemaphoreP, MemoryP, ClockP, HwiP**

**Application/mmWave API/mmWaveLink**

void **HWA\_init** (void)

HWA\_Handle **HWA\_open** (uint32\_t index, int32\_t \*errCode)

int32\_t **HWA\_close** (HWA\_Handle handle)

int32\_t **HWA\_reset** (HWA\_Handle handle)

int32\_t **HWA\_configCommon** (HWA\_Handle handle, HWA\_CommonConfig \*commonConfig)

int32\_t **HWA\_configParamSet** (HWA\_Handle handle, uint8\_t paramsetIdx, HWA\_ParamConfig \*paramConfig, HWA\_SrcDMAConfig \*dmaConfig)

**…..**

**…..**

**…..**

**API**

Figure : HWA Driver Design

## Platforms supported

### HWA 1.0

The driver is supported on R4F architecture and AR14XX SoC on all AR14XX EVMs. The EVMs don’t matter really because there are no external to SoC components required for testing.

### HWA 2.0

The driver is supported on R5F and DSP on TPR12 only

## External Interfaces (API)

Please refer to the doxygen documentation under the hwa/docs directory for information on the internal and external APIs/structures.

Following sub-sections provide information on some of these APIs and their implementation.

### HWA 1.0

### Initialization

This function must be called once per system and before any other HWA driver APIs. It resets the HWA H/W instances in the system.

### Open

This function performs the following when this function is called for the first time:

1. Malloc the driver state context
2. Register HWA interrupts
3. enable access to hardware accelerator config memory
4. disable accelerator and clock, reset the accelerator, leave HWA disabled but enable the clock
5. reset PARAM set registers to 0

Any subsequent calls to this function will return the already allocated handle.

### Close

This function performs the following:

1. Close and cleanup the resources used by the HWA Instance
2. Reset the HWA and disable the state machine
3. The HWA handle is no longer valid and should not be used after the API has been invoked

### Config Common Registers

This function sets the following registers based on the configMask set by the application/higher layer:

1. Set the REG\_NUMLOOPS: 0-4094; 4095 is infinite
2. Set the REG\_PARAMSTART and REG\_PARAMSTOP
3. For FFT mode
   1. Set/Unset the REG\_FFT1D\_EN (1D FFT or not)
   2. Set the REG\_BPMPATTERN, REG\_BPMRATE (0 is invalid value)
   3. Set the REG\_INTERF\_THRESH (use 0xFFFFFF to disable the threshold)
   4. Set the REG\_TWID\_DITHER\_EN
   5. Set the REG\_LFSRSEED
   6. Set the REG\_FFTSUMDIV
4. For CFAR mode
   1. set the REG\_CFAR\_THR\_SCALE (the value to be specified here is as expected by register/HW)
5. Set the I and Q values for the CMULT scale

### Config Paramsets

This function sets the paramset registers. Refer to the doxygen for HWA param structure to get more details on the units and limits of the individual elements and the corresponding register bits that are set. The API sets only the register bits relevant to the selected mode – FFT, CFAR, CMULT.

### Config RAM

This one API provides access to configure both WINDOW RAM and INTERNAL RAM, one at a time. It uses memcpy functionality to copy the configuration from application filled buffer to the corresponding RAM inside the HWA.

### Enable Paramset done interrupt

This API provides a mechanism to enable completion interrupt for a given paramset.

1. If the requested interrupt is for R4F, then it sets the REG\_INTREN and saves the application’s callback fn/arg. This callback is called in the ISR context when this paramset completion interrupt is received by the R4F.
2. If the requested interrupt is for DMA, it sets the REG\_DMATRIGEN and sets the DMA channel number in REG\_ACC2DMA\_TRIGDST

### Enable Done interrupt

This API provides a mechanism to enable completion interrupt after all paramsets from start index to stop index have been executed by the state machine in hardware for the specified loops. The enable control for this interrupt is at the VIM level for R4F and hence HwiP APIs are used to enable this interrupt. The API function also saves the application’s callback fn/arg. This callback is called in the ISR context when this completion interrupt is received by the R4F.

### HWA 2.0

### **Initialization**

This function must be called once per system and before any other HWA driver APIs. It resets the HWA H/W instances in the system.

### Open

This function performs the following when this function is called for the first time:

1. Set up the driver state context
2. Register HWA two thread done interrupts, and two paramset done interrupts
3. enable access to hardware accelerator common register, paramset memory and data RAM
4. clear the PARAM\_DONE\_SET\_STATUS register and TRIGGER\_SET\_STATUS register
5. clear the histogram RAM
6. clear all the clip status registers
7. disable accelerator and clock, reset the accelerator, leave HWA disabled but enable the clock
8. reset PARAM set registers to 0

Any subsequent calls to this function will return the already allocated handle.

### Close

This function performs the following:

1. Close and cleanup the resources used by the HWA Instance
2. Reset the HWA and disable the state machine
3. The HWA handle is no longer valid and should not be used after the API has been invoked

### Config Common Registers

This function sets the following registers based on the configMask set by the application/higher layer, the configMask is one of defined HWA\_COMMONCONFIG\_MASK values,

1. Set up PARAM\_RAM\_LOOP and PARAM\_RAM\_IDX registers.
2. Set up PARAM\_RAM\_LOOP\_ALT and PARAM\_RAM\_IDX\_ALT registers.
3. Set up the context switch trigger mode, and trigger source if it is DMA trigger in the CS\_CONFIG register for context switch
4. For FFT mode
   1. Set up BPM\_RATE and BPM\_PATTERN registers
   2. Set/unset the twiddle dither register DITHER\_TWID\_EN
   3. Set up the LFSR\_REED register
   4. Set up the right-shifting for sum statistic in the register FFTSUMDIV.
   5. Set up shift and scale to all 12 accumulator output in register DC\_EST\_CTRL
   6. Set up the programmed I/Q values used in DC subtraction for all 12 outputs in the registers DC\_I0\_SW to DC\_I11\_SW, and DC\_Q0\_SW to DC\_Q11\_SW
   7. Set up the interference magnitude threshold values for all 12 outputs in the registers INTF\_LOC\_THRESH\_MAG0\_VAL to INTF\_LOC\_THRESH\_MAG11\_VAL
   8. Set up the interference magnitude difference threshold value for 12 outputs in the registers INTF\_LOC\_THRESH\_MAGDIFF0\_VAL to INTF\_LOC\_THRESH\_MAGDIFF11\_VAL
   9. Set up the scale and shift applied to interference magnitude and magnitude difference sum in the register INTF\_STATS\_CTRL
   10. Set up the complex scalars for complex multiply scale multiply and vector multiply in the registers ICMULT\_SCALE0 to ICMUL\_SCALE11 and QCMULT\_SCALE0 to QCMUL\_SCALE11.
   11. Set up the delta fractional frequency increment per param-set loop used in complex multiplier mode 10 in the register TWID\_INCR\_DELTA\_FRAC
   12. Set up the number of samples after channel combination, and the channel combination mask in the register CHAN\_COMB\_SIZE and eight mask registers: CHAN\_COMB\_VEC\_0 to CHAN\_COMB\_VEC\_7
   13. Set up number of zeros to be inserted in zero insertion, and zero insert mask in the register ZERO\_INSERT\_NUM and eight zero insert mask registers, ZERO\_INSERT\_MASK\_0 to ZERO\_INSERT\_MASK\_7.
   14. Set up offset to be added to dimension 1 and dimension 2 maximum results for 2D maximum in the register MAX2D\_OFFSET\_DIM1 and MAX2D\_OFFSET\_DIM2
   15. Set up the CDF\_CNT\_THRESH register for CDF calculation.
   16. Set up window parameters in register INTF\_MITG\_WINDOW\_PARAM0 to register INTF\_MITG\_WINDOW\_PARAM4
5. For CFAR mode
   1. Set up the threadhold scale factor in the register CFAR\_THRESH register
6. For Local maximum Mode
   1. Set up the threshold value for dimension C and B in the register LM\_THRESH\_VAL
   2. Set up the base address in stats RAM for the threshold values for dimension B or C in the register LM\_2DSTATS\_BASE\_ADDR
7. Compression/decompression Mode
   1. Set up K-param values used in EGE compression/decompression in CMP\_EGE\_K0123 and CMP\_EGE\_K5678 registers.

### Config Paramsets

This function sets the paramset registers. Refer to the doxygen for HWA param structure to get more details on the units and limits of the individual elements and the corresponding register bits that are set. The API sets only the register bits relevant to the selected mode – FFT, CFAR, Local Maximum, compression.

### Config/read/clear RAM

There is one API, which provides access to configure the windows, vector multiply, shuffle or frequency derotation RAM one at a time. It uses memcpy functionality to copy the configuration from application filled buffer to the corresponding RAM inside the HWA.

There is one API, which provides access to read data from 2D statistics RAM ( 2DSTAT\_ITER\_VAL, 2DSTAT\_ITER\_IDX, 2DSTAT\_SAMPLE\_VAL, 2DSTAT\_SAMPLE\_IDX), and histogram RAM.

There is one specific API to read histogram threshold RAM.

For now, only histogram RAM clear API is provided.

### Read data from common register

There are several APIs provide access to the common registers, to retrieve the HWA results

1. CFAR detected peak count in the CFAR\_PEAKCNT register
2. 12 DC accumulator results from the register DC\_ACC\_I/Q\_0\_VAL to DC\_ACC\_I/Q\_11\_VAL registers.
3. DC estimation results from register the register DC\_EST\_I/Q\_0\_VAL to DC\_EST\_I/Q\_11\_VAL
4. Number of samples that exceeded the threshold in a chip, or in a frame from the register INTF\_LOC\_COUNT\_ALL\_CHIRP or INTF\_LOC\_COUNT\_ALL\_FRAME.
5. Interference magnitude or magnitude difference threshold values in the register INTF\_LOC\_THRESH\_MAG0\_VAL to INTF\_LOC\_THRESH\_MAG1\_VAL, or the registers INTF\_LOC\_THRESH\_MAGDIFF0\_VAL to INTF\_LOC\_THRESH\_MAGDIFF11\_VAL.
6. Interference magnitude sum or magnitude difference sum values in the register INTF\_STATS\_MAG\_ACC\_0\_VAL to INTF\_STATS\_MAG\_ACC \_11\_VAL, or the registers INTF\_STATS\_MAGDIFF\_ACC\_0\_VAL to INTF\_STATS\_MAGDIFF\_ACC \_11\_VAL

### Read/clear clip status register

The clip read API provides access to the following clip status registers, and another API to clear the following clip status register,

1. both I/Q of DC accumulators 0 to 11 in the DC\_ACC\_CLIP\_STATUS register,
2. both I/Q DC estimation clip status in the DC\_EST\_CLIP\_STATUS register,
3. DC subtraction clip status in the DC\_SUB\_CLIP register
4. Interference magnitude or magnitude difference accumulator clip status in the INTF\_STATS\_ACC\_CLIP\_SATUS
5. Sum of magnitude or magnitude difference values clip status in THE INTF\_STATS\_SUM\_MAG\_VAL\_CLIP\_STATUS or INTF\_STATS\_SUM\_MAGDIFF\_VAL\_CLIP\_STATUS
6. Interference magnitude difference or magnitude threshold clip status in INTF\_STATS\_THRESH\_CLIP\_STATUS register
7. Twid\_incr\_delta\_frac accumulator clip status in the TWID\_INCR\_DELTA\_FRAC\_CLIP\_STATUS register
8. Channel combination clip status in the CHANNEL\_COMB\_CLIP\_STATUS register
9. FFT clip status in the FFT\_CLIP register
10. Input or output formatter clip status in the IP\_OP\_FORMATTER\_CLIP\_STATUS register

### Context switch

There are several APIs to the enable/disable, trigger context switch

1. One API enable/disable the context switching feature, will allow context switch to ALT thread if it is enabled in the paramset.
2. One API for software trigger context switch
3. One API for DMA trigger context switch.

### Paramset done polling

1. One API to check signel paramset done
2. One API to check multiple paramset done

### Enable Paramset done interrupt

This API provides a mechanism to enable completion interrupt for a given paramset, in either background or ALT thread

1. If the requested interrupt is for DSP or R5F, application needs to specify it is either interrupt 1 or interrupt 2. Then the API saves the application’s callback fn/arg, based on which interrupt. This callback is called in the ISR context when this paramset completion interrupt is received by the core
2. If the requested interrupt is for DMA, it sets the DMATRIG\_EN and sets the DMA channel number in HWA2DMA\_TRIGDST in the paramset, in this case, API will ignore the interrupt 1 or interrupt 2 input.

### Enable Done interrupt

This API provides a mechanism to enable completion interrupt for either background thread or ALT thread, after all paramsets in the threads have been executed by the state machine in hardware for the specified loops. The API function also saves the application’s callback fn/arg. This callback is called in the ISR context when this completion interrupt is received by the core for either background or ALT thread.

### Software reset

Three software reset APIs are provided:

1. One API for resetting either Interference stats module or for all 12 DC estimation accumulators
2. One API for resetting the param set counter / execution counter used in Complex multiplier mode 8.
3. One API for resetting the param set counter used in Complex multiplier mode 10.

## Error Handling

Every external API does parameter checking and returns a valid error code on error detection

## Driver Resources

### HWA 1.0

HWA driver uses OSAL malloc API to allocate memory of sizes:

* sizeof(HWA\_Driver)
* sizeof(HWA\_InterruptCtx) \* numHwaParamSets

### HWA 2.0

HWA driver does not need to use malloc to allocate memory.

## Application programming sequence

This section here depicts a typical programming sequence that an application could follow to use the HWA driver (both 1.0 and 2.0 drivers). Please refer to the HWA drivers unit test code for more details.

1. HWA\_Init
2. HWA\_Open
3. For each paramSet
   1. HWA\_ConfigParamSet
   2. DMA Programming
      1. If (triggerType == DMA), then program the 2 input DMA: one for data and one for triggering
      2. If completionInterrupt is to DMA, then program the output DMA for data
   3. HWA\_EnableInterrupt
      1. If R4F for HWA 1.0, or R5F/DSP for HWA 2.0 interrupt is desired, supply the callback fn/arg, for HWA 2.0, also supply the interrupt index
      2. If DMA interrupt is desired, then program the DMA first and then provide the channel number
      3. Both R4F and DMA interrupts can be enabled by calling HWA\_EnableInterrupt twice with the corresponding parameters.
4. HWA\_ConfigCommon
5. HWA\_ConfigRAM
6. HWA\_Enable
7. HWA\_SoftwareTrigger (Optional)
   1. Should be done if (triggerMode==Software for any paramset)
8. Wait for the completion interrupt, if enabled.

Note that due to the limitation imposed by the specific implementation of HWA VBUSP in HWA 1.0 only, Window RAM and Internal RAM (via HWA\_ConfigRAM) can be configured only once before the HWA is triggered even once. If the re-configuration of the RAMs is desired for any reason after the current state machine is executed and done interrupt is received, then application must first call the “HWA\_reset” API before calling the HWA\_ConfigRAM.

# Standards, Conventions and Procedures

## Documentation Standards

The driver’s software is documented using doxygen. Test code is not documented with doxygen because of future restructuring for MCPI test framework compliance.

## Coding Standards

MCPI coding standards are used:

<https://mcu-twiki.design.ti.com/bin/view/MCU/OneMCUSW/CodingStandard>

## Software development tools

Refer to the mmWave SDK release notes for details on the software needed for compilation and debugging of this driver.

## Safety Standards

MISRA-C compliance is under study and will be implemented in future as part of the mmWave SDK requirement.