Application Report
SWRA656B–September 2019–Revised May 2020

Migrating to xWR68xx and xWR18xx Millimeter Wave Sensors

Nitin Sakhuja and Chethan Kumar Y.B.

ABSTRACT

This application report provides guidance for porting mm-wave hardware and application software to the xWR68xx ES2.0 and the xWR18xx devices.

Contents

List of Figures

List of Tables

Trademarks

1 Introduction

The information presented here is applicable to any of the following scenarios:

- Have hardware/software currently deployed on xWR6843 ES1.0 and want to migrate it to xWR6843 ES2.0
- Have hardware/software currently deployed on xWR1642 and want to migrate it to xWR6843 ES2.0
- Have hardware/software currently deployed on xWR1642 and want to migrate it to xWR1843
- Have hardware/software currently deployed on xWR6843AOP ES1.0 and want to migrate to xWR6843AOP ES2.0

The information presented in this document covers:

- Comparison of the base and the new target device along-with a description of how those differences impact existing hardware and software.
- SDK version required for the new target device and updates needed in application build infrastructure (makefiles and/or CCS projects, linker command files, and so forth)
- Updates needed in application source code, for example, API updates, new structure parameters, and so forth.
- Example source code comparison snapshots are provided for easy reference.

For information specific to your current and target device, see the following sections.

Table 1. Migration Reference

Current Device	Target Device	Section
xWR6843 ES1.	xWR6843 ES 2.0	Migrating from xWR6843 ES1.0 to xWR6843 ES2.0 : Section 2.2
xWR1642	xWR6843 ES2.0	Migrating from xWR1642 to xWR6843 ES2.0 : Section 2.3
xWR1642	xWR1843	Migrating from xWR1642 to xWR1843 : Section 3
xWR6843AoP ES1.0	xWR6843AoP Es2.0	Migrating from xWR6843AoP ES1.0 to xWR6843AoP ES2.0 :Section 4

2 xWR6843 ES2.0 Hardware/Software Migration

This section provides Hardware/software migration information for porting hardware and application code from the xWR6843 ES1.0 device to the xWR6843 ES2.0 device. The information provided here is meant to cover the major changes for migrating to a particular MMWAVE-SDK release at the time of writing. For more details, see the *Migration* section in the MMWAVE-SDK Release Notes.

2.1 Hardware Changes From xWR6843 ES1.0 to xWR6843 ES2.0

The changes described in this section are relevant when migrating xWR6843 ES1.0 hardware to xWR6843 ES2.0. Figure 1 shows the device symbolization change from ES1.0 to ES2.0 on device part marking.

Left side device marking shows ES1.0 silicon and right side device marking shows ES2.0 silicon. For more details on the device marking, see the *xWR6843 Device Errata, Silicon Revisions 1. and 2.0*.

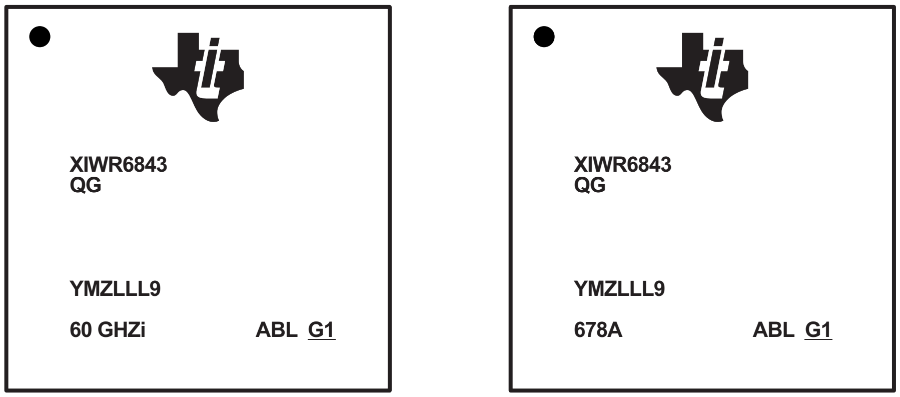


Figure 1. Silicon Device Marking Difference Between xWR6843 ES1.0 and ES2.0

Table 2. xWR6843 ES1.0 to xWR6843 ES2.0 Hardware Changes

No	Summary	xWR6843 ES1.0	xWR6843 ES2.0
1	QSPI interface speed has been improved. This enables faster boot loading, note that supported flashes are listed in the *Flash Variants Supported by the mmWave Sensor*.	Max 40 MHz	Max 80 MHz
2	Boot loader enhancement has been made. This allows faster boot and stability across devices	Boot loader code used to do the APLL calibration	Closed loop APLL calibration will be done by BSS
3	Tx beam forming is introduced (Antenna design need to be considered for the TX beam forming)	No support	Supported
4	Memory compression (Depending upon the compression ratio of the RADAR data cube larger memory would be available for code and remaining data)	No support	Supported
5	Calibration is supported (This improves the performance and stability of the device across temperature)	No Calibration	Calibration supported
6	Clock gating at power-up and IP clock gating based on use-case, this should improve the power saving	No clock gating	Clock gated on unused peripherals. Device low level drivers un-gates the clock depending upon the peripheral used

2.2 Software Migration From xWR6843 ES1.0 to xWR6843 ES2.0

The changes described in this section are relevant for migrating the xWR6843 ES1.0 software based on the SDK 3.2.1 to xWR6843 ES2.0.

If you are migrating from previous SDK 3.x releases, it is recommended to first port to SDK 3.2.1 and ensure that the ported software works successfully on ES1.0 device before porting to SDK 3.3 (which does not support ES1.0 device). This simplifies the debug process by isolating the changes required for ES2.0. For initially migrating to SDK 3.2.1 from older SDK releases, see the *Release Notes* archive in SDK 3.3 package.

NOTE: MMWAVE-SDK 3.3 is the first baseline SDK release for IWR6843ES2.0 device and the scope of the migration notes provided in this section is limited to migrating to MMWAVE-SDK 3.3.

When migrating existing xWR6843 ES2.0 applications to SDK releases beyond MMWAVE SDK 3.3, you should follow the incremental migration instructions provided in the corresponding SDK release notes.

Addendum for AWR6843 ES2.0: MMWAVE-SDK 3.4 is the first baseline SDK release that supports AWR6843 ES2.0. Users migrating to AWR6843 ES2.0 should first follow the steps provided below to migrate to SDK 3.3 and then follow the incremental migration notes provided in the release notes for MMWAVE-SDK 3.4.

Table 3. xWR6843 ES1.0 to xWR6843 ES2.0 Software Migration

No	Summary	Components Impacted	Required Changes
1	MMWAVE-SDK 3.3.0 or above required for IWR6843 ES2.0. MMWAVE-SDK 3.4 or above required for AWR6843 ES2.0 (Please read the AWR6843 addendum in Section 2.2)	Makefile **OR** CCS projects	Application code must be re-compiled with MMWAVE-SDK 3.3.0 or above to run on xWR6843 ES2.0 as prior SDK versions are not compatible with ES2.0. Conversely, SDK 3.3 is not compatible with xWR6843 ES1.0 devices. **Makefile: No change required if you are using SDK makefiles**, as this is automatically handled in the SDK 3.3 environment setup script: C:\ti\mmwave_sdk_03_03_xx_xx\packages\scripts\windows\setenv.bat **OR** **CCS Projectspec:** If the application is compiled using CCS projectspecs, you need to update the products property in DSS and MSS projectspecs as shown below. <property name="products" value="com.ti.rtsc.SYSBIOS:6.73.01.01;com.ti.MMWAVE_SDK:3.3.0.03;"/> **Example:** For reference CCS projects for xWR6843 ES2.0, see the 68xx – mmWave SDK Demo available in MMWAVE Industrial Toolbox.
2	Change the value of SHMEM_ALLOC parameter in MetaImage (flashable) binary generation step.	Makefile OR CCS projects (mss).	The value of SHMEM_ALLOC parameter should be set to 0x00000006 for ES2.0 (it was 0x02000006 for ES1.0 device). **Makefile: No change required if you are using SDK makefiles** build, as this is automatically handled in the SDK 3.3 device specific makefiles. **OR** **CCS Projectspec:** If the application is compiled using CCS projectspecs, update the postBuildStep in MSS projectspec to replace the value 0x02000006 with 0x00000006. **Example:** For reference CCS projects for xWR6843 ES2.0 , see the 68xx – mmWave SDK Demo available in MMWAVE Industrial Toolbox.

Table 3. xWR6843 ES1.0 to xWR6843 ES2.0 Software Migration (continued)

No	Summary	Components Impacted	Required Changes
3	API update for MMWave_open SDK 3.3 requires new parameter to be passed to MMWave_open	MSS/DSS start-up code	**MMWave_open**: Application must set the value of calibMonTimeUnit parameter before calling MMWave_open as shown below. The image below shows reference code updates in the SDK 68xx mmw demo **File:** mmwave_sdk_03_03_xx_xx\packages\ti\demo\xwr68xx\mmw\mss\mss_main.c **Code Snapshot: see Section 6.1**
4	API update for ADCBuf_open SDK 3.3 requires new parameter to be passed to ADCBuf_open	MSS/DSS start-up code	**ADCBUF_open**: Application must set the value of socHandle in the ADCBufparams structure before calling ADCBUF_open as shown below. The image below shows reference code updates in the SDK 68xx mmw demo. **File:** mmwave_sdk_03_03_xx_xx\packages\ti\demo\utils\mmwdemo_adcconfig.c **Code Snapshot: see** Section 6.2
5	API update for CANFD_init SDK 3.3 requires new parameter to be passed to CANFD_init	Drivers	**CANDF_init**: Applications using CANFD driver must pass instance ID to the CANFD_init API as shown below. Only a value of 0 is supported at this time. The image below shows reference code updates in the SDK CANFD driver test. **File:** mmwave_sdk_03_03_xx_xx\packages\ti\drivers\canfd\test\xwr68xx\main.c **Code Snapshot: see Section 6.3**
6	SDK 3.3 removes support for Bus error interrupt from the DMA driver for xWR6843 ES2 as that interrupt is not hooked up to the device.	Drivers	Application would get an error code back from the xwr68xx driver if DMA_enableInterrupt API is called for DMA_IntType_BER. You can either remove the call to the above API can either be removed or ignore the error; however you should review the DMA usage to make sure there is no invalid memory access via MSS DMA engine.
7	General note on CLI configuration file	Sensor Configuration	For applications that re-use the mmWave demo/CLI framework, ensure that the configuration commands (for example, profileCfg, chirpCfg, frameCfg, and so forth) follow the format provided in sample configuration files provided in the mmw demo directory: C:\ti\mmwave_sdk_03_03_xx_xx\packages\ti\demo\xwr68xx\mmw\profiles. for more details, see the *Configuration File Format* section in the mmwwave SDK User's Guide. Section 7
8	BSS clock un-gate required in Secondary bootloader	Secondary Bootloader	**Note: This update is not related to the main application**. It is needed only if you are using a custom secondary bootloader in your system. Secondary Bootloader must ungate BSS clock using SOC gate/ungate API before downloading image to RadarSS/BSS memory as shown below. The image below shows reference code updates in the SDK secondary bootloader example. **File:** C:\ti\mmwave_sdk_03_03_xx_xx\packages\ti\utils\sbl\platform\sbl_xwr68xx.c **Code Snapshot: see** Section 6.4

2.3 Migrating From xWR1642 to xWR6843 ES2.0

This section provides migration guidance to port hardware and software running on the xWR1642 to the xWR6843 ES2.0 device.

2.3.1 Device Comparison

Table 4 lists the key features of the xWR1642 and the xWR6843 devices that need to be considered from a migration perspective. For more information, see the device-specific data sheets and the *IWR14xx/16xx/18xx/68xx Industrial Radar Family Technical Reference Manual*.

Figure 2 shows the device symbolization change from xWR1642 to xWR6843 on the device part marking.

The left side device marking shows the xWR1642 silicon and the right side device marking shows the xWR6843 silicon. For more details on the device marking, see the device-specific Errata.

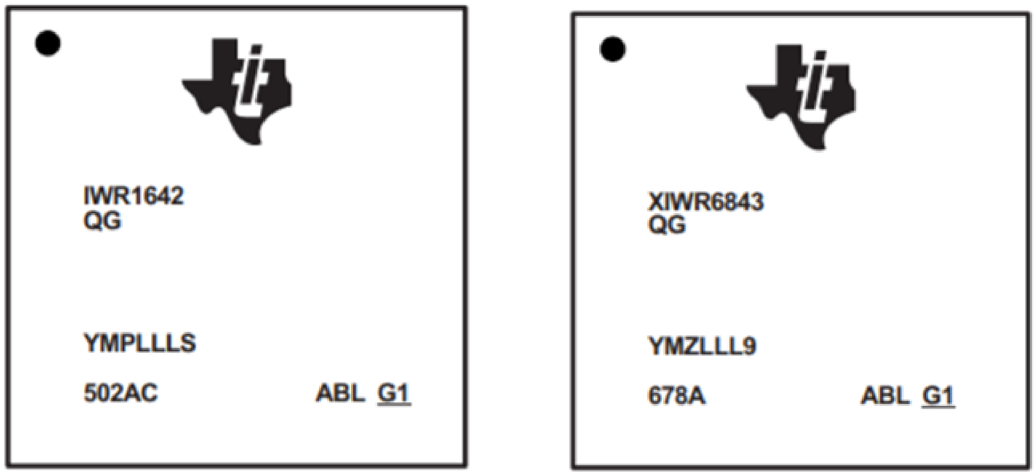


Figure 2. Silicon Device Marking Difference Between xWR1642 and xWR6843

- *IWR1642 Device Errata*
- *AWR1642 Device Errata*
- *IWR6843 Device Errata*
- *AWR6843 Device Errata*

Table 4. Feature List Comparison Between xWR1642 and xWR6843

No	Device Feature Differences	xWR1642	xWR6843	Hardware and Software Impact
1	Frequency Range	76-81 GHz (4GHz continuous)	60-64 GHz (4GHz continuous)	Changes needed in the Antenna design and RF front end configuration
2	Number of Transmit Channels	2	3 [(1)]	3rd Transmitter Antenna need to be designed. Update TX bitmap in chirpCfg
3	Maximum Sampling Rate	6.25 MHz complex	12.5 MHz complex	Higher IF bandwidth & Sampling rates are available on xWR6843
4	Max I/F (Intermediate Frequency)	5 MHz	10 MHz	
5	On-chip memory	1.5MB	1.75MB	Software can leverage the additional memory if needed.
6	Radar Accelerator	Not Applicable	Hardware accelerator for FFT, filtering, and CFAR processing	xWR6843 has flexibility of data processing on Hardware accelerator or DSP
7	Tx beam forming	No support	Supported	xWR6843 has phase shifters which supports the steerable beams. Note: Antennas need to be designed to support TX beam forming operation
8	Memory compression	No support	Supported	Depending upon the compression ratio of the RADAR data cube larger memory would be available for code and remaining data for the xWR6843
9	QSPI interface speed	40 MHz Max	80 MHz Max	QSPI interface has been improved. This enables faster boot loading. Note that supported flashes are listed in the *Flash Variants Supported by the mmWave Sensor*
10	MMWAVE-SDK support	SDK 2.1 (LTS) and above	SDK 3.3.0 and above	General software porting required compiling for xWR6843. For more information, see the migration notes in Section 2.3.4,

(1) 3 Tx simultaneous operation is supported only with 1-V LDO bypass and PA LDO disable mode. In this mode, the 1-V supply needs to be fed on the VOUT PA pin.

2.3.2 Hardware Migration Notes

Package to PCB transition:

- On the RF front end device package to PCB transitions on the RF balls need to be changed from xWR1642 to xWR6843. xWR1642 RF balls package transitions are listed in http://www.ti.com/lit/an/spracg5/spracg5.pdf
- For xWR6843 RF substrate material using the RO4835LOPRO design follow the RF BGA to ball transitions shown in the reference design xWR6843ISK. This design relies on GCPW transmission line for the antenna connectivity.

The reference design is given in the below link:

- Schematic, BOM and assembly files are available at http://www.ti.com/lit/zip/swrc355
- Altium EVM design files are available at http://www.ti.com/lit/zip/swrc355

- For xWR6843 RF substrate material using RO3003 design follow the RF BGA to ball transitions shown in the reference design xWR6843ISK-ODS. This design relies on Micro-strip transmission line for the antenna connectivity.

The reference design is given in the below link:

- Schematic, BOM and assembly files are available at http://www.ti.com/lit/zip/swrr165
- Altium EVM design files are available at http://www.ti.com/lit/zip/swrc356
- After the package to PCB transitions are done, the reference plane (50Ω impedance) is available at the edge of the package. From this point onward, any custom antenna could be connected through transmission line.

2.3.3 Antenna Migration

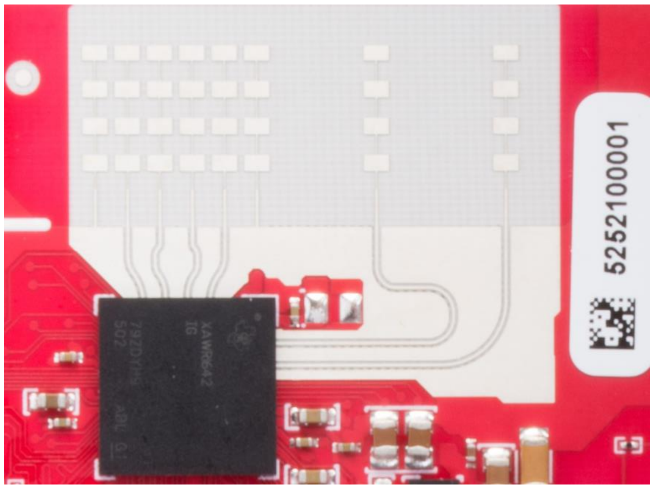


Figure 3. xWR1642 Antenna

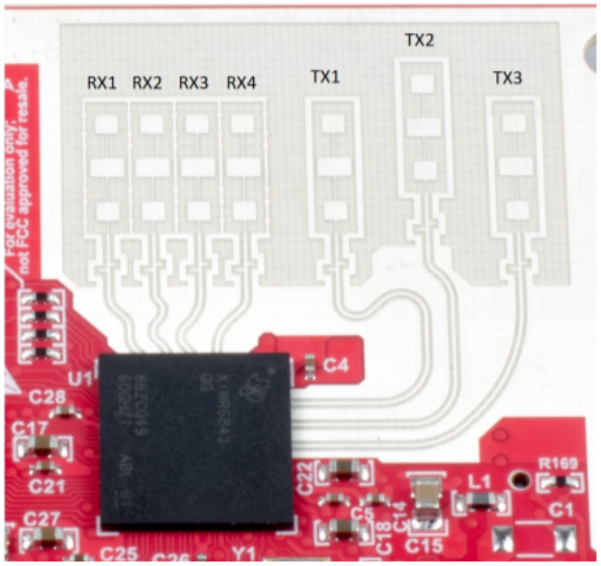


Figure 4. xWR6843 Antenna

From the xWR1642 to the xWR6843, antennas are needed to be redesigned for 60 GHz. For more information, see the above design file package that provides the antenna details for two configurations. For more information, see the *MMWAVEICBOOST and Antenna Module User's Guide*.

Hardware design checklist:

- The xWR1642 has the hardware design (Schematic, Layout, bring-up/wakeup) checklist, which is available at http://www.ti.com/lit/zip/swrr151.
- The xWR6843 hardware design (Schematic, Layout, Bring up/wakeup) checklist is available at http://www.ti.com/lit/zip/swrr161.

2.3.4 Software Migration Notes

Table 5 lists the changes required to port existing xWR1642 application code to xWR6843 ES2.0.

NOTE: MMWAVE-SDK 3.3 is the first baseline SDK release for xWR6843ES2.0 device and the scope of the migration notes provided in this section is limited to migrating to MMWAVE-SDK 3.3.

When migrating existing xWR6843 ES2.0 applications to SDK releases beyond MMWAVE SDK 3.3, you should follow the incremental migration instructions provided in the corresponding SDK release notes.

Addendum for AWR6843 ES2.0: MMWAVE-SDK 3.4 is the first baseline release that supports AWR6843 ES2.0. Users migrating to AWR6843 ES2.0 should follow the steps provided below to migrate to SDK 3.3 and then follow the incremental migration notes provided in the release notes for MMWAVE-SDK 3.4.

Table 5. xWR1642 to xWR6843 ES2.0 Software Migration

No	Summary	Components impacted	Required Changes
1	MMWAVE-SDK 3.3.0 or above required for xWR6843 ES2.0. MMWAVE-SDK 3.4 or above required for AWR6843 ES2.0 (Please read the AWR6843 addendum in Section 2.3.4)	Makefile **OR** CCS projects	Application code must be re-compiled with MMWAVE-SDK 3.3.0 or above to run on xWR6843 ES2.0. **Makefile: No change required if you are using SDK makefiles**, as this is automatically handled in the SDK 3.3 environment setup script: C:\ti\mmwave_sdk_03_03_xx_xx\packages\scripts\windows\setenv.bat **OR** **CCS Projectspec:** If the application is compiled using CCS projectspecs, you need to update the products property in DSS and MSS projectspecs as shown below. <property name="products" value="com.ti.rtsc.SYSBIOS:6.73.01.01;com.ti.MMWAVE_SDK:3.3.0.03;"/> **Example:** For reference CCS projects for xWR6843 ES2.0, see the 68xx – mmWave SDK Demo available in: MMWAVE Industrial Toolbox.
2	Change device type	Makefile **OR** CCS projects	**Makefile:** For SDK makefile based build, set MMWAVE_SDK_DEVICE=iwr68xx/awr68xx in setenv.bat. C:\ti\mmwave_sdk_03_03_xx_xx\packages\scripts\windows\setenv.bat **OR** **CCS Projectspec:** If the application is compiled using CCS projectspecs, change the define SOC_XWR16XX to SOC_XWR68XX in DSS and MSS projectspecs. **Example:** For reference CCS projects for xWR6843 ES2.0, see the 68xx – mmWave SDK Demo available in: MMWAVE Industrial Toolbox.
3	Update RadarSS firmware file path	Makefile **OR** CCS projects (mss)	Need to use iwr6xxx_radarss_rprc.bin in the metaimage generation step. **Makefile: No change required if you are using SDK makefiles**, as this is automatically handled in the SDK 3.3 environment setup script based on the MMWAVE_SDK_DEVICE variable. **OR** **CCS Projectspec:** If the application is compiled using CCS projectspecs, replace xwr16xx_radarss_rprc.bin with iwr6xxx_radarss_rprc.bin in the metaimage generation steps (postbuild steps) **Example:** For reference CCS projects for xWR6843 ES2.0, see the 68xx – mmWave SDK Demo available in: MMWAVE Industrial Toolbox.

Table 5. xWR1642 to xWR6843 ES2.0 Software Migration (continued)

No	Summary	Components impacted	Required Changes
4	Use xWR68xx platform linker command file	Makefile **OR** CCS projects	**Makefile: No change required if you are using SDK makefiles**, as this is automatically handled in the SDK 3.3 environment setup script based on the MMWAVE_SDK_DEVICE variable. **OR** **CCS Projectspec:** If the application is compiled using CCS projectspecs, update the include paths for r4f_linker.cmd and c674x_linker.cmd to COM_TI_MMWAVE_SDK_INSTALL_DIR/packages/ti/platform/xwr68xx/r4f_linker.cmd and COM_TI_MMWAVE_SDK_INSTALL_DIR/packages/ti/platform/xwr68xx/c674x_linker.cmd, respectively. **Example:** For reference CCS projects for xWR6843 ES2.0 , see the 68xx – mmWave SDK Demo available in: MMWAVE Industrial Toolbox.
5	Include xWR68xx driver and CLI libs	Makefile **OR** CCS projects	**Makefile: No change required if you are using SDK makefiles**, as this is automatically handled in the SDK 3.3 environment setup script based on the MMWAVE_SDK_DEVICE variable. **OR** **CCS Projectspec:** If the application is compiled using CCS projectspecs, update the linker include paths to select the *_xwr68xx.aer4f and *_xwr68xx.xe674 lib versions, for example: -llibsoc_xwr68xx.ae674, -llibsoc_xwr68xx.xe674, -llibcli_xwr68xx.aer4f
6	Update sensor front-end configuration parameters	CLI config file (.cfg) and/or source code	• Update the start frequency in profileCfg CLI command and/or MMWave_addProfile API parameters in code to use 60-64GHz frequency band. • Update TX channel bitmap in chirpCfg CLI command and/or API to account for the 3rd TX. **Example:** For more information, see the sample config files in C:\ti\mmwave_sdk_03_03_xx_xx\packages\ti\demo\xwr68xx\mmw\profiles.
7	Replace 16xx SOC definitions with 68xx equivalents	MSS/DSS source code	Replace **SOC_XWR16XX_*** definitions/macros in source code with corresponding **SOC_XWR68XX_*** definitions. **For instance:** Replace SOC_XWR16XX_MSS_ADCBUF_BASE_ADDRESS with SOC_XWR68XX_MSS_ADCBUF_BASE_ADDRESS, Similarly, in Pinmux configuration code, Replace SOC_XWR16XX_PINN5_PADBE with SOC_XWR68XX_PINN5_PADBE and so forth. The image below shows reference code difference between the SDK 16xx and 68xx mmw demos **File:**mmwave_sdk_03_03_xx_xx\packages\ti\demo\xwr68xx\mmw\mss\mss_main.c **Code Snapshot: see Section 6.6**
8	Update Calibration frequency limits in MMWave_open	MSS/DSS start-up code	The calibration frequency limits need to be updated before caling MMWave_open as shown below. The image below shows reference code difference between the SDK 16xx and 68xx mmw demos **File:** mmwave_sdk_03_03_xx_xx\packages\ti\demo\xwr68xx\mmw\mss\mss_main.c **NOTE:** The current radars firmware does not support these values for the 68xx device so these values are set to zero. These should be updated when support is available in the radars firmware. **Code Snapshot: see** Section 6.5

Table 5. xWR1642 to xWR6843 ES2.0 Software Migration (continued)

No	Summary	Components impacted	Required Changes
9	API update for MMWave_open SDK 3.3 requires new parameter to be passed to MMWave_open	MSS/DSS start-up code	**MMWave_open:** Application must set the value of calibMonTimeUnit parameter before calling MMWave_open as shown below. The image below shows reference code updates in the SDK 68xx mmw demo **File:**mmwave_sdk_03_03_xx_xx\packages\ti\demo\xwr68xx\mmw\mss\mss_main.c **Code Snapshot: see** Section 6.1
10	API update for ADCBuf_open SDK 3.3 requires new parameter to be passed to ADCBuf_open	MSS/DSS start-up code	**ADCBUF_open:** Application must set the value of socHandle in the ADCBufparams structure before calling ADCBUF_open as shown below. The image below shows reference code updates in the SDK 68xx mmw demo **File:**mmwave_sdk_03_03_xx_xx\packages\ti\demo\utils\mmwdemo_adcconfig.c **Code Snapshot: see Section 6.2**
11	API update for CANFD_init SDK 3.3 requires new parameter to be passed to CANFD_init	Drivers	**CANDF_init:** Applications using CANFD driver must pass instance ID to the CANFD_init API as shown below. Only a value of 0 is supported at this time. The image below shows reference code updates in the SDK CANFD driver test **File:**mmwave_sdk_03_03_xx_xx\packages\ti\drivers\canfd\test\xwr68xx\main.c **Code Snapshot: see** Section 6.3
12	SDK 3.3 removes support for Bus error interrupt from the DMA driver for xWR6843 ES2 as that interrupt is not hooked up to the device.	Drivers	Application would get an error code back from the xwr68xx driver if DMA_enableInterrupt API is called for DMA_IntType_BER. You can either remove the call to the above API or ignore the error; however, review the DMA usage to make sure there is no invalid memory access via MSS DMA engine.
13	General note on CLI configuration file	Sensor Configuration	For applications that re-use the mmWave demo framework, ensure that the configuration commands (profileCfg, chirpCfg, frameCfg, and so forth) follow the format provided in the sample configuration files provided in the mmw demo directory: C:\ti\mmwave_sdk_03_03_xx_xx\packages\ti\demo\xwr68xx\mmw\profiles. For more information, see the *Configuration File Format* figure in the mmwwave SDK User's Guide. See Section 7
14	BSS clock un-gate required in Secondary bootloader	Secondary Bootloader	**Note: This update is not related to the main application.** It is needed only if you are using a custom secondary bootloader in your systemSecondary Bootloader must ungate BSS clock using SOC gate/ungate API before downloading image to RadarSS/BSS memory as shown below. The image below shows reference code updates in the SDK secondary bootloader example. **File:** C:\ti\mmwave_sdk_03_03_xx_xx\packages\ti\utils\sbl\platform\sbl_xwr68xx.c **Code Snapshot: see** Section 6.4

3 xWR1843 Hardware/Software Migration

This section provides migration guidance to port Hardware and software from the xWR1642 to the xWR1843 device. The information provided here is meant to cover the major changes for migrating to a particular MMWAVE-SDK release at the time of writing. For more information, see the *Migration* section in the MMWAVE-SDK Release Notes.

3.1 Migrating From xWR1642 to xWR1843

3.1.1 Device Comparison

Table 6 lists the key features of the xWR1642 and the xWR1843 devices that need to be considered from Hardware and software migration perspective. For more information, see the device-specific data sheets and the *Industrial mmWave Radar Family Technical Reference Manual* in Section 7.

Figure 5 and Figure 6 show the device symbolization change from xWR1642 to xWR1843 on device part marking.

The left side device marking shows the xWR1642 silicon and the right side device marking shows the xWR1843 silicon. For more details on the device marking, see the device-specific Errata.

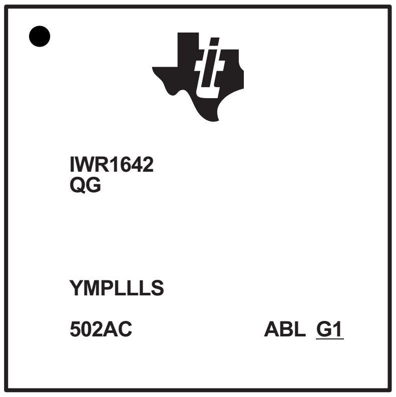


Figure 5. xWR1642 Device Marking

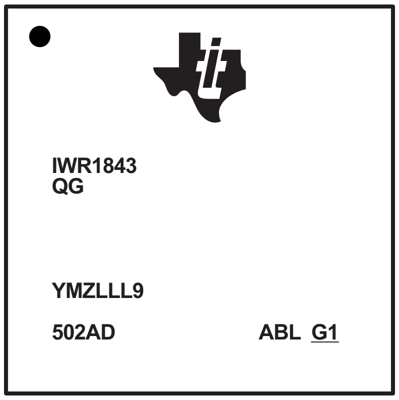


Figure 6. xWR1843 Device Marking

- *IWR1642 Device Errata*
- *AWR1642 Device Errata*
- *IWR1843 Device Errata*
- *AWR1843 Device Errata*

Table 6. Device Feature Comparision Table

No	Device Feature Differences	xWR1642	xWR1843	Hardware and Software Impact
1	Number of Transmit Channels	2	3 (1)	3rd Transmitter Antenna need to be designed. Update TX bitmap in chirpCfg
2	Maximum Sampling Rate	6.25 MHz complex	12.5 MHz complex	Higher IF bandwidth & Sampling rates are available on xWR1843
3	Max I/F (Intermediate Frequency)	5 MHz	10 MHz	
4	On-chip memory	1.5MB	2.0MB	Software can leverage the additional memory if needed.
5	Radar Accelerator	Not Applicable	Hardware accelerator for FFT, filtering, and CFAR processing	xWR1843 has flexibility of data processing on Hardware accelerator or DSP
6	Tx beam forming	No support	Supported	xWR1843 has phase shifters which supports the steerable beams. Note: Antennas need to be designed to support TX beam forming operation
7	MMWAVE-SDK support	SDK 2.1 (LTS) and above	SDK 3.3.0 and above	General software porting required compiling for xWR1843. For more information, see the Migration Notes.

(1) Three Tx Simultaneous operation is supported only with 1-V LDO bypass and PA LDO disable mode. In this mode, the 1-V supply needs to be fed on the VOUT PA pin.

3.1.2 Hardware Migration Notes

3.1.2.1 Antenna Addition

From xWR1642 to xWR1843, the third antenna needs to be introduced. For more information, see the design file package that provides the antenna details. Detailed field of view and radiations can be found in the user's guides listed below.

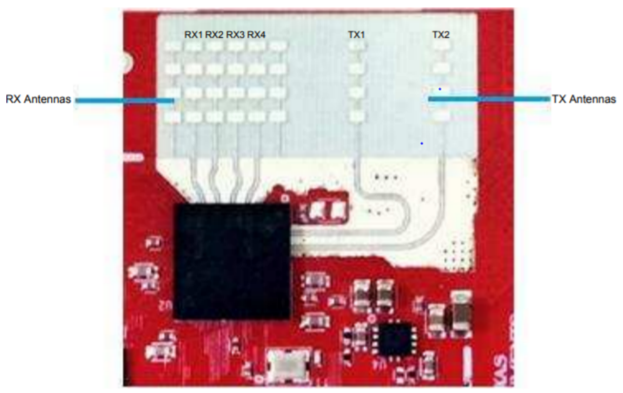


Figure 7. xWR1642 Antenna Image

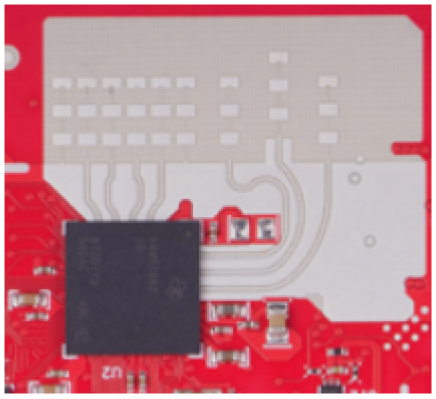


Figure 8. xWR1843 Antenna Image

- xWR1642BOOST Layout and Design Files
- *xWR1642 EVM (xWR1642BOOST) Single-Chip mmWave Sensing Solution User's Guide*
- xWR1843BOOST Hardware Files
- *xWR1843 Evaluation Module (xWR1843BOOST) Single-Chip mmWave Sensing Solution User's Guide*

3.1.3 Hardware Design Checklist

xWR1642 has the hardware design (schematic, Layout, bring-up/wakeup) checklist is available at http://www.ti.com/lit/zip/swrr151 and for the xWR1843 hardware design (schematic, Layout, Bring up/wakeup) checklist is available at http://www.ti.com/lit/zip/spracl2.

3.1.4 Software Migration Notes

Table 7 lists the changes required to port existing xWR1642 application code to xWR1843.

NOTE: The scope of the migration notes provided in this section is limited to migrating to MMWAVE-SDK 3.3.

When migrating existing xWR1843 applications to SDK releases beyond MMWAVE-SDK 3.3, you should follow the incremental migration instructions provided in the corresponding SDK release notes.

Table 7. xWR1642 to xWR1843 Software Migration

No	Summary	Components Impacted	Required Changes
`1	MMWAVE-SDK 3.2.1 or above required for xWR1843 **NOTE:** It is recommended to use SDK 3.3.0 or above to include the latest API updates.	Makefile **OR** CCS projects	Application code must be re-compiled with MMWAVE-SDK 3.3.0 or above to run on xWR1843 **Makefile: No change required if you are using SDK makefiles**, as this is automatically handled in the SDK 3.3 environment setup script: C:\ti\mmwave_sdk_03_03_xx_xx\packages\scripts\windows\setenv.bat **OR** **CCS Projectspec:** If the application is compiled using CCS projectspecs, you need to update the products property in DSS and MSS projectspecs as shown below. <property name="products" value="com.ti.rtsc.SYSBIOS:6.73.01.01;com.ti.MMWAVE_SDK:3.3.0.03;"/> **Example:** For reference CCS projects for xWR1843, see the 18xx – mmWave SDK Demo available in: MMWAVE Industrial Toolbox.
2	Change device type	Makefile **OR** CCS projects	**Makefile:** For SDK makefile based build, set MMWAVE_SDK_DEVICE=iwr18xx/awr18xx in setenv.bat. C:\ti\mmwave_sdk_03_03_xx_xx\packages\scripts\windows\setenv.bat **OR** **CCS Projectspec:** If the application is compiled using CCS projectspecs, change the define SOC_XWR16XX to SOC_XWR18XX in DSS and MSS projectspecs. **Example:** For reference CCS projects for xWR1843, see the 18xx – mmWave SDK Demo available in: MMWAVE Industrial Toolbox.
3	Update RadarSS firmware file path	Makefile **OR** CCS projects (mss)	Need to use xWR18xx_radarss_rprc.bin in the metaimage generation step. **Makefile: No change required if you are using SDK makefiles**, as this is automatically handled in the SDK 3.3 environment setup script based on the MMWAVE_SDK_DEVICE variable. **OR** **CCS Projectspec:** If the application is compiled using CCS projectspecs, replace xwr16xx_radarss_rprc.bin with xWR18xx_radarss_rprc.bin in the metaimage generation steps (postbuild steps) **Example:** For reference CCS projects for xWR1843, see the 18xx – mmWave SDK Demo available in: MMWAVE Industrial Toolbox.

Table 7. xWR1642 to xWR1843 Software Migration (continued)

No	Summary	Components Impacted	Required Changes
4	Use xWR18xx platform linker command file	Makefile **OR** CCS projects	**Makefile: No change required if you are using SDK makefiles**, as this is automatically handled in the SDK 3.3 environment setup script based on the MMWAVE_SDK_DEVICE variable. **OR** **CCS Projectspec:** If the application is compiled using CCS projectspecs, update the include paths for r4f_linker.cmd and c674x_linker.cmd to: COM_TI_MMWAVE_SDK_INSTALL_DIR/packages/ti/platform/xwr18xx/r4f_linker.cmd and COM_TI_MMWAVE_SDK_INSTALL_DIR/packages/ti/platform/xwr18xx/c674x_linker.cmd, respectively. **Example:** For reference CCS projects for xWR1843, see the 18xx – mmWave SDK Demo available in: MMWAVE Industrial Toolbox.
5	Include xWR18xx driver and CLI libs	Makefile **OR** CCS projects	**Makefile: No change required if you are using SDK makefiles**, as this is automatically handled in the SDK 3.3 environment setup script based on the MMWAVE_SDK_DEVICE variable. **OR** **CCS Projectspec:** If the application is compiled using CCS projectspecs, update the linker include paths to select the *_xwr18xx.aer4f and *_xwr18xx.xe674 lib versions, for example: -llibsoc_xwr18xx.ae674, -llibsoc_xwr18xx.xe674, -llibcli_xwr18xx.aer4f
6	Update sensor front-end configuration parameters	CLI config file (.cfg) and/or source code	Update TX channel bitmap in chirpCfg CLI command and/or API to account for the 3rd TX. **Example:** For more information, see the sample config files in C:\ti\mmwave_sdk_03_03_xx_xx\packages\ti\demo\xwr18xx\mmw\profiles.
7	Replace 16xx SOC definitions with 18xx equivalents.	MSS/DSS source code	Replace **SOC_XWR16XX_*** definitions/macros in source code with corresponding **SOC_XWR18XX_*** definitions. **For instance:** Replace SOC_XWR16XX_MSS_ADCBUF_BASE_ADDRESS with SOC_XWR18XX_MSS_ADCBUF_BASE_ADDRESS, Similarly, in Pinmux configuration code: Replace SOC_XWR16XX_PINN5_PADBE with SOC_XWR18XX_PINN5_PADBE and so forth. The image below shows reference code difference between the SDK 16xx and 18xx mmw demos **File:**mmwave_sdk_03_03_xx_xx\packages\ti\demo\xwr18xx\mmw\mss\mss_main.c **Code Snapshot: see** Section 6.7.
8	API update for MMWave_open SDK 3.3 requires new parameter to be passed to MMWave_open	MSS/DSS start-up code	**MMWave_open:** Application must set the value of calibMonTimeUnit parameter before calling MMWave_open as shown below. The image below shows reference code updates in the SDK 68xx mmw demo (same applies to 18xx mmw demo) **File:**mmwave_sdk_03_03_xx_xx\packages\ti\demo\xwr68xx\mmw\mss\mss_main.c **Code Snapshot: see** Section 6.1.
9	API update for ADCBuf_open SDK 3.3 requires new parameter to be passed to ADCBuf_open	MSS/DSS start-up code	**ADCBUF_open:** Application must set the value of socHandle in the ADCBufparams structure before calling ADCBUF_open as shown below. The image below shows reference code updates in the SDK 68xx mmw demo (same applies to 18xx mmw demo). **File:**mmwave_sdk_03_03_xx_xx\packages\ti\demo\utils\mmwdemo_adcconfig.c **Code Snapshot: see** Section 6.2.

Table 7. xWR1642 to xWR1843 Software Migration (continued)

No	Summary	Components Impacted	Required Changes
10	API update for CANFD_init SDK 3.3 requires new parameter to be passed to CANFD_init	Drivers	**CANDF_init:** Applications using CANFD driver must pass instance ID to the CANFD_init API as shown below. Only a value of 0 is supported at this time. The image below shows reference code updates in the SDK CANFD driver test (same for 18xx). **File:**mmwave_sdk_03_03_xx_xx\packages\ti\drivers\canfd\test\xwr618xx\main.c **Code Snapshot: see** Section 6.3.
11	General note on CLI configuration file	Sensor Configuration	For applications that re-use the mmWave demo framework, ensure that the configuration commands (profileCfg, chirpCfg, frameCfg, and so forth) follow the format provided in the sample configuration files provided in the mmw demo directory: C:\ti\mmwave_sdk_03_03_xx_xx\packages\ti\demo\xwr18xx\mmw\profiles. For more information, see the *Configuration File Format* section of the mmwwave SDK User's Guide. See Section 7.

4 xWR6843AoP ES2.0 Migration

This section provides migration guidance to port Hardware and software from the xWR6843AoP ES1.0 to the xWR6843AoP Es2.0 device. The information provided here is meant to cover the major changes for migrating to a particular MMWAVE-SDK release at the time of writing. For more information, see the *Migration* section in the MMWAVE-SDK Release Notes.

4.1 Hardware Changes From xWR6843AoP ES1.0 to xWR6843AoP ES2.0

The changes described in this section are relevant when migrating xWR6843AoP ES1.0 hardware to xWR6843AoP ES2.0. Figure 9 shows the device symbolization change from ES1.0 to ES2.0 on device part marking.

Left side device marking shows ES1.0 silicon and right side device marking shows ES2.0 silicon. For more details on the device marking, see the *xWR6843 Device Errata, Silicon Revisions 1. and 2.0*.

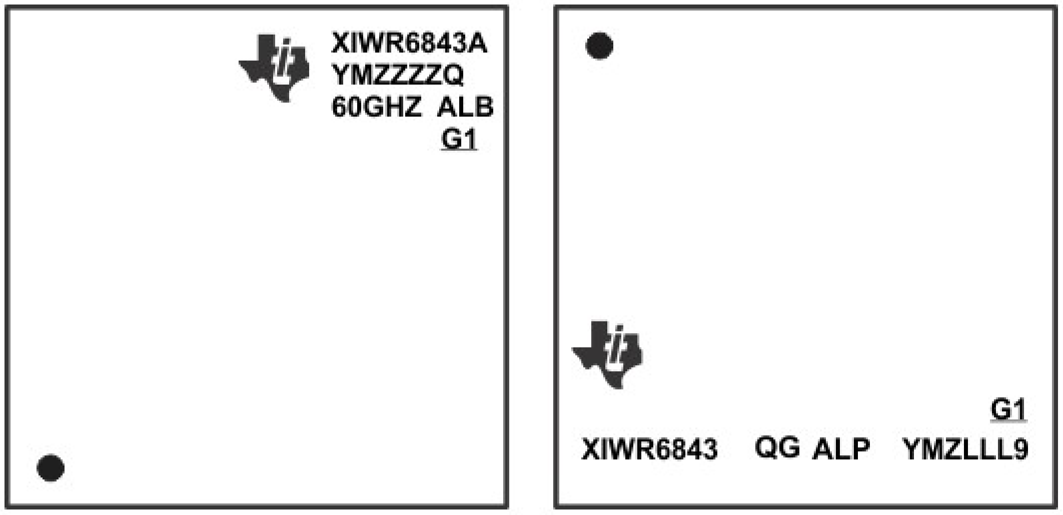


Figure 9. Silicon Device Marking Difference Between xWR6843AoP ES1.0 and ES2.0

Table 8. xWR6843AoP ES1.0 to xWR6843AoP ES2.0 Hardware Changes

No	Summary	xWR6843AoP ES1.0	xWR6843AoP ES2.0
1	QSPI interface speed has been improved. This enables faster boot loading, note that supported flashes are listed in the *Flash Variants Supported by the mmWave Sensor*.	Max 40 MHz	Max 80 MHz
2	Boot loader enhancement has been made. This allows faster boot and stability across devices	Boot loader code used to do the APLL calibration	Closed loop APLL calibration will be done by BSS
3	Tx beam scanning is introduced	No support	Supported
4	Memory compression (Depending upon the compression ratio of the RADAR data cube larger memory would be available for code and remaining data)	No support	Supported
5	Calibration is supported (This improves the performance and stability of the device across temperature)	No Calibration	Calibration supported
6	Clock gating at power-up and IP clock gating based on use-case, this should improve the power saving	No clock gating	Clock gated on unused peripherals. Device low level drivers un-gates the clock depending upon the peripheral used
7	RF Improvements –RX NF (Improved range and accuracy)	Baseline	Improved (Please refer to the datasheet for exact number)
8	RF Improvements –CLK PN (Improved accuracy)	Baseline	Improved (Please refer to the datasheet for exact number)
9	Package change	Baseline	Improved package (Please refer to the datasheet for detailed package information)
10	Changes in Antenna virtual Array	Baseline	Improvement in package routing caused changes in the antenna elements, hence there is change in virtual antenna array between ES1 and ES2.0. See Table 9

4.2 Software Migration From xWR6843AoP ES1.0 to xWR6843AoP ES2.0

The changes described in this section are relevant for migrating the xWR6843AoP ES1.0 software based on the SDK 3.2.0.6 to xWR6843AoP ES2.0 and SDK 3.4.

Besides the addition of the Antenna on Package and a different antenna pattern, xWR6843AoP ES2.0 re-uses the same silicon. Hence software migration from xWR6843AoP ES1.0 to xWR6843AoP ES2.0 broadly includes the following steps in order:

1. Initial migration of software to xWR6843ES2.0 (from MMWAVE-SDK 3.2.0.6 to MMWAVE-SDK 3.4). (Referred to below as **Platform Software Updates**)
2. Angle of Arrival Processing updates for the updated antenna pattern on xWR6843AoP ES2.0. (Referred to below as **AoA Software Updates**)

NOTE: MMWAVE-SDK 3.4.0 is the first baseline SDK release for xWR6843AoP ES2.0 device and the scope of migration notes provided in this section is limited to migrating to MMWAVE-SDK 3.4.0

When migrating existing xWR6843 AoP ES2.0 applications to SDK releases beyond MMWAVE SDK 3.4, you should follow the incremental migration instructions provided in the corresponding SDK release notes.

4.2.1 xWR6843AoP ES2.0 - Platform Software Updates

Table 9. xWR6843AoP ES2.0 Software - Platform Updates

No	Summary	Components Impacted	Required Changes
1	MMWAVE-SDK 3.4.0 or above required for xWR6843AoP ES2.0	Makefile **OR** CCS projects	Application code must be re-compiled with MMWAVE-SDK 3.4.0 or above to run on xWR6843AoP ES2.0 as prior SDK versions are not compatible with ES2.0. Conversely, SDK 3.4.0 is not compatible with xWR6843AoP ES1.0 devices. **Makefile: No change required if you are using SDK makefiles**, as this is automatically handled in the SDK 3.4 environment setup script: C:\ti\mmwave_sdk_03_04_xx_xx\packages\scripts\windows\setenv.bat **OR** **CCS Projectspec:** If the application is compiled using CCS projectspecs, you need to update the products property in DSS and MSS projectspecs as shown below. <property name="products" value="com.ti.rtsc.SYSBIOS:6.73.01.01;com.ti.MMWAVE_SDK:3.4.0.03;"/> **Example:** For reference CCS projects for xWR6843AoP ES2.0, see the 68xx AoP – mmWave SDK Demo available in MMWAVE Industrial Toolbox.
2	Change the value of SHMEM_ALLOC parameter in MetaImage (flashable) binary generation step.	Makefile **OR** CCS projects (mss).	The value of SHMEM_ALLOC parameter should be set to 0x00000006 for ES2.0 (it was 0x02000006 for ES1.0 device). **Makefile: No change required if you are using SDK makefiles** build, as this is automatically handled in the SDK 3.4 device specific makefiles. **OR** **CCS Projectspec:** If the application is compiled using CCS projectspecs, update the postBuildStep in MSS projectspec to replace the value 0x02000006 with 0x00000006. **Example:** For reference CCS projects for xWR6843AoP ES2.0 , see the 68xx AoP – mmWave SDK Demo available in MMWAVE Industrial Toolbox.

Table 9. xWR6843AoP ES2.0 Software - Platform Updates (continued)

No	Summary	Components Impacted	Required Changes
3	Update RadarSS firmware file name	Makefile **OR** CCS projects (mss)	The RadarSS binary for xwr6xxx devices is now called xwr6xxx_radarss_rprc.bin instead of iwr6xxx_radarss_rprc.bin. **Makefile: No change required if you are using SDK makefiles**, as this is automatically handled in the SDK 3.4 environment setup script based on the MMWAVE_SDK_DEVICE variable. **OR** **CCS Projectspec:** If the application is compiled using CCS projectspecs, replace iwr6xxx_radarss_rprc.bin with xwr6xxx_radarss_rprc.bin in the metaimage generation steps (postbuild steps) **Example:** For reference CCS projects for xWR6843AoP ES2.0 , see the 68xx AoP – mmWave SDK Demo available in MMWAVE Industrial Toolbox.
4	API update for MMWave_open SDK 3.3 and above requires a new parameter to be passed to MMWave_open	MSS/DSS start-up code	**MMWave_open**: Application must set the value of calibMonTimeUnit parameter before calling MMWave_open as shown below. The image below shows reference code updates in the SDK 68xx mmw demo **File:** mmwave_sdk_03_04_xx_xx\packages\ti\demo\xwr68xx\mmw\mss\mss_main.c **Code Snapshot: see Section 6.1**
5	API update for ADCBuf_open SDK 3.3 and above requires a new parameter to be passed to ADCBuf_open	MSS/DSS start-up code	**ADCBUF_open**: Application must set the value of socHandle in the ADCBufparams structure before calling ADCBUF_open as shown below. The image below shows reference code updates in the SDK 68xx mmw demo. **File:** mmwave_sdk_03_04_xx_xx\packages\ti\demo\utils\mmwdemo_adcconfig.c **Code Snapshot: see** Section 6.2
6	API update for CANFD_init SDK 3.3 and above requires new parameter to be passed to CANFD_init	Drivers	**CANDF_init**: Applications using CANFD driver must pass instance ID to the CANFD_init API as shown below. Only a value of 0 is supported at this time. The image below shows reference code updates in the SDK CANFD driver test. **File:** mmwave_sdk_03_04_xx_xx\packages\ti\drivers\canfd\test\xwr68xx\main.c **Code Snapshot: see Section 6.3**
7	SDK 3.3 and above removes support for Bus error interrupt from the DMA driver for xWR6843 ES2 as that interrupt is not hooked up to the device.	Drivers	Application would get an error code back from the xwr68xx driver if DMA_enable Interrupt API is called for DMA_IntType_BER. You can either remove the call to the above API or ignore the error; however you should review the DMA usage to make sure there is no invalid memory access via MSS DMA engine.
8	General note on CLI configuration file	Sensor Configuration	For applications that re-use the mmWave demo/CLI framework, ensure that the configuration commands (for example, profileCfg, chirpCfg, frameCfg, and so forth) follow the format provided in sample configuration files provided in the mmw demo directory: C:\ti\mmwave_sdk_03_04_xx_xx\packages\ti\demo\xwr64xx\mmw\profiles. for more details, see the *Configuration File Format* section in the mmwwave SDK User's Guide. Section 7

Table 9. xWR6843AoP ES2.0 Software - Platform Updates (continued)

No	Summary	Components Impacted	Required Changes
9	BSS clock un-gate required in Secondary bootloader	Secondary Bootloader	**Note: This update is not related to the main application**. It is needed only if you are using a custom secondary bootloader in your system. The Secondary Bootloader must ungate BSS clock using SOC gate/ungate API before downloading image to RadarSS/BSS memory as shown below. The image below shows reference code updates in the SDK secondary bootloader example. **File:** C:\ti\mmwave_sdk_03_04_xx_xx\packages\ti\utils \sbl\platform\sbl_xwr68xx.c **Code Snapshot: see** Section 6.4
10	SDK 3.4 mmWave layer enables all valid init time and runtime calibrations for xwr6xxx devices	MSS/DSS start-up code	Application should pass valid values for **freqLimitLow** and **freqLimitHigh** in **mmWave_Open** API and can now enable periodic calibrations in **mmWave_Start** API The image below shows reference code updates in the SDK 68xx mmw demo. **File:** mmwave_sdk_03_04_00_03\packages\ti\demo\xwr68xx\mmw\mss\mss_main.c **Code Snapshot: see Section 6.8**
11	Object detection DPC accepts antenna geometry to enable wider configurations of Tx/Rx antennas	DPCconfiguration	This field is mandatory only for HWA-based Object detection DPC when compiled to use the new AoA 2D algorithm (in the xwr64xx AoP mmw demo). For DSP-based DPC and for HWA-based DPC that uses standard AoA DPU, this field is unused. The image below shows the reference code in the SDK 64xx mmw demo. File: mmwave_sdk_03_04_00_03\packages\ti\demo\xwr64xx\mmw\main.c **Code Snapshot: see Section 6.14**
12	Object Detection HWA DPC now accepts Range FFT Scaling Parameters	DPC configuration	Range HWA-based DPU and Object detection HWA-based DPCs now allow you to set the scaling values for butterfly stages and converting from internal 24-bit to 16- bit output The image below shows the reference code in the SDK 64xx mmw demo. File: mmwave_sdk_03_04_00_03\packages\ti\demo\xwr64xx\mmw\main.c **Code Snapshot: see Section 6.9**
13	Objectdetection Range HWA DPC now allows user to specify the radar cube format	DPC Configuration	ObjDetRangeHWA DPC allows user to specify the radar cube format to allow flexibility in integrating various DSP based algorithms/processing chains Note: mmW demos support only DPIF_RADARCUBE_FORMAT_ The image below shows the reference code in the SDK 68xx mmw demo. File: mmwave_sdk_03_04_00_03\packages\ti\demo\xwr68xx\mmw\mss\mss_main.c **Code Snapshot: see Section 6.10**
14	Updates related to saving/restoring device calibration parameters (Phase shift calibration parameters)	For more details on this and other calibration related updates, see the MMWAVE-SDK 3.4.0 release notes in the Migration Notes.	

4.2.2 xWR6843AoP ES2.0 - AoA Software Updates

Figure 10 and Figure 11 compare the antenna geometries of xWR6843AOP ES1.0 and xWR6843AOP ES2.0.

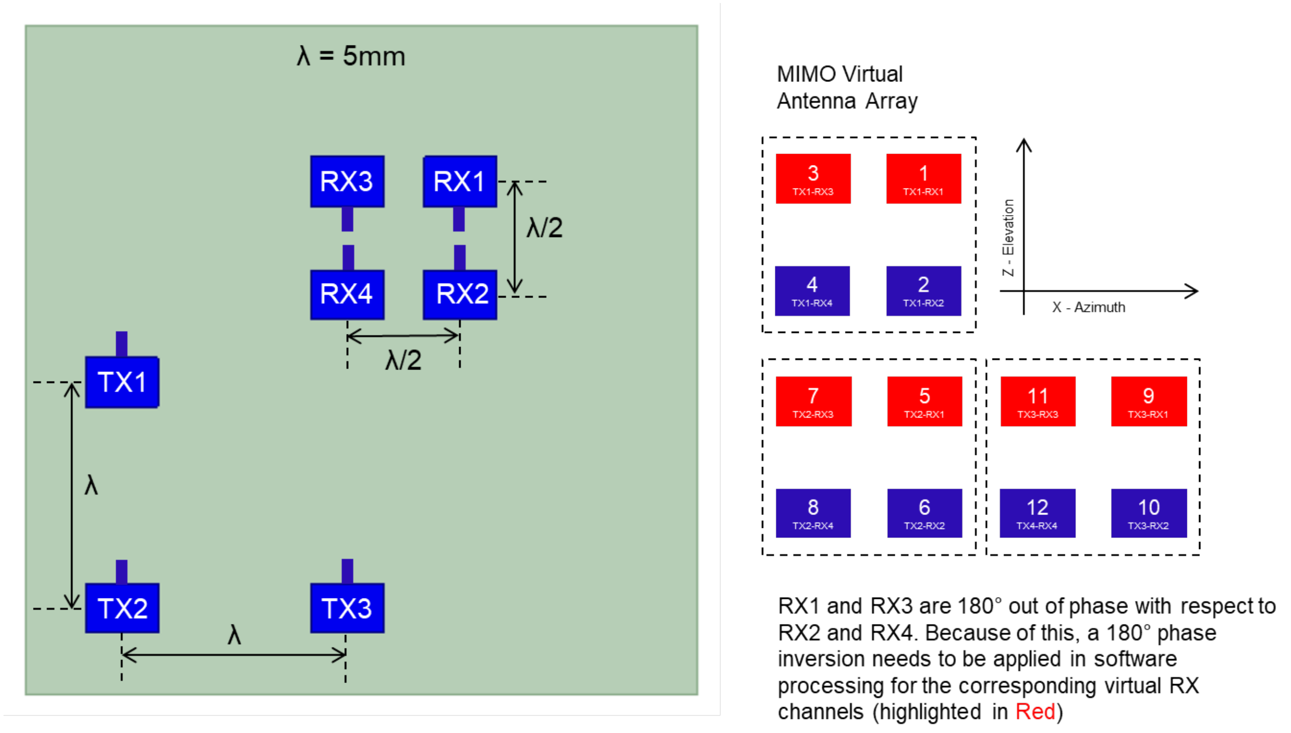

λ = 5mm
RX3
RX1
RX4
RX2
λ/2
λ/2
TX1
λ
TX2
TX3
λ
MIMO Virtual Antenna Array
3 TX1-RX3
1 TX1-RX1
4 TX1-RX4
2 TX1-RX2
Z - Elevation
X - Azimuth
7 TX2-RX3
5 TX2-RX1
11 TX3-RX3
9 TX3-RX1
8 TX2-RX4
6 TX2-RX2
12 TX4-RX4
10 TX3-RX2
RX1 and RX3 are 180° out of phase with respect to RX2 and RX4. Because of this, a 180° phase inversion needs to be applied in software processing for the corresponding virtual RX channels (highlighted in Red)

Figure 10. xWR6843AoP ES1.0 Antenna Geometry and Resulting MIMO Virtual Antenna Array

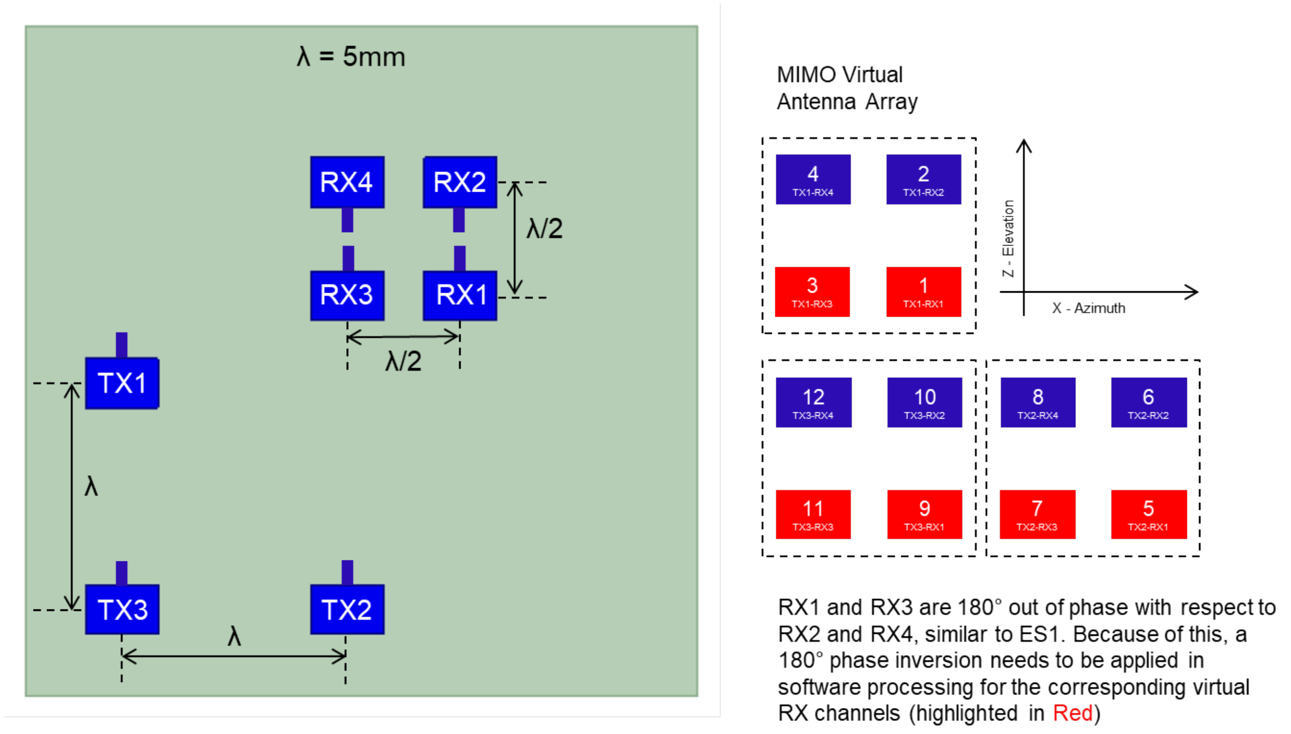

λ = 5mm
RX4
RX2
RX3
RX1
λ/2
λ/2
TX1
λ
TX3
TX2
λ
MIMO Virtual Antenna Array
4 TX1-RX4
2 TX1-RX2
3 TX1-RX3
1 TX1-RX1
Z - Elevation
X - Azimuth
12 TX3-RX4
10 TX3-RX2
8 TX2-RX4
6 TX2-RX2
11 TX3-RX3
9 TX3-RX1
7 TX2-RX3
5 TX2-RX1
RX1 and RX3 are 180° out of phase with respect to RX2 and RX4, similar to ES1. Because of this, a 180° phase inversion needs to be applied in software processing for the corresponding virtual RX channels (highlighted in Red)

Figure 11. xWR6843AoP ES2.0 Antenna Geometry and Resulting MIMO Virtual Antenna Array

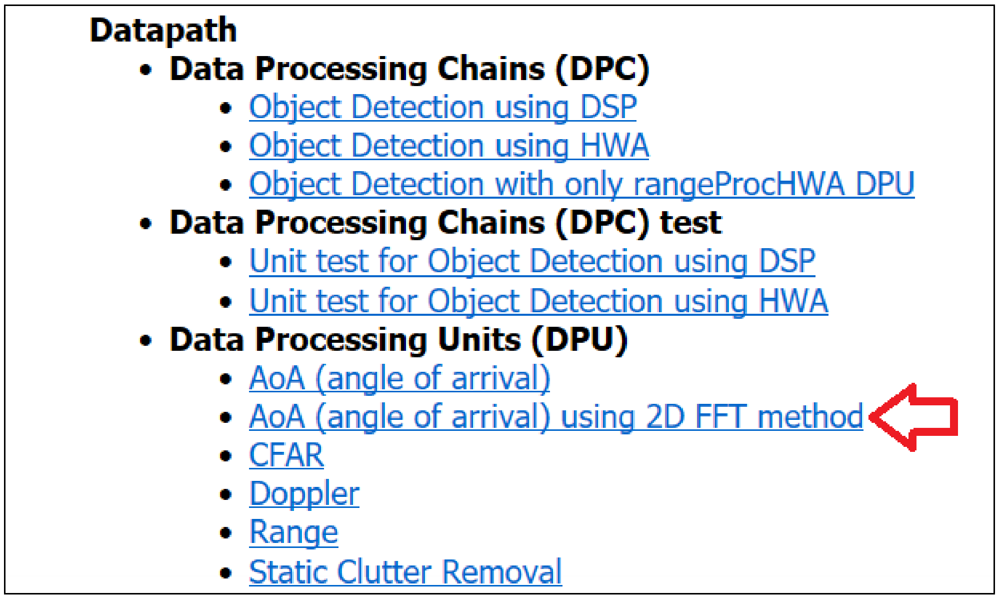
The key antenna updates in xWR6843AOP ES2, as shown above are:

- **RX Antennas:** RX1 and RX2 are swapped on xWR6843AOP ES2. Similarly RX3 and RX4 are swapped
- **TX Antennas:** TX2 and TX3 are swapped on xWR6843AOP ES2.
- **Line Feed:** The RX line feeds on xWR6843AOP ES2 are same as on ES1 i.e. RX1 and RX2 are fed from opposite ends, which results in a 180° phase difference between RX1 and RX2. Similarly, RX3 and RX4 are out of phase by 180°. To compensate for the opposite line feeds, a 180º phase inversion needs to be applied in software processing for the corresponding virtual channels as shown in Figure 11.

MMWAVE-SDK 3.2.0.6 and MMWAVE-SDK 3.4 include the AoA2dProc DPU which performs Angle of Arrival processing for the xWR6843 AoP antenna array using the Hardware Accelerator. The AoA2dProc DPU (Datapath Processing Unit) is used in the xWR64xx AoP mmw demo for angle of arrival processing.

To understand the AoA updates needed for xWR6843AOP ES2, it is recommended to understand the antenna geometry concept defined in AoA2dProc DPU.

1. Navigate to C:\ti\mmwave_sdk_03_04_xx_xx\docs and and open the file mmwave_sdk_module_documentation.html in a browser.
2. Click on the AoA using 2D FFT method link as highlighted in the picture below:



Datapath
- **Data Processing Chains (DPC)**
  - Object Detection using DSP
  - Object Detection using HWA
  - Object Detection with only rangeProcHWA DPU
- **Data Processing Chains (DPC) test**
  - Unit test for Object Detection using DSP
  - Unit test for Object Detection using HWA
- **Data Processing Units (DPU)**
  - AoA (angle of arrival)
  - AoA (angle of arrival) using 2D FFT method
  - CFAR
  - Doppler
  - Range
  - Static Clutter Removal

Figure 12. AoA2dProc HTML Documentation

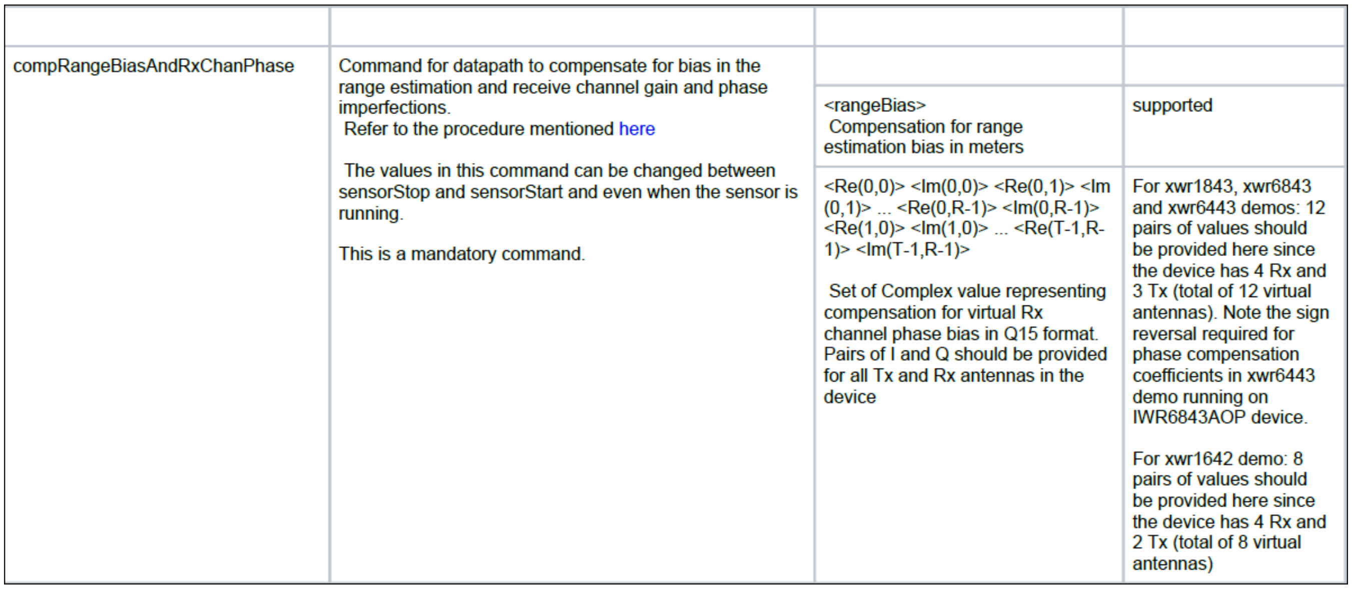
3. Scroll down to the section named Antenna Geometry Definition, which explains how the generic antenna geometry structure is defined and used by the HWA AoA2dProc DPU code. The antenna geometry for a specific antenna (for example, xWR6843AoP ES2.0) is defined in the corresponding C structure in **mmwave_sdk_03_04_xx_xx\packages\ti\board\antenna_geometry.c**.

The image below shows the antenna geometry structure update for xWR6843AoP ES2.0 as compared to xWR6843AoP ES1.0 in MMWAVE-SDK 3.2.0.6.
Code Snapshot: see Section 6.13

The antenna geometry structure is passed to the Object Detection DPC during initialization in **mmwave_sdk_03_04_xx_xx\packages\ti\demo\xwr64xx\mmw\main.c**
Code Snapshot: see Section 6.14

RX Channel Phase Compensation: To compensate for the opposite line feeds as shown in Section 6.12, a 180° phase inversion is applied to the corresponding RX channels (including virtual channels) using the compRangeBiasAndRxChanPhase CLI command available in the mmw demo.

Figure 13, from the MMWAVE-SDK user's guide, explains the structure of this command.



compRangeBiasAndRxChanPhase	Command for datapath to compensate for bias in the range estimation and receive channel gain and phase imperfections. Refer to the procedure mentioned here The values in this command can be changed between sensorStop and sensorStart and even when the sensor is running. This is a mandatory command.		
		<rangeBias> Compensation for range estimation bias in meters	supported
		<Re(0,0)> <Im(0,0)> <Re(0,1)> <Im(0,1)> ... <Re(0,R-1)> <Im(0,R-1)> <Re(1,0)> <Im(1,0)> ... <Re(T-1,R-1)> <Im(T-1,R-1)> Set of Complex value representing compensation for virtual Rx channel phase bias in Q15 format. Pairs of I and Q should be provided for all Tx and Rx antennas in the device	For xwr1843, xwr6843 and xwr6443 demos: 12 pairs of values should be provided here since the device has 4 Rx and 3 Tx (total of 12 virtual antennas). Note the sign reversal required for phase compensation coefficients in xwr6443 demo running on IWR6843AOP device. For xwr1642 demo: 8 pairs of values should be provided here since the device has 4 Rx and 2 Tx (total of 8 virtual antennas)

Figure 13. RX Channel Phase Compensation: CompRangeBiasAndRxChanPhase CLI Command

To understand the CompRangeBiasAndRxChanPhase values configured in the example AoP profile configuration provided in MMWAVE-SDK, see Section 6.15.

5 Helpful Resources

The following resources provide example source code, makefile and CCS projects for the xWR6843 ES2.0 and the xWR1843 devices.

Resource Name	File-System Path / Web URL	Content Reference
MMWAVE-SDK 3.3 mmw demo	68xx - C:\ti\mmwave_sdk_03_03_xx_xx\packages\ti\demo\xwr68xx\mmw 18xx - C:\ti\mmwave_sdk_03_03_xx_xx\packages\ti\demo\xwr18xx\mmw	Source code, Makefiles, Configuration files (.cfg)
MMWAVE-SDK 3.4 mmw demo	68xx - C:\ti\mmwave_sdk_03_04_xx_xx\packages\ti\demo\xwr68xx\mmw 64/68xxAoP C:\ti\mmwave_sdk_03_04_xx_xx\packages\ti\demo\xwr64xx\mmw	
MMWAVE Industrial Toolbox	MMWAVE Industrial Toolbox 68xx ISK – mmWave SDK Demo – DSP Version 64/68xx AoP - mmWave SDK Demo 68xx AoP 18xx – mmWave SDK Demo And various other demos included in Industrial Toolbox	Reference CCS Projectspecs for mmWave SDK mmw demos and other application specific demos.

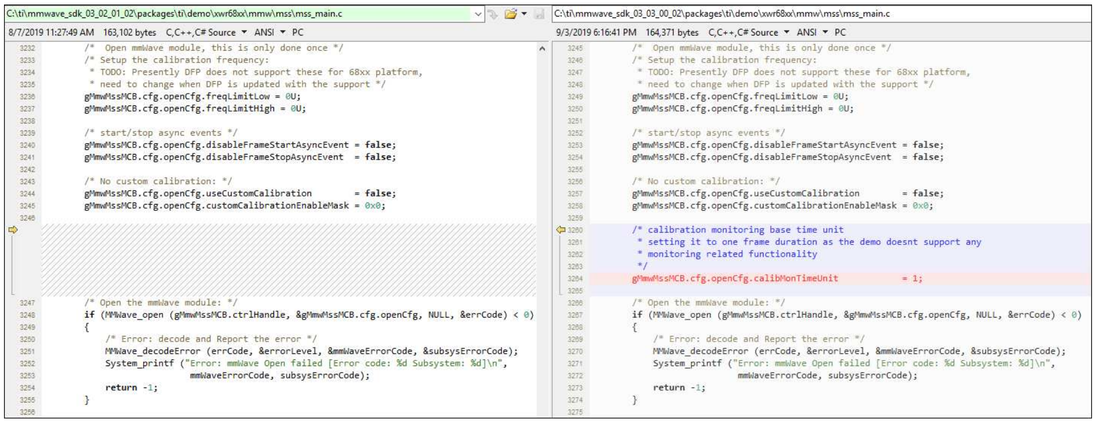
6 Code Snapshots

This section provides code snapshots for the migrations notes presented in the previous sections.

6.1 *SDK 3.3 API Change for MMWave_open*

MMWAVE-SDK 3.2.1 vs MMWAVE-SDK 3.3.0

File: mmwave_sdk_03_03_00_0x\packages\ti\demo\xwr68xx\mmw\mss_main.c



C:\ti\mmwave_sdk_03_02_01_02\packages\ti\demo\xwr68xx\mmw\mss\mss_main.c

8/7/2019 11:27:49 AM 163,102 bytes C,C++,C# Source ANSI PC

```
        /*  Open mmWave module, this is only done once */
        /* Setup the calibration frequency:
         * TODO: Presently DFP does not support these for 68xx platform,
         * need to change when DFP is updated with the support */
        gMmwMssMCB.cfg.openCfg.freqLimitLow = 0U;
        gMmwMssMCB.cfg.openCfg.freqLimitHigh = 0U;

        /* start/stop async events */
        gMmwMssMCB.cfg.openCfg.disableFrameStartAsyncEvent = false;
        gMmwMssMCB.cfg.openCfg.disableFrameStopAsyncEvent  = false;

        /* No custom calibration: */
        gMmwMssMCB.cfg.openCfg.useCustomCalibration        = false;
        gMmwMssMCB.cfg.openCfg.customCalibrationEnableMask = 0x0;

        /* Open the mmWave module: */
        if (MMWave_open (gMmwMssMCB.ctrlHandle, &gMmwMssMCB.cfg.openCfg, NULL, &errCode) < 0)
        {
            /* Error: decode and Report the error */
            MMWave_decodeError (errCode, &errorLevel, &mmWaveErrorCode, &subsysErrorCode);
            System_printf ("Error: mmWave Open failed [Error code: %d Subsystem: %d]\n",
                            mmWaveErrorCode, subsysErrorCode);
            return -1;
        }
```

C:\ti\mmwave_sdk_03_03_00_02\packages\ti\demo\xwr68xx\mmw\mss\mss_main.c

9/3/2019 6:16:41 PM 164,371 bytes C,C++,C# Source ANSI PC

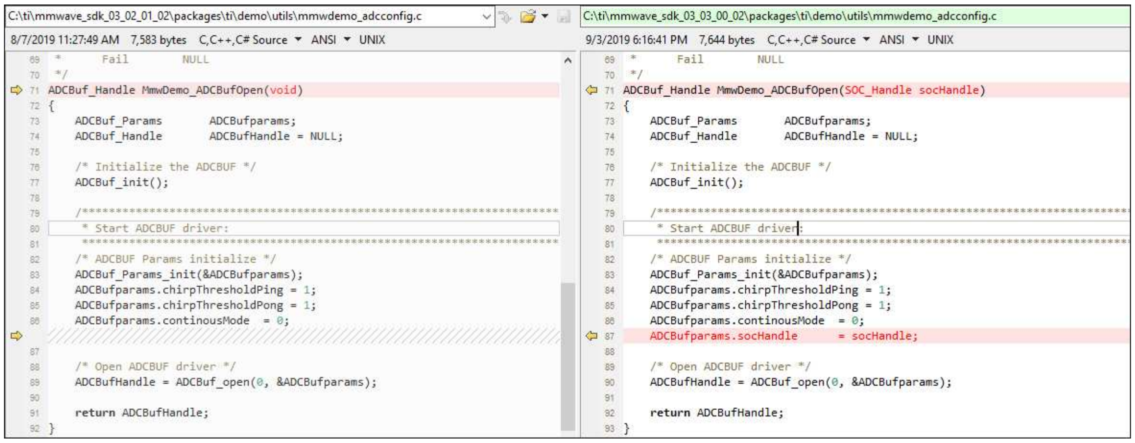
```
        /*  Open mmWave module, this is only done once */
        /* Setup the calibration frequency:
         * TODO: Presently DFP does not support these for 68xx platform,
         * need to change when DFP is updated with the support */
        gMmwMssMCB.cfg.openCfg.freqLimitLow = 0U;
        gMmwMssMCB.cfg.openCfg.freqLimitHigh = 0U;

        /* start/stop async events */
        gMmwMssMCB.cfg.openCfg.disableFrameStartAsyncEvent = false;
        gMmwMssMCB.cfg.openCfg.disableFrameStopAsyncEvent  = false;

        /* No custom calibration: */
        gMmwMssMCB.cfg.openCfg.useCustomCalibration        = false;
        gMmwMssMCB.cfg.openCfg.customCalibrationEnableMask = 0x0;

        /* calibration monitoring base time unit
         * setting it to one frame duration as the demo doesnt support any
         * monitoring related functionality
         */
        gMmwMssMCB.cfg.openCfg.calibMonTimeUnit            = 1;

        /* Open the mmWave module: */
        if (MMWave_open (gMmwMssMCB.ctrlHandle, &gMmwMssMCB.cfg.openCfg, NULL, &errCode) < 0)
        {
            /* Error: decode and Report the error */
            MMWave_decodeError (errCode, &errorLevel, &mmWaveErrorCode, &subsysErrorCode);
            System_printf ("Error: mmWave Open failed [Error code: %d Subsystem: %d]\n",
                            mmWaveErrorCode, subsysErrorCode);
            return -1;
        }
```

Figure 14. SDK 3.3 API Change for MMWave_open

6.2 *SDK 3.3 API Change for ADCBuf_open*

MMWAVE-SDK 3.2.1 vs MMWAVE-SDK 3.3.0

File: mmwave_sdk_03_03_00_0x\packages\ti\demo\xwr68xx\mmw\mss_main.c



C:\ti\mmwave_sdk_03_02_01_02\packages\ti\demo\utils\mmwdemo_adcconfig.c

8/7/2019 11:27:49 AM 7,583 bytes C,C++,C# Source ANSI UNIX

```
 *      Fail        NULL
 */
ADCBuf_Handle MmwDemo_ADCBufOpen(void)
{
    ADCBuf_Params       ADCBufparams;
    ADCBuf_Handle       ADCBufHandle = NULL;

    /* Initialize the ADCBUF */
    ADCBuf_init();

    /*****************************************************************************
     * Start ADCBUF driver:
     *****************************************************************************
    /* ADCBUF Params initialize */
    ADCBuf_Params_init(&ADCBufparams);
    ADCBufparams.chirpThresholdPing = 1;
    ADCBufparams.chirpThresholdPong = 1;
    ADCBufparams.continousMode  = 0;

    /* Open ADCBUF driver */
    ADCBufHandle = ADCBuf_open(0, &ADCBufparams);

    return ADCBufHandle;
}
```

C:\ti\mmwave_sdk_03_03_00_02\packages\ti\demo\utils\mmwdemo_adcconfig.c

9/3/2019 6:16:41 PM 7,644 bytes C,C++,C# Source ANSI UNIX

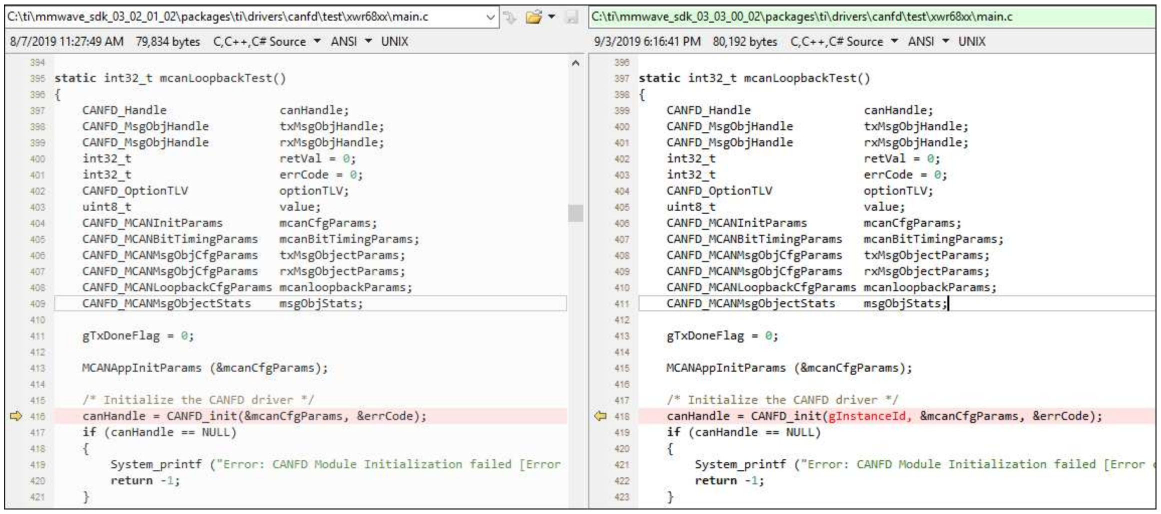
```
 *      Fail        NULL
 */
ADCBuf_Handle MmwDemo_ADCBufOpen(SOC_Handle socHandle)
{
    ADCBuf_Params       ADCBufparams;
    ADCBuf_Handle       ADCBufHandle = NULL;

    /* Initialize the ADCBUF */
    ADCBuf_init();

    /*****************************************************************************
     * Start ADCBUF driver:
     *****************************************************************************
    /* ADCBUF Params initialize */
    ADCBuf_Params_init(&ADCBufparams);
    ADCBufparams.chirpThresholdPing = 1;
    ADCBufparams.chirpThresholdPong = 1;
    ADCBufparams.continousMode  = 0;
    ADCBufparams.socHandle      = socHandle;

    /* Open ADCBUF driver */
    ADCBufHandle = ADCBuf_open(0, &ADCBufparams);

    return ADCBufHandle;
}
```

Figure 15. SDK 3.3 API Change for ADCBuf_open

6.3 SDK 3.3 API Change for CANFD_init

MMWAVE-SDK 3.2.1 vs MMWAVE-SDK 3.3.0

File: mmwave_sdk_03_03_00_0x\ packages\ti\drivers\canfd\test\xwr68xx\main.c



C:\ti\mmwave_sdk_03_02_01_02\packages\ti\drivers\canfd\test\xwr68xx\main.c
8/7/2019 11:27:49 AM 79,834 bytes C,C++,C# Source ANSI UNIX

```
static int32_t mcanLoopbackTest()
{
    CANFD_Handle                canHandle;
    CANFD_MsgObjHandle          txMsgObjHandle;
    CANFD_MsgObjHandle          rxMsgObjHandle;
    int32_t                     retVal = 0;
    int32_t                     errCode = 0;
    CANFD_OptionTLV             optionTLV;
    uint8_t                     value;
    CANFD_MCANInitParams        mcanCfgParams;
    CANFD_MCANBitTimingParams   mcanBitTimingParams;
    CANFD_MCANMsgObjCfgParams   txMsgObjectParams;
    CANFD_MCANMsgObjCfgParams   rxMsgObjectParams;
    CANFD_MCANLoopbackCfgParams mcanloopbackParams;
    CANFD_MCANMsgObjectStats    msgObjStats;

    gTxDoneFlag = 0;

    MCANAppInitParams (&mcanCfgParams);

    /* Initialize the CANFD driver */
    canHandle = CANFD_init(&mcanCfgParams, &errCode);
    if (canHandle == NULL)
    {
        System_printf ("Error: CANFD Module Initialization failed [Error
        return -1;
    }
```

C:\ti\mmwave_sdk_03_03_00_02\packages\ti\drivers\canfd\test\xwr68xx\main.c
9/3/2019 6:16:41 PM 80,192 bytes C,C++,C# Source ANSI UNIX

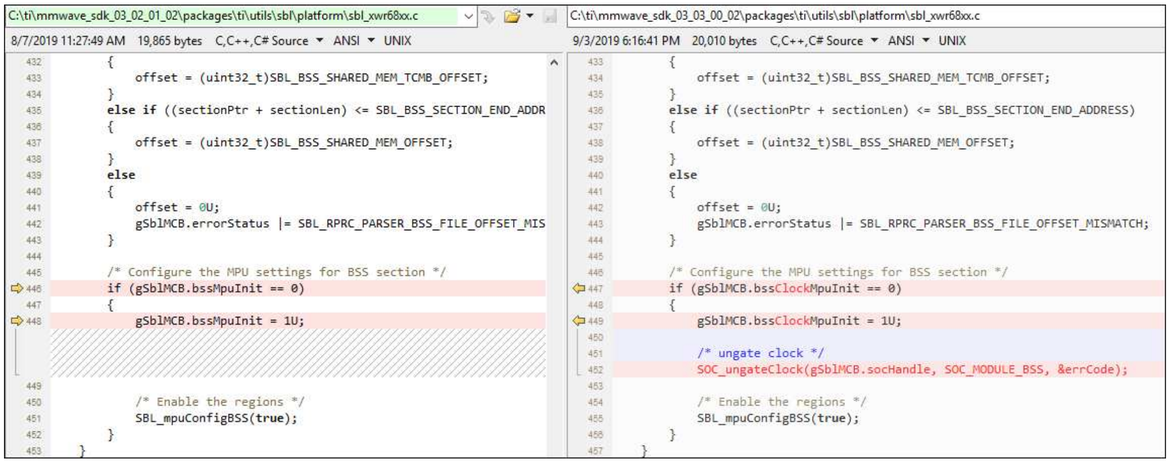
```
static int32_t mcanLoopbackTest()
{
    CANFD_Handle                canHandle;
    CANFD_MsgObjHandle          txMsgObjHandle;
    CANFD_MsgObjHandle          rxMsgObjHandle;
    int32_t                     retVal = 0;
    int32_t                     errCode = 0;
    CANFD_OptionTLV             optionTLV;
    uint8_t                     value;
    CANFD_MCANInitParams        mcanCfgParams;
    CANFD_MCANBitTimingParams   mcanBitTimingParams;
    CANFD_MCANMsgObjCfgParams   txMsgObjectParams;
    CANFD_MCANMsgObjCfgParams   rxMsgObjectParams;
    CANFD_MCANLoopbackCfgParams mcanloopbackParams;
    CANFD_MCANMsgObjectStats    msgObjStats;

    gTxDoneFlag = 0;

    MCANAppInitParams (&mcanCfgParams);

    /* Initialize the CANFD driver */
    canHandle = CANFD_init(gInstanceId, &mcanCfgParams, &errCode);
    if (canHandle == NULL)
    {
        System_printf ("Error: CANFD Module Initialization failed [Error
        return -1;
    }
```

Figure 16. SDK 3.3 API Change for CANFD_init

6.4 SDK 3.3 68xx Secondary Bootloader Update

MMWAVE-SDK 3.2.1 vs MMWAVE-SDK 3.3.0

File: mmwave_sdk_03_03_00_02\packages\ti\utils\sbl\platform\sbl_xwr68xx.c



C:\ti\mmwave_sdk_03_02_01_02\packages\ti\utils\sbl\platform\sbl_xwr68xx.c
8/7/2019 11:27:49 AM 19,865 bytes C,C++,C# Source ANSI UNIX

```
        {
            offset = (uint32_t)SBL_BSS_SHARED_MEM_TCMB_OFFSET;
        }
        else if ((sectionPtr + sectionLen) <= SBL_BSS_SECTION_END_ADDR
        {
            offset = (uint32_t)SBL_BSS_SHARED_MEM_OFFSET;
        }
        else
        {
            offset = 0U;
            gSblMCB.errorStatus |= SBL_RPRC_PARSER_BSS_FILE_OFFSET_MIS
        }

        /* Configure the MPU settings for BSS section */
        if (gSblMCB.bssMpuInit == 0)
        {
            gSblMCB.bssMpuInit = 1U;


            /* Enable the regions */
            SBL_mpuConfigBSS(true);
        }
    }
```

C:\ti\mmwave_sdk_03_03_00_02\packages\ti\utils\sbl\platform\sbl_xwr68xx.c
9/3/2019 6:16:41 PM 20,010 bytes C,C++,C# Source ANSI UNIX

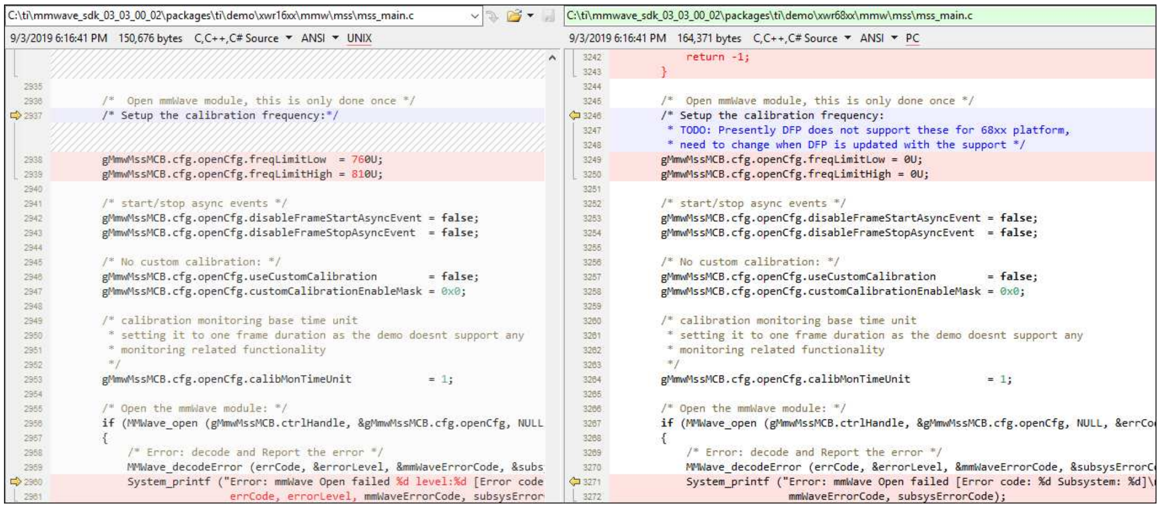
```
        {
            offset = (uint32_t)SBL_BSS_SHARED_MEM_TCMB_OFFSET;
        }
        else if ((sectionPtr + sectionLen) <= SBL_BSS_SECTION_END_ADDRESS)
        {
            offset = (uint32_t)SBL_BSS_SHARED_MEM_OFFSET;
        }
        else
        {
            offset = 0U;
            gSblMCB.errorStatus |= SBL_RPRC_PARSER_BSS_FILE_OFFSET_MISMATCH;
        }

        /* Configure the MPU settings for BSS section */
        if (gSblMCB.bssClockMpuInit == 0)
        {
            gSblMCB.bssClockMpuInit = 1U;

            /* ungate clock */
            SOC_ungateClock(gSblMCB.socHandle, SOC_MODULE_BSS, &errCode);

            /* Enable the regions */
            SBL_mpuConfigBSS(true);
        }
    }
```

Figure 17. SDK 3.3 68xx Secondary Bootloader Update

6.5 SDK 3.3 16xx vs 68xx: Calibration Frequency Update

MMWAVE-SDK 3.3.0 mmw demo (16xx vs 68xx)

File: mmwave_sdk_03_03_00_0x\packages\ti\demo\xwr68xx\mmw\mss_main.c



C:\ti\mmwave_sdk_03_03_00_02\packages\ti\demo\xwr16xx\mmw\mss\mss_main.c

9/3/2019 6:16:41 PM 150,676 bytes C,C++,C# Source ANSI UNIX

```
        /*  Open mmWave module, this is only done once */
        /* Setup the calibration frequency:*/

        gMmwMssMCB.cfg.openCfg.freqLimitLow  = 760U;
        gMmwMssMCB.cfg.openCfg.freqLimitHigh = 810U;

        /* start/stop async events */
        gMmwMssMCB.cfg.openCfg.disableFrameStartAsyncEvent = false;
        gMmwMssMCB.cfg.openCfg.disableFrameStopAsyncEvent  = false;

        /* No custom calibration: */
        gMmwMssMCB.cfg.openCfg.useCustomCalibration        = false;
        gMmwMssMCB.cfg.openCfg.customCalibrationEnableMask = 0x0;

        /* calibration monitoring base time unit
         * setting it to one frame duration as the demo doesnt support any
         * monitoring related functionality
         */
        gMmwMssMCB.cfg.openCfg.calibMonTimeUnit            = 1;

        /* Open the mmWave module: */
        if (MMWave_open (gMmwMssMCB.ctrlHandle, &gMmwMssMCB.cfg.openCfg, NULL
        {
            /* Error: decode and Report the error */
            MMWave_decodeError (errCode, &errorLevel, &mmWaveErrorCode, &subs
            System_printf ("Error: mmWave Open failed %d level:%d [Error code
                                errCode, errorLevel, mmWaveErrorCode, subsysError
```

C:\ti\mmwave_sdk_03_03_00_02\packages\ti\demo\xwr68xx\mmw\mss\mss_main.c

9/3/2019 6:16:41 PM 164,371 bytes C,C++,C# Source ANSI PC

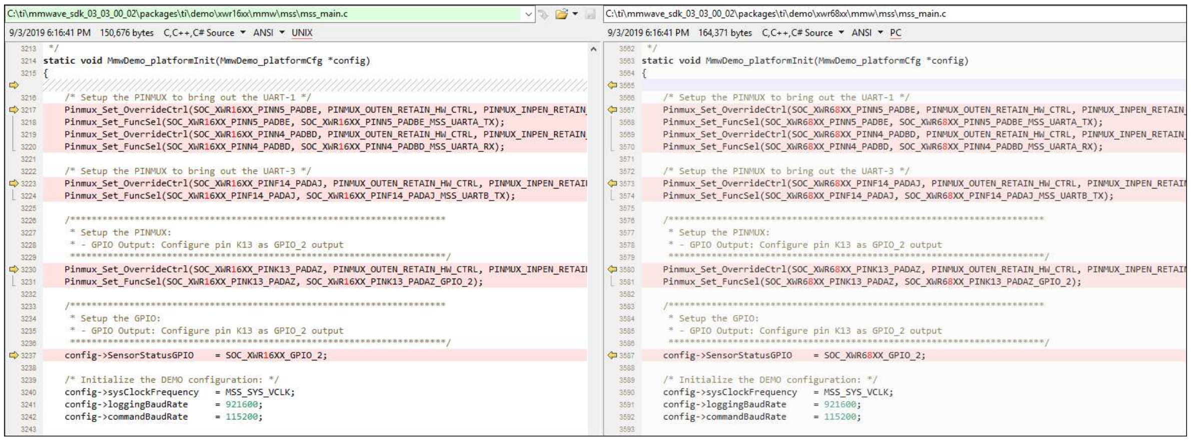
```
            return -1;
        }

        /*  Open mmWave module, this is only done once */
        /* Setup the calibration frequency:
         * TODO: Presently DFP does not support these for 68xx platform,
         * need to change when DFP is updated with the support */
        gMmwMssMCB.cfg.openCfg.freqLimitLow = 0U;
        gMmwMssMCB.cfg.openCfg.freqLimitHigh = 0U;

        /* start/stop async events */
        gMmwMssMCB.cfg.openCfg.disableFrameStartAsyncEvent = false;
        gMmwMssMCB.cfg.openCfg.disableFrameStopAsyncEvent  = false;

        /* No custom calibration: */
        gMmwMssMCB.cfg.openCfg.useCustomCalibration        = false;
        gMmwMssMCB.cfg.openCfg.customCalibrationEnableMask = 0x0;

        /* calibration monitoring base time unit
         * setting it to one frame duration as the demo doesnt support any
         * monitoring related functionality
         */
        gMmwMssMCB.cfg.openCfg.calibMonTimeUnit            = 1;

        /* Open the mmWave module: */
        if (MMWave_open (gMmwMssMCB.ctrlHandle, &gMmwMssMCB.cfg.openCfg, NULL, &errCo
        {
            /* Error: decode and Report the error */
            MMWave_decodeError (errCode, &errorLevel, &mmWaveErrorCode, &subsysErrorC
            System_printf ("Error: mmWave Open failed [Error code: %d Subsystem: %d]\
                                mmWaveErrorCode, subsysErrorCode);
```

Figure 18. SDK 3.3 16xx vs 68xx: Calibration Frequency Update

6.6 SDK 3.3 16xx vs 68xx: SoC Definition Updates

MMWAVE-SDK 3.3.0 mmw demo (16xx vs 68xx)

File: mmwave_sdk_03_03_xx_xx\packages\ti\demo\xwr68xx\mmw\mss_main.c



C:\ti\mmwave_sdk_03_03_00_02\packages\ti\demo\xwr16xx\mmw\mss\mss_main.c

9/3/2019 6:16:41 PM 150,676 bytes C,C++,C# Source ANSI UNIX

```
 */
static void MmwDemo_platformInit(MmwDemo_platformCfg *config)
{

    /* Setup the PINMUX to bring out the UART-1 */
    Pinmux_Set_OverrideCtrl(SOC_XWR16XX_PINN5_PADBE, PINMUX_OUTEN_RETAIN_HW_CTRL, PINMUX_INPEN_RETAIN
    Pinmux_Set_FuncSel(SOC_XWR16XX_PINN5_PADBE, SOC_XWR16XX_PINN5_PADBE_MSS_UARTA_TX);
    Pinmux_Set_OverrideCtrl(SOC_XWR16XX_PINN4_PADBD, PINMUX_OUTEN_RETAIN_HW_CTRL, PINMUX_INPEN_RETAIN
    Pinmux_Set_FuncSel(SOC_XWR16XX_PINN4_PADBD, SOC_XWR16XX_PINN4_PADBD_MSS_UARTA_RX);

    /* Setup the PINMUX to bring out the UART-3 */
    Pinmux_Set_OverrideCtrl(SOC_XWR16XX_PINF14_PADAJ, PINMUX_OUTEN_RETAIN_HW_CTRL, PINMUX_INPEN_RETAI
    Pinmux_Set_FuncSel(SOC_XWR16XX_PINF14_PADAJ, SOC_XWR16XX_PINF14_PADAJ_MSS_UARTB_TX);

    /**********************************************************************
     * Setup the PINMUX:
     * - GPIO Output: Configure pin K13 as GPIO_2 output
     **********************************************************************/
    Pinmux_Set_OverrideCtrl(SOC_XWR16XX_PINK13_PADAZ, PINMUX_OUTEN_RETAIN_HW_CTRL, PINMUX_INPEN_RETAI
    Pinmux_Set_FuncSel(SOC_XWR16XX_PINK13_PADAZ, SOC_XWR16XX_PINK13_PADAZ_GPIO_2);

    /**********************************************************************
     * Setup the GPIO:
     * - GPIO Output: Configure pin K13 as GPIO_2 output
     **********************************************************************/
    config->SensorStatusGPIO    = SOC_XWR16XX_GPIO_2;

    /* Initialize the DEMO configuration: */
    config->sysClockFrequency   = MSS_SYS_VCLK;
    config->loggingBaudRate     = 921600;
    config->commandBaudRate     = 115200;

```

C:\ti\mmwave_sdk_03_03_00_02\packages\ti\demo\xwr68xx\mmw\mss\mss_main.c

9/3/2019 6:16:41 PM 164,371 bytes C,C++,C# Source ANSI PC

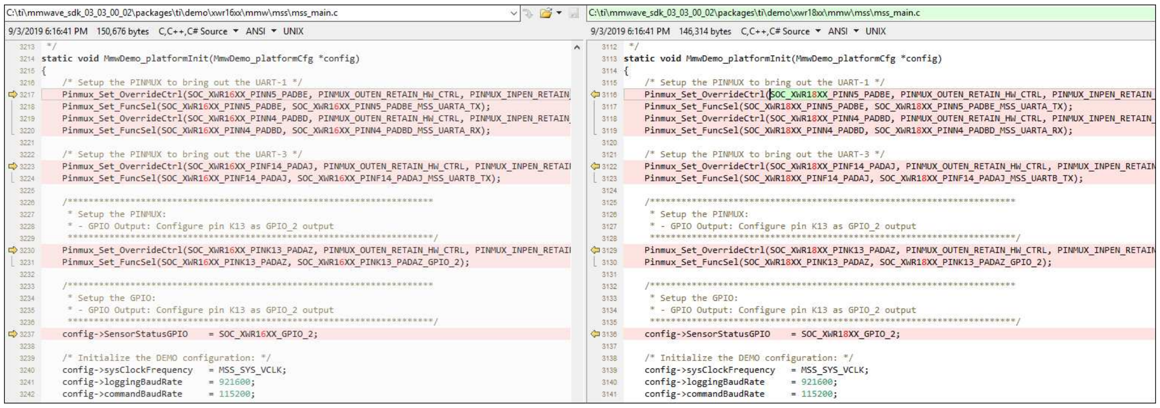
```
 */
static void MmwDemo_platformInit(MmwDemo_platformCfg *config)
{

    /* Setup the PINMUX to bring out the UART-1 */
    Pinmux_Set_OverrideCtrl(SOC_XWR68XX_PINN5_PADBE, PINMUX_OUTEN_RETAIN_HW_CTRL, PINMUX_INPEN_RETAIN
    Pinmux_Set_FuncSel(SOC_XWR68XX_PINN5_PADBE, SOC_XWR68XX_PINN5_PADBE_MSS_UARTA_TX);
    Pinmux_Set_OverrideCtrl(SOC_XWR68XX_PINN4_PADBD, PINMUX_OUTEN_RETAIN_HW_CTRL, PINMUX_INPEN_RETAIN
    Pinmux_Set_FuncSel(SOC_XWR68XX_PINN4_PADBD, SOC_XWR68XX_PINN4_PADBD_MSS_UARTA_RX);

    /* Setup the PINMUX to bring out the UART-3 */
    Pinmux_Set_OverrideCtrl(SOC_XWR68XX_PINF14_PADAJ, PINMUX_OUTEN_RETAIN_HW_CTRL, PINMUX_INPEN_RETAI
    Pinmux_Set_FuncSel(SOC_XWR68XX_PINF14_PADAJ, SOC_XWR68XX_PINF14_PADAJ_MSS_UARTB_TX);

    /**********************************************************************
     * Setup the PINMUX:
     * - GPIO Output: Configure pin K13 as GPIO_2 output
     **********************************************************************/
    Pinmux_Set_OverrideCtrl(SOC_XWR68XX_PINK13_PADAZ, PINMUX_OUTEN_RETAIN_HW_CTRL, PINMUX_INPEN_RETAI
    Pinmux_Set_FuncSel(SOC_XWR68XX_PINK13_PADAZ, SOC_XWR68XX_PINK13_PADAZ_GPIO_2);

    /**********************************************************************
     * Setup the GPIO:
     * - GPIO Output: Configure pin K13 as GPIO_2 output
     **********************************************************************/
    config->SensorStatusGPIO    = SOC_XWR68XX_GPIO_2;

    /* Initialize the DEMO configuration: */
    config->sysClockFrequency   = MSS_SYS_VCLK;
    config->loggingBaudRate     = 921600;
    config->commandBaudRate     = 115200;

```

Figure 19. SDK 3.3 16xx vs 68xx: SoC Definition Updates

6.7 SDK 3.3 16xx vs 18xx: SoC Definition Updates

MMWAVE-SDK 3.3.0 mmw demo (16xx vs 18xx)

File: mmwave_sdk_03_03_xx_xx\packages\ti\demo\xwr18xx\mmw\mss_main.c



C:\ti\mmwave_sdk_03_03_00_02\packages\ti\demo\xwr16xx\mmw\mss\mss_main.c
9/3/2019 6:16:41 PM 150,676 bytes C,C++,C# Source ANSI UNIX

```
 */
static void MmwDemo_platformInit(MmwDemo_platformCfg *config)
{
    /* Setup the PINMUX to bring out the UART-1 */
    Pinmux_Set_OverrideCtrl(SOC_XWR16XX_PINN5_PADBE, PINMUX_OUTEN_RETAIN_HW_CTRL, PINMUX_INPEN_RETAIN
    Pinmux_Set_FuncSel(SOC_XWR16XX_PINN5_PADBE, SOC_XWR16XX_PINN5_PADBE_MSS_UARTA_TX);
    Pinmux_Set_OverrideCtrl(SOC_XWR16XX_PINN4_PADBD, PINMUX_OUTEN_RETAIN_HW_CTRL, PINMUX_INPEN_RETAIN
    Pinmux_Set_FuncSel(SOC_XWR16XX_PINN4_PADBD, SOC_XWR16XX_PINN4_PADBD_MSS_UARTA_RX);

    /* Setup the PINMUX to bring out the UART-3 */
    Pinmux_Set_OverrideCtrl(SOC_XWR16XX_PINF14_PADAJ, PINMUX_OUTEN_RETAIN_HW_CTRL, PINMUX_INPEN_RETAI
    Pinmux_Set_FuncSel(SOC_XWR16XX_PINF14_PADAJ, SOC_XWR16XX_PINF14_PADAJ_MSS_UARTB_TX);

    /**************************************************************************
     * Setup the PINMUX:
     * - GPIO Output: Configure pin K13 as GPIO_2 output
     **************************************************************************/
    Pinmux_Set_OverrideCtrl(SOC_XWR16XX_PINK13_PADAZ, PINMUX_OUTEN_RETAIN_HW_CTRL, PINMUX_INPEN_RETAI
    Pinmux_Set_FuncSel(SOC_XWR16XX_PINK13_PADAZ, SOC_XWR16XX_PINK13_PADAZ_GPIO_2);

    /**************************************************************************
     * Setup the GPIO:
     * - GPIO Output: Configure pin K13 as GPIO_2 output
     **************************************************************************/
    config->SensorStatusGPIO    = SOC_XWR16XX_GPIO_2;

    /* Initialize the DEMO configuration: */
    config->sysClockFrequency   = MSS_SYS_VCLK;
    config->loggingBaudRate     = 921600;
    config->commandBaudRate     = 115200;
```

C:\ti\mmwave_sdk_03_03_00_02\packages\ti\demo\xwr18xx\mmw\mss\mss_main.c
9/3/2019 6:16:41 PM 146,314 bytes C,C++,C# Source ANSI UNIX

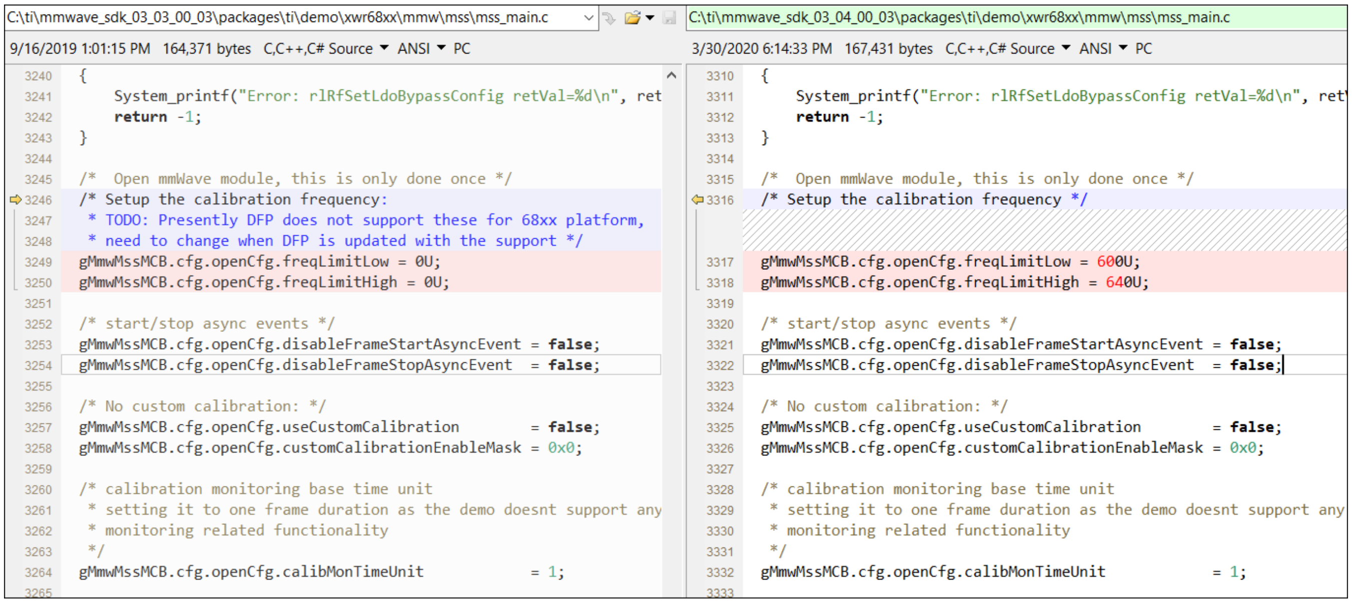
```
 */
static void MmwDemo_platformInit(MmwDemo_platformCfg *config)
{
    /* Setup the PINMUX to bring out the UART-1 */
    Pinmux_Set_OverrideCtrl(SOC_XWR18XX_PINN5_PADBE, PINMUX_OUTEN_RETAIN_HW_CTRL, PINMUX_INPEN_RETAIN_
    Pinmux_Set_FuncSel(SOC_XWR18XX_PINN5_PADBE, SOC_XWR18XX_PINN5_PADBE_MSS_UARTA_TX);
    Pinmux_Set_OverrideCtrl(SOC_XWR18XX_PINN4_PADBD, PINMUX_OUTEN_RETAIN_HW_CTRL, PINMUX_INPEN_RETAIN_
    Pinmux_Set_FuncSel(SOC_XWR18XX_PINN4_PADBD, SOC_XWR18XX_PINN4_PADBD_MSS_UARTA_RX);

    /* Setup the PINMUX to bring out the UART-3 */
    Pinmux_Set_OverrideCtrl(SOC_XWR18XX_PINF14_PADAJ, PINMUX_OUTEN_RETAIN_HW_CTRL, PINMUX_INPEN_RETAIN
    Pinmux_Set_FuncSel(SOC_XWR18XX_PINF14_PADAJ, SOC_XWR18XX_PINF14_PADAJ_MSS_UARTB_TX);

    /**************************************************************************
     * Setup the PINMUX:
     * - GPIO Output: Configure pin K13 as GPIO_2 output
     **************************************************************************/
    Pinmux_Set_OverrideCtrl(SOC_XWR18XX_PINK13_PADAZ, PINMUX_OUTEN_RETAIN_HW_CTRL, PINMUX_INPEN_RETAIN
    Pinmux_Set_FuncSel(SOC_XWR18XX_PINK13_PADAZ, SOC_XWR18XX_PINK13_PADAZ_GPIO_2);

    /**************************************************************************
     * Setup the GPIO:
     * - GPIO Output: Configure pin K13 as GPIO_2 output
     **************************************************************************/
    config->SensorStatusGPIO    = SOC_XWR18XX_GPIO_2;

    /* Initialize the DEMO configuration: */
    config->sysClockFrequency   = MSS_SYS_VCLK;
    config->loggingBaudRate     = 921600;
    config->commandBaudRate     = 115200;
```

Figure 20. SDK 3.3 16xx vs 18xx: SoC Definition Updates

6.8 SDK 3.4 xWR68xx Calibration Frequency Update

MMWAVE-SDK 3.3 vs MMWAVE-SDK 3.4

File: mmwave_sdk_03_04_00_0x\packages\ti\demo\xwr68xx\mmw\mss_main.c



C:\ti\mmwave_sdk_03_03_00_03\packages\ti\demo\xwr68xx\mmw\mss\mss_main.c
9/16/2019 1:01:15 PM 164,371 bytes C,C++,C# Source ANSI PC

```
{
    System_printf("Error: rlRfSetLdoBypassConfig retVal=%d\n", ret
    return -1;
}

/*  Open mmWave module, this is only done once */
/* Setup the calibration frequency:
 * TODO: Presently DFP does not support these for 68xx platform,
 * need to change when DFP is updated with the support */
gMmwMssMCB.cfg.openCfg.freqLimitLow = 0U;
gMmwMssMCB.cfg.openCfg.freqLimitHigh = 0U;

/* start/stop async events */
gMmwMssMCB.cfg.openCfg.disableFrameStartAsyncEvent = false;
gMmwMssMCB.cfg.openCfg.disableFrameStopAsyncEvent  = false;

/* No custom calibration: */
gMmwMssMCB.cfg.openCfg.useCustomCalibration        = false;
gMmwMssMCB.cfg.openCfg.customCalibrationEnableMask = 0x0;

/* calibration monitoring base time unit
 * setting it to one frame duration as the demo doesnt support any
 * monitoring related functionality
 */
gMmwMssMCB.cfg.openCfg.calibMonTimeUnit            = 1;
```

C:\ti\mmwave_sdk_03_04_00_03\packages\ti\demo\xwr68xx\mmw\mss\mss_main.c
3/30/2020 6:14:33 PM 167,431 bytes C,C++,C# Source ANSI PC

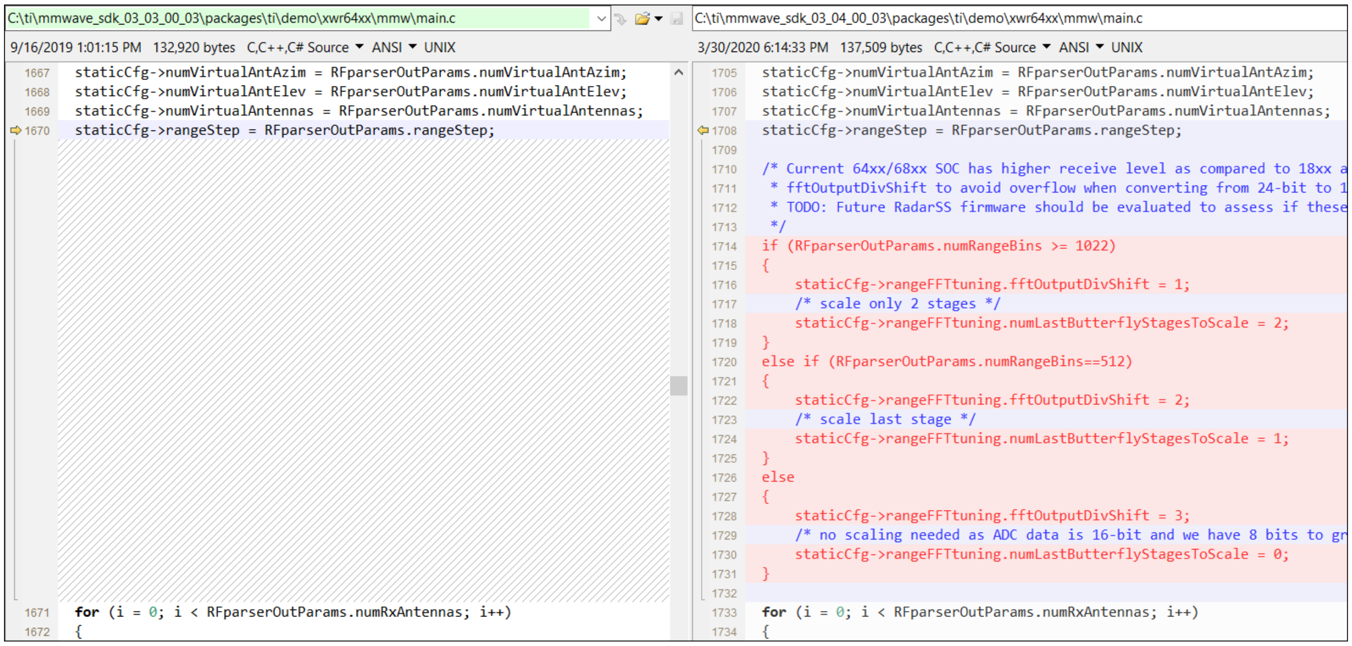
```
{
    System_printf("Error: rlRfSetLdoBypassConfig retVal=%d\n", ret
    return -1;
}

/*  Open mmWave module, this is only done once */
/* Setup the calibration frequency */

gMmwMssMCB.cfg.openCfg.freqLimitLow = 600U;
gMmwMssMCB.cfg.openCfg.freqLimitHigh = 640U;

/* start/stop async events */
gMmwMssMCB.cfg.openCfg.disableFrameStartAsyncEvent = false;
gMmwMssMCB.cfg.openCfg.disableFrameStopAsyncEvent  = false;

/* No custom calibration: */
gMmwMssMCB.cfg.openCfg.useCustomCalibration        = false;
gMmwMssMCB.cfg.openCfg.customCalibrationEnableMask = 0x0;

/* calibration monitoring base time unit
 * setting it to one frame duration as the demo doesnt support any
 * monitoring related functionality
 */
gMmwMssMCB.cfg.openCfg.calibMonTimeUnit            = 1;
```

Figure 21. SDK 3.4 xWR68xx Calibration Frequency Update

6.9 SDK 3.4 Object Detect HWA DPC Range FFT Scaling

MMWAVE-SDK 3.3 vs MMWAVE-SDK 3.4

File: mmwave_sdk_03_04_00_0x\packages\ti\demo\xwr64xx\mmw\main.c



C:\ti\mmwave_sdk_03_03_00_03\packages\ti\demo\xwr64xx\mmw\main.c

9/16/2019 1:01:15 PM 132,920 bytes C,C++,C# Source ANSI UNIX

```
staticCfg->numVirtualAntAzim = RFparserOutParams.numVirtualAntAzim;
staticCfg->numVirtualAntElev = RFparserOutParams.numVirtualAntElev;
staticCfg->numVirtualAntennas = RFparserOutParams.numVirtualAntennas;
staticCfg->rangeStep = RFparserOutParams.rangeStep;

for (i = 0; i < RFparserOutParams.numRxAntennas; i++)
{
```

C:\ti\mmwave_sdk_03_04_00_03\packages\ti\demo\xwr64xx\mmw\main.c

3/30/2020 6:14:33 PM 137,509 bytes C,C++,C# Source ANSI UNIX

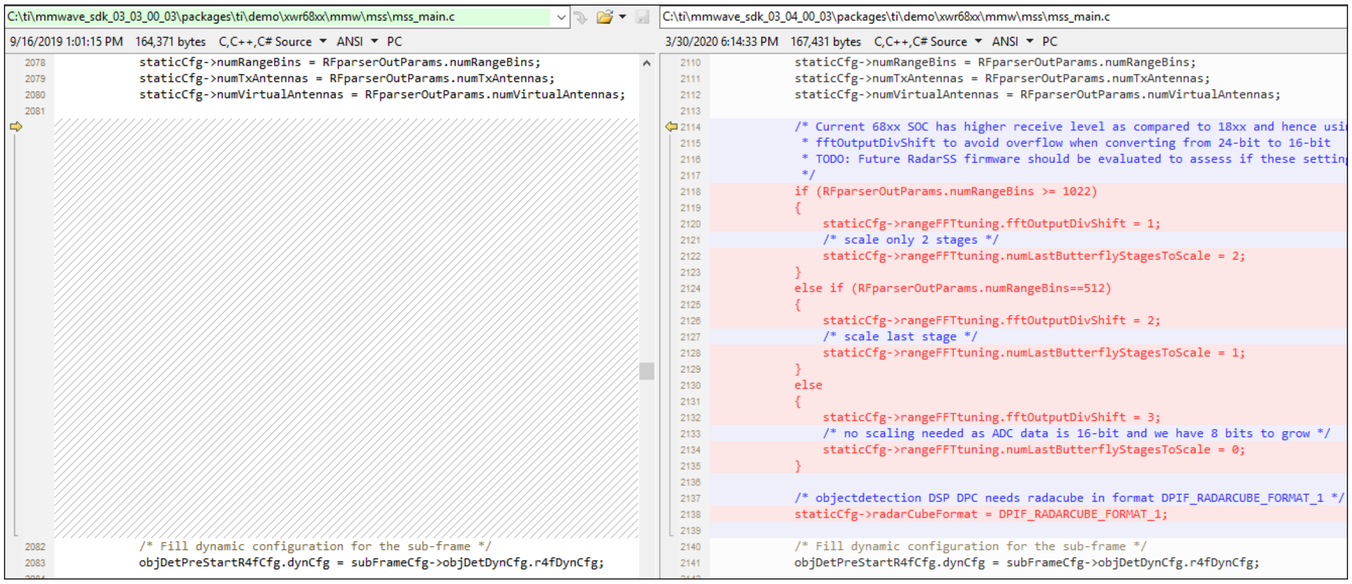
```
staticCfg->numVirtualAntAzim = RFparserOutParams.numVirtualAntAzim;
staticCfg->numVirtualAntElev = RFparserOutParams.numVirtualAntElev;
staticCfg->numVirtualAntennas = RFparserOutParams.numVirtualAntennas;
staticCfg->rangeStep = RFparserOutParams.rangeStep;

/* Current 64xx/68xx SOC has higher receive level as compared to 18xx a
 * fftOutputDivShift to avoid overflow when converting from 24-bit to 1
 * TODO: Future RadarSS firmware should be evaluated to assess if these
 */
if (RFparserOutParams.numRangeBins >= 1022)
{
    staticCfg->rangeFFTtuning.fftOutputDivShift = 1;
    /* scale only 2 stages */
    staticCfg->rangeFFTtuning.numLastButterflyStagesToScale = 2;
}
else if (RFparserOutParams.numRangeBins==512)
{
    staticCfg->rangeFFTtuning.fftOutputDivShift = 2;
    /* scale last stage */
    staticCfg->rangeFFTtuning.numLastButterflyStagesToScale = 1;
}
else
{
    staticCfg->rangeFFTtuning.fftOutputDivShift = 3;
    /* no scaling needed as ADC data is 16-bit and we have 8 bits to gr
    staticCfg->rangeFFTtuning.numLastButterflyStagesToScale = 0;
}

for (i = 0; i < RFparserOutParams.numRxAntennas; i++)
{
```

Figure 22. SDK 3.4 Object Detection DPC FFT Range Scaling Configuration

6.10 SDK 3.4 Object Detect Range HWA DPC Radar Cube Format

MMWAVE-SDK 3.3 vs MMWAVE-SDK 3.4

File: mmwave_sdk_03_04_00_0x\packages\ti\demo\xwr68xx\mmw\mss\mss_main.c



C:\ti\mmwave_sdk_03_03_00_03\packages\ti\demo\xwr68xx\mmw\mss\mss_main.c

9/16/2019 1:01:15 PM 164,371 bytes C,C++,C# Source ANSI PC

```
        staticCfg->numRangeBins = RFparserOutParams.numRangeBins;
        staticCfg->numTxAntennas = RFparserOutParams.numTxAntennas;
        staticCfg->numVirtualAntennas = RFparserOutParams.numVirtualAntennas;

        /* Fill dynamic configuration for the sub-frame */
        objDetPreStartR4fCfg.dynCfg = subFrameCfg->objDetDynCfg.r4fDynCfg;
```

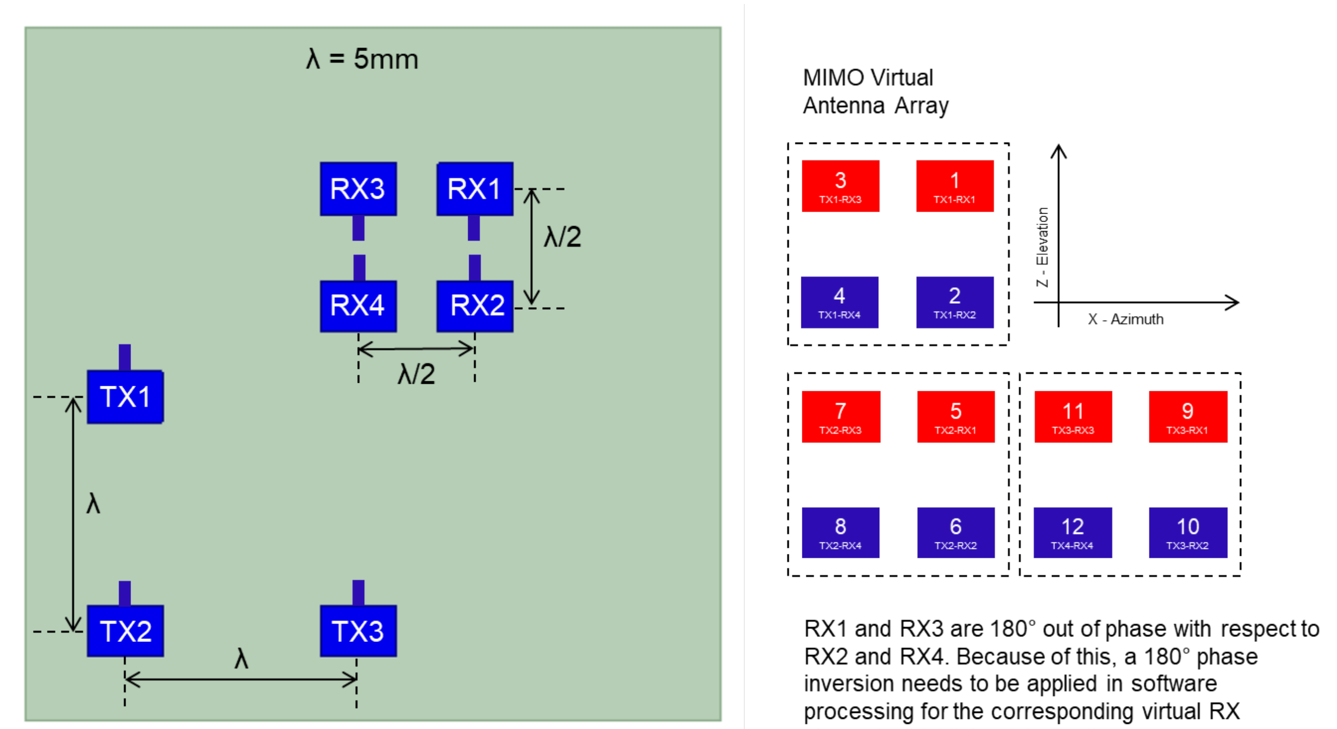
C:\ti\mmwave_sdk_03_04_00_03\packages\ti\demo\xwr68xx\mmw\mss\mss_main.c

3/30/2020 6:14:33 PM 167,431 bytes C,C++,C# Source ANSI PC

```
        staticCfg->numRangeBins = RFparserOutParams.numRangeBins;
        staticCfg->numTxAntennas = RFparserOutParams.numTxAntennas;
        staticCfg->numVirtualAntennas = RFparserOutParams.numVirtualAntennas;

        /* Current 68xx SOC has higher receive level as compared to 18xx and hence usi
         * fftOutputDivShift to avoid overflow when converting from 24-bit to 16-bit
         * TODO: Future RadarSS firmware should be evaluated to assess if these settin
         */
        if (RFparserOutParams.numRangeBins >= 1022)
        {
            staticCfg->rangeFFTtuning.fftOutputDivShift = 1;
            /* scale only 2 stages */
            staticCfg->rangeFFTtuning.numLastButterflyStagesToScale = 2;
        }
        else if (RFparserOutParams.numRangeBins==512)
        {
            staticCfg->rangeFFTtuning.fftOutputDivShift = 2;
            /* scale last stage */
            staticCfg->rangeFFTtuning.numLastButterflyStagesToScale = 1;
        }
        else
        {
            staticCfg->rangeFFTtuning.fftOutputDivShift = 3;
            /* no scaling needed as ADC data is 16-bit and we have 8 bits to grow */
            staticCfg->rangeFFTtuning.numLastButterflyStagesToScale = 0;
        }

        /* objectdetection DSP DPC needs radacube in format DPIF_RADARCUBE_FORMAT_1 */
        staticCfg->radarCubeFormat = DPIF_RADARCUBE_FORMAT_1;

        /* Fill dynamic configuration for the sub-frame */
        objDetPreStartR4fCfg.dynCfg = subFrameCfg->objDetDynCfg.r4fDynCfg;
```

Figure 23. SDK 3.4 Object Detect Range HWA DPC FFT Radar Cube Format

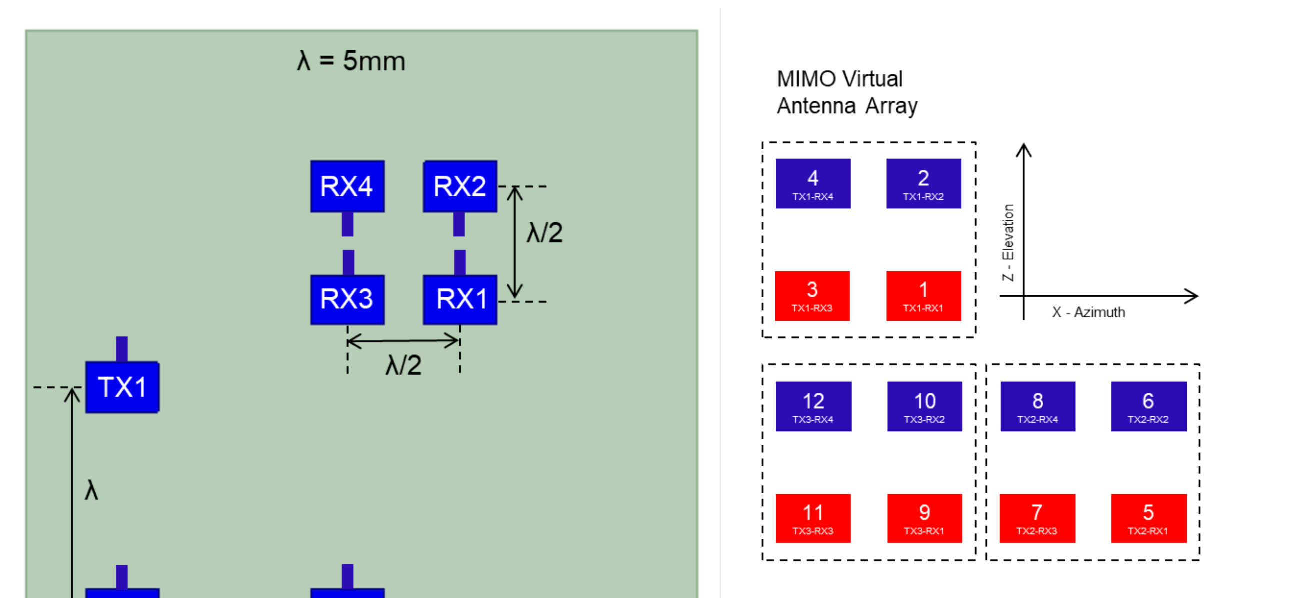
6.11 xWR6843AoP ES1.0 Antenna Geometry



RX1 and RX3 are 180° out of phase with respect to RX2 and RX4. Because of this, a 180° phase inversion needs to be applied in software processing for the corresponding virtual RX channels (highlighted in Red)

Figure 24. xWR6843AoP ES1.0 Antenna Geometry

6.12 xWR6843AoP ES2.0 Antenna Geometry



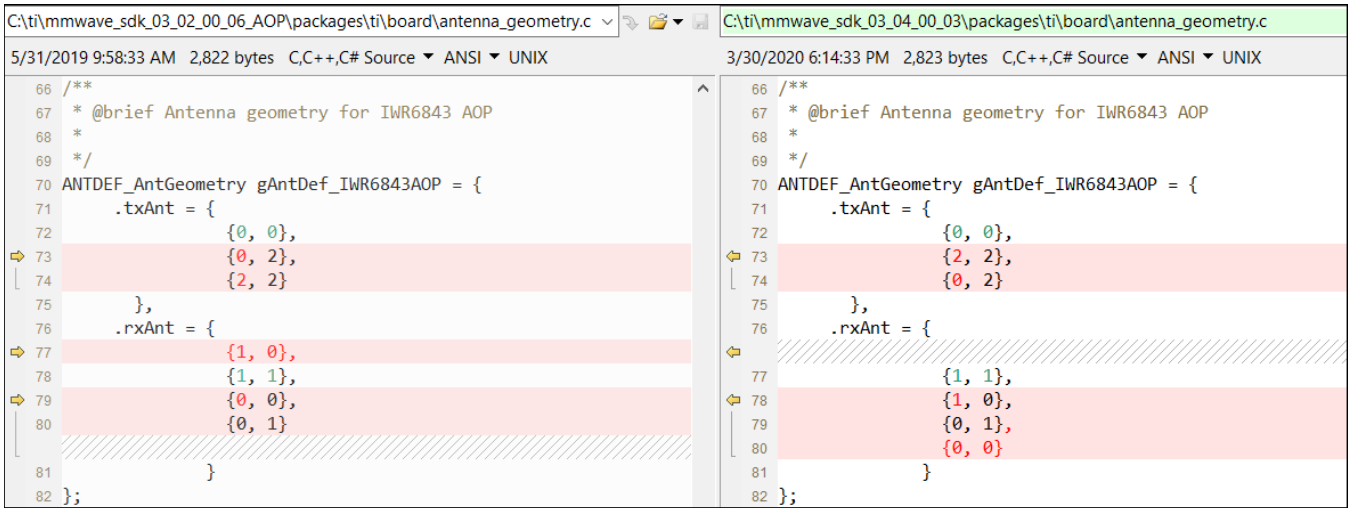
RX1 and RX3 are 180° out of phase with respect to RX2 and RX4, similar to ES1. Because of this, a 180° phase inversion needs to be applied in software processing for the corresponding virtual RX channels (highlighted in Red)

Figure 25. xWR6843AoP ES2.0 Antenna Geometry

6.13 xWR6843AoP ES2.0 Antenna Geometry Code Update

MMWAVE-SDK 3.2.0.6 vs MMWAVE-SDK 3.4

File: mmwave_sdk_03_04_00_0x\packages\ti\board\antenna_geometry.c



C:\ti\mmwave_sdk_03_02_00_06_AOP\packages\ti\board\antenna_geometry.c
5/31/2019 9:58:33 AM 2,822 bytes C,C++,C# Source ANSI UNIX

```
/**
 * @brief Antenna geometry for IWR6843 AOP
 *
 */
ANTDEF_AntGeometry gAntDef_IWR6843AOP = {
      .txAnt = {
                  {0, 0},
                  {0, 2},
                  {2, 2}
     },
      .rxAnt = {
                  {1, 0},
                  {1, 1},
                  {0, 0},
                  {0, 1}
               }
};
```

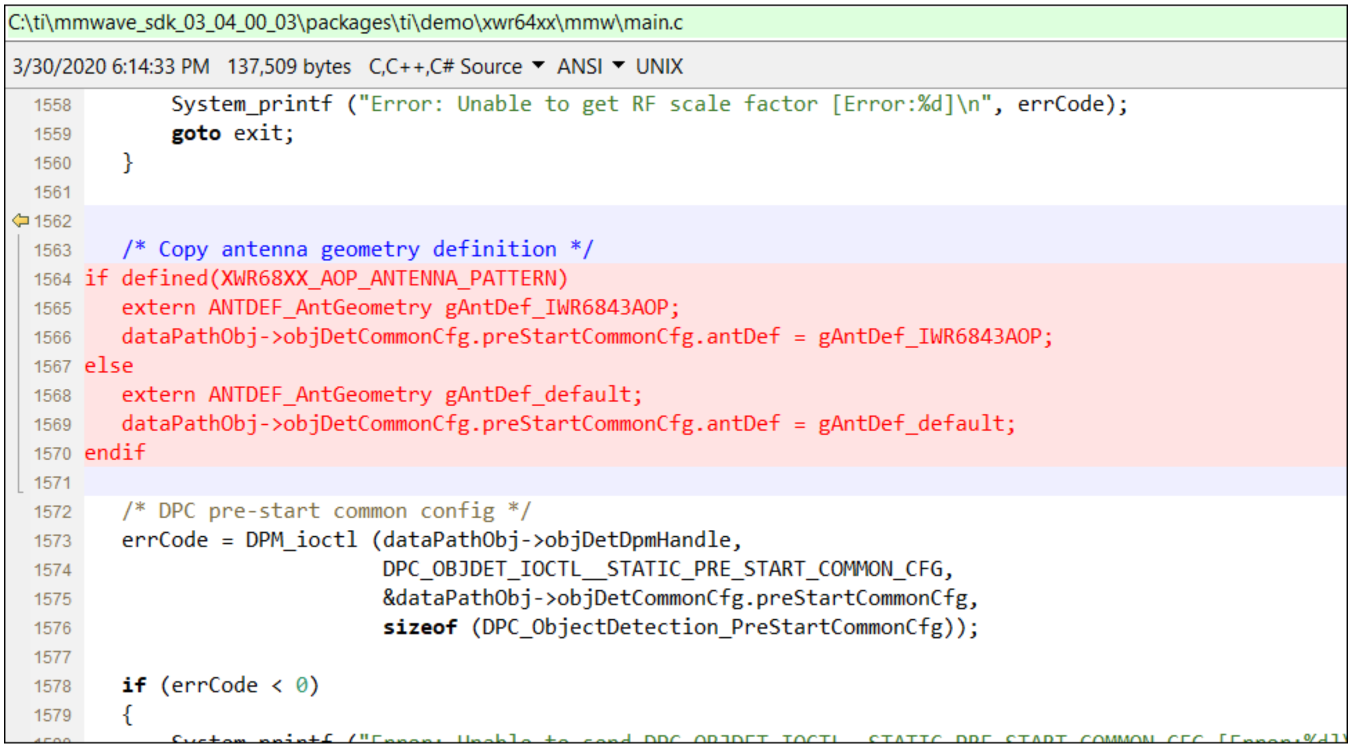
C:\ti\mmwave_sdk_03_04_00_03\packages\ti\board\antenna_geometry.c
3/30/2020 6:14:33 PM 2,823 bytes C,C++,C# Source ANSI UNIX

```
/**
 * @brief Antenna geometry for IWR6843 AOP
 *
 */
ANTDEF_AntGeometry gAntDef_IWR6843AOP = {
      .txAnt = {
                  {0, 0},
                  {2, 2},
                  {0, 2}
     },
      .rxAnt = {
                  {1, 1},
                  {1, 0},
                  {0, 1},
                  {0, 0}
               }
};
```

Figure 26. SDK 3.2.0.6 Vs SDK 3.4: Antenna Geometry Update for xWR6843AoP ES2.0

6.14 Antenna Geometry Structure Usage in mmw demo

File: mmwave_sdk_03_04_00_0x\packages\ti\demo\xwr64xx\mmw\main.c



C:\ti\mmwave_sdk_03_04_00_03\packages\ti\demo\xwr64xx\mmw\main.c
3/30/2020 6:14:33 PM 137,509 bytes C,C++,C# Source ANSI UNIX

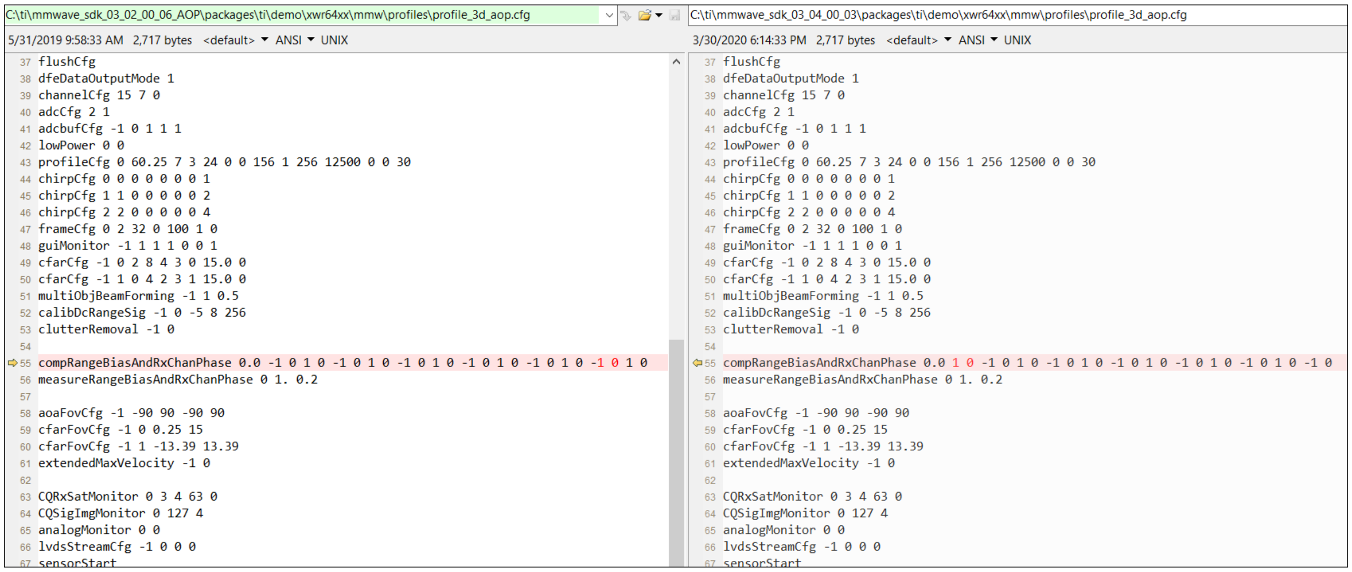
```
        System_printf ("Error: Unable to get RF scale factor [Error:%d]\n", errCode);
        goto exit;
    }


    /* Copy antenna geometry definition */
if defined(XWR68XX_AOP_ANTENNA_PATTERN)
    extern ANTDEF_AntGeometry gAntDef_IWR6843AOP;
    dataPathObj->objDetCommonCfg.preStartCommonCfg.antDef = gAntDef_IWR6843AOP;
else
    extern ANTDEF_AntGeometry gAntDef_default;
    dataPathObj->objDetCommonCfg.preStartCommonCfg.antDef = gAntDef_default;
endif

    /* DPC pre-start common config */
    errCode = DPM_ioctl (dataPathObj->objDetDpmHandle,
                         DPC_OBJDET_IOCTL__STATIC_PRE_START_COMMON_CFG,
                         &dataPathObj->objDetCommonCfg.preStartCommonCfg,
                         sizeof (DPC_ObjectDetection_PreStartCommonCfg));

    if (errCode < 0)
    {
```

Figure 27. Antenna Geometry Structure Usage in mmw demo

6.15 xWR6843AoP ES2.0 RX Channel Phase Compensation

MMWAVE-SDK 3.2.0.6 vs MMWAVE-SDK 3.4

File: mmwave_sdk_03_04_00_03\packages\ti\demo\xwr64xx\mmw\profiles\profile_3d_aop.cfg



C:\ti\mmwave_sdk_03_02_00_06_AOP\packages\ti\demo\xwr64xx\mmw\profiles\profile_3d_aop.cfg

5/31/2019 9:58:33 AM 2,717 bytes <default> ANSI UNIX

```
flushCfg
dfeDataOutputMode 1
channelCfg 15 7 0
adcCfg 2 1
adcbufCfg -1 0 1 1 1
lowPower 0 0
profileCfg 0 60.25 7 3 24 0 0 156 1 256 12500 0 0 30
chirpCfg 0 0 0 0 0 0 0 1
chirpCfg 1 1 0 0 0 0 0 2
chirpCfg 2 2 0 0 0 0 0 4
frameCfg 0 2 32 0 100 1 0
guiMonitor -1 1 1 1 0 0 1
cfarCfg -1 0 2 8 4 3 0 15.0 0
cfarCfg -1 1 0 4 2 3 1 15.0 0
multiObjBeamForming -1 1 0.5
calibDcRangeSig -1 0 -5 8 256
clutterRemoval -1 0

compRangeBiasAndRxChanPhase 0.0 -1 0 1 0 -1 0 1 0 -1 0 1 0 -1 0 1 0 -1 0 1 0 -1 0 1 0
measureRangeBiasAndRxChanPhase 0 1. 0.2

aoaFovCfg -1 -90 90 -90 90
cfarFovCfg -1 0 0.25 15
cfarFovCfg -1 1 -13.39 13.39
extendedMaxVelocity -1 0

CQRxSatMonitor 0 3 4 63 0
CQSigImgMonitor 0 127 4
analogMonitor 0 0
lvdsStreamCfg -1 0 0 0
sensorStart
```

C:\ti\mmwave_sdk_03_04_00_03\packages\ti\demo\xwr64xx\mmw\profiles\profile_3d_aop.cfg

3/30/2020 6:14:33 PM 2,717 bytes <default> ANSI UNIX

```
flushCfg
dfeDataOutputMode 1
channelCfg 15 7 0
adcCfg 2 1
adcbufCfg -1 0 1 1 1
lowPower 0 0
profileCfg 0 60.25 7 3 24 0 0 156 1 256 12500 0 0 30
chirpCfg 0 0 0 0 0 0 0 1
chirpCfg 1 1 0 0 0 0 0 2
chirpCfg 2 2 0 0 0 0 0 4
frameCfg 0 2 32 0 100 1 0
guiMonitor -1 1 1 1 0 0 1
cfarCfg -1 0 2 8 4 3 0 15.0 0
cfarCfg -1 1 0 4 2 3 1 15.0 0
multiObjBeamForming -1 1 0.5
calibDcRangeSig -1 0 -5 8 256
clutterRemoval -1 0

compRangeBiasAndRxChanPhase 0.0 1 0 -1 0 1 0 -1 0 1 0 -1 0 1 0 -1 0 1 0 -1 0 1 0 -1 0
measureRangeBiasAndRxChanPhase 0 1. 0.2

aoaFovCfg -1 -90 90 -90 90
cfarFovCfg -1 0 0.25 15
cfarFovCfg -1 1 -13.39 13.39
extendedMaxVelocity -1 0

CQRxSatMonitor 0 3 4 63 0
CQSigImgMonitor 0 127 4
analogMonitor 0 0
lvdsStreamCfg -1 0 0 0
sensorStart
```

Figure 28. SDK 3.2.0.6 Vs SDK 3.4: RX Channel Phase Compensation

7 References

- Texas Instruments: *IWR1642 Device Errata*
- Texas Instruments: *AWR1642 Device Errata*
- Texas Instruments: *IWR1843 Device Errata*
- Texas Instruments: *AWR1843 Device Errata*
- Texas Instruments: IWR6843 Device Errata
- Texas Instruments: AWR6843 Device Errata
- xWR1642BOOST Layout and Design Files
- xWR6843AOPEVM Schematic, Assembly and Bill of Materials
- Texas Instruments: *xWR1642 EVM (xWR1642BOOST) Single-Chip mmWave Sensing Solution User's Guide*
- Texas Instruments: *MMWAVEICBOOST and Antenna Module User's Guide*
- *xWR6843 Checklist for Schematic Review, Layout Review, Bringup/Wakeup*
- xWR1843BOOST Hardware Files
- Texas Instruments: *xWR1843 Evaluation Module (xWR1843BOOST) Single-Chip mmWave Sensing Solution User's Guide*
- xWR6843 Product Page (Device data sheet, Silicon Errata)
- xWR6843AoP Product Page (Device data sheet, Silicon Errata)
- xWR1843 Product Page (Device data sheet, Silicon Errata)
- xWR1642 Product Page (Device data sheet, Silicon Errata)
- Texas Instruments: *IWR14xx/16xx/18xx/68xx Industrial Radar Family Technical Reference Manual*
- MMWAVE-SDK Product Page
- MMWAVE-SDK 3.3.0 download page (Release notes, User guide and SDK download)
- MMWAVE-SDK 3.4.0 downloadpage (Release notes, User guide and SDK download)

Revision History

NOTE: Page numbers for previous revisions may differ from page numbers in the current version.

Changes from A Revision (November 2019) to B Revision **Page**

- IWR68xx, IWR18xx, IWR1642, and IWR6843 were changed to xWR68xx, xWR18xx, xWR1642, and xWR6843, respectively, throughout the document. 1
- Updates were made in Section 1. 3
- Updates were made in Section 2.2. 5
- Updates were made in Section 2.3.1. 7
- Updates were made in Section 2.3.2. 8
- Update was made in Section 2.3.4. 10
- Updates were made in Section 3.1.1. 13
- Update was made in Section 3.1.4. 15
- Added new Section 4. 17
- Updates were made in Section 3.1.1. 19
- Updates were made in Section 4.2.1. 19
- Updates were made in Section 5. 24
- Added new Section 6.8. 28
- Updates were made in Section 6.9. 29
- Added new Section 6.10. 29
- Added new Section 6.11. 30
- Added new Section 6.12. 30
- Updates were made in Section 6.13. 31
- Updates were made in Section 6.14. 31
- Updates were made in Section 6.15. 32