MMWAVE SDK Release Notes

Product Release 3.2.1

Release Date: Aug 8, 2019

Release Notes Version: 1.0

CONTENTS

1. Introduction

The mmWave SDK enables the development of millimeter wave (mmWave) radar applications using TI mmWave sensors (see list of supported Platform/Devices). The SDK provides foundational components which will facilitate end users to focus on their applications. In addition, it provides few demo applications which will serve as a guide for integrating the SDK into end-user mmWave application.

Key mmWave SDK features:

- Building blocks
  - Full driver availability
  - Layered approach to programming analog front end
  - Catalog of mmwave algorithms optimized for C674x DSPs
- Demonstrations and examples
  - TI RTOS based
  - Out of box demo with easy configurability via TI cloud based GUI
  - Representation of "point cloud" and benchmarking data from demo via GUI
  - Profiles tuned to common end user scenarios such as Range, Range resolution, Velocity, Velocity resolution
- Documentation

mmWave SDK works along with the following external tools:

- Host tools including Pin Mux, Flashing utilities
- Code Composer Studio™ IDE for RTOS development

2. Release overview

2. 1. What is new

- Support for devices mentioned in the "Platform and Device Support" section below
- New features can be found in New Features section.
- Bug fixes
- Tools update

2. 2. Platform and Device Support

The devices and platforms supported with this release include:

Supported Devices	Supported EVM
AWR1843 ES1.0	AWR1843BOOST - AWR1843 Evaluation Module RevC
AWR1642 ES2.0	AWR1642BOOST - AWR1642 Evaluation Module RevB
AWR1642_HS ES 2.0 *	
AWR1443 ES3.0	AWR1443BOOST - AWR1443 Evaluation Module RevB
IWR6843 ES1.0	IWR6843ISK+MMWAVEICBOOST - IWR6843 Evaluation Module
IWR1843 ES1.0	IWR1843BOOST - IWR1843 Evaluation Module RevC
IWR1642 ES2.0	IWR1642BOOST - IWR1642 Evaluation Module RevB
IWR1642_HS ES 2.0*	
IWR1443 ES3.0	IWR1443BOOST - IWR1443 Evaluation Module RevB

* High Secure (HS) devices need additional MMWAVE-SECDEV package

xWR terminology is used in sections that are common for AWR and IWR devices

Silicon versions other than the ones in the table above are not supported

This release of mmWave SDK supports the foundation components for the devices mentioned in the table above . At system level, the mmWave SOC/EVM may interface with other TI ecosystem SOCs/Launchpads/EVMs and software for these other devices will not be a part of the mmWave SDK foundation components.

2. 3. Component versions

Components inside mmwave_sdk that have their own versions are shown below.

Component		Version	Type	Comment
mmwave sdk		3.2.1	Source and Binary	Overall package release version
RadarSS firmware (patch) for xwr14xx, xwr16xx, xwr18xx		1.2.5.2	Binary	RadarSS firmware is in ROM. Only the patch is included in the mmwave sdk release
RadarSS firmware for xwr68xx		6.0.7.0	Binary	
mmWaveLink Framework		1.2.5	Source and Binary	
FTDI		2.12	Binary	
Image Creator	gen_bincrc32	1.0	Windows and Linux binary	
	out2rprc	2.0	Windows binary	Need mono to run this on Linux
	Crc multicore image	1.0	Windows and Linux binary	
	Multicore image generator	1.0	Windows and Linux binary	
	create_ConfigRPRC	1.0	Windows and Linux binary	

2. 4. Tools dependency

For building and using mmwave sdk the following tool versions are needed.

Tool	Version	Download link
CCS	7.4 or later	download link
TI SYS/BIOS	6.73.01.01	Included in mmwave sdk installer
TI ARM compiler	16.9.6.LTS	Included in mmwave sdk installer
TI CGT compiler	8.3.3	Included in mmwave sdk installer
XDC	3.50.08.24	Included in mmwave sdk installer
C64x+ DSPLIB	3.4.0.0	Included in mmwave sdk installer
C674x DSPLIB	3.4.0.0	Included in mmwave sdk installer
C674x MATHLIB (little-endian, elf/coff format)	3.1.2.1	Included in mmwave sdk installer
Mono JIT compiler	4.2.1	Only for Linux builds
mmWave Radar Device support package	1.6.1 or later	Upgrade to the latest using CCS update process (see SDK user guide for more details)
TI Emulators package	7.0.188.0 or later	Upgrade to the latest using CCS update process (see SDK user guide for more details)
MMWAVE-SECDEV	2.0.0	Needed for xWR16xx high secure (HS) devices only Can be requested from link
Pinmux tool (optional)	Latest	Used to generate pinmux configuration for custom board https://dev.ti.com/pinmux (Cloud version)
Doxygen (optional)	1.8.11	Only needed if regenerating doxygen docs
Graphviz (optional)	2.36.0 (20140111.2315)	Only needed if regenerating doxygen docs

The following tools are needed at runtime

Runtime tool	Version	Link
Uniflash	Latest	Uniflash tool is used for flashing xWR1xxx devices Cloud version (Recommended): https://dev.ti.com/uniflash Offline version: http://www.ti.com/tool/uniflash
mmWave Demo Visualizer	Latest	TI Gallery APP for configuring mmWave sensors and visualizing the point cloud objects generated by the mmWave SDK demo https://dev.ti.com/mmWaveDemoVisualizer

2. 5. Licensing

Please refer to the mmwave_sdk_software_manifest.html, which outlines the licensing status for mmwave_sdk package.

3. Release content

3. 1. New Features

- mmWave Suite enhancement
  - mmWaveLib Enhancements
    - Added new functions in CFAR that will output noise floor in addition to thee detected peaks.
    - Moved makefiles of mmwavelib unit test benches to mmwavelib\test folder for code re-organization
  - mmWave data processing layers
    - AoA DSP DPU: added max Velocity algorithm
  - mmwavelink Enhancements
    - Added 2 new async events: RL_MMWL_AE_MISMATCH_REPORT and RL_MMWL_AE_INTERNALERR_REPORT to report communication related errors
    - Added GET API to retrieve errors details about RadarSS CPU and ESM fault
    - RX Gain Phase Monitoring – Added more reporting fields in corresponding async event to improve interference immunity.
    - Provided an option to program dummy chirps in the end of frame in frame config API. This is an option to mask ADC data for few chirps in the end of frame.
    - Added new API rlDeviceSetRetryCount to change the retry count from default value to any user desired value that is <=RL_API_CMD_RETRY_COUNT. Application could use this API to set retry count to zero to skip the command retry step.
  - mmWave Enhancements
    - Enable boot time “TX phase calibration” by default for xwr18xx via rlRfInitCalibConfig
- mmWave Demos enhancement
  - Added max velocity feature to the AoA DSP DPU thereby enabling that feature in xwr68xx and xwr16xx demos
  - xwr18xx demo: Invokes the LDO Bypass API (with RF LDO bypassed and PA LDO input enabled) as part of initialization routine as per recommendation for the TI xWR1843BOOST EVM.
  - Added handling of 2 new async events: RL_MMWL_AE_MISMATCH_REPORT and RL_MMWL_AE_INTERNALERR_REPORT which could be generated by mmwavelink due to RadarSS related communication errors.
  - mmWave Visualizer enhancement
    - Added support for calibDcRangeSig command from the real-time advanced window
    - Added extra time to chirp idle time to handle the minimum chirp requirement of 15 usec
- mmWave Utilities
  - Added utility module to generate Gaussian distributed random complex number with specified variance
- Components/Tools
  - RadarSS: Updated the RadarSS component for xwr14xx/xwr16xx/xwr18xx (see exact version above). Users should refer to the RadarSS release notes included under mmwave_sdk_<ver>/firmware/radarss folder for features and enhancements done in this component.

3. 2. Migration section

This section describes the changes that are relevant for users migrating to the mmWave SDK 3.2.1 release from 3.2.0 release. See release notes archive in the SDK release package for migrating from other older releases.

Summary	Component/s	Subcomponent	Behavior of impact
The minimum duration (idle time + ramp end time) of a chirp can not be less than 15us (normal mode of operation) + 10us (if RadarSS WDT is enabled) + 10us (if per chirp phase-shifter API is enabled). The best case minimum chirp cycle time is 15us and worst case minimum chirp cycle time is 35us.	Demos		Application should validate their chirp design against these rules to avoid RadarSS related faults/errors. mmW Demos do not enable RadarSS WDT by default and per chirp phase shifter API is also not showcased. So follow the rules for normal mode of operation when designing chirp to be executed using mmW demo.
The ESM_MONITORING_EN flag in rlDigMonPeriodicConf_t structure have been made reserved.	Control	mmWaveLink	Application should ignore the reserved flag.
The bit 14 and 25 of esmGrp1Err field in rlBssEsmFault_t structure is updated with programmable filter parity and ECC DB fatal error indications respectively.	Control	mmWaveLink	Application should handle this error as a fatal error from the mmWave Device.
The ESM_MONITORING flag in rlDigPeriodicReportData_t structure have been made reserved.	Control	mmWaveLink	Application should ignore the reserved status flags.
The bit 2 of statusFlags field in rlMonTxBpmRep_t structure is updated with phase shifter monitoring status.	Control	mmWaveLink	Application should handle this status report if device supports phase shifter.
New asynchronous event message ID has been added for notification of CRC /Checksum failure in the received message (Async-event) from the device.	Control	mmWaveLink	Application needs to handle this msgID RL_MMWL_ASYNC_EVENT_MSG and SBID-RL_MMWL_AE_MISMATCH_REPORT in the ‘rlAsyncEvent’ callback implementation.

Value option '0x2' for rlTestPattern_t-> testPatGenCtrl is not validated and removed.	Control	mmWaveLink	Application should not use this option to generate the test pattern.
One of reserved field of rlFrameCfg_t structure is now used as 'numDummyChirpsAtEnd' parameter to configure the dummy chirp at the end of frame.	Control	mmWaveLink	Application should be aware of this parameter change while using rlSetFrameConfig/rlGetFrameConfig API.
In case MutexLock callback returns non-zero value in the interrupt context (reading async-event) then mmWaveLink generates internal Async-event with error code to notify the Host about this failure.	Control	mmWaveLink	Application needs to handle this msgID RL_MMWL_ASYNC_EVENT_MSG and SBID-RL_MMWL_AE_INTERNALERR_REPORT in the 'rlAsyncEvent' callback implementation.
The bits [11:8] STATUS_PWRDET_RXn in statusFlags in rlMonRxIntAnaSigRep_t structure have been made reserved as RX power detector monitoring has been disabled.	Control	mmWaveLink	Application should ignore the reserved flag.
The bit 26 PCR test enable in enMask in rlMonAnaEnables_t is made reserved as this test has been de-featured.	Control	mmWaveLink	Application should ignore the reserved bits.
The bit 26 PCR test status in digMonLatentFault in rlDigLatentFaultReportData_t is made reserved as this test has been de-featured.	Control	mmWaveLink	Application should ignore the reserved flag.
The bit 2 PCR test enable in testEn2 in rlperiodicTest_t as this test has been de-featured.	Control	mmWaveLink	Application should ignore the reserved bits.
Added new fields loopbackPowerRF1, loopbackPowerRF2, loopbackPowerRF3, rxNoisePower1 and rxNoisePower2 in rlMonRxGainPhRep_t structure.	Control	mmWaveLink	Application should refer to the doxygen of this report structure to consume this new information
Added reportMode field in rlRxGainPhaseMonConf_t structure.	Control	mmWaveLink	Application should be aware of this parameter change while using the API.
Added reportMode field in rlTxGainPhaseMismatchMonConf_t structure.	Control	mmWaveLink	Application should be aware of this parameter change while using the API.
Moved rlRfTempData_t structure from mmwavelink.h to rl_sensor.h	Control	mmWaveLink	No impact to application
Some reserved bytes in async reports can contain non-zero values.	Control	mmWaveLink	All reserved bytes in reports are to be masked when consuming these reports
anaTxPos filed in rlTestSource_t is reserved.	Control	mmWaveLink	Application should ignore the reserved field.
Report mode 0 is not supported for the following monitoring APIs: 1. rlRfTxIntAnaSignalsMonConfig 2. rlRfRxIntAnaSignalsMonConfig, 3. rlRfPmClkLoIntAnaSignalsMonConfig, 4. rlRfGpadcIntAnaSignalsMonConfig, 5. rlRfPllContrlVoltMonConfig, 6. rlRfDualClkCompMonConfig.	Control	mmWaveLink	Application should be aware of this parameter change while using these APIs.
Bit3 - VIM Test of rlStartComplete_t status field has been de-featured.	Control	mmWaveLink	Application should ignore the reserved flag.
Bit3 - VIM Test of rlMonDigEnables_t enMask field has been de-featured.	Control	mmWaveLink	Application should ignore the reserved bits.
Bit0 - SYNTH_VCO_OPENLOOP fault injection in synthFault in rlAnaFaultInj_t has been de-featured	Control	mmWaveLink	Application should ignore the reserved bits.
In rlMonPllConVoltRep_t statusFlags , Bit3 - VC01 and bit6 - VC02 slope monitors have been de-featured	Control	mmWaveLink	Application should ignore the reserved flag.
Bit3 in statusFlags in rlMonPmclkloIntAnaSigRep_t is now STATUS_LVDS_PMCLKLO to match the spec.	Control	mmWaveLink	Application should be aware of this parameter change while using these APIs.
Bit 2 in statusFlags in rlMonGpadcIntAnaSigRep_t is now STATUS_GPADC_REF2 to match the spec	Control	mmWaveLink	Application should be aware of this parameter change while using these APIs.

3. 3. Issues fixed

This section captures the issues that were fixed in this release for mmWave Suite/Demos. For RadarSS related issues that are fixed as part of this release can be found in RadarSS release notes included under mmwave_sdk_<ver>/firmware/radarss folder.

Issue Type	Key	Summary
Bug	UNIFLASH-1195	mmwave device IWR6843: Unable to flash 2 MetaImages via the command line package
Bug	MMWL-150	Fixed an issue with No fault reported even after a CRC is corrupted
Bug	MMWL-153	Fixed MSS monitor latent fault report documentation to match with ICD

Bug	MMWL-159	mmWaveLink: Wrong function name in the log string of rlDriverAddDevice function.
Bug	MMWL-162	Tx power monitor configuration comment is in compatible with code
Bug	MMWL-166	Fixed bug in handling retry in case of NACK
Bug	MMWL-167	Checksum failure for the ACK results in next command being ignored by Radar Device
Bug	MMWL-169	Fix for AE length info being reported to application upon CRC failure.
Bug	MMWL-170	Should read limited no. of bytes to look for SYNC pattern
Bug	MMWL-173	Added a logic to recover from CNYS corruption scenario
Bug	MMWL-177	Added max retry count limit in case of NACK retry
Bug	MMWL-178	Added an API to control number of retry for command
Bug	MMWL-180	Added a return in case of OS Interface Error when Mutex Lock callback returns Error.
Bug	MMWL-193 MMWL-199	Updated doxygen comments and incorporated code inspection review comments
Bug	MMWL-202	Fixes for LDRA violations.
Bug	MMWL-215	Test Source configuration structure changed to match the ICD definition.
Bug	MMWL-217	mmWaveLink update for flushing 256 + CRC Size bytes instead of 256 + CRC Size + SYNC bytes in case of ACK SYNC corrupted
Bug	MMWSDK-1890	Visualizer 3.2 not compatible with mmWave SDK 3.1, 3.0 for xWR68xx
Bug	MMWSDK-1898	SPIB works with CS0 slave, but it is not working with CS1 and CS2 slaves
Bug	MMWSDK-1929	Definition of valid burstPeriodicity in user guide doesn't match mmwavelink documentation
Bug	MMWSDK-1932	Visualizer v3.2 generates invalid cfg for xwr68xx OOB
Bug	MMWSDK-1938	mmw demo: cannot reconfig demo if the first config results in RF related error in mmWave_config
Bug	MMWSDK-1948	xwr64xx demo: use constant or permanent string for overridePlatformString
Bug	MMWSDK-1952	mmW demo doesn't handle mmwave start related failures cleanly

3. 4. Known Issues

3. 4. 1. mmWave Suite/Demos Known Issues

The following issues are known at the time of this release.

Issue Type	Key	Summary	Comments
Bug	MMWSDK-1872	EDMA transfer can stall while running DPC test code on AWR18xx	When the HWA based DPC test code is configured to repeat 10 identical frames, the EDMA transfer can stall at random while repeating identical frame executions. It appears that the problem is related to EDMA read issue while crossing the 4K boundary. Some of the EDMA source and destination addresses in L2 memory happened to be just before the 4K boundary, and the EDMA transfer from HWA to L2 after several frame repetitions became stalled during the transfer from HWA to L2 memory. After forcing the addresses of input/ouput buffers in L2 memory just above 4K boundary, the problem appeared to go away. These EDMA transfers are configured as A-sync transfers with bcount = 1.

Bug	MMWSDK-1871	If the chirp idle times between different bursts/subframes are different in advanced subframe profile for xwr64xx and xwr68xx devices, it can lead to RF CPU fault	The RadarSS in DFP 6.1 cannot handle the case where certain profile config has <10us chirp idle time and others have >10us chirp idle times and all these profile configs are active in the advanced frame configuration. Workaround (choose any one): 1. Increase the idle time of the profiles to ensure they're always >= 10us. (OR) 2. Issue a Dynamic Power Save Configuration API with all fields disabled before frame trigger. (rlRfDynamicPowerSave)
Bug	MMWSDK-1870	LVDS streaming of S/W data can get stuck sometimes depending on timing and/or buffer placements	This issue is under debug and rootcause/workaround is unknown. When such situation happens, underlying CBUFF IP doesn't provide the completion event for that session and in mmW demo, this can lead to debug assert as the previous frame would be marked incomplete and demo will prevent the next frame to proceed.
Bug	MMWSDK-1542	AoA DPU: RX phase calibration does not work when measurement is done with less than the possible max antenna size (#tx < 3, #rx < 4 in case of IWR6843)	Documented procedure in past releases always mentioned that all the available antennas on the device be turned on for measurement - so this is not creating any deviation from that. This is listed as known issue so that user are aware of the limitation.
Bug	MMWSDK-1497	Intermittent failure in "monitoring results" for mmwavelink unit test for awr16xx	This issue is seen in noisy lab environment only. One out of many reports for noise figure has failure status. Observed noise figure from that report are logged at the end of the test run and can be used for debugging further, in case this is seen in other scenarios.
Bug	MMWSDK-1363	Range processing hwa DPU crashes when number of RX antenna is 4, and range fft size is 1024	For 1 TX 4 RX and numRangeBins = 1024, the BdstIndex for EDMA copy will go beyond its limit of 32768. The calculation is follows: BytesPerChirp = numRangeBins * numRxAnt * sizeof(cmplx16ImRe_t) = 16KB. For 1 TX antenna, due to ping/pong scheme, the jump will be 2 * BytesPerChirp = 32KB. The same case is solved by manually setting destination address in rangeProc DSP based implementation. For rangeProcHWA, the manually setting of destination address is not doable.
Bug	MMWSDK-1157	Rare failure seen in UART loopback driver unit test - HW limitation	
Bug	MMWSDK-1078	Limitation in processing chain + LVDS instrumentation use case	See limitations section below
Task	MMWSDK-533	GUI of mmw demo running slow from Firefox browser	Workaround: Please switch to Chrome browser.
Story	MMWSDK-319	CAN driver: DMA mode is not supported	
Story	MMWSDK-252	UART driver has not tested for Data Length 5 and 6	

3. 4. 2. RadarSS Known Issues

3. 4. 2. 1. RadarSS firmware (patch) 1.2.5.2 for xwr14xx, xwr16xx, xwr18xx

Users should refer to the RadarSS release notes included under mmwave_sdk_<ver>/firmware/radarss folder for known issues in this release of RadarSS firmware.

3. 4. 2. 2. RadarSS firmware 6.0.7.0 for xwr68xx

Users should refer to the RadarSS release notes included under mmwave_sdk_<ver>/firmware/radarss folder for known issues in this release of RadarSS firmware.

3. 5. Limitations

3. 5. 1. mmWave Suite/Demos Limitations

Some of these limitations are captured in the "known issues" list shown in previous section.

1	CAN driver: • DMA and FIFO mode are not supported
2	CANFD driver: • DMA and Timestamping are not supported

3	CBUFF/CSI2/LVDS:
	• Driver does not support the following functionality:
	  • Multiple packets
	  • 3 channels
	• CSI2: ADC streaming has only been tested under 1 configuration in csi_stream usecase

4	CRC driver: "Auto" mode is not implemented.
5	DMA driver: MPU and Parity Feature not implemented.
6	EDMA driver: Privilege feature not implemented.
7	HWA driver: Any modes/algorithm outside the scope of mmWave demo are not tested (however they are implemented in the driver).
8	I2C driver: Verified loopback mode on all mmWave device TI EVM (however all features are implemented in the driver) and master mode using address scanning on all devices. Note that default xWR1642 BOOST EVM does not have a direct connection to I2C devices on the board from the xwr1642 device and this I2C scan test in driver will fail until board modifications are done.
9	QSPI/QSPI Flash driver: ▪ dual-Read/Quad read in configuration mode is not supported ▪ setting write protections bits is not supported
10	SPI (MIBSPI) Limitations: • For xWR14xx, MIBSPI is only supported on SPIA, hence driver only supports SPIA. SPIB is not supported in xWR14xx. In xWR16xx, both instances are MIBSPI and are supported within the driver. • When MIBSPI mode is used in 4-pin slave mode, for every CHARLEN (8 bits or 16 bits), CS signal(from Master) has to be toggled and 2 VBUSP cycles need to be inserted. This needs to be taken care on SPI master device.
11	DMA based transactions are not supported for CRC and Mailbox driver.
12	mmW demo: See demo's doxygen page for more details.
13	Processing chain + LVDS instrumentation: ▪ This feature is not available for xWR14xx due to ADC Buffer being unavailable for streaming while datapath processing is active. ▪ For xWR16xx, xWR18xx, xWR68xx, CQ cannot be streamed out reliably when datapath processing is also enabled. The data corruption for CQ data over LVDS lanes is seen more pronounced when multiple chirps/chirp event is enabled. Note that, for this reason, default mmW demo does not allow LVDS streaming and multiple chirps/chirp event to be enabled in the same configuration.

3. 5. 2. RadarSS Limitations

3. 5. 2. 1. RadarSS firmware (patch) 1.2.5.2 for xwr14xx, xwr16xx, xwr18xx

Users should refer to the RadarSS release notes included under mmwave_sdk_<ver>/firmware/radarss folder for "Unsupported Features and APIs" in this release of RadarSS firmware.

3. 5. 2. 2. RadarSS firmware 6.0.7.0 for xwr68xx

Users should refer to the RadarSS release notes included under mmwave_sdk_<ver>/firmware/radarss folder for limitations in this release of RadarSS firmware.

4. Test reports

Results of the unit tests can be found in the docs/test folder. The test folder has separate folders for all the SoC variants. System level test is run using demos.

5. Installation instructions

mmwave_sdk installer is available as a Windows Installer and a Linux installer.

- **mmwave_sdk_<version>-Windows-x86-Install.exe: Windows installer verified on Windows 7 and Windows 10 machines**
- **mmwave_sdk_<version>-Linux-x86-Install.bin: Linux installer verified on Ubuntu 14.04 & Ubuntu 16.04 64 bit machines.**

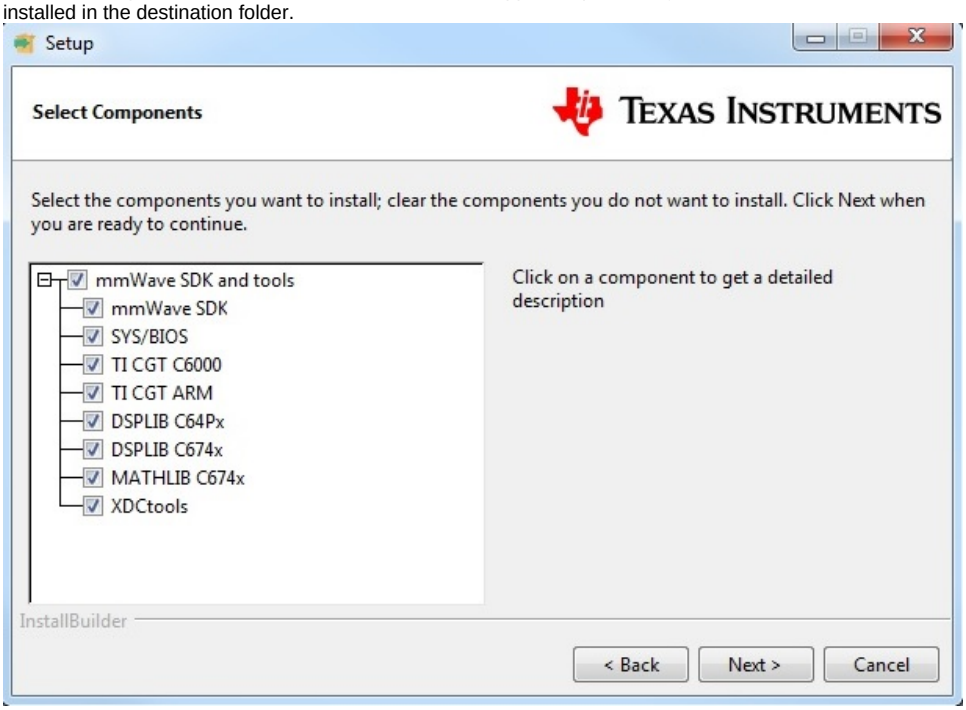
5. 1. Installation in GUI mode

Depending on your development environment run the appropriate installer

- In Windows environment, double clicking the Windows installer from Windows explorer should start the installation process
- If in Linux environment,
    - On 64-bit machines: Since mmwave_sdk_<version>-Linux-x86-Install.bin is a 32-bit executable, install modules that allows Linux 32bit binaries to execute: "sudo dpkg --add-architecture i386"
    - Enable execute permission for the Linux installer by running "chmod +x mmwave_sdk_<version>-Linux-x86-Install.bin" command
    - Run the installer using "./mmwave_sdk_<version>-Linux-x86-Install.bin" command
    - On 64-bit machines if the GUI does not show up you may need to install additional packages: "sudo apt-get install libc6:i386 libgtk2.0-0:i386 libxtst6:i386"

Installation steps:

- Setup
- Choose Destination Location: Select the folder to install (default is c:\ti on windows and ~/ti on linux). The installation folder selected should not have spaces in its full path.
- Select Components: The installer includes all the tools needed for building the mmWave SDK. You should see a screen like below (except that each component will also have version information appended). The only reason to deselect a tool is if the exact tool version is already installed in the destination folder.



- Review installation decisions
- Ready to install
- Once installation starts all the selected components will be installed (if a component with the same version exists in the destination folder it will be overwritten)
- Installation complete

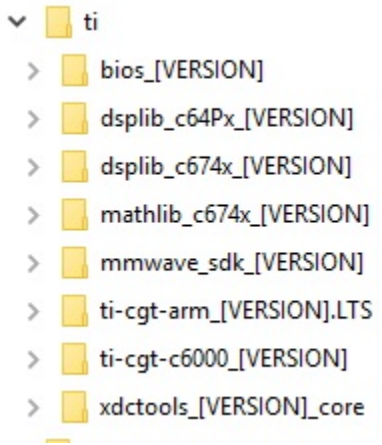
5. 2. Installation in unattended command line mode

The installers can be run in command line mode without user intervention

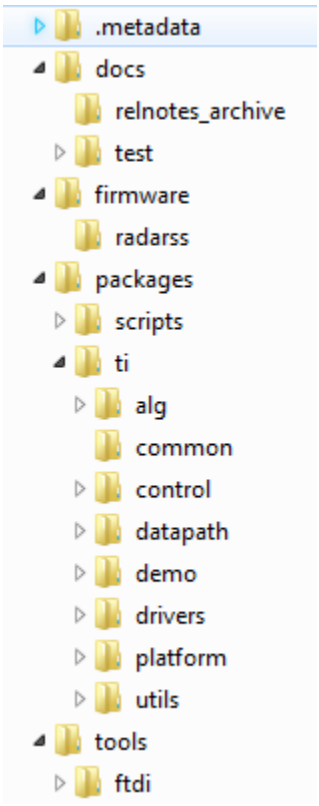
- In Windows environment
    - Run the installer using "mmwave_sdk_<version>-Windows-x86-Install.exe --prefix <installation folder> --mode unattended" command. This will install all the components in the installer.
        - Please note that even though the command may finish immediately it takes sometime for all the folders to show up in the destination folder (double check if you have the folder structure in "Post Installation" section before proceeding)
        - For command line help including information about selective installation of components run the following command "mmwave_sdk_<version>-Windows-x86-Install.exe --help"
- In Linux environment:
    - On 64-bit machines: Since mmwave_sdk_<version>-Linux-x86-Install.bin is a 32-bit executable, install modules that allows Linux 32bit binaries to execute: "sudo dpkg --add-architecture i386"
    - Enable execute permission for the Linux installer by running "chmod +x mmwave_sdk_<version>-Linux-x86-Install.bin" command
    - Run the installer using "./mmwave_sdk_<version>-Linux-x86-Install.bin --prefix <installation folder> --mode unattended" command. This will install all the components in the installer.
        - For command line help including information about selective installation of components run the following command "./mmwave_sdk_<version>-Linux-x86-Install.bin --help"

5. 3. Post Installation

After the installation is complete the following folder structure is expected in the installation folder (except that each component will have appropriate version number in place of the VERSION placeholder shown below)



Under the mmwave_sdk <ver> folder you should have the following directory structure.

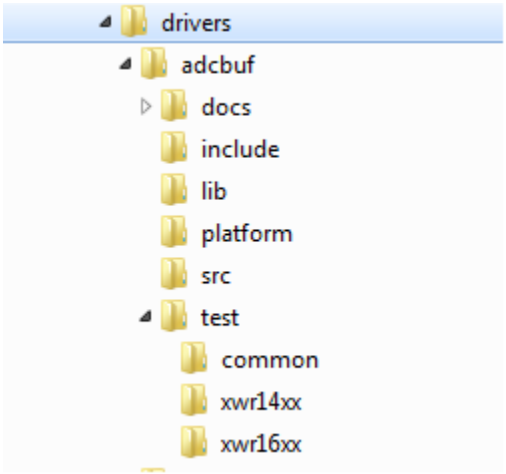


6. Package Contents

The mmwave sdk release package contains the following major components/folders.

6. 1. Drivers

Drivers can be found under mmwave_sdk_<ver>/packages/ti/drivers folder. The directory structure of all drivers is similar to the one shown below for adcbuf (some drivers do not have a unit test as shown in the table below)



- docs: Driver API documentation done with doxygen
- include: Include files
- lib: Prebuilt libraries
- platform: Platform files
- src: Driver Source files
- test/<platform>: Unit test src files and prebuilt unit test binary for supported platforms
- test/common: Unit test src files common for all platforms
- driver base folder has external header file, make files

Content of each driver is indicated in the table below.

Component	Source & prebuilt library	API Document (doxygen)	Unit test (source & prebuilt binary)
ADCBUF	X	X	X
CAN	X	X	X
CANFD	X	X	X
CBUFF/LVDS	X	X	X
CRC	X	X	X
CRYPTO[1]	X	X	X
CSI2	X	X	X
DMA	X	X	X
EDMA	X	X	X
ESM	X	X	
GPIO	X	X	X
HWA	X	X	X
I2C	X	X	X
MAILBOX	X	X	X
OSAL	X	X	
PINMUX	X	X	
QSPI	X	X	X

QSPIFLASH	X	X	X
SOC	X	X	
SPI	X	X	X
UART	X	X	X
WATCHDOG	X	X	X

[1] CRYPTO is only supported on high secure (HS) devices

6. 2. Control

Control modules can be found under mmwave_sdk_<ver>/packages/ti/control folder. Content of each of the control module is shown below

Component	Source & Prebuilt Library	API Document (doxygen)	Unittest (source & prebuilt binary)
datapath manager (dpm)	X	X	X
mmwavelink framework	X	X	X
mmwave high level api	X	X	X

6. 3. Datapath

Datapth modules can be found under mmwave_sdk_<ver>/packages/ti/datapath folder. Content of each of the control module is shown below

Component	Source & Prebuilt Library	API Document (doxygen)	Unittest (source & prebuilt binary)
RangeProc DPU	X	X	X
Doppler DPU	X	X	X
Static Clutter DPU	X	X	X
CFAR CA DPU	X	X	X
AoA DPU	X	X	X
Datapath EDMA	X	X	
Object Detection DPC [1]	X	X	X

[1] No pre-built library for Object Detection DPC

6. 4. Algorithm

Algorithms can be found under mmwave_sdk_<ver>/packages/ti/alg folder. Currently algorithms applicable for mmwave functionality are provided under this folder:

Component	Source & Prebuilt Library	API Document (doxygen)	Unittest (source & prebuilt binary)
gtrack	X	X	X
mmwavelib	X	X	X

6. 5. Usecases

Usecases can be found under mmwave_sdk_<ver>/packages/ti/drivers/test folder.

Component	Source	API Document (doxygen)	Unittest (source & prebuilt binary)
csi_stream (IWR14xx only)	X	X	X
mem_capture	X	X	X

6. 6. Demos

Demos can be found under mmwave_sdk_<ver>/packages/ti/demo/<platform>. The following demos are included in the mmwave sdk package. Details on running demos can be found in the mmwave_sdk_user_guide.

Component	Source & Prebuilt Binary	Demo document (doxygen)	Demo GUI
mmw[1]	X	X	X

[1] Demo is supported on all devices except for xwr14xx in this release

6. 7. Misc folders

Following folders are also part of mmwave_sdk_<ver>/packages/ti folder.

- common: Common header files needed across all components
- platform: platform specific files
- utility: Contains
  - ccs debug utility which is the MSS/DSSbinary that needs to be flashed when connecting/developing using CCS (details can be found in mmwave_sdk_user_guide)
  - cli which is the cli helper utility used by the demos
  - cycleprofiler which is the helper utility used for profiling the various components inside the SDK
  - hsiheader which is a helper utility that creates a header for the data to be shipped over LVDS lanes.
  - mathutil is used to perform some common operations such as log2, rounding, saturation based on the core they need to run on (R4F, C674x)
  - secondary boot loader (sbl)
  - testlogger which is the helper utility for driver unit tests

6. 8. Scripts

Build scripts can be found in mmwave_sdk_<ver>/packages/scripts folder. Build instructions can be found in mmwave_sdk_user_guide.

6. 9. Firmware

RadarSS firmware for all supported devices is included under mmwave_sdk_<ver>/firmware/radarss folder. Procedure to flash the radarss is covered in the mmwave_sdk_user_guide.

6. 10. Tools

The following tools are included in the release in binary form. These can be found under mmwave_sdk_<ver>/tools folder.

- **Ftdi:** These Windows PC drivers are needed when interfacing to the board via FTDI port on MMWAVE-DEVPACK or MMWAVEICBOOST

6. 11. Docs

mmwave_sdk_<ver>/docs folder contains important documents related to the release such as

- mmwave_sdk_software_manifest.html: Software Manifest
- mmwave_sdk_release_notes.pdf: Release Notes (this document)
- mmwave_sdk_user_guide.pdf: User guide
- mmwave_sdk_module_documentation.html: Links to individual module's documentation

mmwave_sdk_<ver>/docs/relnotes_archive contains release notes from previous releases. Release notes contain migration information.

mmwave_sdk_<ver>/docs/test folder contains test results for each SoC. Each SoC folder in turn may contain multiple test group folders (such as module_test, alglib_test) which have the following files

- Report.html: Detailed Test report with links to logs
- *.log: Test logs for unit tests

7. Related documentation/links

Other than the documents included in the mmwave_sdk package the following documents/links are important references.

- SoC links:
  - Automotive mmWave Sensors
  - Industrial mmWave Sensors
- Evaluation Modules (EVM) links:
  - Automotive Evaluation modules (Booster Pack, DEVPACK)
  - Industrial Evaluation modules (Booster Pack, ISK)