MMWAVE SDK Release Notes

Product Release 1.1.0

Release Date: Oct 23, 2017

Release Notes Version: 1.0

CONTENTS

1 Introduction
2 Release overview
- 2.1 What is new
- 2.2 Platform and Device Support
- 2.3 Component versions
- 2.4 Tools dependency
- 2.5 Licensing

3 Release content
- 3.1 New Features
- 3.2 Migration section
- 3.3 Issues fixed
- 3.4 Known Issues
- 3.5 Limitations

4 Test reports
5 Installation instructions:
6 Package Contents
- 6.1 Drivers
- 6.2 Control
- 6.3 Algorithm
- 6.4 Demos
- 6.5 Misc folders
- 6.6 Scripts
- 6.7 Firmware
- 6.8 Tools
- 6.9 Docs

7 Related documentation/links

1. Introduction

The mmWave SDK enables the development of millimeter wave (mmWave) radar applications using xWR1443 and xWR1642 SOCs. 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

- mmWave suite enhancements
- Demo enhancements
- Memory optimizations
- Package/Build enhancements
- Tools changes

More details can be found in NewFeatures section.

2. 2. Platform and Device Support

The device and platforms supported with this release include:

Supported Devices	Supported EVM
AWR1443 ES2.0	AWR1443BOOST - AWR1443 Evaluation Module
AWR1642 ES1.0	AWR1642BOOST - AWR1642 Evaluation Module
IWR1443 ES2.0	IWR1443BOOST - IWR1443 Evaluation Module
IWR1642 ES1.0	IWR1642BOOST - IWR1642 Evaluation Module

> xWR14xx terminology is used in sections that are common for AWR14xx and IWR14xx
>
> xWR16xx terminology is used in sections that are common for AWR16xx and IWR16xx
>
> xWR14xx ES1.0 is not supported in this release.

> 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		1.1.0	Source and Binary	Overall package release version
RadarSS firmware		1.9.1	Binary	
mmWaveLink Framework		0.9.1	Source and Binary	
FTDI		2.12	Binary	
Image Creator	gen_bincrc32	1.0	Windows and Linux binary	
	out2rprc for xwr14xx	3.3	Windows binary	Need mono to run this on Linux
	out2rprc for xwr16xx	2.0	Windows binary	Need mono to run this on Linux
	Crc multicore image for xwr16xx	1.0	Windows and Linux binary	
	Multicore image generator for xwr16xx	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	7.1 or later	download link Please note that CCS v7.1 or later is mandatory. CCSv6.x cannot be used
TI SYS/BIOS	6.52.00.12	Included in mmwave sdk installer
TI ARM compiler	16.9.1.LTS	Included in mmwave sdk installer
TI CGT compiler	8.1.3	Included in mmwave sdk installer
XDC	3.50.00.10	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	3.2.8	Only for Linux builds
mmwave device support packages	1.5.3 or later	Upgrade to the latest using CCS update process (see SDK user guide for more details)
TI Emulators package	6.0.0576.0 or later	Upgrade to the latest using CCS update process (see SDK user guide for more details)
Pinmux tool (optional)	Latest	Used to generate pinmux configuration for custom board https://dev.ti.com/pinmux (Cloud version)
Doxygen (optional)	1.8.6	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
  - Drivers
    - ADCBuf drivers exposes a new API to get the ADC buffer channel address
    - Added capability for sending S/W triggered user data over CBUFF/LVDS interface
    - Added capability to transfer user defined headers along with the H/W and S/W triggered data over HSI interface
    - Enhanced the ESM driver to allow application to register callback functions on specific ESM errors
    - SPI driver now exposes new stats function
    - SPI data throughput can be improved by using special options in the SPI driver. Refer to the SPI driver doxygen for more details on caveats around these options.
    - Added DMA functionality to UART driver
    - Added Watchdog driver for MSS
  - mmWaveLink
    - Updated to support the RadarSS firmware version as noted above
  - mmWave
    - Enhanced mmWave APIs to separate one-time RadarSS API calls from reconfig capable API calls
    - Added support for advanced frame config
  - mmWavelib
    - Floating-point CFAR-CA:
      - mmwavelib_cfarfloat_caall supports CFAR cell average, cell accumulation, SO, GO algorithms, with input signals in floating point formats;
      - mmwavelib_cfarfloat_caall_opt implements the same functionality as mmwavelib_cfarfloat_caall except with less cycles, but the detected objects will not be in the ascending order.
      - mmwavelib_cfarfloat_wrap implements the same functionality as mmwavelib_cfarfloat_caall except the noise samples for the samples at the edges are the circular rounds samples at the other edge.
      - mmwavelib_cfarfloat_wrap_opt implements the same functionality as mmwavelib_cfarfloat_wrap except with less cycles, but the detected objects will not be in the ascending order.
    - Floating-point AOA estimation:
      - mmwavelib_aoaEstBFSinglePeak implements Bartlett beamformer algorithm for AOA estimation with single object detected, it also outputs the variance of the detected angle.
      - mmwavelib_aoaEstBFSinglePeakDet implements the save functionality as mmwavelib_aoaEstBFSinglePeak without the variance of detected angle calculation.
      - mmwavelib_aoaEstBFMultiPeak also implements the Bartlett beamformer algorithm but with multiple detected angles, it also outputs the variances for every detected angles.
      - mmwavelib_aoaEstBFMultiPeakDet implements the same functionality as mmwavelib_aoaEstBFMultiPeak but with no variances output for every detected angles.
    - DBscan Clustering:
      - mmwavelib_dbscan implements density-based spatial clustering of applications with noise (DBSCAN) data clustering algorithm.
      - mmwavelib_dbscan_skipFoundNeiB also implements the DBSCAN clustering algorithm but when expanding the cluster, it skips the already found neighbors.
    - Clutter Removal:
      - mmwavelib_vecsum: Sum the elements in 16-bit complex vector.
      - mmwavelib_vecsubc: Subtract const value from each element in 16-bit complex vector.
    - Windowing:
      - mmwavelib_windowing16xl6_evenlen: Supports multiple-of-2 length(number of input complex elements), and mmwavelib_windowing16x16 supports multiple-of-8 length.
      - mmwavelib_windowing16x32: This is updated to support multiple-of-4 length(number of input complex elements). It was multiple-of-8 previously.
    - Floating-point windowing:
      - mmwavelib_windowing1DFltp: support fixed-point signal in, and floating point signal out windowing, prepare the floating point data for 1D FFT.
      - mmwavelib_chirpProcWin2DFxdpinFltOut, mmwavelib_dopplerProcWin2DFxdpinFltOut: prepare the floating point data for 2D FFT, with fixed point input. The difference is mmwavelib_chirpProcWin2DFxdpinFltOut is done per chip bin, while mmwavelib_dopplerProcWin2DFxdpinFltOut is done per Doppler bin.
      - mmwavelib_windowing2DFltp: floating point signal in, floating point signal out windowing to prepare the floating point data for 2D FFT.
    - Floating-point vector arithmetic(power accumulation): accumulates signal powers in floating point version.
    - FFT utility: mmwavelib_dftSingleBinWithWindow calculates single bin DFT with windowing.
- mmWave demo
  - Added support for advanced frame config in mmW demo for xWR16xx.
  - Processing chain enhancement for xwr14xx mmw demo
  - Added new procedure and command in mmW demo to enable user to measure and compensate for range bias and rx ant phase error

        - Process multiple chirps per chirp available interrupt in xwr16xx demo
        - Implemented calls for basic CPU sleep (wake on interrupt) on R4F and DSP during idle task in mmW demos
        - Added high level summary of system memory distribution across code and data buffers in mmW demo doxygen
        - Added a perl based script to convert SDK 1.0 based mmW config files into compatible SDK 1.1 config files
        - [Experimental] Added algorithm in xwr16xx mmW demo to detect moving objects up to 2x the profile supported limit for velocity
        - [Experimental] Added Clutter Removal Algorithm to mmW demo
    - Memory optimization
        - Enabled build options (compiler, linker) in default SDK make infrastructure to create binaries that are memory optimized
        - Enabled build options in BIOS CFG to build C674x BIOS libraries that are memory optimized
        - Use thumb target R4Ft to build R4F BIOS libraries that are memory optimized
    - Package enhancements
        - mmWave SDK package now includes all the tools and components needed to build
        - Perl is no longer needed for building mmWave SDK
    - Tools
        - Upgraded FTDI drivers to the latest version to enable support on windows 10 as well
        - gen_bincrc32 is now windows and linux executable instead of perl utility
        - out2rprc for xwr14xx detects section misalignment and returns an error.

3. 2. Migration section

This section describes the changes that are relevant for users migrating to the mmWave SDK 1.1.0 release from 1.0.0 release.

Summary	Component/s	Subcomponent	Behavior of Impact
DMA functionality is added to UART driver	Driver	UART	Feature added: DMA can now be used for sending and receiving data to/from UART. Default is DMA support enabled for 16xx R4F and DSP whereas no DMA for 14xx R4F (due to limited code space in 14xx). Read/Write APIs provide the finer control on whether DMA is used for that particular operation Existing application code change: Existing application code which do not intend to use DMA will not see the API change as UART_Params_init() will take care of defaults. The application will however need to link to DMA library (which most likely it will be already doing it as part of other application/system needs)
Extend the ESM driver to register FIQ callback functions	Drivers	ESM	ESM_init now returns a handle which application can use to register notification callback on an intended ESM group error. If the application is not interested in registering a callback, then it doesnt need to change any code to save the ESM handle and can continue calling ESM_init as in SDK 1.0 release
Implement advanced frame config in mmWave API layer	mmWave		New MMWave_DFEDataOutputMode is added. MMWave_config structure has a new union member to provide the advanced frame config. Since chirps are common element between advanced frame config and the legacy frame config, the defines for MMWave_DFEDataOutputMode_CHIRP and its associated structure chirpModeCfg has been changed to MMWave_DFEDataOutputMode_FRAME and frameCfg to denote the legacy frame mode.
Enhance error code returned by mmWave APIs to provide more debugging information	mmWave		The errCode in/out parameter filled by mmWave APIs will now be an encoded error code with the following information:- - Error or Informational message - mmWave error code - Subsystem error code There exists a #MMWave_decodeError which can be used to determine the exact error code and error level. Existing application: Most likely the existing application won't need any change except if it wants to add more logic in decoding mmWave filled error codes. Note that this errCode is the last pointer input parameter in every API. Each API function still returns -1 on error. example: extern int32_t MMWave_config (MMWave_Handle mmWaveHandle, MMWave_CtrlCfg* ptrControlCfg, int32_t* errCode);
Restructure EDMA in 1D FFT processing to process N chirps at a time	xWR16xx Demos	Processing Chain, Config File	Existing CFG files will continue to work as is. New Feature: This new multi-chirp mode can be invoked by setting chirpThreshold to 0 (automatic) or >1.

xWR14xx mmW demo code optimization	Demos	xWR14xx mmW Demo	- L3 code overlay for one-time init code (same concept as 16xx DSP) - Most system_printf() calls were removed/replaced. The system_printfs() informing errors were replaced by debug_assert except for the cases where the errors were configuration/GUI recoverable errors (in these cases the system_printf() was not removed).
Split mmWave_config API into two APIs - a one time mmWave_open and a repetitive mmWave_config function	mmWave, Demos		Any demo using mmWave API layer will need to migrate to these new APIs. Previously, mmWave exposed 3 main APIs to start the sensor - a one time mmWave_init and repetitive mmWave_config and mmWave_start functions. Now, the functionality in mmWave_config is split into 2 functions - a one time mmWave_open (which will do all functionality until RfInit) and a repetitive mmWave_config function (By repetitive , I mean it can be called again after a mmWave_stop). Other functions of mmWave have some changes due to re-organization but the functionality has remained the same. Note that it is recommended to not call RF init more than once per power cycle. In our generic demos (both mmW and capture), this translates to open time config (chCfg, adcCfg, lowpower etc) cannot be changed using stop/reconfig/start sequence (after the first sensorStart).
Add support for advanced frame config in dsp_edma and demo/GUI	xWR16xx mmW Demos	Processing Chain, Config File, Output stream	Advanced frame is an optional feature that is added to xwr16xx mmW demo and can be invoked using new CLI commands. The users of legacy frame mode (i.e. defacto mode in SDK 1.0) will see some changes to the CLI commands and to the output stream. A perl script is provided in the demos/profile directory for users to easily convert their existing SDK 1.0 based CFG to SDK 1.1 compatible CFG so that they can be run against the SDK 1.1. demo code/binary.
Functionality added to S/W trigger CBUFF transfer for shipping out user buffer	Drivers, Demos	CBUFF, Capture Demos	New APIs for CBUFF driver - see driver doxygen for more details. Highlights: - CBUFF_open/CBUFF_close are now replaced with CBUFF_createSession/CBUFF_deleteSession. - A new pair of APIs are introduced - CBUFF_activateSession and CBUFF_deactivateSession - to switch between different sessions created by the application. Only one session can be in progress i.e. active at a time. setHSI command in the capture demo is now enhanced to allow configuring these multiple sessions and output data type. Known caveat: ---------- CBUFF/LVDS: Refer to mmWave SDK user guide on data size restriction
API changes due to new RadarSS firmware	Demos, mmWave, mmWaveLink		Removed Redundant/Depcrecated fields from mmWaveLink data structures: 1) anaChannelCfg in rlLowPowerModeCfg_t 2) txEnable in rlContModeCfg_t
Driver API changes for compliance with static code analysis	Drivers, mmWaveLib		See doxygen for these modules for APIs changes: mmWaveLib: mmwavelib_dftSingleBin() EDMA: transferCompletionCallbackFxn_t(), EDMA_errorInfo_t()
Provide a bundled package with mmWave SDK that includes all the dependent tools	Build		Easier installation process. Less need for environment configuration to be supplied by user. The mmwave sdk package now includes all the TI tools that are needed for building mmwave sdk. So there is no longer need for downloading/installing tools separately or setting up environment variables for each. The package includes the scripts that setup environment variables automatically based on the installation folder.
add c674 dsplib to tools dependency since mmwavelib floating point lib depends on this	Build		SDK builds now need a new define for C674X DSP Lib path variable. Since SDK 1.1 will have a bundled package, tools installation and path setting will be automatically done by the SDK installer and no changes are needed to be done by the user.
Removed PERL as dependency in SDK build process	Build		A perl based application was used to generate the CRC needed for the flash version of the application bin file.This perl based application has been replaced with a C-based application/executable. This change is already absorbed as part of the SDK package and generateBin helper utilities have been updated accordingly. This change will be transparent to users who choose to use the generateBin utility. For users creating their own version of generateBin utility should replace the perl file with gen_bincrc32 executable.
Build warnings are treated as errors	Build		By default, the SDK build uses "–emit_warnings_as_errors" option to help users identify certain common mistakes in code that are flagged as warning but could lead to unexpected results. If user desires to disable this feature, then please set the flag MMWAVE_DISABLE_WARNINGS_AS_ERRORS to 1 in setenv.bat or setenv.sh and invoke that file again to update the build environment.

Use default build options in SDK makefiles for optimizing code size of libraries and application	Build	Pre-built Libraries, SDK common makfiles, mmW demo BIOS CFG file	SDK default build option is -ms0 for C674x components and application and thumb for R4F components and application. All pre-built binaries in SDK (except for mmWaveLib) will be built using this option. Any customer application using mmwave_sdk.mak for compiler and linker options will automatically see this change and will be built using these option. SYSBIOS provides a new target called R4Ft for building the BIOS code in thumb mode. This option saves code memory at a nominal cost in cycles.
out2rprc will now return error for unaligned sections for xwr14xx and SDK batch files will tap on that error	Build		When such error is detected by out2rprc utility, no valid bin file will be generated by the SDK build infrastructure.

3. 3. Issues fixed

The following issues from previous releases were fixed in this release

Issue Type	Key	Summary
Bug	MMWSDK-871	Vizualizer GUI produces config with out of bounds sampling rate for xWR16xx
Bug	MMWSDK-815	Peak grouping configuration does not work when min or max range indices are bigger than 255.
Bug	MMWSDK-758	CBUFF Module includes CSI2 header files
Bug	MMWSDK-712	Build fails when DOWNLOAD_FROM_CCS flag is set to NO
Bug	MMWSDK-704	update capture demo post processing matlab script to strip out headers
Bug	MMWSDK-703	update sampling rate in capture demo as per device spec
Bug	MMWSDK-678	AWR14xx binary utility (out2rprc) issue when TCMB is used for code sections
Bug	MMWSDK-677	Mismatch in ADC data format in XWR16xx Capture Demo
Bug	MMWSDK-670	multiObjBeamForming command in the profileCfg does not take effect.
Bug	MMWSDK-669	Wrong string for error print in MMW DSS application
Bug	MMWSDK-665	out2rprc not returning error for unaligned sections for xwr14xx
Bug	MMWSDK-631	Capture Demo application goes into Abort while enabling Continuous mode
Bug	MMWSDK-629	mmwave_sdk_setupenv has a typo in the help for set XWR16XX_RADARSS_IMAGE_BIN
Bug	MMWSDK-628	Chirp Config parameters sent over CLI doesn't get converted properly in cli_mmave.c
Bug	MMWSDK-595	Address issue of ghost objects appearing at random angles from a real object due to multiObjBeamForming feature
Bug	MMWSDK-591	mmw demo azimuth heatmap does not rescale after changing range depth/width and using just the stop/start button
Bug	MMWSDK-586	xWR14xx demo displays many invalid detections due to loss of precision
Bug	MMWSDK-575	GUI shows Y-axis truncation of range profile for given configuration for xWR16xx
Bug	MMWSDK-574	MMWave_stop fails when finite frames are configured due to information return code from BSS
Bug	MMWSDK-573	UART Driver: Incorrect handling of the interrupts
Bug	MMWSDK-562	xWR14xx mmw demo should abort when memory allocation fails
Bug	MMWSDK-218	Significant bias of few cm seen in range measurement in mmw demo.

3. 4. Known Issues

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

Issue Type	Key	Summary	Comments
Bug	MMWSDK-860	Mailbox Driver Handle definition conflicts with same definition in SysBIOS	Workaround: Edit the definition in mmWave SDK mailbox driver and recompile the driver
Bug	MMWSDK-840	Azimuth heat map is no longer at zero Doppler for the improved scheme (DATA_PATH_CHAIN_COMBINED_LOGMAG) implemented in MMWSDK-586	
Bug	MMWSDK-763	ghost objects seen with OOB demo [on about 50% of boards] for near by objects due to violation of the "far field" assumption	
Bug	MMWSDK-651	Small chirp times causes crash (MIPS overflow) in 1D processing.	
Task	MMWSDK-533	GUI of mmw demo running slow from Firefox browser	Workaround: Please switch to Chrome browser.
Bug	MMWSDK-319	CAN driver: DMA mode is not supported	
Story	MMWSDK-252	UART driver has not tested for Data Length 5 and 6	

3. 5. Limitations

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

1	ADCBuf driver: CQ feature is not implemented.
2	CAN driver: • DMA and FIFO mode are not supported
3	CANFD driver: • DMA and Timestamping are not supported

4	CBUFF/CSI2/LVDS:
	• Limited functionality is tested using stream demo
	  ◦ Only ADC data format and user buffer with and without headers have been verified.
	  ◦ Continuous mode support is experimental.
	• Driver does not support the following functionality:
	  ◦ Multiple packets
	  ◦ 3 channels

5	CRC driver: "Auto" mode is not implemented.
6	DMA driver: MPU and Parity Feature not implemented.
7	EDMA driver: Privilege feature not implemented.
8	HWA driver: Any modes/algorithm outside the scope of mmWave demo are not tested (however they are implemented in the driver).
9	I2C driver: Verified only loopback mode on mmWave device TI EVM (however all features are implemented in the driver).
10	QSPI/QSPI Flash driver: ▪ dual-Read/Quad read in configuration mode is not supported ▪ setting write protections bits is not supported

11	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.
12	DMA based transactions are not supported for CRC and Mailbox driver.
13	capture demo: ▪ Data equivalent to the L3 memory available on a given device can be collected in a given execution. ▪ CSI-2: The demo has only been verified in Raw 16 mode with the Data format set to CBUFF_DataFmt_ADC_DATA. ▪ LVDS: The demo has only been verified in the 16bit output format in DDR mode with the Data format set to CBUFF_DataFmt_ADC_DATA and with software triggered session for user data (headers are always enabled). ▪ Testing was performed using the configuration available in the demo directory (See SDK User Guide for more details)
14	mmW demo: See demo's doxygen page for more details.

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.

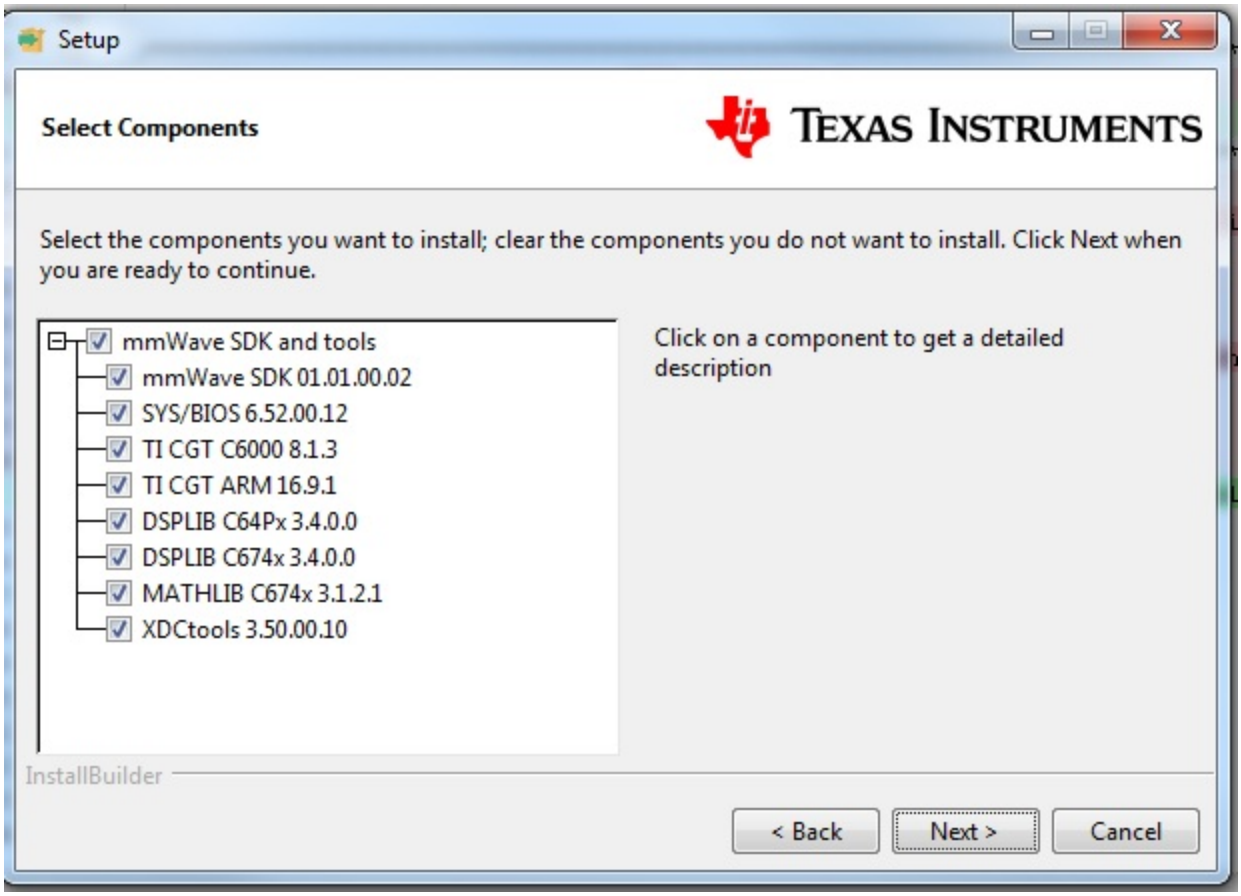
- **mmwave_sdk_<version>-Windows-x86-Install.exe: Windows installer verified on Windows 7 and Windows 10 machines**
- **mmwave_sdk_<version>-Linux-x86-Install.bin: Linux installer verified on an Ubuntu 14.04 & Ubuntu 16.04 64 bit machines.**

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,
  - 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

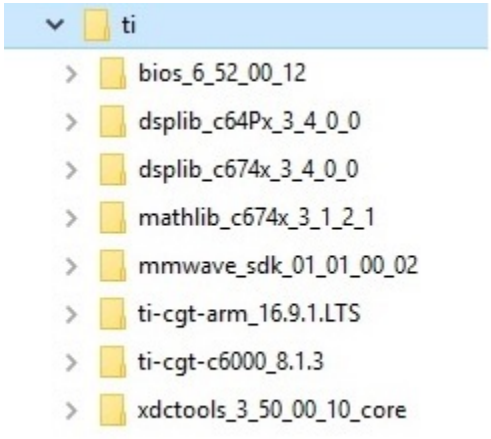
Installation steps:

- Setup
- License Agreement: Accept the license agreement
- Choose Destination Location: Select the folder to install (default is c:\ti on windows and ~/ti on linux)
- Select Components: The installer now includes all the tools needed for building the mmWave SDK. You should see a screen like below (the exact version in the installation may be different from the one shown below). The only reason to deselect a tool is if the tool is already installed in the destination folder.

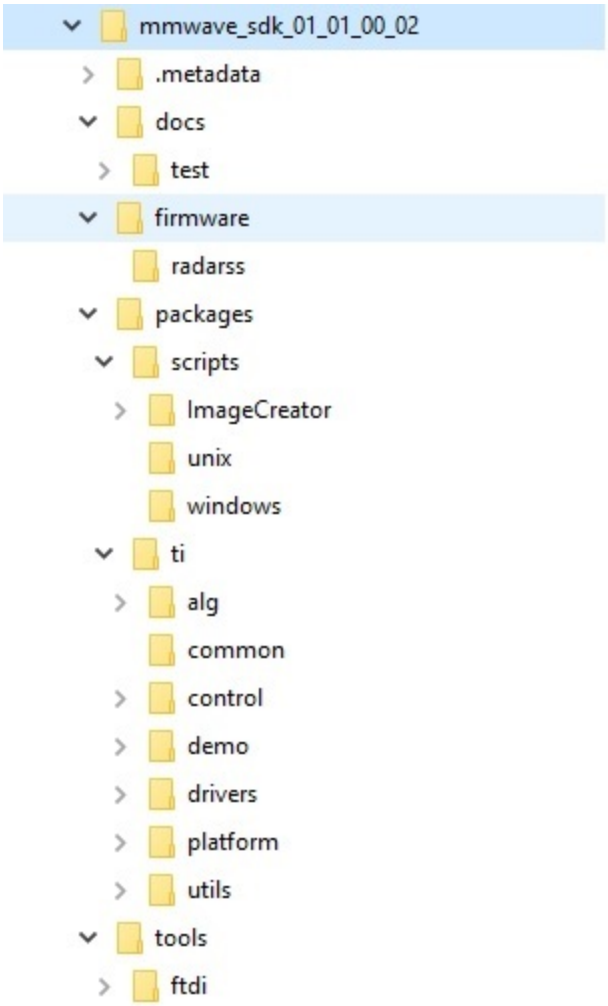


- Review installation decisions
- Ready to install
- Once installation starts all the selected components will be installed (if a component with the same version exists in the destination folder it will be overwritten)
- Installation complete

After the installation is complete the following folder structure is expected in the installation folder (please note that the version numbers may not be same as the one illustrated below)



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

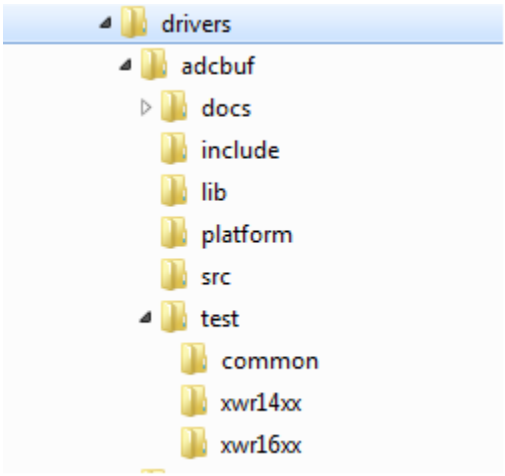


6. Package Contents

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 that <platform: xwr14xx, xwr16xx>
- 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
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

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)
mmwavelink framework	X	X	X

mmwave high level api	**X**	**X**	**X**

6. 3. 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)
mmwavelib	**X**	**X**	**X**

6. 4. 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
capture	**X**	**X**	
mmw	**X**	**X**	**X**

6. 5. 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
  - testlogger which is the helper utility for driver unit tests

6. 6. Scripts

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

6. 7. 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. 8. 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 MMWAVE-DEVPACK

6. 9. 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
- mmwave_sdk_user_guide.pdf: User guide

mmwave_sdk_<ver>/docs/relnotes_archive contains release notes from previous releases

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:
  - AWR1443
  - AWR1642
  - IWR1443
  - IWR1642
- EVM links:
  - AWR1443BOOST
  - AWR1642BOOST
  - IWR1443BOOST
  - IWR1642BOOST
  - MMWAVE-DEVPACK