CC3200 SimpleLink™ Wi-Fi® and Internet-of-Things Solution, a single-chip wireless MCU

Software Development Kit (SDK) v1.2.0 Release Notes

TABLE OF CONTENTS

1 Introduction

This document describes the Software Development Kit (SDK) version 1.2.0 for use with the CC3200 SimpleLink Wi-Fi MCU device mounted on the CC3200 LaunchPad evaluation kit.

2 Getting Started

Please follow the on-line CC3200 Quick Start Guide to start using the CC3200 LaunchPad development platform.

Please download the CC3200 Getting Started Guide to get started with your project development.

2.1 Procedure to Upgrade from SDKv1.1.0 to SDKv1.2.0

To upgrade from SDKv1.1.0 to SDKv1.2.0, servicepack "servicepack-1.0.1.6-2.6.0.5" needs to be flashed on CC3200. Service pack "servicepack-1.0.1.6-2.6.0.5" is hosted in *CC3100_CC3200_ServicePack-1.0.1.6-2.6.0.5-windows-installer.exe* downloadable from http://www.ti.com/tool/cc3200sdk. Please refer to UNIFLASH Quick start guide on details of flashing (http://processors.wiki.ti.com/index.php/CC31xx_%26_CC32xx_UniFlash) the service pack.

3 Main changes from SDK Package 1.1.0

3.1 Networking

3.1.1 SSL – Domain Server name verification

The option to verify the domain server name is enabled.
This feature require updated host driver.

3.1.2 SSL - Support SHA384 on certificate chain verification

Enabling SHA384 during certificate chain verification
Clarification: As a server, the device is not supporting client authentication with SHA384 on the chain

3.1.3 SSL - Add support for new chipper suites in client mode

Enable the following chipper suites in client mode:

- TLS1_2_TLS_RSA_WITH_AES_128_CBC_SHA256
- TLS1_2_TLS _RSA_WITH_AES_256_CBC_SHA256
- TLS1_2_TLS _ECDHE_RSA_WITH_AES_128_CBC_SHA256
- TLS1_2_TLS _ECDHE_ECDSA_WITH_AES_128_CBC_SHA256

3.1.4 Wi-Fi Enterprise security – Disable server authentication

Enabling the option to disable the server authentication using Wi-Fi Enterprise security
Disable the authentication is available by using WlanConnect command only (it's not kept in the profiles)
Note: Disable need to be sent prior to every WlanConnect command

3.1.5 Reduce SFLASH access

Remove some write access to the SFLASH while exiting shutdown

3.1.6 DHCP Enhancements

Enabling the option to retrieve the DHCP lease time and default gateway address
Improve the DHCP handshake robustness

3.1.7 DNS Enhancements

DNS enhancements were added in order to improve the robustness
Enabling the option to temporary set the amount of retries per DNS query
(For reference: 5 retries = ~1.5Sec, 11 retries = ~5Sec Default 32 retries = ~18Sec)
Enabling the option to retrieve the secondary DNS server address and temporary set it

3.1.8 Host Driver fixes and code optimizations

- Robustness fixes
- Allow easier integration with Network Application Libraries
- Adding Timeout mechanism
    - The driver has the ability to detect *communications failures (timeouts) between the host and the SL device*
    - This mechanism enables the driver to identify the timeouts and notify to the application
    - To enable this service, the application need to implement a function to get a timestamp and to register to the sl_GeneralEvtHdlr
- Improving the Error indications
    - The driver has the ability to detect and notify the application upon driver operations failures during SL API call that may cause the application to get stuck
- To get this service, the application needs to register to the sl_GeneralEvtHdlr (user.h)

3.2 SDK Content

Please refer to Section 11 (Revision History) for changes in the SDK components.

4 Release Content

Item	Version	Type
Device	CC3200R1 [ES1.33 Chip Id : 0x4000010]	Production device
Development boards	CC3200-LAUNCHXL Rev3.2 onwards	Orderable from TI
SDK Installer	CC3200SDK-1.2.0-windows-installer.exe For Windows 7,8 and Windows XP	Provided with a click wrap license
Firmware	2.6.0.5.31.1.4.0.1.1.0.3.34	*servicepack-1.0.1.6-2.6.0.5* is provided through *CC3100_CC3200_ServicePack-1.0.1.6-2.6.0.5-windows-installer.exe* downloadble from http://www.ti.com/tool/cc3200sdk
Network Processor host driver	Version 1.0.1.6	Source code
MCU Peripherals Drivers	Version 1.2.0	Source code
Power Management Framework Library	Version 1.2.0	Source code
Supported IDE	IAR version : 7.30 onwards CCS version : 6.1.1	Accessible separately. Not a part of this package
Demo	Embedded HTML web-site	Pre-flashed on LaunchPad and source code provided
User guides	CC3200 Getting started guide CC3200LAUNCHXL User Guide Peripheral Driver Library User's Guide Power Management Framework Guide Application notes for sample applications Simplelink OTA (Over-The-Air) update User's Guide Quick start Guide TI-RTOS User Guide SimpleLink Host Driver Programmer's Guide	PDF PDF Doxygen CHM PDF PDF PDF PDF Doxygen HTML
Tools	USB driver for CC3200LAUNCHXL for Windows7, 8 and Windows XP	Executable

5 Directory structure of SDK

Double-Click on the package installation file and follow the installation guidelines to copy the directories (and files) to the preferred location.
The first level directory structure is as shown in the table below.

Directory Name	Content
Docs	• Getting Started Guide for application development • Launchpad User Guide • SimpleLink Host Driver Programmer's Guide • Peripheral Driver Library User's Guide • Power Management Framework Guide • Application notes for sample applications • Simplelink OTA (Over-The-Air) update User's Guide • Quick start Guide • TI-RTOS User Guide • Programmer's Guide • API User's Guide for MQTT and HTTP libraries
Examples	Example application in source code
Driverlib	• Peripheral driver library source files • The driverlib.a is provided in CCS, GCC and IAR directories
Inc	• Register definition header files
Middleware	• Power management framework providing an easy to use infrastructure for power aware solution
Oslib	• Interface file to configure Free-RTOS or TI-RTOS
SimpleLink	• The SimpleLink Network Processor host driver code. • Pre-built simplelink library (simplelink.a) for OS, NON_OS, PM_framework and NON_OS_PM are available in CCS and IAR directories. • For GCC compiler libsimplelink.a and libsimplelink_nonos.a are available in gcc folder. • Optimized library for nonos configuration are present in all the directories(IAR, CCS, and GCC)
Netapps	• SMTP client library source files • XMPP client library source files • HTTP server library source files • HTPP client library source files • TFTP client library source files • JSON parser library source files • MQTT client library source files • MQTT client server library source files
third_party	• FatFS source files

	• FreeRTOS source files
ti_rtos	• Abstraction layer files for TI-RTOS
tools	• **ccs_patch –** Files required for CCS-FTDI-LP connection • **gcc_scripts –** Script files required for GCC compiler. • **ftdi-** Contains CC3200 FTDI-USB driver for Windows • **Iar_patch –** Files required for IAR-FTDI-LP connection

6 MCU and Associated Peripherals

CC3200 Device provides a user programmable Cortex M4 core in conjunction with user programmable peripheral. Drivers and example applications are provided in the SDK package for the following peripheral interfaces -

ADC	Analog to digital converter.
AES	Advanced encryption standard
CAMERA	Parallel interface (8 bits) that can be used to connect to camera sensors.
CRC	Cyclic redundancy check
DES	Data encryption standard.
GPIO	General purpose input/output
I2C	Standard I2C interface to communicate with various sensors
Interrupt	Interrupt module
I2S	Audio signal processor
PinMux	Pinmux setting for Pads
PRCM	Power reset and clock module
SDHost	Secure digital host controller
SHAMD5	Secure hash algorithm, message digest algorithm
SPI	Serial peripheral interface
Systick	System tick functionality
Timer	General purpose timers
UART	Standard UART interface
UDMA	Direct memory access
WDT	Watchdog timer

7 Networking

7.1 Package Quality

7.1.1 Interoperability - IOP

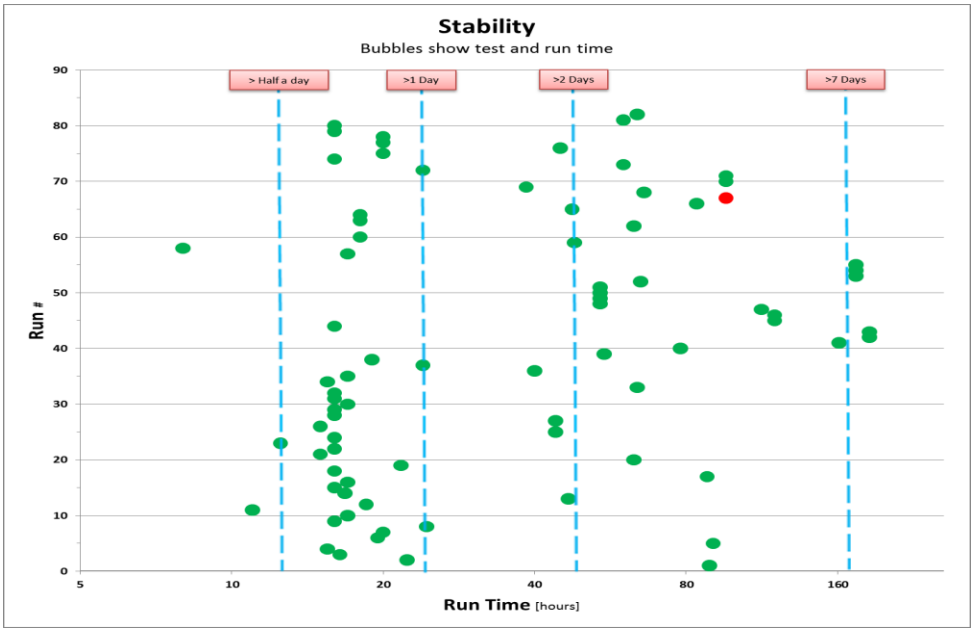
- STA mode was tested for connection, traffic and power consumption with more than 200 AP
- AP mode was tested for connection and traffic with more than 50 STA

7.1.2 Robustness

- Use cases were tests for 1000 of cycles – for example:
  - Connect/Disconnect
  - On/Off
  - Connect, Send Packet, Disconnect

7.1.3 Stability

- Stability in all traffic scenarios was tested for at least 12 hours (major use cases were tested for at least 24hours) and upto 7 days – User may rarely experience:
  - Traffic Stops
  - System freeze
- STA was kept running different traffic scenarios in open AIR for long duration
- Green Dot - STA was stopped after a set time
- The Failure tests (red dot) is representing TCP Tx with 163Byte payload in congested environment, the test fail due to socket disconnect after 67Hours (3 other instances passed >70hours)



7.2 Features

7.2.1 Wi-Fi

Standards	802.11b/g/n (20MHz SISO) **Station** and Wi-Fi Direct **Client**
Supported Channels	1-13 The default regulatory domain is US (1-11)
Personal Security	WEP, WPA and WPA2
Enterprise Security	WPA-2 Enterprise EAP Fast, EAP PEAPv0 MSCHAPv2, EAP PEAPv0 TLS, EAP PEAPv1 TLS, EAP TLS, EAP TTLS TLS, EAP TTLS MSCHAPv2
Provisioning	SmartConfig™ technology Wi-Fi Protected Setup (WPS2) Access Point mode with internal HTTP Web Server
Standards	802.11b/g **Access Point** and Wi-Fi Direct **Group Owner**
Clients	1
Personal Security	WEP, WPA and WPA2

7.2.2 Networking protocols

IP	IPv4
Transport	UDP TCP RAW ICMP
Cross-Layer	DHCP ARP DNS
Application	mDNS DNS-SD HTTP 1.0 web server
Transport Layer Security	SSLV3 SSL_RSA_WITH_RC4_128_SHA SSLV3 SSL_RSA_WITH_RC4_128_MD5 TLSV1 TLS_RSA_WITH_RC4_128_SHA TLSV1 TLS_RSA_WITH_RC4_128_MD5 TLSV1 TLS_RSA_WITH_AES_256_CBC_SHA TLSV1 TLS_DHE_RSA_WITH_AES_256_CBC_SHA TLSV1 TLS_ECDHE_RSA_WITH_RC4_128_SHA TLSV1 TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA TLSV1_1 TLS_RSA_WITH_RC4_128_SHA

	TLSV1_1 TLS_RSA_WITH_RC4_128_MD5 TLSV1_1 TLS_RSA_WITH_AES_256_CBC_SHA TLSV1_1 TLS_DHE_RSA_WITH_AES_256_CBC_SHA TLSV1_1 TLS_ECDHE_RSA_WITH_RC4_128_SHA TLSV1_1 TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA TLSV1_2 TLS_RSA_WITH_RC4_128_SHA TLSV1_2 TLS_RSA_WITH_RC4_128_MD5 TLSV1_2 TLS_RSA_WITH_AES_256_CBC_SHA TLSV1_2 TLS_DHE_RSA_WITH_AES_256_CBC_SHA TLSV1_2 TLS_ECDHE_RSA_WITH_RC4_128_SHA TLSV1_2 TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA TLS1_2 TLS_RSA_WITH_AES_128_CBC_SHA256 TLS1_2 TLS _RSA_WITH_AES_256_CBC_SHA256 TLS1_2 TLS _ECDHE_RSA_WITH_AES_128_CBC_SHA256 TLS1_2 TLS _ECDHE_ECDSA_WITH_AES_128_CBC_SHA256 client
User application sockets	Up to 8 open sockets Up to 2 secured application sockets: - One server (listen socket and accept socket) + client (data socket) - Up to two clients (data socket)

7.2.3 Advanced Features

802.11 Transceiver	Transmit and Receive raw Wi-Fi packets with full control over payload. Wi-Fi disconnect mode. Can be used for general-purpose applications (e.g. tags, sniffer, RF tests)
Traffic Filters	Embedded filters to reduce power consumption and Wake-on-LAN trigger packets (IP and MAC layer)

7.2.4 Power modes

MCU Power Management framework library allows user applications to exploit the following device power management modes:

- Sleep
- Low Power Deep Sleep
- Hibernate

The Wi-Fi network processor core supports the following low power policies

Low Power mode	Uses 802.11 Power Save and Device Deep Sleep Power with three user configurable policies
Configurable Power Policies	• Normal (Default) - Best tradeoff between traffic delivery time and power performance • Low power –Used only for Transceiver mode application

	(Disconnect mode) • Long Sleep Interval – wakes up for the next DTIM after a configurable sleep interval, up to 2 seconds. This policy is only applicable for client socket mode

8 Advanced Information

8.1 System/Software Capabilities

- Host SPI interface max speed: 20MHz
- Robustness tests
  - Start/Stop with Wi-Fi Connect/Disconnect and data Tx burst was tested for 5000 cycles and found to be stable
  - Wi-Fi Connect/Disconnect without data was tested for 5000 cycles and found stable
- TCP/IP
  - TCP Window size is 32KB for production device
  - The memory resources are divided among all user sockets and the TCP windows size might change accordingly
  - IP Fragmentation is not supported for Tx UDP and RAW sockets
  - In connection mode Tx and Rx traffic should be done after IP is acquired
  - Max Tx payload for Raw packet with IP header is 1460 bytes
  - Max Tx payload for Raw Transceiver (disconnected mode) is 1476 bytes (including data and header)
  - Min Tx payload for Raw Transceiver (disconnect mode) is 14 bytes (including Data and Header)
  - Closing socket should be done in a proper way (for example not to close a socket while there is blocking receive command on it) - a timeout can be used in this scenarios
  - TCP socket keep alive timeout is set to 5Min (non configurable)
- SSL/TLS
  - Elliptic-curve based ciphers (e.g. ECDH) implies a longer connection time
  - Supported modes
    - Up to one Server (Listen Socket and Accept Socket) + Client (Data socket)
    - Up to Two clients (Data socket)
  - Certificate Authority (CA) certificates needs to be installed if server authentication is required
  - Client mode –
    - Signature authentication check – must be less or equal to 2048
    - Key exchange and challenge – must be less or equal to 2048
    - Client authentication – must be less or equal to 2048
  - Server mode –
    - Signature authentication check – must be less or equal to 2048
    - Key exchange and challenge – must be less or equal to 2048
    - Client authentication – must be less or equal to 2048
  - Packets will be truncated above 1386Bytes (two TCP packets will be transmitted)
- Tx Power
  - Tx power in AP mode takes effect only after reset
- Wi-Fi Direct
  - In Group Owner mode FAST connection policy should be set to TRUE

- Rx Filters
  - BSSID can't be filtered while STA is connected (if filtered will cause disconnection)
- Power Management
  - The device will remain in active after initialization until the host reads all events
- File System
  - Up to 100 user files
  - File size is limited to ~1Mbyte[1,040,384byte] (No error will be returned while trying to create a larger size)
- Setting device Mode
  - Changing the device role (STA<->AP<->P2P) requires to reset the device
  - Setting network configurations after setting the device role (without reset) can lead to system halt
  - If SetRole to station was issued during AP mode it won't accept connected STA (See first bolt, reset is required)
  - Setting the device mode is persistent and SFLASH endurance must be considered on use cases that requires switching between roles
  - Network configuration is applicable to the current role of the device
- Default State - With no other configuration the default state of the device is as follows:
  - STA mode
  - Regulatory domain is US (channel 1-11)
  - Connection policy – AutoStart and AutoSmartConfig
  - DHCP - Enable
- HTTP Server
  - Support HTTP 1.0
  - Built-in ROM WEB Pages
  - Additional WEB pages could be stored on the File System
  - Dynamic content through proprietary Token mechanism (limited to 64 Characters)
- HTTP internal WEB Pages – main limitations
  - Values entered are not validated – for example:
  - Adding longer/short key in password fields (will be accepted)
  - Typing letters in DHCP lease time (instead of numbers)
  - WPA password is requested to be entered in Hex format when it should be ASCII
  - The length of the AP SSID field is limited to 15 characters (instead of 32)
  - The length of the AP Password field is limited to 24 characters (instead 63)
  - The length of the Device name is limited to 15 characters (instead of 32)
  - Adding/configuring Hidden SSID is not supported
  - HTTP authentication (user name and password) should be disabled in order not to get into a lock state
  -
- WEP
  - Supporting only WEP open using ASCII pre shared key however a small code can be used to support Hex format (more details and code example included in the programmer's guide)

- WPS
  - Up to 4 seconds delay between association and EAPOL-Start
- SmartConfig
  - Not supported with 5GHz AP (802.11a/n/ac)
  - Not supported for MIMO-capable configuration devices
  - Not supported with non-standard proprietary modulation schemes
  - Only Group 0 is supported in auto start mode
  - In Auto Start Mode the key is transferred not encrypted
- Enterprise Security
  - WPA2-TLS connection is not supported with v3 certificates
  - Connection is successful with expired date certificates
- Serial Flash

  CC3200 supports JEDEC specification compliant Serial Flash devices with 4Kbyte sector size erase. The following parts were validated:
  - Micron N25Q128-A13BSE40 128Mbit
  - Spansion S25FL208K 8Mbit
  - Winbond W25Q16V 16Mbit
  - Adesto AT25DF081A 8Mbit
  - Macronix MX25L12835F-M2 128Mbit
  - Macronix MX25R6435 64Mbit
  - ISSI IS25LQ016 16Mbit

Updated list and more recommendation can be found in: CC3100 & CC3200 Serial Flash Guide

9 Networking sample applications

The release package includes several sample applications developed for the CC3200 Cortex M4 processor. The applications come with

- Detailed Application Note (readme documents)
- Project files for IAR, CCS or Both as well as GCC makefiles for a few applications

9.1 Antenna Selection

This is a reference implementation for antenna-selection scheme running on the CC3200 MCU, to enable improved radio performance inside buildings

9.2 Connection Policies

This application demonstrates the usage of the CC3200 profiles and connection-policies.

9.3 Send Email

This application sends an email using SMTP to a user-configurable email address at the push of a button.

9.4 Enterprise Network Connection

This application demonstrates the procedure for connecting the CC3200 to an enterprise network.

9.5 File System

This application demonstrates the use of the file system API to read and write files from the serial Flash.

9.6 Get Time

This application connects to an SNTP cloud server and receives the accurate time.

9.7 Get Weather

This application connects to 'Open Weather Map' cloud service and receives weather data.

9.8 Getting Started in AP Mode

This application configures the CC3200 in AP mode. It verifies the connection by pinging the connected client.

9.9 Getting Started in STA Mode

This application configures the CC3200 in STA mode. It verifies the connection by pinging the connected Access Point.

9.10 HTTP Server

This application demonstrates using the on-chip HTTP Server APIs to enable static and dynamic web page content.

9.11 MDNS

This application registers the service for broadcasting and attempts to get the service by the name broadcasted by another device.

9.12 Mode Configuration

This application demonstrates switching between STA and AP modes.

9.13 NWP Filters

This application demonstrates the configuration of Rx-filtering to reduce the amount of traffic transferred to the host, and to achieve lower power consumption.

9.14 P2P (Wi-Fi Direct)

This application configures the device in P2P (Wi-Fi Direct) mode and demonstrates how to communicate with a remote peer device.

9.15 Provisioning with SmartConfig

This application demonstrates the usage of TI's SmartConfig™ Wi-Fi provisioning technology. The *Wi-Fi Starter Application* for iOS and Android is required to use this application. It can be downloaded from following link: http://www.ti.com/tool/wifistarter or from the Apple App store and Google Play.

9.16 Provisioning with WPS

This application demonstrates the usage of WPS Wi-Fi provisioning with CC3200.

9.17 Scan Policy

The application demonstrates the scan-policy settings in CC3200.

9.18 SSL/TLS

The application demonstrates the usage of certificates with SSL/TLS for application traffic privacy and device or user authentication

9.19 TCP Socket

The application demonstrates simple connection with TCP traffic.

9.20 Transceiver Mode

The application demonstrates the CC3200 transceiver mode of operation.

9.21 UDP Socket

The application demonstrates simple connection with UDP traffic.

9.22 XMPP Client

The application demonstrates instant messaging using a cloud based XMPP server.

9.23 File Download

This application demonstrates file download from a cloud server to the on board serial Flash.

9.24 Hibernate

This application demonstrates the hibernate ultra-low-power mode in a networking application using an UDP client.

9.25 Serial Wi-Fi

This application implements a wireless terminal over a Wi-Fi network and an “AT commands” like control interface

9.26 Out of Box Application

This application demonstrates the out of box experience where user can view different demo and SDK web links on their web-browser.

9.27 Wi-Fi Audio

This application demonstrates Bi-directional wireless audio on a CC3200 LaunchPad. It uses two LaunchPad boards in STA mode and streams the audio from one LaunchPad to the other.

9.28 Camera Application

This example demonstrates image capture using the CC3200 fast parallel interface

9.29 Sensor Profile

This application shows how to use hibernate mode through the Power Management Framework

9.30 Idle Profile

This Application exercises low power modes (LPDS) using Power Management Framework (middleware).

9.31 Watchdog System Demo

This application illustrates full system recovery, using watchdog, including network subsystem.

9.32 Websocket Camera

This application illustrates Websocket HTTP Server functionality with camera JPEG streaming to HTML 5 based web client

9.33 TFTP Client

This application illustrates Trivial File Transfer Protocol client by reading/writing file on TFTP server

9.34 HTTP Client Demo

This application illustrates the usage of HTTP Client library to enable the device as an HTTP Client

9.35 Idle Profile (Non OS)

This application exercises the low power modes (LPDS) using Power Management Framework in a Non OS environment.

9.36 MQTT Client

This application showcases the device acting as a MQTT client in a fully functional MQTT network

9.37 MQTT Server

This application showcases the device acting as an MQTT Server capable of managing multiple local clients as well as allowing the local clients to communicate with remote MQTT clients.

9.38 OTA Update

This application illustrates Over-The-Air (OTA) update of Service Pack, User application and User files.

9.39 Power Measurement

This application enables the user to measure current consumption for various power modes

10 MCU sample applications

10.1 LED Blink

This application uses the GPIO DriverLib APIs to blink an LED on the CC3200 Launchpad.

10.2 Timer Demo

This application demonstrates the CC3200 timer DriverLib APIs. It uses 16 bit timers to generate interrupts which in turn toggle the state a GPIO driving LEDs.

10.3 Watchdog Demo

This application demonstrates the use of the Watch Dog timer (WDT) DriverLib APIs. It shows how the watchdog timer resets the device on system failure.

10.4 UART Demo

This application uses the UART DriverLib APIs to demonstrate typing echo on a UART terminal.

10.5 Interrupt Application

This application uses the Interrupt DriverLib APIs to demonstrate the CC3200 MCU interrupt preemption and tail-chaining capabilities.

10.6 I2C Demo

This application uses the I2C DriverLib APIs to read and write the I2C peripherals on the CC3200 LaunchPad.

10.7 MCU Sleep

This application demonstrates the Sleep mode of the CC3200 MCU.

10.8 uDMA Application

This application uses the UDMA DriverLib APIs to show various DMA mode functionalities.

10.9 FreeRTOS Demo

This application demonstrates using FreeRTOS for multiple task creation and inter-task communication using queues.

10.10 AES Demo

This application uses the AES Driverlib APIs to exercise various AES encryption modes.

10.11 DES Demo

This application uses the DES Driverlib APIs to exercise various AES encryption modes.

10.12 CRC Demo

This application uses the CRC Driverlib APIs to exercise various CRC calculation modes.

10.13 SHA-MD5 Application

This application uses the SHA-MD5 Driverlib APIs to exercise various SHA-MD5 modes.

10.14 ADC Demo Application

This application demonstrates the C3200 ADC module using the Driverlib APIs.

10.15 PWM Demo

This application demonstrates the PWM mode of the CC3200 General Purpose Timers (GPT).

10.16 SD Host Demo

This application demonstrates the functionality of the Secure Digital Host (SD Host) controller of CC3200, which interfaces with standard SD memory cards in 1-bit transfer mode.

10.17 SDHost FatFS Demo

This application uses the FatFS library for block level read/write access to SD card, using the SD Host controller on CC3200.

10.18 SPI Demo

This application shows the initialization sequence that enables the CC3200 SPI module in full duplex 4-wire master and slave modes.

10.19 UART DMA

This application demonstrates using the CC3200 UART interface with uDMA and interrupts.

10.20 Timer Count Capture

This application demonstrates measuring the frequency of an external signal using the CC3200 Timer count capture feature.

10.21 Application Bootloader

This application showcases the secondary bootloader operations to manage updates to application image.

10.22 Dynamic Library Loader

This application exercises an approach to enable dynamic loading of an application-binary from non-volatile memory while the program is being executed.

11 Revision History

SDK Version	Updates from previous version
1.2.0	**Version Updates**: • Updated Network Processor Host Driver to version – 1.0.1.6 • Updated Driverlib source to version 1.2.0 **Simplelink**: • Please refer to the section 12.2 for the detailed changes **Simplelink-Extlib**: • Added fastboot configuration for flc library **Driverlib**: • Updated files are prcm.c/h, camera.c, rom_patch.h, spi.c, timer.c, version.h • Please refer to the section 12.3 and section 13.2 for the detailed changes **Netapps**: • Fixed known issues for MQTT library • Updated http server library to make it compatible with the latest browser update. • Wrapper layer for JSON library is added **Middleware:** • Updated middleware library to support multiple UART instances **Reference examples:** • Removed the content for Provisioning AP application. This would be released as an add-on package. • Removed the Deep-Sleep network application • Modified the sleep_deepsleep application to remove the deep-sleep scenario.

	• Fixed known issues for Wi-Fi audio application • Fixed known issues for UDP socket application • Fixed CRC application to work with word-size input data • Fixed AES demo application for user-defined key • Added fastboot configuration for application bootloader, refer to the section 8.3 of *"CC3200 Over-The-Air (OTA) Update Application Note"*
1.1.0	**New Additions:** • Examples - idle_profile_nonos, mqtt_client, mqtt_server, ota_update, dynamic_lib_loader, application_bootloader, http_client_demo • Libraries - netapps/json, netapps/mqtt, netapps/http/client, simplelink_extlib **Version Updates:** • Updated Network Processor Host Driver to version – 1.0.0.10 • Updated Driverlib source to version 1.1.0 **Simplelink:** • Added bug fix to ensure correct operation of SPI transfer, when combinations of DMA and CPU transfers are invoked. • Added macro DISABLE_DEBUGGER_RECONNECT • To be defined for deployment scenarios and also if the system can be reset between debugger sessions. • Added new library build configuration – Simplelink_Non_OS_PM **Driverlib:** • Updated files are: aes.c/.h, camera.c/.h, i2s.c/.c, i2c.c, pin.c, prcm.c/.h, sdhost.c, uart.c, des.c, timer.c/h • Please refer to the section 12 for the detailed changes. **Netapps:** • Fixed memory leak issues in http server. **OSlib:** • Added task_yield in osi_MsgQWrite in osi_tirtos.c. **Middleware:** • Power management framework

	○ Automatic entry into “sleep” low power mode added to power management framework when idling ○ Conditional saving of PSP or MSP performed based on the control register value • Timer framework ○ In cc_timer.c, bug fix to handle the software timer list handling appropriately. • Device drivers ○ In rtc_hal.c, ▪ RTC timer operations optimized as the register accesses from the 32KHz domain was slower. ▪ Slow clock counter read from the 40MHz domain and the workaround to ensure correct value read also incorporated. ○ UART driver updated, ▪ Added echo feature ▪ Added read termination based on configured termination character. ▪ DMA operation bug fix **Reference examples:** • In all relevant applications, updates done to socket event handler due to changes in Host driver • Removed peripheral reset in crypto examples (AES, CRC, DES, SHA) as the engines are a shared resource • In camera application, PIN_01 muxed to non-I2C mode (GPIO). • In interface files, macros updated to ensure the interrupt registration is done using osi* APIs in case of OS application. • In gpio_if, the naming convention changed to use GPIO naming with GPIO APIs. • In startup_ccs, macro added to avoid user configured vector table in case of TIRTOS. • In timer_if, updated API to specify parameter in millisec. • In udma_if, the channel select condition check corrected. • In freertos_demo example, modifications done to run the same code with TIRTOS as well. • In get-weather example, HTTP client APIs used to demonstrate the API usage. • In idle profile example, ○ Bug fix to correct length of key passed

	○ Added sleep clock enable to WDT peripheral ○ Interrupt mode operations made default for UART • In oob example, updated html pages to indicate correct device version and AP profile parameters. • In sdhost example, Added internal Pull-ups on Data and CMD. • In sdhost_fatfs example, ○ Aligned with new FatFS ○ Fixed corruption issue with large file writes/reads ○ Added internal Pull-ups on Data and CMD • In sensor profile example, ○ Bug fix to correct length of key passed ○ Added sleep clock enable to WDT peripheral ○ Interrupt mode operations made default for UART • In websock_camera example, ○ Added pinmux to remove dependency on J2 and J3 jumpers ○ Enhanced camera interface - added option to set resolution • In wifi_audio_app example, ○ Improved the I2S interface - added option to configure sampling rate and bits per sample ○ Enhanced the Audio Codec Interface
1.0.0	• Removing filters while configuring the device to default state • Updated the “file_download” example to remove the use of temporary file • Modified uniflash session files to use the relative paths • Enabled automatic FTDI driver installation capability • Increased SPI Speed • Added netapps folder containing xmpp, smtp, http, tftp protocol library • Added watchdog_system_demo, websocket_camera and tftp_client examples • Modified CCS project files to have include path from SDK ROOT • Changed XMPP server IP Address in xmpp_client application • Changed CCS target configuration cc3200.ccxml file to use CC3200 Debug Interface • Added error handling in all the applications • Reference linker command files added to support 256KB RAM available in production device • Update Network Processor Host Driver to version 1.0.0.1

	• Updated Driverlib source to version 1.0.2 • Moved xmpp and email library to netapps folder • Moved AP configuration macro and networking status bit enum to common header file “common.h” • Updated to FreeRTOS version 8.0.1 • TI-RTOS support for IAR and GCC IDEs
0.5.2	Added a function to configure the firmware to default state across all applications.
0.5.1	First Release

12 Enhancements/Changes (With respect to SDK 1.1.0 release)

This section lists all the enhancements and changes done.

12.1 Enhancements/changes in Sample Applications/Examples

ID	MCU00016011
Title	Addition of "fast boot" configuration to application_bootloader
Description	Application bootloader identifies the user application image to boot using boot info file and finally loading and executing the image. The ‘FAST BOOT’ option • **Doesn't turn on the networking subsystem for loading the application image during the normal boot.** • **Reduces the overall loading time of the application image during the normal boot.** • **Improves the overall init time of the application.** By "normal boot" it is meant any boot like POR, hibernate, WDT reset, MCU reset etc. The only exception being one boot cycle just after an OTA update. The trade-off here is the application bootloader run time size which increases from 16KB to 32KB when ‘FAST BOOT’ is enabled. To work with this application bootloader the user (or OTA) application needs to be re-compiled and linked to run from 0x2000_8000. For further details refer to the section 8.3 of *“CC3200 Over-The-Air (OTA) Update Application Note”*

12.2 Enhancements/changes in Host driver

ID	MCU00016005
Title	Added new API sl_Stop_WithNwpLpdsPoll
Description	This API should only be used for a special use case of LPDS. This special use case is when the Networking sub-system has to be stopped before the APPS MCU enters LDPS. Note that the existing/standard sl_Stop API should not be used for this special use case. Continue to use the (already existing) sl_Stop API for rest of the cases. **Note the errata related to this API in section 14.5 and bug id: MCU0015994**

ID	MCU00016006
Title	sl_GeneralEvtHdlr to be registered
Description	This handler must be registered. This handler will enable the application to be notified on general/fatal driver errors and therefore it is mandatory. If the application already registered to this handler the handler should check for the new errors.

ID	MCU00016007
Title	_SlSpawnEntryFunc_t return type change
Description	In the new driver the spawn entry function returns a value 1.0.0.10 **:** typedef void (*_SlSpawnEntryFunc_t)(void* pValue); 1.0.1.6 : typedef short (*_SlSpawnEntryFunc_t)(void* pValue);

12.3 Enhancements/changes in driverlib

ID	MCU00013767
Title	Enhancement in PRCMHibernateEnter, PRCMLPDSEnter and PRCMMCUReset
Description	Add “wfi” and and “while(1)” in PRCMHibernateEnter, PRCMLPDSEnter and PRCMMCUReset API's after the actual request is made. This is to make the API’s more robust.

ID	MCU00014239
Title	Add new API PRCMLPDSEnterKeepDebugIf
Description	Added a new API to PRCM driver for entry into LPDS w/ Test PD kept enabled. This is in addition to the compile time switch that was provided in PRCMLPDSEnter

13 Fixed items in this release (With respect to SDK 1.1.0)

13.1 Issues fixed in Sample Applications

ID	MCU00012150
Title	printing of multiple socket error for a single socket event
Description	break statements are missing while handling socket events, which is leading to printing of multiple socket event instead of just one

ID	MCU00012235
Title	wifi_audio_app doesn't work if only enable audio playback
Description	wifi_audio_application doesn't work if only enable audio playback means if initialize RecordPlay with I2S_MODE_TX.

ID	MCU00015969
Title	Missing packet ID while sending the data over UDP socket
Description	This fix is needed due to specific UDP packet format required by the Iperf. Iperf may recognize the packet as Out of Order.

ID	MCU00015996
Title	Websocket application not working with latest chrome browser
Description	A new header introduced by the chrome was not being handled

ID	MCU00015997
Title	CRC application giving wrong output in case of word length input
Description	The word length input was always considered as the byte length input resulting in wrong calculation.

ID	MCU00015998
Title	AES application: wrong key length for user defined key
Description	Application was discarding the last 8 bits

ID	MCU00015999
Title	MQTT library not freeing memory in case of failed MQTT connection
Description	Memory was not being freed which was resulting in memory leak

ID	MCU00016000
Title	MQTT library: socket in not closed in case of DNS failure
Description	The application might run out of sockets in case of repetitive DNS failures

ID	MCU00012149
Title	MQTT library doesn't invoke application callback for mqtt disconnect event
Description	Disconnect callback was only being called in case of non-blocking mode

13.2 Issues fixed in driverlib

ID	MCU00015099
Title	CameraXClkSet : Fix related to CAM_XCLK_DIV_BYPASS

Description	Existing :
	case CAM_XCLK_DIV_BYPASS: ulReg |= 0x0000000F;
	
	Fixed :
	case CAM_XCLK_DIV_BYPASS: ulReg |= 0x0000001F;

ID	MCU00015100
Title	Remove PRCMSOCReset and PRCMDeepSleepEnter and related content
Description	Remove the below API's and macros as these are not supported and documented for 32XX devices. • PRCMSOCReset • PRCMDeepSleepEnter • PRCM_DSLP_MODE_CLK • PRCM_SRAM_DSLP_RET Remove support for Configuring deep sleep SRAM retention register in driverlib API's PRCMSRAMRetentionEnable and PRCMSRAMRetentionDisable

ID	MCU00016002
Title	PRCMLPDSEnter : Fix for lockup on “Z/pre-production” device
Description	Issue : Network sub-system gets locked after entering in LPDS mode on Z device Fix : Not to Disable the Tdflash in "Z" device as the TdFlash belongs to Network subsystem

ID	MCU00016001
Title	PRCMHibernateWakeUpGPIOSelect : Fix related to wake-up type configuration

Description	Existing :
	ulRegValue |= (ulType << (ucLoop*2));
	
	Fixed :
	ulRegValue = (ulRegValue & (~(0x3 << (ucLoop*2)))) | (ulType <<(ucLoop*2));

ID	MCU00013766
Title	PRCMCC3200MCUInit : Fix related to I2C and GPIO semaphore configuration
Description	I2C and GPIO semaphore configuration are lost after LPDS. Add code for reconfiguring these semaphores in wake after LPDS mode. This PRCMCC3200MCUInit API should be called by application in post LPDS wake up path also. Without this change Interrupts and DMA requests for GPIO, I2C were not working after LPDS exit.

ID	MCU00013764
Title	SPIDmaDisable : Fix related to configuration
Description	Need to add "~" to ulFlags before writing to register in SPIDmaDisable. Without this API was not working.

ID	MCU00010588
Title	TimerConfigure : Fix related to ASSERT when built with DEBUG macro
Description	Calling TimerConfigure() with DEBUG macro was causing assert.

13.3 Wi-Fi issues fixed in this release (with respect to 1.0.0.10 SP)

ID	MCS00133231
Title	SSL: Client Hello doesn't advertise SHA256 in the extension list
Description	Some servers requires SHA256 extension and will fail to connect if not present

ID	MCS00134538
Title	SSL: not able to connect with SHA-256 with some specific servers
Description	Fix for servers that sends a challenge request with SHA-256 (i.e www.dropbox.com)

ID	MCS00134809
Title	DHCP: The device might stop responding when changing DHCP client settings while it is connected to the Wi-Fi network
Description	After changing the DHCP setting (dynamic/static) the device must restarted in order for the configuration to take effect. The fix was to prevent the device from stop responding after the change of the settings (before restart)

ID	MCS00135125
Title	File System: Files above 500 KB can't be created in Fail Safe mode
Description	File size can be up to 1MB – the fix allow the fail-safe file to be up to 1MB as well.

	File that created by Uniflash require and updated version of it

ID	MCS00135183
Title	System Robustness: Improve start/stop robustness for SPI Big Endian
Description	Robustness issue observed after continuously performance start/stop fora ~10hours on system supporting Big Endian

ID	MCS00135236
Title	Host: General fixes
Description	Http server events fix Internal Spawn fixes SL Studio - use internal spawn by default

ID	MCS00135280
Title	WPS: System might halt if AP is disconnected
Description	In some rare cases when connecting to an AP using WPS the system might halt if the AP is been disconnected (reconfigured or shutdown)

ID	MCS00135328
Title	Host: false detection of sync pattern during read operation
Description	In some rare cases there might be a false detection of sync pattern during read operation

ID	MCS00134030
Title	Host: Blocking sl_Send with Tx size 0 prevents CW
Description	CW (continuous wave) didn't work if the Tx size was set to zero

ID	MCS00135346
Title	PM: NWP will not enter sleep due to packets that weren't released
Description	The NWP will not enter to sleep due to some DNS packets that weren't released

14 Errata

The following section covers known issues and performance limitations with CC3200 devices and the SDK.

14.1 OS Abstraction Layer

ID	MCU0008224
Title	Task_Delete Function in osi.h results in System Crash
Description	Calling osi_TaskDelete(NULL) by a task to kill itself cause a crash as there is no NULL check in implementation.
Impact	FreeRTOS based application that using osi_TaskDelete() API are impacted
Workaround	Delete Task using a handle as below: OsiTaskHandle tHandle = NULL; osi_TaskDelete(&tHandle);

14.2 Network Processor Performance

Item	Production device
MCU-NWP SPI link	20 MHz
Init time from hibernate until device ready*	75 mSec
Init time from hibernate until WPA2 connection*	120 mSec
Maximum UDP throughput, open socket	16 Mbps
Maximum TCP throughput, open socket	13 Mbps
Maximum TLS/SSL throughput with RC4_128 cipher	9 Mbps
Maximum TLS/SSL throughput with AES_256 cipher	12 Mbps
Minimum TLS/SLL connection time with ECC cipher	1.3 Sec
Minimum TLS/SSL connection time with RSA cipher	130 mSec

* Excluding user application boot time

14.3 Wi-Fi known issues

ID	MCS00123349
Title	WiFi Security: CC3100 and CC3200 Supports only WEP with Key Index 0 (==> AP Key index 1)
Description	When using WEP security – only WEP index 0 is supported
Impact	Can't use more than one key in WEP security
Workaround	None

ID	MCS00106970
Title	WiFi Security: Traffic Stop while WPA EAP-TLS Enterprise and Reauthentication enabled
Description	In WPA EAP-TLS security the traffic stopped when Reauthentication packet is received
Impact	Traffic stopped
Workaround	Disabled Reauthentication or set it to a very long time

ID	MCS00130040
Title	WiFi Direct Reliability: 65% Success rate when Peer device is initiator of connection
Description	Negotiation with other peer not always successful at first chance
Impact	The first connection doesn't success
Workaround	Try to connect again

ID	MCS00131174
Title	Scan: Results list contain duplicate networks
Description	The SimpleLink might returns duplicate networks when the network list is not totally filled and the get scan results ask for fewer entries than what was actually found.
Impact	duplicate networks in Scan results list
Workaround	Read the maximum entries at once (20 entries) or to read one by one starting from the end to the beginning and check for duplicates. Once a duplicate was found the list is completed

14.4 Networking

ID	MCS00127876
Title	sl_NetAppDnsGetHostByName return with no answer in high traffic
Description	In high Rx traffic some DNS packets can get lost
Impact	No answer on request
Workaround	Run the API again

ID	MCS00128353
Title	UDP/RAW socket data payload is limited to MTU size
Description	Tx IP Fragmentation is not supported for UDP and RAW Tx
Impact	Packet bigger than MTU size will lead that portion of the packet will be discard
Workaround	Use packet size <= MTU size

ID	MCS00128959

Title	DHCP: SL continues using its previous IP address if an invalid IP in the DHCPACK (before lease time expired)
Description	DHCPACK arrives to SL with invalid address in the DHCPACK params address field but also the IP destination is the same invalid address (MAC address is the valid SL address). SL does not listen to other IPs address as destination but his own therefore this DHCPACK is not processed and SL continue to use his old address until the lease time expires
Impact	The device will continue to use the previous IP address
Workaround	N/A

ID	MCS00129407
Title	NS: SL device should discard ICMP Req datagram with problem in IP Header
Description	According to the RFC – if the gateway or host processing a ICMP Req datagram and finds a problem with the header parameters such that it cannot complete processing the datagram it must discard the datagram
Impact	Low impact – The SL device sends ICMP reply message
Workaround	N/A

ID	MCS00131564
Title	NS: Error -105 when trying to open 4 TCP server sockets while the internal HTTP server is running
Description	While the HTTP server is running one of the TCP server is been used and limit the number of user TCP Servers Error -105 - SL_ENOBUFS [No buffer space available]
Impact	Med impact – Only 3 TCP servers can be used while the HTTP is running
Workaround	Disable the internal HTTP Server if 4 TCP Server need to be used

ID	MCS00131966
Title	NS: blocking accept on secure socket doesn't return
Description	procedure: open secured socket bind listen select on socket => select not return when other side connected
Impact	High impact – Select doesn't return
Workaround	Don't use select method for accept on secure socket

ID	MCS00131612
Title	Transceiver mode: Can't configuring the channel of a RAW Socket if it's already open
Description	Changing the channel while a RAW socket is open to receive by using SetSockOpt

	command can halt the Host. The command response on SetSockOpt doesn't return. As a result, the host is might get stuck if it configured to blocking mode
Impact	Host might get stuck
Workaround	Close the socket and open it again with the correct channel

ID	MCS00135767
Title	Command sl_WlanSet is not working on Big Endian systems
Description	The command is not working well on Big Endian systems and will return an documented error. The command is used to set Tx power, Scan parameters, regulatory domain
Impact	Undocumented error and the command is not taking any affect
Workaround	Program the setting in the image (Uniflash) and avoid using the API
Remarks	Fix is expected in the next service pack release

14.5 Host

ID	MCS00127283
Title	Free RTOS OS is not stable when running UDP traffic and Ping
Description	Known issue with free RTOS that can cause deadlock
Impact	Deadlock in OS
Workaround	Use TI RTOS

ID	MCS00130291
Title	WPS PIN Connect might fail if pin code is not null-terminated
Description	If the PIN code from the HOST is not null terminated the string can be wrongly used and in some cases the connection doesn't succeed
Impact	Connection doesn't succeed
Workaround	Add null termination to the PIN code string

ID	MCS00131563
Title	Host: Set/Get time is limited up to year 2038
Description	time.h (standard time library) is limiting the structure to a signed 32-bit integer, and this number is interpreted as the number of seconds since 00:00:00 UTC on 1 January 1970
Impact	Low impact – Certifications that are bounded by date will expire after year 2038
Workaround	N/A

ID	MCU00015994
Title	Reset observed when the APPS MCU enters LPDS after the nwp is shutdown
Description	A new API sl_Stop_WithNwpLpdsPoll was added to support a special use case of

	LPDS. This special use case is when the Networking sub-system has to be stopped before the APPS MCU enters LDPS. Note that the existing/standard sl_Stop API should not be used for this special use case. However, with this new API, there is issue observed. Issue: At times, reset of device is observed when the device enters LPDS.
Impact	Application which shuts down the NWP before entering LPDS
Workaround	N/A

14.6 Power Management

ID	MCS00128947
Title	In Enterprise network the device will Frequently Wakeup due to IPV4 BRDCST Rx frames
Description	On enterprise network there a lot of BRDCST packets
Impact	Increase in power consumption
Workaround	Add a filter to block the broadcast packets (will be different for each enterprise network)
Remarks	Fix is Not expected – the filter is specific to the network

15 Notes for CC3200 MCU Application Developers

15.1 Simplelink build - Deployment Scenarios Vs. Development mode

To be able to support the usage of the reloading of application image using the debugger without having to reset the device (LaunchPad),

- the implementation in the cc_pal* (simplelink) file required a NwpPowerOnPreamble routine to stop networking services and
- Delay in NwpPowerOn() function to be introduced for proper operation.

This was required as a core reset from the debugger will only reset the APPs processor and the networking engine would still be active. Hence, on the next debug session the networking engine has to be gracefully stopped and started again. Some additional delays had to be added to do this, which results in greater overall current consumption. As this additional steps and delay are required only for the debug purpose, they should not be a part of the deployment applications.

For ease of use, the CC3200 SDK latest package provides separate configurations of simplelink library for development (debug) and deployment scenario. The following 4 should be used for the deployment scenario (does not include the additional steps and delay) as per the use-case.

- NON_OS – Simplelink for NON-OS environment
- OS – Simplelink for OS environment
- PM_Framework – Simplelink with Power Management Framework for OS environment
- NON_OS_PM – Simplelink with Power Management Framework hook

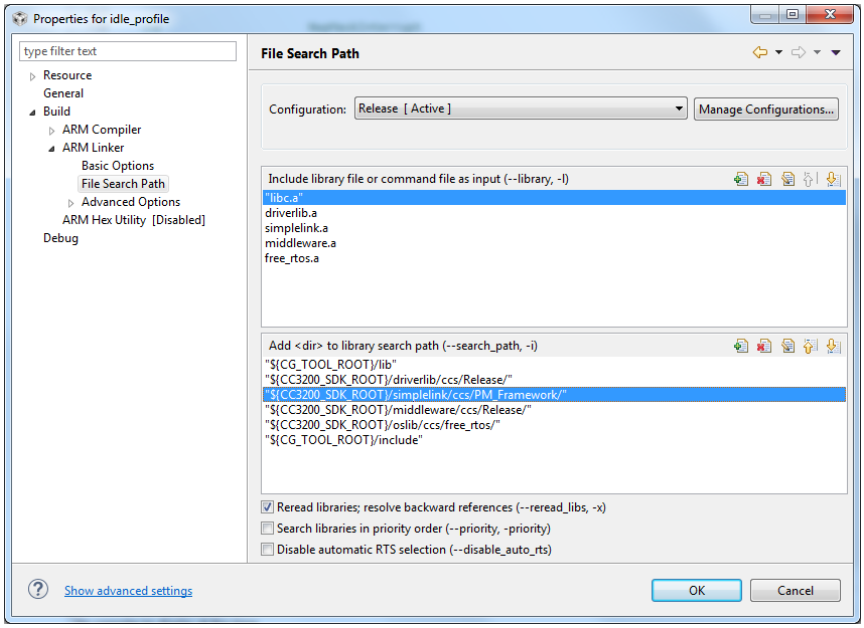
All the above configurations are pre-built in the SDK package along with their generated output library. All of the networking examples link to one of these configurations by default. The corresponding debug configurations are also present as part of the project.

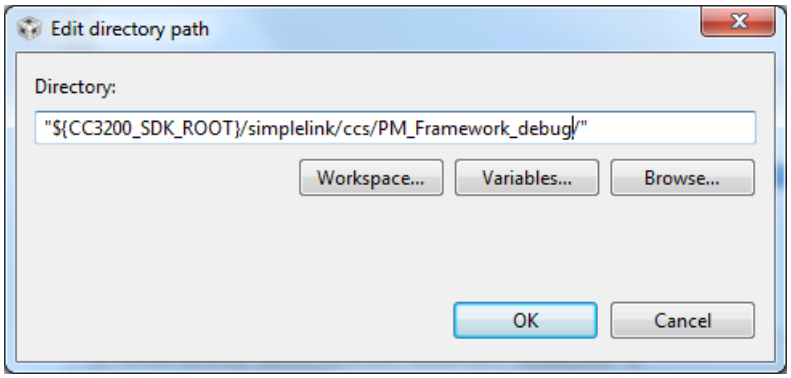
- NON_OS_debug – Simplelink for NON-OS environment with debug support
- OS_debug – Simplelink for NON-OS environment with debug support
- PM_Framework_debug – Simplelink with Power Management Framework for OS environment (debug support)
- NON_OS_PM_debug – Simplelink with Power Management Framework hook (debug support)

These configurations are not pre-built and the user will have to build them first before linking them to their application. Note that it will result in overall greater current consumption. It is advised to link networking example to the debug configurations of simplelink library while debugging as it will enhance the user experience. Please go through the following steps to link your application to the debug configuration.

15.1.1 Relinking to simplelink library in CCS

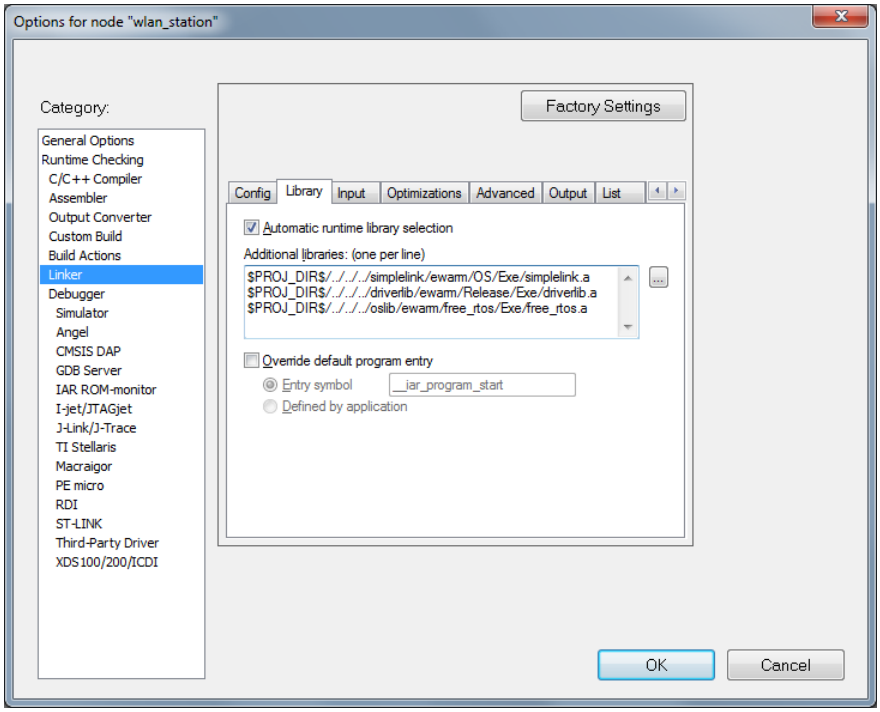
- Make sure to compile the relevant debug configuration for simplelink library
- Right click on the application and go to "*Properties →ARM Linker →File Search Path*"
- Edit the simplelink library path as per the following screenshots

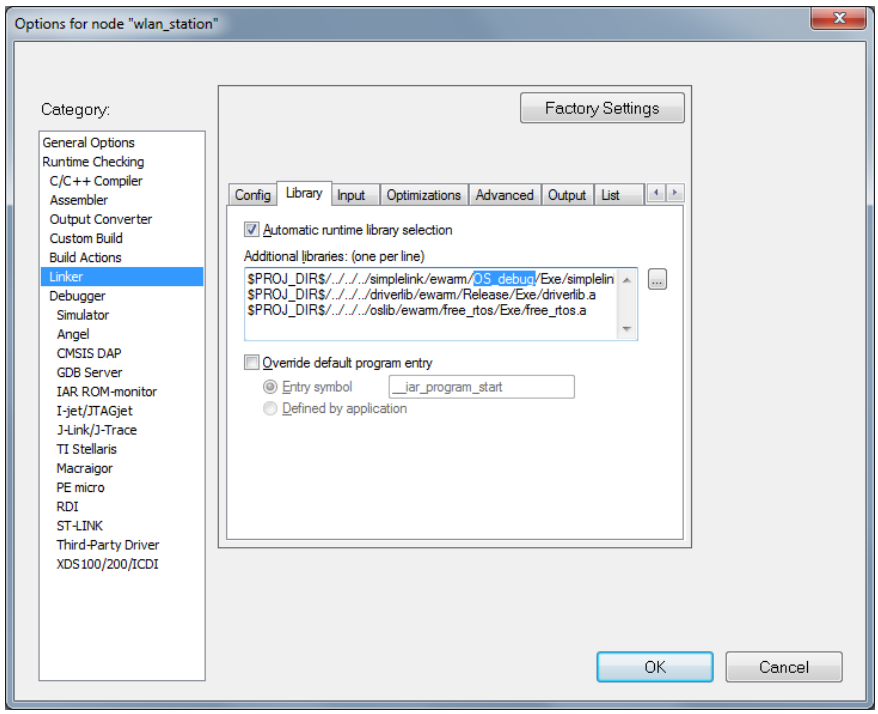




15.1.2 Relinking to simplelink library in IAR

- Make sure to compile the relevant debug configuration for simplelink library
- Right click on the application and go to *"Options → Linker → Library"*
- Edit the simplelink library path as per the following screenshots





15.1.3 Relinking to simplelink library in GCC

- Make sure to compile the relevant debug configuration for simplelink library

- Open the application's *"Makefile"* and go the linking portion
- Make the linking changes as per the following snapshots

```
#
# Rules for building the wlan_station example.
#
${BINDIR}/wlan_station.axf: ${OBJDIR}/main.o
${BINDIR}/wlan_station.axf: ${OBJDIR}/pinmux.o
${BINDIR}/wlan_station.axf: ${OBJDIR}/gpio_if.o
${BINDIR}/wlan_station.axf: ${OBJDIR}/uart_if.o
${BINDIR}/wlan_station.axf: ${OBJDIR}/startup_${COMPILER}.o
${BINDIR}/wlan_station.axf: ${ROOT}/simplelink/${COMPILER}/${BINDIR}/libsimplelink.a
${BINDIR}/wlan_station.axf: ${ROOT}/driverlib/${COMPILER}/${BINDIR}/libdriver.a
${BINDIR}/wlan_station.axf: ${ROOT}/oslib/${COMPILER}/${BINDIR}/FreeRTOS.a
SCATTERgcc_wlan_station=wlan_station.ld
ENTRY_wlan_station=ResetISR
```

```
#
# Rules for building the wlan_station example.
#
${BINDIR}/wlan_station.axf: ${OBJDIR}/main.o
${BINDIR}/wlan_station.axf: ${OBJDIR}/pinmux.o
${BINDIR}/wlan_station.axf: ${OBJDIR}/gpio_if.o
${BINDIR}/wlan_station.axf: ${OBJDIR}/uart_if.o
${BINDIR}/wlan_station.axf: ${OBJDIR}/startup_${COMPILER}.o
${BINDIR}/wlan_station.axf: ${ROOT}/simplelink/${COMPILER}/${BINDIR}/libsimplelink_debug.a
${BINDIR}/wlan_station.axf: ${ROOT}/driverlib/${COMPILER}/${BINDIR}/libdriver.a
${BINDIR}/wlan_station.axf: ${ROOT}/oslib/${COMPILER}/${BINDIR}/FreeRTOS.a
SCATTERgcc_wlan_station=wlan_station.ld
ENTRY_wlan_station=ResetISR
```

15.2 Patch required for IAR

For IAR 7.2 or lower, IAR patch provided under tools folder in SDK is required. This dependency is removed for IAR 7.3 or higher. Make sure the IAR patch should not be applied, if working with IAR version 7.3 or higher, as it is not compatible.

If multiple FTDI devices are connected to a PC, which is being used for downloading the code from IAR debugger, a prompt listing all the FTDI connected to the PC will pop up. There is no definite way of identifying the intended target. So, it is preferred that only one device should be connected to the PC while executing the code from the debugger.

15.3 Connecting back debugger on waking up from LPDS

The Test Power Domain is shutdown whenever the system enters LPDS (by default). In order to avoid this and allow for connecting back the debugger after waking up from LPDS, the macro KEEP_TESTPD_ALIVE has to be defined while building the "driverlib" library.

In addition to this step, the networking sub-system has to be placed in active mode by executing the below statement

```
//iRetVal = sl_WlanPolicySet(SL_POLICY_PM, SL_NORMAL_POLICY, NULL, 0);
iRetVal = sl_WlanPolicySet(SL_POLICY_PM, SL_ALWAYS_ON_POLICY, NULL, 0);
```

This is recommended for development purposes only, as it adds to the current consumption of the system.

Note: The debugger can connect back reliably only when the application processor is active.

15.4 Pinmux pre-requisites

15.4.1 Pinmuxing I2C

The pins on which the I2C functionality is made available by default (MODE1) are

- SCL - PIN_1 (GPIO_10) and
- SDA - PIN_2 (GPIO_11)

If the SCL and SDA pins are chosen to be muxed onto any other pins (other than PIN_1 and PIN_2), it is mandatory to mux the PIN_1 and PIN_2 to some other functionality for correct operation (for ex., GPIO (MODE0)).

15.4.2 Pinmuxing UART

When operating in UART_LOAD mode (SOP[2:0] =100), the PIN_55 and PIN_57 is muxed to the UART functionality by default.

- UART0_TX - PIN_55 (GPIO_01, MODE3) and
- UART0_RX - PIN_57 (GPIO_02, MODE3)

If the UART Tx and UART Rx pins are chosen to be muxed onto any other pins (other than PIN_55 and PIN_57), it is mandatory to mux the PIN_55 and PIN_57 to some other functionality for correct operation (for ex., GPIO (MODE0)) – only while operating in UART_LOAD mode.

Note: Muxing the PIN_55 and PIN_57 to some other functionality is not required while working in FUNCTIONAL mode (SOP[2:0]=000) or 2-wire JTAG mode (SOP[2:0]=001). Hence, it is recommended to choose the settings in development mode similar to deployment scenarios. In such cases, the FUNCTIONAL mode is recommended for using the debugger.

15.5 Updating Simplelink spawn task stack size

By default the simplelink spawn task stack size is set to 2048 bytes. While this size is sufficient for the applications available in the SDK, this value may need to be changed depending on the callback implementation/call flows.
In order to change the stack size, the “oslib” library has to be rebuilt with the macro “SPAWN_TASK_STACK” set to the optimum stack size value
(for ex. -DSPAWN_TASK_STACK=3072).

Implementation in the file “osi.h” in oslib is as below:

```
#ifndef SPAWN_TASK_STACK
```

```
#define STACK_LEN           (2048) /*Stack Size*/
#else
#define STACK_LEN           (SPAWN_TASK_STACK)
#endif
```

15.6 Configuring Default Settings for Network Processor

In order to execute the applications in the SDK one after another, the networking engine is configured to a default mode by invoking the simplelink APIs in a predefined sequence. This is done to ensure that the implementation of the current application is not impacted by the sequence of APIs invoked by applications prior to this one. In particular some APIs can cause information to be saved in non-volatile memory for use in subsequent power ON cycles.

The function does the following:
- Set the mode to STATION
- Configures connection policy to Auto and AutoSmartConfig
- Deletes all the stored profiles
- Enables DHCP
- Disables Scan policy
- Sets Tx power to maximum
- Sets power policy to normal
- Unregister mDNS services
- Remove all filters

For deployment scenarios, invoking the function "ConfigureSimpleLinkToDefaultState" is not mandatory.

15.7 Implementing Library Callbacks

The user applications implement the callbacks that are registered with the libraries. While using the Simplelink and MQTT libraries, invoking the core library APIs from a callback should be avoided and can lead to lockup scenarios.
It is recommended to signal another task from the callback routines invoked from the library and invoke the core library API calls from that task.

15.8 Serial Flash Access

Serial Flash should only be accessed through the file system APIs. Refer to the "file_operations" example for more information.

15.9 Post LPDS Settings

"PRCMCC3200MCUInit" API must be called after coming out of LPDS. If this API is not called post LPDS then the Interrupts and DMA requests for GPIO and I2C modules would not work as the initial

configuration done for them is lost after LPDS. “PRCMCC3200MCUInit” API has these settings and would take care of reconfiguration of these modules

The *Idle Profile* application in the SDK is making use of this API after coming out of LPDS in lp3p0_restore_soc_data API.

15.10 Default GPIO settings

All GPIO pins default to Mode-1 on power-up, unless programmed by the MCU. Set all un-used pins except the serial flash SPI (pin#11, 12, 13, 14) and JTAG pins (16, 17, 19, 20) to Mode-0 to prevent any bus contention. For e.g. by default I2C is muxed onto Pins 01 and 02 and if the application muxes I2C onto some other pins say 03 and 04. This would cause contention.

To set any pin (PIN_X) to Mode-0 use the API

```
PinModeSet(PIN_X,PIN_MODE_0);
```

Specifically to resolve contention on I2C lines Add following code snippet to void PinMuxConfig(void)

```
PinModeSet(PIN_01,PIN_MODE_0);
PinModeSet(PIN_02,PIN_MODE_0);
```

15.11 JTAG Pins Settings

Pin #17 (Pin Alias TDO) is driven high during hibernate when device is operating in SOP mode "Fn2WJ (Functional mode with a 2-wire SWD mapped to fixed pins)". If this pin is used to connect some other device which has a conflicting drive then there could be current leakage in hibernate mode.

In order to "Hi-Z" Pin #17 while entering into hibernate mode and bringing the pin back to normal mode after hibernate below code needs to be exercised. Add below lines of code to start of PRCMHibernateEnter() API. This ensures Pin #17 is "Hi-Z" when device enters hibernate.

```
HWREG(0x4402FC14) |= 0x100;
HWREG(0x4402FC14) &= ~0x2;
```

Add below lines of code to start of PRCMCC3200MCUInit() API. This ensures Pin #17 is brought to normal state when device wakes up after hibernate.

```
HWREG(0x4402FC14) &= ~0x100;
HWREG(0x4402FC14) |= 0x2;
```