



## BCP Driver

---

# Release Notes

Applies to Product Release: 02.01.00.07  
Publication Date: July, 2018

### Document License

This work is licensed under the Creative Commons Attribution-NoDerivs 3.0 Unported License. To view a copy of this license, visit <http://creativecommons.org/licenses/by-nd/3.0/> or send a letter to Creative Commons, 171 Second Street, Suite 300, San Francisco, California, 94105, USA.

### Contributors to this document

Copyright (C) 2012, 2015 Texas Instruments Incorporated - <http://www.ti.com/>



---

Texas Instruments, Incorporated  
20450 Century Boulevard  
Germantown, MD 20874 USA

---

VP00102-Form-1  
Revision D

## Contents

Overview.....	1
LLD Dependencies .....	1
New/Updated Features and Quality .....	1
Resolved Incident Reports (IR) .....	4
Known Issues/Limitations.....	4
Licensing .....	4
Delivery Package.....	4
Installation Instructions.....	4
Test and Example .....	5
BCP Test.....	5
BCP Example.....	8
Customer Documentation List.....	9

# BCP Driver version 02.01.00.07

## Overview

This document provides the release information for the latest BCP driver which should be used by drivers and application that interface with BCP IP.

BCP Driver module includes:

- Compiled library (Big and Little) Endian of BCP Driver.
- Source code.
- API reference guide
- Design Documentation

## LLD Dependencies

LLD is dependent on following external components delivered in PDK package:

- CSL
- CPPI LLD
- QMSS LLD

## New/Updated Features and Quality

This is an **engineering release**, tested by the development team. New and updated features are in reference to version BCP Driver 01.00.00.

### Release 2.01.00.07

- Updated buildlib.xs to add RULES\_MAKE macro to support build based on custom Rules.make

### Release 2.01.00.06

- Updated packaging of library size text files and removed chm files from package. Added support for SECTTI and parallel builds.

### Release 2.01.00.05

- Fix for IR SDOCM00114659: BCP LLD: Extra memory register writes may lead to incorrect configuration. With the fix, register accessing functions of the BCP correctly

address the specific register indicated by the Application, and the execution time for the function is reduced.

- Fix for IR SDOCM00115151: BCP example and unit tests not executing: require enable of TSC. TSC needs to be explicitly enabled in example and unit test programs before use.
- Fix for IR SDOCM00115152: BCP LLD: initialize queueType to known value in Bcp\_rxOpen() function. The queueType variable is initialized to known-value QMSS\_PARAM\_NOT\_SPECIFIED.

#### **Release 2.01.00.04**

- Fix for IR SDOCM00114088: Bcp\_rxOpen fails (due to forcing of queueType to 0 when the Rx Queue number is specified by the application). If the queue number is specified by the Application, the queueType should be appropriately specified as well.

#### **Release 2.01.00.03**

- Fix for IR SDOCM00112895: Incorrect Implementation of osalDeleteSem functions in FFTC and BCP PDK examples and tests causes small memory leak
- Fix for IR SDOCM00105728: Missing the extern “C” construct in bcp\_osal.h

#### **Release 2.01.00.02**

- Addressing IR SDOCM00107359 to enable debug of LLD with optimization.

#### **Release 2.01.00.01:**

- Incorporated new osal functions for QMSS accumulator queues for IR SDOCM00107127: Separate protection OSALs for QMSS LLD.

#### **Release 2.01.00.00:**

- Fix for IR SDOCM00105932. BCP Example and Unit Test Update for K2L to remove SRIO usage for K2L device.

#### **Release 2.00.00.04:**

- Fix for IR SDOCM00102300. BCP driver not compatible with QMSS using RMv2

#### **Release 2.00.00.03:**

- Fix for IR SDOCM00100869. Update to handle second Queues assigned from second QMSS instance

#### **Release 2.00.00.02:**

- Fix for IR SDOCM00098765. Update during EVM bringup.

#### **Release 2.00.00.01:**

- Renamed the device specific folders tci6634 to k2k as per new naming conventions.
- Support for TCI6636K2H device (k2h).

#### **Release 2.00.00.00:**

- Updates to work with auto generated CSL device file.
- Fixed issues with test bench and driver to work with later BCP CSL files and Simulator.
- New APIs are added for the new register fields.

**Release 2.00.00.1000:**

- Source code is migrated to use new CSL macro names and test code is also migrated.
- Tests are not working on C6634 simulator.
- One library to support multiple devices and updated RTSC scripts accordingly.
- Driver directory structure modified to support Keystone-II platforms. Currently support is added for C6634 device.
- Tested with new CSL, QMSS, CPPI, BIOS & XDC tools which supports C6634 device.

## Resolved Incident Reports (IR)

Table 1 provides information on IR resolutions incorporated into this release.

**Table 1 Resolved IRs for this Release**

IR Parent/ Child Number	Severity Level	IR Description
PRSDK-2194	NA	Add RTOS Installer script to autoset SDK_INSTALL_PATH

## Known Issues/Limitations

**Table 2 Known Issue IRs for this Release**

IR Parent/ Child Number	Severity Level	IR Description
SDOC M00093305	S2 - Major	BCP tests with interrupt not working

## Licensing

Please refer to the software Manifest document for the details.

## Delivery Package

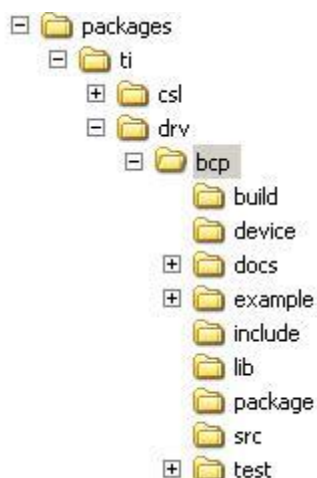
There is no separate delivery package. The BCP Driver is being delivered as part of PDK.

## Installation Instructions

The LLD is currently bundled as part of Platform Development Kit (PDK). Refer installation instruction to the release notes provided for PDK.

## Directory structure

The following is the directory structure after the BCP driver package has been installed:



The following table explains the contents of the BCP package:-

Directory Name	Description
ti/drv/bcp	The top level directory contains the following:- <ol style="list-style-type: none"> <li><u><i>XDC Build and Package files</i></u> These files (<code>config.bld</code>, <code>package.xdc</code> etc) are the XDC build files which are used to create the BCP package.</li> <li><u><i>Exported Driver header file</i></u> Header files which are provided by the BCP driver and should be used by the application developers for driver customization and usage.</li> </ol>
ti/drv/bcp/build	The directory contains internal XDC build related files which are used to create the BCP Driver package.
ti/drv/bcp/device	The directory contains the device specific files for the BCP device driver.
ti/drv/bcp/docs	The directory contains the BCP driver documentation.
ti/drv/bcp/example	The “example” directory in the BCP driver has a usage example which explains how the BCP driver API’s are used to send and receive data.
ti/drv/bcp/include	The “include” directory has private BCP driver header files. These files should not be used by application developers.
ti/drv/bcp/lib	The “lib” folder has pre-built Big and Little Endian libraries for the BCP driver along with their <u><i>code/data size information</i></u> .
ti/drv/bcp/package	Internal BCP driver package files.
ti/drv/bcp/src	Source code for the BCP Driver.

## Test and Example

The section documents information about the test and example code located in the BCP driver.

### BCP Test

BCP Unit test code provided in the package is used by the development team in validating the BCP driver.

The unit test code provided test the following use cases:

- LTE PDSCH/PUSCH using BCP: Given a LTE PDSCH/PUSCH test vector and payload, this test uses BCP LLD APIs to construct BCP headers required for various DL/UL engine sub modules. It sends data for processing to BCP using BCP driver HLD APIs; waits on the result (using polling) and finally validates the result against reference output data to indicate if the test passed or failed. LTE DL test modified to use interrupts.
- WCDMA FDD HSDPA/HSUPA using BCP: Given a WCDMA DL/UL test vector and payload, this test uses BCP LLD APIs to construct BCP headers required for various DL/UL engine sub modules. It sends data for processing to BCP using BCP driver HLD APIs; waits on the result (using polling) and finally validates the result against reference output data to indicate if the test passed or failed.
- WiMax DL/UL test using BCP: Given a WiMax DL/UL test vector and payload, this test uses BCP LLD APIs to construct BCP headers required for various DL/UL engine sub modules. It sends data for processing to BCP using BCP driver HLD APIs; waits on the result (using polling) and finally validates the result against reference output data to indicate if the test passed or failed.
- Rel 99 UL/DL processing using BCP.
- High priority accumulation interrupts tested using LTE DL test case.
- Remote BCP Use case using SRIO Type 9 messages:

Demonstrates the use of BCP driver APIs on BCP local and remote (non-BCP) devices to be able to send/receive packets successfully over SRIO using Type 9 messages. A sample SRIO transport layer implementation using Type 9 messages has been provided too.

*Note:* this remote unit test case is not applicable to TCI6630 devices since SRIO is not present.

**Test setup requirements:**

- 2 PCs connected via Ethernet
- CCS v5
- TCI6618 v0.4.0 installed on both PCs
- BCP driver 1.0.0.5

**Test setup:**

To demonstrate the usage of BCP on TCI6618 from a remote TCI6616 device, the test application creates 2 tasks. On BCP local device, i.e., TCI6618, a BCP server task is created that waits for BCP requests from remote devices (TCI6616) and once received and processed by BCP H/w, the BCP output is sent back to TCI6616 device from where the request had come. On BCP remote device, i.e., TCI6616, a BCP client task is created that



sends BCP requests to remote device using SRIO Type 9 packets and waits on BCP output from TCI6618. Once BCP output is received, the client task validates the output and detects the remote transfer a success/failure.

User would have to build and run the application on 2 PCs, one as server (BCP local device) and the other as client (BCP remote device). To build the application as BCP server running on TCI6618, build the test application with “SIMULATION\_SUPPORT” , “BCP\_LOCAL\_DEVICE” and “RUN\_REMOTE\_USECASE\_TEST” compilation flags enabled. To build the application as BCP client task running on TCI6616, build the test application with “SIMULATION\_SUPPORT” and “RUN\_REMOTE\_USECASE\_TEST” compilation flags enabled.

### Simulator setup:

The simulator configuration file (tisim\_tci6618\_pv.cfg) used by the Simulator model needs to be modified to ensure that the test application works. The configuration file by default is present in the following directory: C:\Program Files\Texas Instruments\ccsv4\simulation\_csp\_tn\bin\configurations.

On PC where you want to run TCI6618 BCP server application:

- Open the configuration file in an editor
- Enable the SRIO Ethernet adapter in the simulator by changing the simulator configuration file as follows. This is done by changing the line from:-

<code>"INPUT1 SOCK, OFF;"</code>
----------------------------------

To

<code>"INPUT1 SOCK, ON;"</code>
---------------------------------

On PC where you want to run TCI6616 BCP client application:

- Open the configuration file in an editor
- Enable the SRIO Ethernet adapter in the simulator by changing the simulator configuration file as follows. This is done by changing the line from:-

<code>"INPUT1 SOCK, OFF;"</code>
----------------------------------

To

<code>"INPUT1 SOCK, ON;"</code>
---------------------------------

- Also ensure that the simulator backend operates as a TCP client and sends all its data to the BCP Server simulator. This is done by making the following changes:

<code>INPUT7     SOCK_TYPE,     SERVER;</code>
--

To

INPUT7	SOCK_TYPE,	CLIENT;
INPUT8	SOCK_ADDRESS,	158.218.100.194;

Note: The IP address 158.218.100.194 should be replaced with the IP address of PC running BCP server application.

### **Running the application:**

1. Build the remote test application for TCI6618 and TCI6616.
2. Launch the debugger on PC that's going to run TCI6618 BCP server application.
3. Load the test application and run it.
4. Launch the debugger on PC that's going to run TCI6616 BCP client application.
5. Load the test application and run it.

### Note:

1. Please ensure that the debugger on TCI6618 PC is running before trying to launch the debugger on TCI6616 PC. This is because the simulator servers are configured to operate on TCI6618 PC and failure to do so will cause the simulator client on TCI6616 PC to fail.
2. The input test vectors for LTE, WCDMA, WiMax unit tests are provided in the "test" directory under BCP driver package. For the unit tests to run successfully, please ensure that the input test vectors ("bcp\test\lte", "bcp\test\wcdma", "bcp\test\wimax") are placed two levels higher than the unit test project .out file location. For example, if the Unit test project out file is located under "C:\MyPDKWorkspace\Bcp\_testProject\Debug" folder, then please copy the input test vectors from PDK package to "C:\MyPDKWorkspace" folder.

## **BCP Example**

The BCP example provided demonstrates the use of various BCP LLD and HLD APIs to:

- Initialize BCP and its MMRs
- Setup BCP CDMAHP and TM sub-module to stream data in/out successfully.
- Add various BCP headers to the packet
- To send/receive data from the BCP and process it

The BCP example provided in this release is a single core example, i.e., to be run on any given one core and it demonstrates the above functionality using an LTE DL/UL, WCDMA DL/UL, Rel 99 DL/UL and WiMax DL/UL packets for illustration.

The BCP example also demonstrates:

- BCP Tx Queue Monitoring: Illustrates how one can maintain a history of the last "N" packets transmitted to BCP using just an additional general purpose hardware queue. This feature has been implemented in the WCDMA HSUPA example code and can be seen in action by enabling the compilation flag "MAINTAIN\_DESC\_HISTORY" in *example\bcp\_wcdma\_ul.c* file.

Note:

1. The input test vectors for LTE, WCDMA, WiMax examples are provided in the “example” directory under BCP driver package. For the example to run successfully, please ensure that the input test vectors (“bcp\example\lte”, “bcp\example\wcdma”, “bcp\example\wimax”) are placed two levels higher than the example project .out file location. For example, if the Example project out file is located under “C:\MyPDKWorkspace\Bcp\_exampleProject\Debug” folder, then please copy the input test vectors from PDK package to “C:\MyPDKWorkspace” folder.

## Customer Documentation List

Table 3 lists the documents that are accessible through the /docs folder on the product installation CD or in the delivery package.

**Table 3      Product Documentation included with this Release**

Document #	Document Title	File Name
1	API documentation (generated by Doxygen)	docs/bcpDocs.chm
2	Design Document	docs/BCP_SDS.pdf