CPPI LLD

Release Notes

Applies to Product Release: 01.00.02.02:
Publication Date: April 30, 2012

Contributors to this document

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

Contents

Overview....1
LLD Dependencies....1
New/Updated Features and Quality....1
Resolved Incident Reports (IR)....8
Known Issues/Limitations....8
Licensing....9
Delivery Package....9
Installation Instructions....9
    Directory structure....9
Customer Documentation List....10

Release Notes

CPPI LLD version 01.00.02.02

Overview

This document provides the release information for the latest CPPI Low Level Driver which should be used by drivers and application that interface with CPPI IP.

CPPI LLD module includes:

- Pre-compiled library for DSP (Big and Little) Endian of CPPI LLD.
- Source code.
- API reference guide
- Design Documentation

LLD Dependencies

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

- CSL
- QMSS LLD
- RM LLD

New/Updated Features and Quality

This is an **engineering release**, tested by the development team.

Release 1.0.2.2

- Add 128 bytes of padding to Cppi_Object. This is necessary to ensure linker doesn't place an unrelated object in the same cache line as Cppi_Object.

Release 1.0.2.1

- CPPI manages its own heap. This allows CPPI to allocate 1K chunks of memory from system heap then manage cache within its own sandbox. Memory is only released to system when Cppi_exit() is called. This removes the restriction that CPPI have a dedicated heap when using shared memory. The shared heap itself must still be shared memory capable such that it manages the cache for its own internal structures, such as HeapMemMP from SYS/BIOS. When running multicore, Osal_cppiCsEnter() must be correctly implemented, such that only one core can touch any part of the cache-aligned 1K chunks at any given time.

- This also allows the use of static memory that can optionally passed in via Cppi_IntiCfg. This may aid system integrators who don't want to use any form of malloc().

Release 1.0.2.0

- *Fixed multicore cache coherence and critical section issues. Please note the documentation in CPPI_QMSS_LLD_SDS.pdf that requires that CPPI use a dedicated heap when using shared memory for the channel heap.*

Release 1.0.1.5

- Modified example project configuration file to support devices with fewer number of cores

Release 1.0.1.4

- Added support for Resource Manager LLD. For all existing applications there are no API modifications required. The Cppi_startCfg API has been added to configure use of the RM LLD if desired.

Release 1.0.1.3

- Release adds examples and unit test code to demonstrate Linux User Mode LLD usage for ARM processor. Support only applicable for devices with ARM processor.

Release 1.0.1.2

- Release includes modifications to support User Mode access for ARM processor. Support only applicable for devices with ARM processor.

Release 1.0.1.1

- Additional device support

Release 1.0.1.0

- **SDOCM00085084 –** Prefix listlib APIs in CPPI LLD to avoid conflicts with other listlib APIs used in the system
- **SDOCM00083379 –** Compiler remarks in cppi_desc.h

Release 1.0.0.16

- Added auto generation of LLD version number and Makefile
- Fixed missing cache writeback in Cppi_channelClose() and Cppi_closeRxFlow() when the channel is not freed due to additional references.

Release 1.0.0.15

- Updated cache invalidation and writeback OSAL APIs to use mfence. Added XMC prefetch buffer invalidation.

Release 1.0.0.14

- Changes for limiting doxygen requirement only during the release

- Copyright modification to TI BSD

Release 1.0.0.13

- Updated test code.
- Added project txt files to enable auto project creation for example and test projects

Release 1.0.0.12

- Queue Proxy is not modeled in the simulator. Added flag **SIMULATOR_SUPPORT** to handle the unsupported feature in qmss_mgmt.h. Ensure the example and test projects define this flag to differentiate between simulator and device. Pre-built library is compiled with this flag turned off.

Release 1.0.0.11

- C66 Target support
- Modifications to the LLD to be device independent.
    - CPPI API changed from
      Cppi_Result Cppi_init (void);
      to
    - Cppi_Result Cppi_init (Cppi_GlobalConfigParams *cppiGblCfgParams);
    - Link device specific file **cppi_device.c** in the driver/application.
    - Add an external reference to device specific configuration
      extern Cppi_GlobalConfigParams cppiGblCfgParams;
    - Deprecated APIs Cppi_setCompletionTag() and Cppi_getCompletionTag()
    - Deprecated rx_size_thresh3_en field from Cppi_RxFlowCfg data structure.
    - Added new descriptor field manipulation APIs
        - Cppi_setDataLen(), Cppi_getDataLen(), Cppi_setSoftwareInfo0(), Cppi_getSoftwareInfo0(), Cppi_setSoftwareInfo1(), Cppi_getSoftwareInfo1(), Cppi_setSoftwareInfo2(), Cppi_getSoftwareInfo2(), Cppi_setOrigBufferpooIndex(), Cppi_getOrigBufferpooIndex(), Cppi_incrementRefCount(), Cppi_decrementRefCount(), Cppi_getRefCount()

    - Software workaround for PS location bug – The PS location is not updated correctly in the host descriptor. Change the get Cppi_getPSData()API to workaround the above issue. The API changed from
      Cppi_Result Cppi_getPSData (Cppi_DescType descType, Cppi_Desc *descAddr, uint8_t **dataAddr, uint32_t *dataLen)
      To
      static inline Cppi_Result Cppi_getPSData (Cppi_DescType descType, **Cppi_PSLoc location**, Cppi_Desc *descAddr, uint8_t **dataAddr, uint32_t *dataLen)

    - Added new APIs to get LLD version ID and Version String
        - uint32_t Cppi_getVersion (void);
        - const char* Cppi_getVersionStr (void);

Release 1.0.0.10

- Prebuilt libraries are both ELF and COFF. Examples and test projects are ELF only.
- Removed cppi instance count. Make sure the application calls Cppi_init()APIs once. When using multicore application, application MUST provide synchronization between cores such that slave cores wait on master core to finish Cppi_init() before calling Cppi_open() API.
    - An example is provided in "sample" example.
    - Deprecated error return codes **CPPI_ALREADY_INITIALIZED**, **CPPI_NOT_INITIALIZED**
- Added cache coherency hooks.
    - Added cache coherency callouts for cache invalidation and writeback. The cache hooks are only in control path. No cache coherency operations are performed in data path.
    - OSAL has been modified to add OSAL implementation of callouts for L1 and L2 caches (L2 is commented out right now). Refer to *cppi_osal.h* and *sample_osal.c*

      When using CPPI in multicore application, the heap for CPPI LLD memory allocation is placed in shared memory. This **MUST** be a separate heap used only by CPPI to avoid false sharing issues when caches are enabled.
    - "sample" example has been modified to configure L2 caches and MPAX for address translation. It is currently commented out under **L2_CACHE** define.
- Changed library optimization level from o3 to o2. Removed deprecated option ml3.
- Removed defines **QT** and **QT_WORKAROUND** from examples and test code.

Release 1.0.0.9

- Migration of LLD from COFF to ELF. Prebuilt libraries are ELF only.

Release 1.0.0.8

- Modifications to LLD to conform to CPPI 4.2.11 spec
    - Changes to global CPPI configuration structure passed in Cppi_open API. Memset the Cppi_CpDmaInitCfg configuration structure to zero if you don't want to change the default values for fields listed below.
        - New timeoutCount variable is added to Cppi_CpDmaInitCfg to configure timeout after buffer starvation.
        - New writeFifoDepth variable is added to Cppi_CpDmaInitCfg to configure write arbitration FIFO depth.

        - Configurable QM base address. Allows configuring of QM base addresses in order to allow overlapping QMs.
    - Receive flow configuration structure changes.
        - The values that can be assigned to rx_dest_tag_lo_sel, rx_dest_tag_hi_sel, rx_src_tag_lo_sel, rx_src_tag_hi_sel has changed. Please refer to Cppi_RxFlowCfg structure for details
        - When configuring rx_size_thresh0, rx_size_thresh1, rx_size_thresh2, specify the actual packet size. The LLD will translate it to the configurable value by right shifting the packet size.
        - In order to avoid confusion when setting the threshold enable mask, the configuration has changed.

          rx_size_thresh_en is deprecated. 4 new fields rx_size_thresh0_en, rx_size_thresh1_en, rx_size_thresh2_en, rx_size_thresh3_en are provided to specify which thresholds must be enable. The LLD will calculate the threshold mask.
    - CPDMA loopback enable
        - Added new APIs to get (Cppi_getCpdmaLoopback()) and set (Cppi_setCpdmaLoopback()) CPDMA loopback enable bit.
- INTD is modeled in simulator. If you are using accumulator to generate interrupts, you need to acknowledge them after processing in order to receive further interrupts.

```
Qmss_ackInterrupt (cfg.channel, 1);
Qmss_setEoiVector (Qmss_IntdInterruptType_HIGH, cfg.channel);
```

  cfg.channel is the accumulator channel used. Refer to the API documentation for further details.

  Modified examples and test project to remove QT define.
- Changed XDC tools version to 3.16.02.32 in examples and test projects.
- Changed sample project to use IPC version to 1.20.00.21
- Removed QT_WORKAROUND from examples for internal linking RAM use and disabling accumulator.

Release 1.0.0.7

- Modified types from XDC to C99
- Changed all source, header, and example code to reflect CSL include path change.

Release 1.0.0.6

- This release is for workarounds for issues found during testing. The workarounds are compiled under **QT_WORKAROUND** define.

- The examples are test case are modified for QT. Define **QT** and **QT_WORKAROUND** (defined by default) to run the examples and testcases on QT.
- Packets are not transmitted out when QM base address in CPPI is configured.
- Internal linking RAM causes CCS to hang. Use external(L2) linking RAM instead.
- Accumulator cannot be disabled. The PDSP firmware does not clear the command causing the API to loop indefinitely.
- Monolithic packets are received with zero packet length. Data and protocol specific data are not present in the received packet.
- Packet length is read as zero when descriptor is popped by reading register C and D.

Release 1.0.0.5

- Modifications to LLD to conform to CPPI 4.2.10 spec
  - Changed on-chip field in the monolithic descriptor to return push policy. On-chip is no longer supported.
    - Moved returnPushPolicy from Cppi_HostDescCfg to Cppi_DescCfg.
  - Changed maximum supported transmit channels for packet accelerator subsystem (PASS) to 9.

Release 1.0.0.4

- Modified examples and test code to remove references deprecated API Qmss_getQueuePendingStatus()

Release 1.0.0.3

- Modifications to LLD to conform to CPPI 4.2.9 spec
  - Teardown is no longer supported.
    - Removed enum Cppi_DescType_TEARDOWN.
    - Removed teardown descriptor definition Cppi_TearDownDesc
    - Removed API Cppi_getTdInfo(), used to get teardown information from descriptor
    - Removed free teardown descriptor queue information freeTdQueue from configuration structure Cppi_CpDmaInitCfg
    - Removed queue information tdQueue on which the teardown descriptor will be queued from transmit channel configuration Cppi_TxChInitCfg structure

- Changed Enum Cppi_CpDma_FFTC_CPDMA to include 2nd instance of FFTC. The new enums are Cppi_CpDma_FFTC_A_CPDMA and Cppi_CpDma_FFTC_B_CPDMA

- Changed CPPI global configuration structure Cppi_GlobalConfigParams to include FFTC_B instance
- Internal linking RAM use is supported. CPPI examples are modified to use internal linking RAM. The same can be done in the application. LLD will configure linking RAM0 address to internal linking RAM address if a value of zero is specified in linkingRAM0Base parameter. LLD will configure linking RAM0 size to maximum internal linking RAM size if a value of zero is specified in linkingRAM0Size parameter
- Device specific sample configuration is built within the driver. They are located within the device directory. There is no need to add/link the file to the project. Remove sample_cppi_cfg.c from example .project files. Remove external reference to sample_cppiGblCfgParams.
- Device specific configuration parameter has been removed from init API. The API has changed to

  Cppi_Result Cppi_init (Void)
- Added queue manager base address to CPPI global configuration structure Cppi_GlobalConfigParams.
- Following IRs against simulator are verified

Release 1.0.0.2

- Modifications to LLD to conform to CPPI 4.2.7 spec
  - Rx flow configuration – Cppi_RxFlowCfg structure changes.
    - Fields rx_swdb_present and rx_tstamp_present are combined and the new field is called rx_einfo_present.
  - Monolithic descriptor – Cppi_MonolithicDesc structure change
    - Added a Reserved field to align the monolithic descriptor to 32 bytes when optional data is present. The data offset should be increased by 4 bytes for monolithic descriptors when the EPIB block is used.
  - Transmit channel configuration – Cppi_TxChInitCfg structure change
    - 3 new fields filterEPIB, filterPS and aifMonoMode must be initialized when configuring a transmit channel.
- Setting original buffer information is decoupled from the Cppi_setData() API. A new API Cppi_setOriginalBufInfo() is provided to set the original buffer information.
- The sample_cppiGblCfgParams structure in sample_cppi_cfg.c is updated for FFTC_B defines.
- Shared memory allocation
  - Shared memory cannot be allocated using the BIOS Memory_alloc() API. If the CPPI resources such as channels, flows are opened from more than 1 core, the handles must be allocated from shared memory. IPC package is used to allocate

shared memory. The “sample” example project in CPPI LLD depicts the use model.

Release 1.0.0.1

- Changed OSAL critical section APIs to be more generic.

  Instead of passing the key as an input parameter to the enter function (as was previous version), changed it such that OSAL creates the handle instead of the caller. OSAL creates the unique handle in CS enter, handle is a return parameter. From the LLD perspective it is an opaque handle that is passed to the CS exit function.
- Changed number of channels, flows from 128 to 129 for AIF CPDMA in sample_cppiGblCfgParams(sample_cppi_cfg.c)
- CPPI LLD help integrated with the CCSv4 Eclipse Help subsystem

Release 1.0.0.0

- Initial release of low level driver

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
SDOCM00091791	Major	CPPI_LLD: padding needed for cppiObject

Known Issues/Limitations

IR Parent/ Child Number	Severity Level	IR Description
BCG_IP_P.BTS_pa_cdmahp.401	Major	CDMAHP RX PS location bit is not updated in CPPI descriptor

Licensing

Please refer to the software Manifest document for the details.

Delivery Package

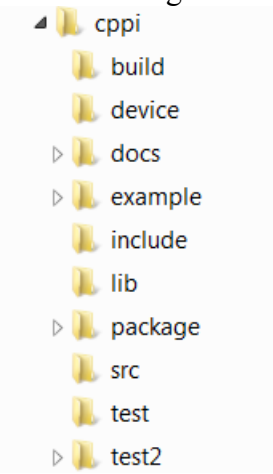
There is no separate delivery package. The CPPI LLD is being delivered as part of PDK.

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 CPPI LLD package has been installed:



The following table explains each individual directory:

Directory Name	Description
`ti/drv/cppi`	The top level directory contains the following:- 1. *Environment configuration batch file* The file "`setupenv.bat`" is used to configure the build environment for the CPPI low level driver. 2. *XDC Build and Package files* These files (`config.bld, package.xdc` etc) are the XDC build files which are used to create the CPPI package. 3. *Exported Driver header file* Header files which are provided by the CPPI low level driver and should be used by the application developers for driver customization and usage.
`ti/drv/cppi/build`	The directory contains internal XDC build related files which are used to

	create the CPPI low level driver package.
ti/drv/cppi/device	The directory contains the device specific files for the CPPI low level driver.
ti/drv/cppi/docs	The directory contains the CPPI low level driver documentation.
ti/drv/cppi/example	The "example" directory in the CPPI low level driver has the infrastructure mode example.
ti/drv/cppi/include	The "include" directory has private CPPI low level driver header files. These files should not be used by application developers.
ti/drv/cppi/lib	The "lib" folder has pre-built Big and Little Endian libraries for the QMSS low level driver along with their *code/data size information*. It also includes the makefile to build LLD for ARMv7 target
ti/drv/cppi/package	Internal CPPI low level driver package files.
ti/drv/cppi/src	Source code for the CPPI low level driver.
ti/drv/cppi/test	The "test" directory in the CPPI low level driver has unit test cases which are used by the development team to test the CPPI low level driver.
ti/drv/cppi/test2	The "test2" directory in the CPPI low level driver has unit test cases for ARM Linux user space LLD which are used by the development team to test the CPPI low level driver.
eclipse	The "eclipse" directory has files required to integrate CPPI low level driver documentation with Eclipse IDE's Help Menu.

Customer Documentation List

Table 2 lists the documents that are accessible through the **/docs** folder on the product installation CD or in the delivery package.

Table 2 Product Documentation included with this Release

Document #	Document Title	File Name
1	API documentation (generated by Doxygen)	docs/cppilldDocs.chm
2	Design Document	docs/CPPI_QMSS_LLD_SDS.pdf
3	Software Manifest	docs/CPPI_LLD_SoftwareManifest.pdf