

QMSS LLD

Release Notes

Applies to Product Release: 02.00.00.18
Publication Date: November 14, 2013

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) 2013 Texas Instruments Incorporated - <http://www.ti.com/>



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)	17
Known Issues/Limitations	17
Licensing.....	17
Delivery Package	17
Installation Instructions.....	17
Customer Documentation List	19

QMSS LLD version 02.00.00.18

Overview

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

QMSS LLD module includes:

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

LLD Dependencies

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

- CSL
- CPPI LLD
- RM LLD

New/Updated Features and Quality

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

Release 2.0.0.18

- Put “far” attributes on qmssLObjIsValid for those projects who recompile sources without `--mem_model:data=far`.

Release 2.0.0.17

- Moved qmssLObjIsValid to “.far:local” and added a `--mem_model:data=far` to enable moving qmssLObjIsValid away from rest of near data. This only affects scalars in the library which are not part of data path.

Release 2.0.0.16

- Added section tags “.neardata:local” to qmssLObjIsValid and “.far:local” to qmssLObj. This allows the user to force these sections to a local/private memory when .far and

.neardata are put in a shared/nonprivate memory. No action is required by user if .far and .neardata are still private.

- Added removeBytes field to qos scheduler and qos scheduler + drop scheduler. This allows shaping of L3/L4 traffic by removing cost of lower level headers from qos accounting. Note: please ensure all C registers get populated on all relevant “push” operations before using this parameter. Typically, but not required, offsetBytes will be set to 0 when removeBytes is used.
- Add writeback for qmssGObj in Qmss_init(). This has no effect when qmssGObj is placed in MSMC because L1D is read-allocate. It is required to support qmssGObj placed in L2 cacheable memory such as DDR, since L2 cache is write-allocate.

Release 2.0.0.15

- Added Qmss_openAccumulatorCh, Qmss_cfgAccumulatorCh, Qmss_stopAccumulatorCh, and Qmss_closeAccumulatorCh. This separates resource management (open/close) from configuration (cfg/stop) enabling cfg/stop to be called from realtime code (while open/close are called from non-realtime code). Note Qmss_programAccumulator and Qmss_disableAccumulator still exist and bundle the resource management and configuration together, which has more latency than configuration alone.

Release 2.0.0.14

- Add shared object library support

Release 2.0.0.13

- Add RM to remaining user mode examples (qmInsRegionTest, qmSCfgTest, qmDescTest, and qmDcfgTest) and removed internal resource management from defines in qmss_test.h.
- Fix error exit in Qmss_insertMemoryRegion. If any resources are taken before error is detected, they are returned to RM after the error is detected.
- Fix error exit in Qmss_insertMemoryRegion for RM errors. It previously exited with open critical section which would result in crash if critical section is mapped in the osal.
- Add Qmss_QueueType_INTC_SET[234]_QUEUE to add all INTC(CIC) accessible queues.
- Add Install for qmInfraDmaMC.out so it can appear in yocto/arrago filesystems.
- Add error check in Qmss_start and Qmss_startCfg to return error if Qmss_init() has not been called on the global object.
- Fix Qmss_getStarvationCount(s) for starvation queues on second QM.
- Add Qmss_setEoiVectorByIntd, Qmss_ackInterruptByIntd in order to address the second intd (to allow second accumulator). Added second INTD/accumulator to RM and qmss_device.c.
- Change logic for returning isAllocated when using rm, such that it will return 0, 1, and 2 like without rm. However, it may return a different value for ≥ 2 depending on whether multiple instances of qmss LLD are created (such as when using ARM user mode). If there

is single owner count (DSP only) it works like without RM. However if there are multiple owners, it returns owner count instead of reference count. Note: this may change to a global reference count in a future release.

- Note: when using RM, the queType field to Qmss_queueOpen MUST be correct. It can be either a valid queue type, or QMSS_PARAM_NOT_SPECIFIED. Do not hardcode to other constants such as 0, because those are otherwise legal queue types.

Release 2.0.0.12

- Remove initialization of qmQueMgmtProxyDataReg (which isn't currently used by LLD) from k2h/k2k because that memory area doesn't exist in hw.
- Cleans up if partial group allocation (Qmss_rmServiceGroups) fails.
- Added Qmss_exit() to clean up the "control" and "linking ram" resources in RM.
- Modify Qmss_insertMemoryRegion to allocate link RAM index via RM so it is globally accounted.
- Modify Qmss_init to not reject due to RM linking ram permissions when initCfg.qmssHwStatus = QMSS_HW_INIT_COMPLETE. This allows "clean" init when ARM set up linking RAM.
- Integrate RM into examples and unit test from ARM (qmQAllocTest.out, qmInfraDmaSC.out, qmInfraDmaMC.out). This requires starting rm server with following command line:
 - rmServer.out rm/device/k2h/global-resource-list.dtb
rm/device/k2h/policy_dsp_arm.dtb
- The following guidelines can help migrate code to support all 16K queues.
 - When using CPPI keep using Qmss_getQueueNumber and make sure you don't discard qMgr field.
 - Otherwise, use Qmss_getQIDFromHandle to the the unmutilated 16-bit queue number. This should be used when passing to TI firmware such as the packet accelerator or QoS.
 - The following grep can help inspect user code for 16K queue problems:
 - `egrep -r -n --include='*.[ch]' "qMgr|qNum" .`
 - If you see qMgr+qNum on adjacent lines the code is good.
 - If you see a qMgr=0 and qNum on adjacent lines the code is forcing to work on first 4K queues and can probably be readily extended to use all 16K.
 - If you see a qNum without a qMgr, you should use the Qmss_getQIDFromHandle APIs instead (assuming the lh part of expression is not CPPI otherwise make sure both qMgr+qNum are used).
 - Add new API Qmss_queueOpenUse which only opens the queue if it has already been opened somewhere else.

- Fix RM integration in Qmss_queueOpenInRange(). It would always return the first queue in the range whether or not it was already open.

Release 2.0.0.11

- Added starvation counter and high priority queues to second QM. Updated qmInfraK2*BiosExampleProject to use those new queues in one of its test cases.
- Add new API Qmss_removeMemoryRegion() to assist with making projects restartable.
- Make examples and unit tests work on both DSP and ARM
 - Note: in order to run on ARM, it requires loading and activating kernel module which opens memory protection for the QM/CPPI to user space. This must be done once per boot of EVM.

```
insmod hplibmod.ko
cat /proc/netapi
```

- Make all QM and CPPI examples “restartable” from the ARM. This means each can be run sequentially (or multiple times) without rebooting the EVM.

Release 2.0.0.10

- Fully integrated Keystone II RM. If RM Server Handle equals NULL the LLD will operate in backwards compatibility mode as if RM does not exist.
 - QMSS QoSched and QoSchedDropSched projects updated to use RM.
 - QMSS InfraMC example project updated to use RM.

Release 2.0.0.9

- Add two new API: Qmss_getQIDFromHandle() and Qmss_getHandleFromQID() which are used to convert between Queue ID that are used by other hardware, and queue handles used by the LLD.
- Add new API: Qmss_getStarvationCounts() which queries a group of 4 starvation counters, since hw clears all 4 even if sw only asked for 1.
- Fix NULL push in qos scheduler firmware and update its version number to 2.0.1.5.

Release 2.0.0.8

- Aligned Resource Manager callouts with new Keystone II RM APIs. Only RM Service Handle equals NULL has been tested with LLD.

Release 2.0.0.7

- Synchronize with keystone 1. Rebase to 1.0.3.19 from 1.0.2.16 (see below).
- Remove cpqi_types.h/qmss_types.h since LLDs use c99 types. No longer need to add ti/drv/qmss and ti/drv/cppi as include paths.

Release 2.0.0.6

- Fixed errors found in user mode LLDs example/test projects building.

Release 2.0.0.5

- Bug fixes. See Resolved Incident Reports section below.
- Renamed the device specific folders as per new naming conventions.
- Support for TCI6636K2H device (k2h).

Release 2.0.0.4

- Updates for using auto-generated cslr_device.h and csl_device_interrupt.h files.

Release 2.0.0.3

- Modification for single LLD library to work for all platforms. Moved the default location of C66x libraries to lib\c66x inside component directory
- Build support for ARMv7 user mode target. Limited build verification in this release.

Release 2.0.0.2

- Update accumulator firmware to allow reclamation feature to support 16K queues.
- Update qmss_device.c to match realignment of TX queues in the HW specs.
- Rename InfrastructureMode and InfrastructureModeMulticore to InfraDmaSC and InfraDmaMC, respectively, to shorten pathnames.
- Reconfigure InfraDmaSC and InfraDmaMC to use the DMA in the second QM and the first QM, respectively. This insures both DMAs are tested.

Release 2.0.0.1

- Fixed issue with load balancing (queues opened with absolute # were not counted as load)
- Added enum for Qmss_PdspID_PDSP8.
- Update QoS firmware to use 4 bit hint field instead of 5 bits (broke descriptors with addresses aligned to 0x10). (SDOCM00088701)
- Update accumulator firmware to use 4 bit hint field (SDOCM00088702), allow for 16K queues, and use correct base address for diversion feature.

Release 2.0.0.0

- Addition of support for two QM groups on Keystone 2.
 - There are two modes, JOINT and SPLIT.
 - In JOINT mode, descriptors can be pushed onto any of the 16K queues and linking RAM and descriptor regions are automatically added to both groups. Since JOINT_LOADBALANCED is 0, applications who set Qmss_InitCfg to 0 will automatically operate in this mode without other changes relative to Keystone 1. No new APIs are required. While some APIs include a queueGroup in structures, it is always ignored in JOINT mode.
 - There are two schedulers for JOINT mode. ROUND ROBIN strictly alternates between the two QMs while LOADBALANCED adds to the QM with the least open queues.
 - In SPLIT mode, the queueGroup that is added to various configuration structures is used to specify which queue group will be used. A new API, Qmss_queueOpenInGroup() allows the group number to be specified when

- opening queues. The existing Qmss_queueOpen() API will always allocate from group 0.
- For debugging purposes its possible to force use of one QM in either split or joint mode by setting (maxQueMgrGroups,maxQueMgr,maxQue) = (1,2,8192) instead of (2,4,16384) in qmss_device.c.
- KeyStone2 devices have new directory structure for devices, example and test folders

Release 1.0.3.19

- Add push proxy feature that allows C+D to be pushed together for devices where errata prevents using the HW proxy.
 - Therefore, the ability to run QoS Scheduler and Drop Scheduler at different rates is removed since it wasn't required by primary user of QoS Scheduler + Drop Scheduler
 - Push Proxy feature is only available together with drop scheduler, so it won't slow down QoS Scheduler alone.
- Remove critical section/mutex from Qmss_queuePush() for the c6600, since it is not necessary since it always does atomic 64 bit stores.
- Change all (void *) cast in qmss_device.c into the actual types. This allows qmss_device.c to be compiled with a C++ compiler.

Release 1.0.3.18

- Readback of QoS scheduler ports doesn't work. This issue was due to an error in memcmp() in test project, and was present in all versions of QoS scheduler.
- QoS Scheduler Drop Scheduler push statistics are changed from interrupt mechanism to queue mechanism. Now, the top level configuration for the drop scheduler can specify up to 4 queue pairs for push stats. Once the MSB of one of the stats becomes set, the stats are copied into a descriptor taken from the specified queue source, then placed into the queue destination.
- qmQoS SchedTestProject fails on BE: Error introduced by drop scheduler, fixed in this release.
- qmQoS SchedDropSchedTestProject fails on BE: Error fixed in this release.

Release 1.0.3.17

- Added "drop scheduler" plus "qos scheduler" firmware (which is in qos_sched_drop_sched_le[] and qos_sched_drop_sched_be[]). This firmware supports 20 lite ports of 1 group of 4 queues together with the drop scheduler which supports a model tail drop and fixed probability RED drop.
- The existing qos scheduler firmware is still supported, and is in qos_sched_le[] and qos_sched_be[]. It is backwards compatible with previous release, except:

- Added a new parameter outThrotThresh (and outThrotType) that stops a port from outputting packets if the output que has more than the threshold. Setting this to 0 is backwards compatible with previous releases. When nonzero, this allows ports to be cascaded while keeping all dropping at the head of the hierarchy.
- Added a check on wrInitialCredit. It must be at least 50 bytes or at least 1 packet. This allows throughput to be optimized. If error is returned, just re-normalize the wr credits such that the minimum meets the above requirement, and the desired ratios are maintained.
- Resolved issues found by static analysis tool.
- Added a new API Qmss_queueBlockOpen which can allocate a contiguous block of aligned queues. It is used by qmQoSchedTest and qmQoSchedDropSchedTest to allocate its queues.

Release 1.0.3.16

- Add qos scheduler APIs to assist in unit conversion: Qmss_convertQoSchedBitRate() and Qmss_convertQoSchedPacketRate(). These can be used to convert a bit per second rate or packet per second rate to a cir/pir/wrr credit value. Note that the format of the credit value has not changed; existing code that calculates credits themselves will still work as is.
- Add more test cases to qos scheduler unit test. 11 deployment scenarios are implemented in test_qosSchedScen.c. These scenarios are automatically run as part of qmQoSchedTestProject. Note that test_qosSched.c contains a #define QOS_SCHED_FAST_SCENARIO. This causes the new scenarios to run for 1/100th of the configured time in order to speed up regression. This define should be removed to run for the fully specified 1 minute per test case.
- Improve robustness of QoS scheduler firmware to invalid configurations. One invalid configuration was found that leads to an infinite loop.

Release 1.0.3.15

- Update SRIO context tracker firmware and qmss_qos to support 6 garbage queues instead of 5, and to make output garbage queues configurable.
 - In order to do this, added Qmss_QoSrioCfg.garbageRetQs to configure the queues.
 - The firmware queues specified through Qmss_QoSrioCfg.queBase reduced from 32 to 21. The garbage return queues were removed and the tx completion queues moved up. Thus, QMSS_QOS_SRIO*Q_OFFSET were updated.
 - Since firmware was changed, it now exports its version number through scratch, which can now be queried through Qmss_getQoSfwVersion().
- Prefix the symbol queueFree to qmssQueueFree to avoid collisions.
- Add multi core critical section around all qmss_qos.c functions that use the firmware's mailbox.

Release 1.0.3.14

- Remove memset() and memcpy() prototypes from qmss_osal.h and replace with #include<string.h> to avoid introducing side effects of removing the prototypes from user code.
- Add Qmss_queuePushDescSizeRaw() and Qmss_queuePopRaw that do not perform address translation. These should only be used by highly optimized applications who manage virtual/physical addresses themselves.
- Add Qmss_getQueuePushHandleCfg() to return the config side address of the queue's D register.

Release 1.0.3.13

- Updated the QoS scheduler firmware and C code. One change is externally visible which is the addition of the Qmss_QoSchedPortCfg.overheadBytes which allows for the Ethernet overhead to be accounted for when scheduling in bytes/bits per second. For example for normal Ethernet frames this would be set to 24 while 0 is backwards compatible with previous versions.
- Additionally, the internal algorithm has been enhanced in the area of weighted round robin handling, robustness to missed timer ticks, and the port scheduler was changed to schedule the CIR of each ports by selecting groups using regular round robin (instead of giving each group its full CIR, potentially starving others)

Release 1.0.3.12

- Remove RM checks from Qmss_ackInterrupt and Qmss_setEoiVector. These checks were redundant to those done when opening the accumulator channel.

Release 1.0.3.11

- Update qmss_device.c for the C6657 device, no functional changes for other devices.

Release 1.0.3.10

- Update firmware for QoS scheduler to fix a crash of the firmware.

Release 1.0.3.9

- Pad qmssGObj to 128 bytes. This prevents the linker from inserting other structures immediately after qmssGObj, which QM would clobber via cache invalidation during its normal operation.

Release 1.0.3.8

- Increase resolution qos scheduler byte credits from a shift of 8 to a shift of 11. This is transparent to user code if QMSS_QOS_SCHED_BYTES_SCALE_SHIFT are used.

Release 1.0.3.7

- Increase number of QoS scheduler lite ports from 6 to 10, increase resolution for packets from a shift of 6 to a shift of 20, and increase resolution for bytes from a shift of 3 to a shift of 8. This is transparent to user code if QMSS_QOS_SCHED_PACKETS_SCALE_SHIFT and QMSS_QOS_SCHED_BYTES_SCALE_SHIFT are used.

Release 1.0.3.6

- Fix doxygen for Qmss_configureQosSrioCluster, Qmss_enableQosSrioCluster, and Qmss_disableQosSrioCluster. No changes to any executable code.

Release 1.0.3.5

- Remove ^Z from bottom of firmware header files which causes compilation problem on Linux. No functional change in any code.

Release 1.0.3.4

- Add SRIO context tracking feature which is in legacy QoS (Qmss_*QosSrioCluster).

Release 1.0.3.3

- Update QoS scheduler firmware to v 1.0.0.2 to fix a minor problem with the pir portion of the scheduler. Greatly enhanced QoS scheduler unit test.

Release 1.0.3.2

- Updated QoS scheduler firmware to v 1.0.0.1 to fix problems with wrong data rates and an infinite loop.

Release 1.0.3.1

- Updated QoS scheduler firmware to fix problems disabling ports and with congestion dropping.

Release 1.0.3.0

- Added a new QoS scheduling algorithm called QoS scheduler. Its unit test/example is in test\test_qosSched.c.

Release 1.0.2.5

- Added an include file in example project to provide platform specific configurations.

Release 1.0.2.4

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

Release 1.0.2.3

- Fix for hint bits in accumulator and QoS firmware. This caused some functions to fail when descriptors are aligned to 4 bits instead of 5. Only 4 bits are required.
- Update Round Robin QoS firmware to fix scheduling.
- Allow queue numbers > 4095 to be specified to QoS
- Fix Qmss_getStarvationCount() API.

Release 1.0.2.2

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

Release 1.0.2.1

- Additional device support

Release 1.0.2.0

- Add queue divert feature to accumulator firmware
- Enhance error checking for Qmss_insertMemRegion
- Fix warnings in examples

Release 1.0.1.0

- Add Round Robin Cluster mode to QoS and update QoS firmware.

Release 1.0.0.17

- Added auto generation of LLD version number and Makefile

Release 1.0.0.16

- Replaced XDC types for Endian define with compiler provided options in Qmss_queuePush()
 - “xdc_target__bigEndian” and “xdc_target__littleEndian” is replaced by “_BIG_ENDIAN” and “_LITTLE_ENDIAN”

Release 1.0.0.15

- Changed Qmss_queuePush() API to use 64 bit atomic writes via the DMA SCR instead of using Queue Proxy.
 - Use “xdc_target__bigEndian” flag for Big Endian
 - Use “xdc_target__littleEndian” flag for Little Endian
 - Updated “qmss_device.c” to include the DMA SCR address

- Updated QOS and accumulator PDSP firmware files for 16, 32, 48 channel for LE and BE. Fixed bug in reclamation queue for monolithic descriptors which was masking queue number to 2048 instead of 8192
- 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
- PDSP firmware change
 - The Accumulator firmware includes an optional reclamation queue which can be used for packet discards. Any descriptor placed on the reclamation queue will be recycled in the same manner as if it had been submitted to CDMA. The descriptor packet information field is used to determine the return queue and the return handling, including options to unlink host descriptors and push to either the front or the back of the return queue. Setting queue to zero disables the reclamation feature
 - Added a new API Qmss_programReclaimQueue().
 - Updated the accumulator PDSP firmware files for 16, 32, 48 channel for LE and BE

Release 1.0.0.13

- Fixed Qmss_queuePopDescSize() API to read the packet size from the status registers instead of management register C.
- 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.
 - QMSS API changed from
`Qmss_Result Qmss_init (Qmss_InitCfg *initCfg);`
to
`Qmss_Result Qmss_init (Qmss_InitCfg *initCfg, Qmss_GlobalConfigParams *qmssGblCfgParams);`
 - Link device specific file **qmss_device.c** in the driver/application.
 - Add an external reference to device specific configuration
`extern Qmss_GlobalConfigParams qmssGblCfgParams;`
 - Changed to enum **qmss_QueueType**

```

Old values
typedef enum
{
    /** Low priority queue */
    Qmss_QueueType_LOW_PRIORITY_QUEUE = 0,
    /** PASS queue */
    Qmss_QueueType_PASS_QUEUE = 640,
    /** INTC pending queue */
    Qmss_QueueType_INTC_QUEUE = 662,
    /** SRIO queue */
    Qmss_QueueType_SRIO_QUEUE = 672,
    /** High priority queue */
    Qmss_QueueType_HIGH_PRIORITY_QUEUE = 704,
    /** starvation counter queue */
    Qmss_QueueType_STARVATION_COUNTER_QUEUE = 736,
    /** Infrastructure queue */
    Qmss_QueueType_INFRASTRUCTURE_QUEUE = 800,
    /** Traffic shaping queue */
    Qmss_QueueType_TRAFFIC_SHAPING_QUEUE = 832,
    /** General purpose queue */
    Qmss_QueueType_GENERAL_PURPOSE_QUEUE = 864
}Qmss_QueueType;

New values

typedef enum
{
    /** Low priority queue */
    Qmss_QueueType_LOW_PRIORITY_QUEUE = 0,
    /** PASS queue */
    Qmss_QueueType_PASS_QUEUE,
    /** INTC pending queue */
    Qmss_QueueType_INTC_QUEUE,
    /** SRIO queue */
    Qmss_QueueType_SRIO_QUEUE,
    /** High priority queue */
    Qmss_QueueType_HIGH_PRIORITY_QUEUE,
    /** starvation counter queue */
    Qmss_QueueType_STARVATION_COUNTER_QUEUE,
    /** Infrastructure queue */
    Qmss_QueueType_INFRASTRUCTURE_QUEUE,
    /** Traffic shaping queue */
    Qmss_QueueType_TRAFFIC_SHAPING_QUEUE,
    /** General purpose queue */
    Qmss_QueueType_GENERAL_PURPOSE_QUEUE
}Qmss_QueueType;

```

- Added new defines for Queue types. Refer to QMSS section under CSL changes
- Changed the accumulator programming APIs to handle high, low and QOS firmware images
 - Deprecated enum Qmss_AccPriorityType
 - Changed APIs
 - Qmss_Result Qmss_programAccumulator (Qmss_AccPriorityType type, Qmss_AccCmdCfg *cfg);
 - Qmss_Result Qmss_disableAccumulator (Qmss_AccPriorityType type, uint8_t channel);
 - Qmss_Result Qmss_configureAccTimer (Qmss_AccPriorityType type, uint16_t timerConstant);

To

- ```

 Qmss_Result Qmss_programAccumulator (Qmss_Pdspld pdspld,
 Qmss_AccCmdCfg *cfg);
 Qmss_Result Qmss_disableAccumulator (Qmss_Pdspld pdspld, uint8_t
 channel);
 Qmss_Result Qmss_configureAccTimer (Qmss_Pdspld pdspld, uint16_t
 timerConstant);

```
- Added new API to allocate queues from within a specified range
    - `Qmss_QueueHnd Qmss_queueOpenInRange (uint32_t startQueNum, uint32_t endQueNum, uint8_t *isAllocated);`
  - Added new APIs to get LLD version ID and Version String
    - `uint32_t Qmss_getVersion (void);`
    - `const char* Qmss_getVersionStr (void);`

### **Release 1.0.0.10**

- Prebuilt libraries are both ELF and COFF. Examples and test projects are ELF only.
- Added a new API to programs the timer constant used by the PDSP firmware to generate the timer tick. Configurable timer ticks are 10us, 20us and 25us. Default is 25us.
 

```

 Qmss_Result Qmss_configureAccTimer (Qmss_AccPriorityType type, uint16_t
 timerConstant)

```
- Updated PDSP firmware files to allow configuration of above mentioned timer tick.
- Removed qmss instance count. Make sure the application calls `Qmss_init()` APIs once. When using multicore application, application **MUST** provide synchronization between cores such that slave cores wait on master core to finish `Qmss_init()` before calling `Qmss_start()` API.
  - An example is provided in “InfrastructureModeMulticore” example.
  - Deprecated error return code **QMSS\_ALREADY\_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 *qmss\_osal.h* and *infrastructure\_multicoreosal.c*
  - ” InfrastructureModeMulticore” 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.
  - Note that PDSP firmware must be downloaded in order to program the accumulator.

### **Release 1.0.0.9**

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

### **Release 1.0.0.8**

- Added a new macro `QMSS_DESC_SIZE(desc)` to get the descriptor size if the popped descriptor contains the descriptor size.

If `Qmss_queuePushDescSize()` API is used to push a descriptor onto a queue, the descriptor when popped will have the descriptor size information in the lower 4 bits. This macro is provided to obtain the descriptor size information. Minimum size is 16 bytes. Maximum size is 256 bytes

- 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 dependency.

- PDSP firmware download intermittently failed in BE mode. Fixed by resetting PDSP's program counter before enabling and using volatile variables for addressing.
- Changed XDC tools version to 3.16.02.32 in examples and test projects.
- Disabling accumulator channel works correctly as long as all interrupts generated by INTD are acknowledged. Removed `QT_WORKAROUND` from examples.
- Internal RAM configuration is fixed. The linking RAM address register is configured with internal linking RAM offset instead of absolute address. Removed `QT_WORKAROUND` from examples.

### **Release 1.0.0.7**

- Modified types from XDC to C99
- Changed all source, header, and example code to reflect CSL include path change in CSL version 1.0.0.13.

### **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.
- 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 QMSS 1.0.0 spec
  - Added new type `Qmss_IntdInterruptType` to acknowledge end of interrupt for QMSS CDMA buffer starvation events
    - API changed from `Qmss_Result Qmss_setEoiVector (Qmss_AccPriorityType type, UInt8 interruptNum);`  
to  
`Qmss_Result Qmss_setEoiVector (Qmss_IntdInterruptType type, UInt8 interruptNum);`
  - Added a new macro `QMSS_DESC_PTR(desc)` to mask off the lower 4 bits of descriptor. If `Qmss_queuePushDescSize()` API is used to push a descriptor onto a queue, the descriptor when popped will contain the descriptor size information in the lower 4 bits. The macro provided will clear out the size information.
  - Setting of threshold on transmit queues to transmit a packet is not required anymore. Transmit pending queue signal is not hooked to threshold.
  - The linking RAM is now 64 bits wide. Declare the data type accordingly when using external linking RAM.
  - Changed maximum supported packet accelerator subsystem (PASS) queues to 12.
  - PDSP firmware files for accumulator and QoS are dated Jan 20th 2010.
- Added APIs to send commands to program QoS PDSP.

### **Release 1.0.0.4**

- Deprecated API `Qmss_getQueuePendingStatus()` used to get queue pending status.

### **Release 1.0.0.3**

- Added new API `Qmss_getQueueHandle` to get queue handle given the queue manager and queue number
- New API is added to pop the packet size along with descriptor address. The API is

Void Qmss\_queuePopDescSize (Qmss\_QueueHnd hnd, Ptr \*descAddr, UInt32 \*packetSize);

- Changed Enum Qmss\_QueueType\_FFTC\_QUEUE to include 2<sup>nd</sup> instance of FFTC. The new enums are Qmss\_QueueType\_FFTC\_A\_QUEUE and Qmss\_QueueType\_FFTC\_B\_QUEUE
- Internal linking RAM use is supported. QMSS 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\_qmss\_cfg.c from example .project files. Remove external reference to sample\_qmssGblCfgParams.
- Device specific configuration parameter has been removed from init API. The API has changed to
 

Qmss\_Result Qmss\_init (Qmss\_InitCfg \*initCfg)
- No need to explicitly include qmss\_acc.h. Including qmss\_drv.h is sufficient.
- PDSP firmware download during init is now supported. It is untested since simulator does not model the download. The pre-built PDSP firmware images for 16 channels, 32 channels and 48 channel accumulator are packaged in the firmware directory.
- New API to set end of interrupt in INTD module. It is untested since simulator does not model INTD. The API is
 

Qmss\_Result Qmss\_setEoiVector (Qmss\_AccPriorityType type, UInt8 interruptNum)
- Modified Infrastructure example to showcase both monolithic and host descriptor use.

### **Release 1.0.0.2**

- A new API is introduced in QMSS LLD. The Qmss\_start() API must be called at least once on every core. It initializes the objects local to each core. This must be the first API called immediately after Qmss\_init(). In case of a core that does not call Qmss\_init() API, Qmss\_start() should be the first API called.
- Qmss\_insertMemoryRegion() API is modified to return the inserted memory region index when successful.

### **Release 1.0.0.1**

- Added Infrastructure Mode Multicore example to demonstrate data transfer and synchronization between cores.
- 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.

- QMSS 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**

| <b>IR Parent/<br/>Child Number</b> | <b>Severity<br/>Level</b> | <b>IR Description</b>                                                                                               |
|------------------------------------|---------------------------|---------------------------------------------------------------------------------------------------------------------|
| SDOCM00105104                      | Major                     | cppiK2KC66BiosTestProject gets symbols in the reserved far sections cannot be accessed as near error during linking |
|                                    |                           |                                                                                                                     |
|                                    |                           |                                                                                                                     |

## **Known Issues/Limitations**

| <b>IR Parent/<br/>Child Number</b> | <b>Severity Level</b> | <b>IR Description</b>                                           |
|------------------------------------|-----------------------|-----------------------------------------------------------------|
| SDOCM00088706                      | Minor                 | Queue diversion feature in accumulator doesn't support both QMs |
|                                    |                       |                                                                 |

## **Licensing**

Please refer to the software Manifest document for the details.

## **Delivery Package**

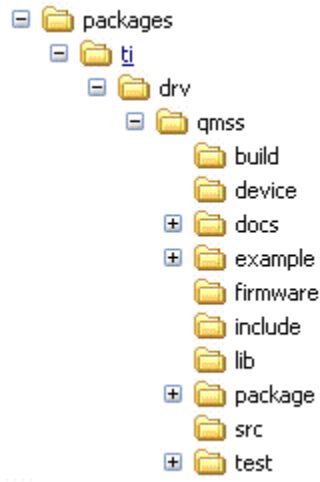
There is no separate delivery package. The QMSS LLD is being delivered as part of PDK.

## **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 QMSS LLD package has been installed:



The following table explains each individual directory:

| Directory Name       | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ti/drv/qmss          | The top level directory contains the following:- <ol style="list-style-type: none"><li><u>Environment configuration batch file</u><br/>The file “setupenv.bat” is used to configure the build environment for the QMSS low level driver.</li><li><u>XDC Build and Package files</u><br/>These files (config.bld, package.xdc etc) are the XDC build files which are used to create the QMSS package.</li><li><u>Exported Driver header file</u><br/>Header files which are provided by the QMSS low level driver and should be used by the application developers for driver customization and usage.</li></ol> |
| ti/drv/qmss/build    | The directory contains internal XDC build related files which are used to create the QMSS low level driver package.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| ti/drv/qmss/device   | The directory contains the device specific files for the QMSS low level driver.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| ti/drv/qmss/docs     | The directory contains the QMSS low level driver documentation.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| ti/drv/qmss/example  | The “example” directory in the QMSS low level driver has the infrastructure mode example.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| ti/drv/qmss/firmware | The “firmware” directory in the QMSS low level driver has the pre-built PSDP firmware files for accumulator and QoS.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| ti/drv/qmss/include  | The “include” directory has private QMSS low level driver header files. These files should not be used by application developers.                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| ti/drv/qmss/lib      | The “lib” folder has pre-built Big and Little Endian libraries for the QMSS low level driver along with their <u>code/data size information</u> .                                                                                                                                                                                                                                                                                                                                                                                                                                                               |

|                     |                                                                                                                                                 |
|---------------------|-------------------------------------------------------------------------------------------------------------------------------------------------|
| ti/drv/qmss/package | Internal QMSS low level driver package files.                                                                                                   |
| ti/drv/qmss/src     | Source code for the QMSS low level driver.                                                                                                      |
| ti/drv/qmss/test    | The “test” directory in the QMSS low level driver has unit test cases which are used by the development team to test the QMSS low level driver. |
| eclipse             | The “eclipse” directory has files required to integrate QMSS 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/qmsslldDocs.chm               |
| 2          | Design Document                          | docs/CPPI_QMSS_LLD_SDS.pdf         |
| 3          | Software Manifest                        | docs/QMSS_LLD_SoftwareManifest.pdf |