Open Event Machine library

Release Notes

Applies to Product Release: 01.10.00.00:
Publication Date: February, 2014

Contributors to this document

Texas Instruments, Incorporated
821 avenue Jack Kilby
06270 Villeneuve-Loubet Cedex,
FRANCE

Contents

Release Notes

OpenEM library version 01.10.00.00

Overview

This document provides the release information for the Open Event Machine (OpenEM) library for KeyStone I devices (TMS320C6670 and TMS320C6678) and KeyStone II devices (TCI6638K2K).

Open Event Machine library includes:

- Compiled library in Big and Little Endian
- Source code
- Examples for ARM and DSP
- API reference guide
- User's guide
- OpenEM white paper
- Compilation guide

OpenEM Library Tools set

OpenEM library has been compiled and validated on following external tools:

- TI TMS320C6000 C/C++ Code Generation Tools version 7_4_1
- KeyStone I
  - TI BIOS MCSDK version 02.01.02.05
    - TI Code Generation XML Processing Scripts
    - TI PDK C6670 version 1.1.2.5
    - TI PDK C6678 version 1.1.2.5
    - TI XDC TOOLS version 3.24.5.48
  - TMDSEVM6670/ TMDSEVM6678 EVMs
- KeyStone II
  - TI MCSDK version 03.00.00.10
    - TI Code Generation XML Processing Scripts
    - TI PDK KEYSTONE II version 1.0.0.10
    - TI XDC TOOLS version 3.24.5.48
  - TI MCSDK version 03.00.00.11
    - mcsdk_linux_3_00_00_11
  - TI MCSDK version 03.00.02.14
    - K2_LINUX_03.08.04_13.09_01
    - K2_RT_LINUX_03.08.04_13.09_01
  - XTCIEVMK2X EVM
  - GCC 4.7.3 for ARM (Linaro GCC 2012.10 or higher)
- Code Composer Studio 5.3.0.90 or higher

New/Updated Features and Quality

This is an official release.

Release 1.10.0.0 (KeyStone I and KeyStone II – DSP and ARM)

- Bug fix in Generic part:
  - SR#1-59236181 : Error when initializing EM public descriptors in RH library triggers K2 communication failure
    - Description:
      - The customer is running into a case where all communication between K2s is failing. The root cause of the issue appears to be a misconfiguration of memory pools associated with the EM public descriptor region.
      - There is code in the ARM TI RH library (ti_em_rh_init.c, init_events()) that sets up the descriptor pools. The EM public event memory region is sized for 4096 descriptors, and the customer is using every one of those 4096 (the last 128 being used for the RXFRAG pool). However, the code skips the memory associated with the first descriptor – this means the last descriptor is outside the address the region, and the address that is associated with that descriptor when it is pushed onto / popped from the RXFRAG queue is garbage.
    - Openem architecture:
      - Each OpenEM descriptor has a preamble occupying the last 2 words of the previous descriptor. That's the reason why the first descriptor of a descriptor region is lost since it doesn't have a previous descriptor. Therefore, the first descriptor of a descriptor region cannot be assigned to any pool. This implies that init_events should check that the capacity of the descriptor region is at least one more than the accumulated capacity of the event pools. Today's implementation doesn't perform this check. As a result, the last descriptors of the last pools will not be part of the descriptor region if the capacity of the descriptor region is less or equal to the accumulated capacity of the event pools. When those pointers to those descriptors are pushed to the free queues, the linking RAM may get corrupted causing fatal errors in PDSP/Packet DMA/...
    - Implementation:
      - Added a check in the Linux RH library and in the DSP examples.
- Feature improvements in Generic part:
  - SR#1-58961691 : Filtering BIP Based On MSB Of Stream ID
    - Description:

- The checks in the OpenEM XGE router now obey to the following behavior:
  - Recycle received Ethernet frames if wrong Ethertype/service type and redirection is disabled.
  - Redirect to miss queues if wrong Ethertype/service type and redirection is enabled

- Implementation:
  - For that purpose, API to enable/disable miss redirection for Ethertype/service type have been added:
    - em_status_t ti_em_xge_rx_miss_enable(ti_em_xge_rx_miss_type_t miss_type)
    - em_status_t ti_em_xge_rx_miss_enable(ti_em_xge_rx_miss_type_t miss_type)
    - typedef enum ti_em_xge_rx_miss_type_e_ {
      ti_em_xge_rx_miss_type_ETHER_TYPE = 0,
      ti_em_xge_rx_miss_type_SERVICE_TYPE = 1}
      ti_em_xge_rx_miss_type_e;

Release 1.9.0.1 (KeyStone I and KeyStone II – DSP and ARM)

- Bug fix in Generic part:
  - OpenEM scheduler does not preserve order from SD to AP queues
    - Description:
      - When an atomic event is popped from the SD queue, the scheduler checks whether the requesting core is already processing an event associated with the same atomic queue. If the core is, the scheduler will send the event to the dispatcher regardless of the state of the corresponding AP queue (where multiple events associated with the same atomic queue may be waiting).
    - Implementation:
      - The scheduler now systematically pushes the event to the AP queue if it is already processing an event associated with the same atomic queue. Events will be popped from the AP queue in the correct order then. This fix impacts the FW scheduler.
  - SR#1-55384031 : Packet Accelerator silently dropping packets sent to queue 640 (PDSP1)
    - Fixed bug in previous implementation (1.9.0.0), the 4 LSB of the packet address, used by the PKTDMA were not configured appropriately, PS word were not considered by the PA.

- Implemented the case when the em_send() API directly sends the packet to PA.

Release 1.9.0.0 (KeyStone II – DSP and ARM)

- Feature improvements in Generic part:
  - Optimize OpenEM data base
    - Description:
      - device route table: 16 bytes per device, 64 Kbytes for 4K devices
      - Since a single device should be able to connect to not more than 256 other devices, we could significantly optimize the device route table by a 2-step mapping:
        - the first table maps from a 12-bit device ID to a 8-bit index -> this table is 4096 x 1 B = 4 kB
        - the second table maps from a 8-bit index to the XGE/RIO parameters -> this table is 256 x 16 B = 4 kB
      - The total would be 8 kB and this could be kept in MSMC.
    - Implementation:
      - TI_EM_DEVICE_NUM changes from 4096 to 256
      - Emti_QUEUE_HANDLE_DEVICE_IDX_MASK changes from 0xFFF00000 to 0xFF000000
      - Emti_QUEUE_HANDLE_DEVICE_IDX_SHIFT changes from 20 to 24
      - stream_idx in ti_em_device_xge_route_t changes from uint8_t[2] to uint8_t. em_send constructs the stream ID in the BIP header as follows:
        - 8 LSB: local device ID as specified by configuration
        - 8 MSB: stream ID as specified by route
    - Description:
      - HW queue state table: 2 bytes per HW queue, 32 Kbytes for 16K HW queues. It should be possible to store the HW queue state table in DDR.
    - Implementation:
      - Move hwQueueStateTbl and sdQueueStateTbl from .tiEmGlobalFast to .tiEmGlobalSlow
  - CR#2: Support of error counters
    - Added following API: em_status_t ti_em_counter_get(ti_em_counter_type_t type, ti_em_counter_value_t* value_ptr, int flag);
    - Added following Enumeration: typedef enum ti_em_counter_type_e_ {ti_em_counter_type_XGE_RECEIVE = 0, ti_em_counter_type_XGE_MISS = 1, ti_em_counter_type_XGE_WRONG_MAC = 2, ti_em_counter_type_XGE_WRONG_PROCESS = 3, ti_em_counter_type_XGE_WRONG_SEQUENCE = 4} ti_em_counter_type_e;

        - Flag is used to reset the counter right after it has been read.
    - Clean up macros for buffer mode
        - TI_EM_BUF_MODE_GLOBAL_TIGHT -> TI_EM_BUF_MODE_TIGHT
        - TI_EM_BUF_MODE_GLOBAL_LOOSE -> TI_EM_BUF_MODE_LOOSE
        - TI_EM_BUF_MODE_LOCAL -> obsolete
- Bug fix in Generic part:
    - SR#1-56260551 - Message 'stuck' waiting for dispatch when using ATOMIC load-balanced (4-core) queues
        - Fixed race condition for atomic queues in scheduler firmware.
        - Changed Linux RH library to use un-cached memory for PDSP communication memory between Linux and the PDSP.
        - Use a DSB instruction instead of a DMB on ARM for memory barrier to ensure correct write operations on DMA bufferable memory.
    - SR#1-55384031 : Packet Accelerator silently dropping packets sent to queue 641 (PDSP1)
        - When OpenEM transfers packets to the Packet Accelerator (PA), OpenEM now formats these packets to be compatible with the RIO format supported by the PA (the ethernet format also supported by the PA is not supported by the OpenEM). For that purpose, OpenEM performs the following actions:
            - In case of FW queue routing sending packets to the PA
                - Distinguishes PA TX queue (hard-coded, device dependent) from other HW queues
                - Sets packet type to "RIO"
                - Writes OpenEM queue ID/process ID to PS info (RIO stream ID)
                - Pushes packet to PA TX queue (requires memory fence).
            - This is a limited implementation which does not support the case when the em_send() API directly sends the packet to PA.
        - This is a limited implementation which does not support the mapping of multiple event queues to the same HW queue.
        - This is a limited implementation in which case the process index written by the FW queue routing in the “protocol specifics” words of the descriptor is hard coded to 0.

Release 1.8.0.1 (KeyStone II – DSP and ARM)

- Bug fix in Generic part:
    - SR#1-55818181 - Issue with un-cached MSMC

- Complete description of the issue will be available in next revision of “TCI6638K2K - Communications Infrastructure KeyStone SOC - Silicon Errata- SPRZ401A”. The problem exists on Keystone 2 - PG1.x devices only. It will be fixed for Keystone 2 - PG2 devices. In order to know the device version, read the register located at address 0x02620018: 0x0b98_102f for PG1.0 and 0x1b98_102f for PG1.1.
- Problem description
    - The K2 device does NOT allow having mappings from 0x80000000-0xffffffff (virtual) to 0x0c000000 (physical, MSMC SRAM) in the SES MPAX. The devices ALLOW having these aliases in the XMC MPAX. Failure to follow this will lead to a lockup condition.
    - Common use case is to make an alias of MSMC-SRAM to give it different permissions or cachability to the CPU, then silently pass these virtual addresses to DMAs (like EDMA, pktdma, wireless accelerators). To avoid the lockup condition, SW must translate these addresses to 0x0c000000 based addresses before passing to the DMA.
    - OpenEM is exposed to this problem since:
        - We recommend storing the event descriptors in un-cached MSMC as OpenEM makes extensive use of Packet DMA and PDSP FW.
        - We recommend storing the “.tiEmGlobalFast” memory section in un-cached MSMC.
        - XGE header buffers and XGE fragment buffers shall be store in un-cached memory.
- The workaround of the MSMC stall issue for OpenEM is as follows:
    - MSMC alias for storing un-cached descriptors is hard-coded between 0x1C00_0000 and 0x1C5F_FFFF.
    - OpenEM translates pointers to descriptors if the (dscPtr & 0xFF00_0000 == 0x1C00_0000) condition is met. OpenEM translates from aliased to native MSMC address when pushing the descriptor pointer to a HW queue or when writing the descriptor pointer to a descriptor: nativeDscPtr = aliasDscPtr & 0FFF_FFFF
    - OpenEM translates pointers to descriptors if the (dscPtr & 0xFF00_0000 == 0x0C00_0000) condition is met. OpenEM translates from native to aliased MSMC address when popping the descriptor pointer from a HW queue or when

reading the descriptor pointer from a descriptor: aliasDscPtr = nativeDscPtr | 1000_0000

- When building OpenEM, this behavior can be enabled/disabled through the compile flag “K2_PG1_MSMC_STALL_WA”. OpenEM 1.9.0.0 targets K2 PG1.x devices and assumes K2_PG1_MSMC_STALL_WA is defined.

Release 1.8.0.0 (KeyStone I and KeyStone II – DSP and ARM)

- Feature improvements in Generic part:
  - Add missing threshold management in PktDMA chaining rx flow configuration
    - In order to use different receive hw queues based on the incoming event size when using Infra-PktDMA chaining, the ti_em_osal_chain_rx_flow_open() function (OSAL on Linux) and ti_em_chain_rx_flow_open (pdk_hal on DSP) have been enhanced with size a, size b, size c and size d parameters used to configure the hw queue thresholds of the reception flow. This does not change the OpenEM configuration API but now the pool_idx_size_a, pool_idx_size_b, pool_idx_size_c and pool_idx_size_d fields of the ti_em_chain_config_t structure are taken into account by the hardware configuration layer.
  - Optimize ti_em_combine
    - There was a need to chain multiple buffers to an event, starting from the first part. This was quite slow with ti_em_combine, because it always needed to traverse the complete list of descriptors. To cope with this issue, the OpenEM has been changed according to following APIs:
      - em_event_t ti_em_combine(ti_em_pair_t pair, ti_em_iterator_t* iterator_ptr)
        - preconditions: iterator has been started on pair.head_event
        - combines pair.head_event and pair.tail_event (last descriptor of pair.head_event points to first descriptor of pair.tail_event)
        - postconditions: current descriptor of iterator matches last descriptor of old pair.head_event (can be resplit right away)
      - ti_em_pair_t ti_em_split(em_event_t event, ti_em_iterator_t* iterator_ptr)
        - preconditions: iterator has been started on event
        - splits event (current descriptor of iterator becomes last descriptor of pair.head_event; descriptor following current descriptor of iterator becomes first descriptor of pair.tail_event)

        - postconditions: current descriptor of iterator matches last descriptor of pair.head_event (can be recombined right away); second and following buffers of pair.tail_event are cache coherent
    - What has changed?
        - new prototype for ti_em_combine: added iterator_ptr as input/output
        - new behavior for ti_em_split: splits behind current descriptor of iterator instead of in front
        - new behavior for ti_em_iterator_start: current descriptor matches first instead of second (allows combining if head is not scattered)
- Bug fix in Generic part:
    - SR# 1-55817571 - XGE chaining broken on 2nd EM Process
        - XGE chaining did not work with the 2nd EM process. The problem had to do with the initialization of the global variable. txQueueBaseIdx is now set during global init phase for every process. And now, TX header buffer size is set during global init phase for header descriptors pool of every process.
    - SR# 1-54125657 - Failures from Emti_xgeSend will not be returned to the caller
        - The return code for Emti_xgeSend call was not propagated back to caller. The em_send now returns what is returned by Emti_moduleChainSend, which is not always EM_OK.
    - Uninitialized variables in XGE router
        - Some variables of the XGE router were not initialized conducting to wrong error detection. The issue has been fixed. A SW workaround consisted in a reset of the PDSP RAMs by the SW prior to activate the FW.
- Bug fix in Linux part:
    - em_send() was not correctly returning an EM_ERR_NOT_FOUND error when running out of divert queues for XGE chaining.

Release 1.7.0.0 (KeyStone I and KeyStone II – DSP and ARM)

- Feature improvements in Generic part:
    - The router firmware checks the Ethertype and the destination MAC address of the received packet before processing.
        - If etherType is correct the packet is processed, otherwise it is pushed in a rx Miss Queue. If the destination MAC address is correct, the packet is processed, otherwise, it is pushed for recycling. Depending on the number of VLAN priorities for EM Chaining and the ingress 10Gbe port, the input queue is different. Counters are updated to save

information on packet received, packet with unexpected ethertype or unexpected Mac address.

- The EM configuration now allows setting in more than 1 VLAN priority for the EM Chaining traffic.
  - Up to 8 tx and Rx queues can be used for chaning traffic, corresponding to 8 VLAN priorities.

- Feature improvements in Linux part:
  - Support any number of Linux joining processes:
    - The Linux ti_em_rh_join() and ti_em_osal_core_join() functions have been enhanced with an extra parameter specifying the OpenEM core Id to join. If -1 is provided, an unique free OpenEM core Id (up to 8) is allocated and returned as the function result. If providing a valid OpenEM core Id, the OpenEM instance is joined using this OpenEM core Id that can be eventually shared between different OpenEM processes. But only one dispatcher can run on a given OpenEM core Id.
    - The OpenEM XGE chaining has been enhanced to use a pool of TI_EM_XGE_TX_DIVERT_QUEUE_NUM (32 by default) divert queues used for XGE packet fragmentation instead of a static assignment per OpenEM core Id. It allows multiple Linux processes that have joined a running OpenEM instance and sharing the same OpenEM core Id to be able to send correctly in parallel fragmented events through XGE chaining. 32 queues have been reserved allowing the 4 ARM cores to perform in parallel up to 8 send operations per core of fragmented events through XGE chaining.
    - If there is no more available divert queues at a given moment, em_send() will returns with an EM_ERR_NOT_FOUND error. It is then up to the OpenEM application of retrying the event send operation when a free divert queue would be available for that purpose.

Release 1.6.0.4 (KeyStone I and KeyStone II – DSP and ARM)

- Bug fix in Generic part
  - XGE router firmware must ignore 8 most significant bits of stream ID field:
    - Description: The router behavior is not correct when using the 8 MSB part of the stream ID.
    - Solution: The stream ID field is 16 bits wide but only the 8 LSB identify the source device. Two different source devices cannot share the same 8 LSB. The router firmware has been fixed to only use the 8 LSB for re-assembly and ignore the 8 MSB (but these 8 MSB are still sent within the BIP header).
  - Use only one TX queue for OpenEM chaining:

        - Description: Chaining between two OpenEM processes using infra-PktDMA may fail if the TI_EM_CHAIN_TX_QUEUE_NUM value is different between the two OpenEM process (like by default between DSP and ARM).
        - Solution: The em_send function always uses only the first PktDMA queue (Infra-PktDMA transmit queue base index) when performing chaining between different processes.
- Bug fix in Linux part:
    - Multi-threaded scheduler feature:
        - Description: The goal is to increase the scheduling throughput by supporting up to 4 threads per Open-EM instance.
        - Solution: The feature is now supported in Linux RH library.

Release 1.6.0.3 (KeyStone I and KeyStone II – DSP and ARM)

- Bug fix in Generic part
    - SR# 1-53102181:
        - Description: Make sure only first OpenEM process initializes XGE Hw.
        - Solution: Feature request is implemented.
    - SR# 1-53101333:
        - Description: em_send skips last descriptor(s) (not included in packet length).
        - Solution: The Open-Em is now able to detect any additional descriptor (for control purpose at application level) when sending an event to a remote device instance. The additional descriptor is not discarded and is recycled in its release queue.
    - SR# 1-53930301:
        - Description: EM Chaining over XGE fragment combining introduces 4 bytes of garbage data into the payload.
        - Solution: The XGE router firmware now removes those 4 bytes prior to rebuild the packet.
    - Using Packet DMA infrastructure for releasing events:
        - Description: Previously only the DSP version was using PktDMA infrastructure to release events.
        - Solution: This feature allows using the Infra-PktDMA release mechanism in Linux. Now the behavior is the same for both Linux and DSP. Also if specifying -1 in the dma_idx field of the ti_em_config_t during the initialization or if setting the TI_EM_NO_PKTDMA_RELEASE flag during the compilation of the OpenEM library, this Infra-PktDMA based release mechanism will not be used and a software release queue

will be used instead. It can be helpful for debug purpose or if user does not want to use a dedicated Infra-PktDMA channel/flow for that purpose.

- Atomicity was not working with post-storing:
  - Description: in fact the atomicity was lost by the chaining routing.
  - Solution: The firmware performing scheduling and chaining has been fixed.
- Preloading was not working after ti_em_alloc_local and em_send.
  - Description: the event type was overwritten with 0 in the firmware.
  - Solution: The firmware performing scheduling and chaining has been fixed.
- Corrupted device route table
  - Description: There is an overlap between adjacent entries in the device route table. Writing to one entry will corrupt the neighboring entry.
  - Solution: Entry size is corrected to prevent overlap.
- Change EM_QUEUE_STATIC_MIN from 0 to 1
  - Description: “0” corresponds to EM_QUEUE_UNDEF, it shall not be used as a static queue.
  - Solution: macro has been modified and scheduler/router firmware has been updated to reflect modification.
- Scattered events implementation change:
  - Description: previous implementation was relying on the fact that the event length matches the total length of all the buffers.
  - Solution: The iterator now stops when the next descriptor pointer equals NULL, not when the accumulated buffer size matches the event size. This applies to all code that iterates, not only ti_em_combine.
- Lowering the HW queue base alignment for the global init
  - Macro name is changed from TI_EM_QUEUE_BASE_ALIGNMENT to TI_EM_HW_QUEUE_STEP
  - Macro value is changed from 128 to 32
- The post-storing does not need a route for be enabled.
  - A check that was verifying the existence of a route has been removed.
- Resource management on the number of channels used for chaining.
  - A public macro is available (TI_EM_CHAIN_TX_QUEUE_NUM). Its default value is 4 for DSP and 1 for Linux.
  - It can be modified to a power of 2. In that case, the Open-EM library shall be recompiled.

- Bug fix in DSP part:
  - SR#1-52814679:
    - Description: First event is lost after initialization
    - Solution: PDSP local memory is cleared before firmware loading in all DSP examples.
- Bug fix in Linux part:
  - SR#1-53101311:
    - Description: Address translation is missing when sending scattered events.
    - Solution: Address translation is now done when constructing a scattered event.

Release 1.6.0.2 (KeyStone I and KeyStone II – DSP and ARM)

- Bug fix in Generic part
  - SR#1-52815021:
    - Description: Open-EM needs to support payloads smaller than 34 bytes for chaining over XGE.
    - Solution: Feature request is implemented.
  - SR #1-52815028:
    - Description: EM Chaining events that are smaller that RX FRAG size arrive at destination from RX FRAG pool.
    - Solution: Starting from 1.5.0.0, all events (regardless of the intended OpenEM process and the structure - scattered or not) are forwarded to the TX infrastructure Packet DMA queue of the intended process.
- Bug fix in Linux part:
  - SR#1-52814679:
    - Description: First event is lost after initialization
    - Solution: PDSP local memory is cleared before firmware loading and use an allocated standard queue for descriptor release in case of firmware error.
  - SR#1-52814671:
    - Description: Failing init_pdsp
    - Solution: The application has to always perform local ARM timer initialization (based on ARM PMU) after setting core affinity.
  - SR#1-52814694:
    - Description: Illegal instruction and one descriptor missing when receiving fragmented event over XGE.
    - Solution: Issue is been fixed with the fixes related to SR #1-52814671 and SR #1-52814679.

- Linux examples always use poll mode (no interrupt mode) for dispatchers running on core 0.
- Fix in Linux Example_1 for small event case (<34 bytes).

Release 1.6.0.1 (KeyStone I and KeyStone II – DSP and ARM)

- Bug fix in OpenEM XGE chaining:
  - Make BIP header big endian : The BIP protocol uses Big Endian Byte-order, but in the OpenEM chaining event packet the CIP header is in little-endian. For transmit, em_send() has been updated, and for the receiving side, PDSP now takes care the endian-ness.

Release 1.6.0.0 (KeyStone I and KeyStone II – DSP and ARM)

- Bug fix in chaining examples:
  - ti_em_chain_rx_flow_open call-back function does not configure correctly the RX flow used by chaining. The error was when bigger buffers than maximum size of pools are pushed to chaining flow. The fix is to configure rx_fdq1, rx_fdq2, rx_fdq3 RX free queue indexes.
  - Add missing divert queues configuration in Linux Example_1.
- Bug fix in XGE example:
  - VLAN tags for XGE examples/tests are currently incorrect and must be fixed according to the IEEE 802.1Q standard.
- Bug fix in the XGE router firmware:
  - Queue index was not properly set by XGE router firmware and resulted in errors when RX fragments were reused.
- Bug fix in the ti_em_dispatch_once function
  - An incorrect mask was removing the higher bits of the device ID before calling the receive function.
- Bug fix in the ti_em_iterator_next function
  - The current size was not correctly managed.
- Port the Example_2 for MCSDK 3.0.0.10 (beta) and MCSDK 3.0.0.11 (GA):
  - Use PDSP #4 for DSP and #2 for Linux.
  - Use flow 25 for connecting.
  - Use queues 6656 and beyond for DSP.
  - Removed useless queue open functions for static queues.
- Feature improvements:
  - SRIO chaining is now available on Keystone II devices.
    - Configuration improvement: the Open-EM device ID is now de-correlated from the device SRIO node ID.

- New field in RIO chaining configuration (ti_em_chain_rio_config_t structure): my_node_idx
- New prototype for the add_rio_route (with node_idx of the remote SRIO device)

- The number of Open-EM firmware loads has been reduced and simplified
  - em_scheduler_router_fw_c667x_pdsp1 and em_scheduler_router_fw_c667x_pdsp2 for Keystone I devices
  - em_scheduler_router_fw_c663x and em_router_fw_c663x for Keystone II devices
- The interrupt scheme is the same for all devices.
  - Interrupts are disabled by default and it's the application's responsibility to enable/disable them on a core basis.
- The coh_mode field has been removed from the ti_em_pool_config_t structure.
- The TI_EM_PRIVATE_EVENT_NUM macro has been obsoleted.
  - To allocate the mandatory 8 kbytes for Open-EM private data, a new macro is available (TI_EM_PDSP_GLOBAL_DATA_SIZE).
- Post-storing is enabled: it allows to
  - Allocate a local event as a working event and to send it (the Open-EM converts it into a global event). Such a local event can be freed.
  - Combine local events with global events and to send then the combined event.
  - Allocated local event buffers are aligned on 8 bytes boundaries.
- Chaining and post-storing make use of 4 PKTDMA TX channels in parallel.
  - ti_em_chain_config_t provides the base index for the multiple PKTDMA TX queues.
  - Multiple TX channels, RX channels and RX flows are configured.
  - The TX channel is randomly selected during the em_send() call based on the queue handle.
- The receive function behavior has changed:
  - The received event in the prototype is always a global event.
    - If the event type was without preloading, the behavior remains the same.
  - In case of preloading (feature available only on DSP):
    - The application can claim a pointer to the preloaded buffer.
    - The application has no more access to the local preloaded event. This one is automatically freed by the dispatcher.
    - It is the application's responsibility to free the global event.

- A new API (ti_em_exit_global) has been added in order to properly stop the Open-EM and its associated private resources (queues, firmwares, rx flows, tx/rx queues).
- The management of Core Id has been enhanced in Linux:
  - Core Ids are now allocated and released dynamically, up to 8 cores can be allocated at the same time on a Linux instance.
  - A process that leaves with ti_em_osal_core_exit() will release its Core Id.
  - Joining processes have now their own Core Id.
  - It allows to use an own set of divert queues for joining processes instead of using the OpenEM Core 0 ones.
  - OSAL database and PDSP communication memory can be now accessed from joining processes.
  - The ti_em_osal_core_init() OSAL function has been renamed to ti_em_osal_core_init_global() since it must be only called once.
- Increase the total amount of Pool Ids that can be stored in QMSS descriptors from 16 to 256 (this is used for chaining on Linux).
- Example_1 on Linux has been enhanced with more runtime options allowing to easily illustrate chaining capabilities like scatter/gather:
  - '-j' to define the number of jobs
  - '-i' to define the transmit interval in microseconds
  - '-z' to define the payload data size to transfer
  - '-m' to enable multi-frame
  - '-f' to set the XGE transmit fragment size
  - '-r' to set the XGE receive fragment size
  - '-b' to set the event buffer size

Release 1.5.0.1 (KeyStone I and KeyStone II – DSP and ARM)

- Support ARM interrupts when event scheduled
- Validated with PREEMPT_RT

Release 1.5.0.0 (KeyStone I and KeyStone II – DSP)

- Move queue type in event descriptor
- Multi-threaded scheduler
- Make it possible to allocate HW queues on any QM
- Use relative rather than absolute queue index for DMA TX queues
- Get rid of host process

- Reduce the number of private descriptors
- Separate TX DMA queues for chaining
- Receive events from HW queues
- Allow to query buffer mode and coherency mode
- Provide helper functions for accessing device, process and queue index in queue handle
- Add missing trace points
- Verify that SW events are not sent to TX queue
- Friendly operation without preload
- Fix Bugs in ti_em_packet_set_buffer_info function. Incorrect mask prevents usage of queue indexes superior to 8192. Provides an example for configuring the 10 Gigabit chaining mechanism (KeyStone II). XGE TX queues are replaced by infrastructure PKTDMA TX queues
- Improved initialization procedure. ti_em_global_init function only allocates queues, flows, channels when needed.
- Provides an example for configuring the 10 Gigabit chaining mechanism (KeyStone II). 10 Gigabit TX queues are replaced by infrastructure PKTDMA TX queues

Release 1.4.0.0 (KeyStone I - DSP)

- Added support for chaining over SRIO on KeyStone I devices

Release 1.3.0.1 (KeyStone II – DSP and ARM)

- Fix Bug in Router Firmware
  - Firmware did not support Device Command queue index superior to 4095.
- Fix Bugs in em_send/em_send_group function
  - Sending to another process TX queue was not done with the correct descriptor size tag.
  - Checks should not be performed on a queue handle for a different process or a different device.
- Fix Bug in ti_em_queue_create_hw_static function
  - Incorrect check on the process index
- Fix Linux Bug in ti_em_rh_init.c
  - Private descriptor mapping's paddr was checked incorrectly.
    - Was checked before copying user specified address.
- Fix Linux Bug
  - PktDMA flows, channels and queues base were incorrect.
- Feature Improvement:
  - Add capability to provide an offset for moving the buffer pointer in the event descriptor.
- Provides a full example with one ARM-linux instance and one DSP instance

- Both instances communicates through router

Release 1.3.0.0 (KeyStone II - DSP and ARM)

- Fix Bug in dispatcher function
  - The clearing of the atomic locality hint was not performed on the correct object.
- Fix Bug in dispatcher function
  - In case of event group notification, the notification events were not sent properly
- Improved performance of the event release process
  - The event size is to be reset to zero bytes before being pushed to a release queue
- Merged code source for DSP and ARM-linux
- Added support of previously defined functions
  - em_status_t em_queue_create_static(const char* name, em_queue_type_t type, em_queue_prio_t prio, em_queue_group_t group, em_queue_t queue)
- Added support of sending to HW queues
- Full support of chaining:
  - Over 10 Gigabit Ethernet (XGE) sub-system
  - Intra-device across processes
    - When adding a route to another process, please note that the flow index to be used for the other process shall be equal to the other process's DMA base Index + 8.
      - There is currently no API to retrieve this information.
  - Router is available as binary load for a Navigator PDSP.

Release 1.2.0.1 (KeyStone I and KeyStone II - DSP)

- Fix Bug in em_send function
  - The cache coherency was not working properly when in chaining with cache coherency managed by the OpenEM.
- Fix Bug in event_machine_sw_router function
  - Memory fences were missing.
  - Access to Command Queue Index was incorrect.
- Fix Bug in em_core_count function
  - Was always returning 8.
  - Now returns the number of cores that did call ti_em_init_local function.
- Features not supported in this release (and in the previous release 1.2.0.0)
  - em_queue_create_static
- Example_0 is back up-to-date

  - Provides an example without chaining
- API documentation has been reformatted.
- Moved em_scheduler firmware location from .tiEmLocal section to .tiEmGlobalSlow section

Release 1.2.0.0 (KeyStone I and KeyStone II - DSP)

- Fix Bug in ti_em_dispatch_once
  - The free info word is not copied correctly.
- Fix generated makefiles
  - Automatically generated makefiles need to be manually modified to remove lines related to the compilation of ".c" files that are not delivered along with the OpenEM.
- ~~Added support of previously defined public functions~~
  - ~~em_status_t em_queue_create_static(const char* name, em_queue_type_t type, em_queue_prio_t prio, em_queue_group_t group, em_queue_t queue)~~
- OpenEM chaining functionality over 10 Gigabit Ethernet (XGE) sub-system
  - Transmit Side
    - XGE and chain headers are inserted
    - Events may be fragmented into fragments
    - Fragments are pushed to XGE Transmit (TX) queue(s)
  - Receive Side
    - Fragments are popped from XGE Receive (RX) queues
    - Fragments may be reassembled into an event
    - XGE and chain headers are removed
    - The events are forwarded to event queues
  - Global queue handles are supported to encode:
    - Device ID (for 4K devices)
    - Queue ID (for 16K queues) with 256 queue sets and 64 queues per set
  - Managing routing tables
    - Device routing table for 4K devices
    - Queue routing table for 256 entries

Release 1.1.0.0 (KeyStone I - DSP)

- Remove Public Dependencies on Private Headers
- Fix Compiler Errors for Queue Groups
  - Declarations of em_queue_group_create and em_queue_group_delete in .h and .c files do not match.

- Fix Compiler Warning on Emti_error
  - Emti_error(em_status_t error, em_escope_t escope, ...) cannot be inline because of variable arguments. Remove variable arguments.
- Fix Bug in ti_em_dispatch_once
  - Non-preloaded event is handled incorrectly
  - if ((lvEventType & TI_EM_EVENT_TYPE_PRELOAD_MSK) != TI_EM_EVENT_TYPE_PRELOAD_OFF)
- Rework OpenEM memory section
  - Default EM database occupies more than 200 kB of MSMC memory. For better MSMC usage, rarely used tables shall be moved to DDR
- Use HW Queue instead of Region Index for Initialization of Private Events
- Change in ti_em_init_global () to reduce flooding.
- ti_em_claim_global Shall Not Modify Event
- New Cache Coherency Handling
  - em_free and em_send handle Cache Coherency only for first Buffer.
  - Iterators handle Cache Coherency for non-first Buffers.
  - Event Coherency Mode and Event Coherency Level are removed.
  - New API to Handle Coherency but not send. Handle Cache Coherency same way as em_send.
- Protocol Specific Words always in Descriptor
  - In all places where PS Location needs to be overwritten, it is written as "in descriptor".
  - New Descriptor Format that leaves space for 2 PS Words in a 16-word descriptor (words 12 and 13).
  - Buffer Free Function Pointer (presence depends on Buffer Free Mode, not supported together with PS words) to EPIB.
  - Two other EM Words are needed for fields that are always present (Event Free Mode, Buffer Free Mode, Buffer Free Policy, Free Push Policy, Free Queue, Buffer Coherency Mode, Buffer Dirty Flag). Use Fixed Negative Descriptor Offsets (words -1 and -2). Limitations: First Descriptor in memory region cannot be used. Not aligned with cache lines, cannot place descriptors in cached memory in the future.

Release 1.0.0.3 (KeyStone II - DSP)

- Support for KeyStone II devices
- Fix Scheduler Policy
  - Event leak possible when using atomic queue

Release 1.0.0.2 (KeyStone I - DSP)

- Support for several pre-compiled variants of the OpenEM library
  - Centralized Scheduling policies
  - TMS320C6670 and TMS320C6678 platforms
- Supported Feature set (detailed concepts are described in the OpenEM white paper):
  - Parallel and Atomic event queues
  - Event Group notion to perform fork-join operations
  - Static and Dynamic Locality schemes
  - Schedule pre-fetching
  - Event preloading
  - Management or not of the cache coherency for the event content(s)
  - Scattered events
- Example on TMDXEVM6670L / TMDXEVM6678L EVMs from Advantech

Resolved Incident Reports (IR)

None.

Known Issues/Limitations

- For Keystone II devices
  - The OpenEM scheduler firmware has been tested on all PDSPs
    - Up to two instances of scheduler firmware (OpenEM instances) may reside in the QMSS.
  - The OpenEM router firmware has been tested on all PDSPs
    - Only one instance of the 10 Gigabit router firmware may reside in the QMSS.
  - As PDSPs 1,2,5,6 interrupts are routed to INTD1 and PDSPs 3,4,7,8 interrupts are routed to INTD2, all PDSPs allocated for scheduling by one OpenEM instance have to belong to either the first group or the second group. OpenEM can use any PDSP that is not used for another purpose and any 8 consecutive INTD channels that are not used for another purpose. Chapter 4.3 of the KeyStone Architecture Multicore Navigator User Guide (SPRUGR9E) is no longer up to date and needs to be discarded.
  - SR#1-55818181 - Issue with un-cached MSMC - OpenEM 1.9.0.0 - K2_PG1_MSMC_STALL_WA build flag is defined.

- For Keystone I devices
  - The OpenEM scheduler firmware has been tested on all PDSPs
  - The Open EM scheduler Firmware can be used in conjunction of the accumulator firmware.
    - When using the 32-channel version, the Open EM scheduler has to be executed on the PDSP #2 of the QMSS
      - Private events section has to be 8 Kbytes located on the QMSS_L2_SCRATCH memory (base address is 0x340BC000)
      - The hw_interrupt_base_idx field shall be set with a value higher or equal to 32.
    - When using the 16-channel version, the Open EM scheduler has to be executed on the PDSP #1 of the QMSS
      - Private events section has to be 8 Kbytes located on the QMSS_L2_SCRATCH memory (base address is 0x340B8000)
      - The hw_interrupt_base_idx field shall be set with a value lower or equal to 32 – TI_EM_CORE_NUM.
    - Using the 48-channel version is prohibited since the Open EM scheduler and this version of accumulator share the same interrupt lines
- Chaining and post-storing
  - The number of multiple TX channels for parallelizing the chaining and post-storing is set to 4.
  - In multi-threaded router-scheduler configuration, only the first router routes the events to the schedulers.

  - XGE router makes use of the first PKDMA TX channel only when sending event to OpenEM processes.
- Interrupts are disabled by default and thus must be explicitly enabled if needed.
- Following public functions are not supported by OpenEM implementation:
  - em_status_t em_queue_delete(em_queue_t queue)
  - em_status_t em_queue_enable(em_queue_t queue)
  - em_status_t em_queue_enable_all(em_eo_t eo)
  - em_status_t em_queue_disable(em_queue_t queue, int num_notif, const em_notif_t* notif_tbl)
  - em_status_t em_queue_disable_all(em_eo_t eo, int num_notif, const em_notif_t* notif_tbl)
  - em_status_t em_eo_delete(em_eo_t eo)
  - em_status_t em_eo_remove_queue(em_eo_t eo, em_queue_t queue, int num_notif, const em_notif_t* notif_tbl)
  - em_status_t em_eo_stop(em_eo_t eo, int num_notif, const em_notif_t* notif_tbl)
  - em_status_t em_queue_group_delete(em_queue_group_t group, int num_notif, const em_notif_t* notif_tbl);
  - em_status_t em_event_group_delete(em_event_group_t event_group);
  - em_status_t em_queue_group_modify(em_queue_group_t group, const em_core_mask_t* new_coremask, int num_notif, const em_notif_t* notif_tbl)
- No EO context supported.
- No EO local start/stop functions.
- No notifications (except for event group).
- No parallel ordered queues.
- Fixed minor/major event type (undefined).
- No 64-bit support.
- For the DSP part, the application shall define build symbols:
  - EM_32_BIT
  - TI_EM_C6670 or TI_EM_C6678 or TI_EM_C6634
  - EM_CHECK_LEVEL (0 - disabled or 1 - enabled)
  - TI_EM_TRACE_LEVEL (0 - disabled or 1 - enabled)
- For the Linux part, the application shall define build symbols:
  - EM_32_BIT
  - TI_EM_C6638
  - DEVICE_K2K
  - EM_CHECK_LEVEL (0 - disabled or 1..3 - enabled)
  - TI_EM_TRACE_LEVEL (0 - disabled or 1..3 - enabled)
  - _GNU_SOURCE=1

  - TI_EM_LINUX
  - TI_EM_GCC
  - TI_EM_ARM_A15
  - _LITTLE_ENDIAN
- Sending events to event queues that are located on a different device via 10 Gigabit Ethernet is not validated on TI OpenEM validation environment. TI verification includes XGE transfers from one device to the same device using a loopback cable.
- OpenEM library is not validated in big-endian mode.
- OpenEM user's guide is to be aligned.

Licensing

The component is released under the BSD License and GPL for the kernel module, please see the SW manifest document at the root of the installation directory for details.

Delivery Package

The OpenEM library is available in standalone installers for Windows and Linux for both KeyStone I and KeyStone II platforms:

- openem_1_10_0_0_k1_SetupWin32.exe , openem_1_10_0_0_k2_SetupWin32.exe: Windows installers for an OpenEM RTSC package which includes source code, pre-compiled libraries and examples.
- openem_1_10_0_0_k1_Linux-x86_Install.bin, openem_1_10_0_0_k2_Linux-x86_Install.bin: Linux installers for an OpenEM RTSC package which includes source code, pre-compiled libraries and examples.

Installation Instructions

The OpenEM library can be installed anywhere. On Windows, the proposed place is C:\ti. If you decide to place the OpenEM library elsewhere, you need to specify that path in the CCSv5.3 Tool Discovery path.

Directory structure

The following is the directory structure after the OpenEM library has been installed as a standalone package. When installed through the BIOS MCSDK package, “docs” and “packages” are kept identical.

Directory name	Description
openem_1_10_0_0/docs	The directory contains the OpenEM documentation including this release notes, the OpenEM SW manifest, the compilation guide, the OpenEM white paper, the user’s guide and the api reference guide.
openem_1_10_0_0/eclipse	The directory contains the Eclipse plugins for CCSv5.1 or later integration
openem_1_10_0_0/package	The directory contains OpenEM RTSC package information.
openem_1_10_0_0/packages/ti/runtime/openem	The top level directory contains the header files which are provided by the OpenEM and should be used by the application developers for library usage.
openem_1_10_0_0/packages/ti/runtime/openem/doc/doxygen/html	The directory contains the OpenEM documentation.
openem_1_10_0_0/packages/ti/runtime/openem/lib/C6670 openem_1_10_0_0/packages/ti/runtime/openem/lib/C6678 openem_1_10_0_0/packages/ti/runtime/openem/lib/C6634	The directory has pre-built Big and Little Endian libraries.
openem_1_10_0_0/packages/ti/runtime/openem/src	The directory contains the source code for the OpenEM main library.
openem_1_10_0_0/packages/ti/runtime/openem/modules	The directory contains the source code for the OpenEM modules (like chaining)
openem_1_10_0_0/packages/ti/runtime/openem/dsp/examples	The directory in the OpenEM contains DSP-based examples and associated platforms to compile.
openem_1_10_0_0/packages/ti/runtime/openem/linux	The directory contains the Linux port files.
openem_1_10_0_0/packages/ti/runtime/openem/linux/modules	The directory contains the Linux kernel modules needed for few kernel services.
openem_1_10_0_0/packages/ti/runtime/openem/linux/osal	The directory contains the OS Abstraction Layer library used to map OpenEM platform needs to Linux services (this library is needed to build OpenEM library on Linux).
openem_1_10_0_0/packages/ti/runtime/openem/linux/rh	The directory contains the OpenEM Runtime Helper library to simplify the OpenEM instance initialization on Linux (this library is optional and is not needed directly by the OpenEM library).
openem_1_10_0_0/packages/ti/runtime/openem/linux/keystone2	This directory contains platform specific header files for KeyStone II architecture.
openem_1_10_0_0/packages/ti/runtime/openem/linux/examples	This directory contains Linux examples and associated platform files to compile.

openem_1_10_0_0/packages/ti/runtime/openem/linux/tools	This directory contains utilities for debug purpose

OpenEM DSP build

See the compilation guide for details

OpenEM DSP usage

Please go to the OpenEM user's guide for usage details.

OpenEM DSP example projects

The directory ([your path]/openem/dsp/examples) contains examples. This can be used as a blue-print for the OpenEM user.

Example_0

- Keystone I and Keystone II
- The DSP Example_0 displays a template on how to use the OpenEM on a stand-alone DSP instance without any chaining to another instance.
- K2_PG1_MSMC_STALL_WA build flag shall be defined in the project when building with OpenEM 1.9.0.0 on K2 devices.

Example_2

- Keystone II
- The DSP Example_2 displays a template on how to use the OpenEM chaining with an OpenEM instance (or process) running on the same device, on the ARM side. It has to be run in parallel with the Linux Example_2.
    - The Linux side on ARM must have booted.
    - The Linux OpenEM process is responsible to start the OpenEM router.
    - The DSP OpenEM process waits that the Linux OpenEM process has finished its initialization.
    - The DSP OpenEM process does its own initialization and indicates to the Linux process when it is done.
    - At this point, both processes are configured and connected to the router.
    - Static HW queues are configured on both processes in order to exchange queue handle tables.
    - The Linux OpenEM process creates jobs that it sends to the DSP process for processing.

    - Once the DSP OpenEM process has processed and verified the content, it sends back to the Linux process. Each process “processes” the same number of jobs.
- A spreadsheet (openem_ressources.xlsx) explains how the resources are split over the processes.
- K2_PG1_MSMC_STALL_WA build flag shall be defined in the project when building with OpenEM 1.9.0.0 on K2 devices.

Example_3

- Keystone I and Keystone II
- The DSP Example_3 displays a template on how to use the OpenEM on a stand-alone DSP instance with a SRIO chaining.
    - The SRIO is configured to perform a loopback before the SERDES level
        - The data is pushed back to the digital domain before the SERDES macros.
    - The loopback enabling is highlighted in the code.
- K2_PG1_MSMC_STALL_WA build flag shall be defined in the project when building with OpenEM 1.9.0.0 on K2 devices.

OpenEM ARM build on Linux

Go into the [install_path]/openem_1_10_0_0/packages/ti/runtime/openem/linux directory.

You must have your GCC cross-compiler set in your current path (the cross compiler prefix may be overridden by using the CROSS_COMPILE environment variable).

The location of the Linux for KeyStone II root filesystem must be specified in the ROOT_DIR environment variable. We recommend to use Linux on KeyStone II with a Linux root filesystem mounted through NFS and to point to the exported NFS directory with ROOT_DIR.

Type to build OpenEM , OpenEM OSAL, OpenEM RH libraries and the provided examples:

```
$ make –f Makefile.c6638 clean all install
```

The OpenEM is built as a shared library (libopenem.so) and thus must be located in $ROOT_DIR/usr/lib at runtime. The example (Example_0) is installed in $ROOT_DIR/opt/openem/examples. OpenEM OSAL and RH libraries are static libraries and are not used at runtime. The built libraries for the Linux host can be found in the [install_path]/openem_1_10_0_0/packages/ti/runtime/openem/lib/c6638/arm-linux directory.

In order to use OpenEM on Linux, you must install a kernel module for a couple of services needed to access hardware resources from user space. For convenience this module is prebuilt for Linux 3.8.4 kernel (the one provided in MCSDK 3.00.00.11) and available in [install_path]/openem_1_10_0_0/ti/runtime/openem/linux/modules. You only have to copy it manually in the Linux root filesystem:

```
$ cp modules/em_mod.ko $ROOT_DIR/lib/modules/3.8.4/extra
```

This module can be rebuilt (for example if using PREEMPT_RT kernel). For that purpose you need to download the linux-keystone kernel in source form from Arago. In this case you have to set the KDIR environment to the Linux kernel build directory and type:

```
$ make –f Makefile.c6638 mod mod_install
```

OpenEM ARM usage

You must mount the Linux root filesystem on the EVM target (or generate an initrd or an ubifs image) with the standard MCSK Linux root filesystem and the OpenEM library, the OpenEM module and the example.

Then log as root on the KeyStone II EVM and install the kernel module:

```
# modprobe em_mod
```

You can now run the OpenEM example 0:

```
# /opt/openem/examples/Example_0
```

You should then get the following type of traces:

```
starting Example 0
EM[trace][C0] main(): running 4 cores
EM[trace][C0] my_global_init(): master core initialization
EM[trace][C0] my_em_init_global(): calling EM global initialization
EM[trace][C0] ti_em_rh_init_global(): set QMSS region init
EM[trace][C0] ti_em_rh_init_global(): initialize QMSS
EM[trace][C0] ti_em_rh_init_global(): load PDSP scheduler firmware
EM[trace][C0] ti_em_rh_init_global(): load PDSP router firmware
EM[trace][C0] ti_em_rh_init_global(): set EM configuration and initialize EM
EM[trace][C0] my_local_init(): local init
EM[trace][C0] ti_em_rh_init_local(): EM local configuration
EM[trace][C0] my_global_init(): objects creation done
EM[trace][C3] my_local_init(): local init
EM[trace][C3] ti_em_rh_init_local(): EM local configuration
EM[trace][C2] my_local_init(): local init
EM[trace][C2] ti_em_rh_init_local(): EM local configuration
EM[trace][C1] my_local_init(): local init
EM[trace][C1] ti_em_rh_init_local(): EM local configuration

==========
Run PASSED!

Statistics for core 0:
Number of jobs: 487
Number of jobs per flow: 9 14 17 16 18 15 16 14 16 16 16 14 16 16 16 14 16 14 17 14 16 14 18 14 16 14 16 15 16 14 16 14
Min/mean/max processing cycles: 2082/2556/11074
Min/mean/max overhead cycles: 934/1159/3431


Statistics for core 1:
Number of jobs: 180
Number of jobs per flow: 10 6 5 5 4 5 5 6 6 6 5 6 6 5 5 6 6 6 4 6 5 6 5 6 5 6 5 6 5 6 6 6
Min/mean/max processing cycles: 2228/2819/15315
Min/mean/max overhead cycles: 1131/7581/25423


Statistics for core 2:
Number of jobs: 176
Number of jobs per flow: 6 6 5 6 6 6 5 6 5 4 6 6 5 5 4 6 5 6 5 6 5 6 5 6 6 6 6 5 5 5 7 5 6
Min/mean/max processing cycles: 2154/2768/14567
Min/mean/max overhead cycles: 1256/7765/25832
```

```
Statistics for core 3:
Number of jobs: 181
Number of jobs per flow: 7 6 5 5 4 6 6 6 5 6 5 6 5 6 7 6 5 6 6 6 6 6 4 6 5 6 6 6 6 5 5 6
Min/mean/max processing cycles: 2168/2810/14173
Min/mean/max overhead cycles: 1142/7438/23201


Statistics for all cores
Total number of jobs: 1024
Total number of jobs per flow: 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32
32
Min/mean/max processing cycles: 2082/2684/15315
Min/mean/max overhead cycles: 934/4527/25832

Start/stop/run cycles: 96481/11932242/11835761

==========
EM[trace][C2] my_exit_local(): exiting...
```

OpenEM ARM example projects

The directory [install_path]/openem_1_10_0_0/packages/ti/runtime/openem/linux/examples contains four Linux examples. These can be used as a blue-print for the OpenEM user. The example configurations can be configured per platform with the files located in [install_path]/openem_1_10_0_0/packages/ti/runtime/openem/linux/examples/configs.

- Example_0 illustrates local send and receive of events (thus without using chaining). Even if the example is not using chaining, event payload (buffers) are allocated using CMA (contiguous physical memory). You can also use standard Linux virtual memory (as allocated with malloc()) by enabling the USE_LOCAL_PAYLOAD_BUFFERS flag in [install_path]/openem_1_10_0_0/packages/ti/runtime/openem/linux/examples/Example_0/src/my_em_init.c
- Example 1 illustrates sending and receiving of events using chaining over XGE mechanism. Note that the XGE chaining in this example is simulated and is in fact using Infrastructure PktDMA underneath to loopback into the same instance.
- Example 2 illustrates sending and receiving of events between one OpenEM ARM process and one DSP process. See DSP Example_2 for more details.
- Example 3 illustrates sending and receiving of events with interrupt support. Note that a set_interrupt_affinity_script.sh is provided in opt/openem on the target to bind the QMSS interrupts on the different ARM cores before running the example.

These examples can be used in a server/client model to illustrate the joining mechanism (except Example_3): you must launch one example first with the '–s' flag. This will create the OpenEM instance and its associated shared database and run the dispatcher on all cores. In another telnet session, you can the launch the same example binary but with the '-c' flags. This one will join the running instance as a client and will perform the event send.

You can also specify the number of cores of the instance with the '-n *core_num*' flag (it is 4 by default).

OpenEM Linux flags

Some new build flags have been added in Linux to change the behavior and performance of both the OpenEM library and examples. These flags must be specified to the compiler in the [install_path]/openem_1_10_0_0/packages/ti/runtime/openem/linux/Makefile.c6638:

- TI_EM_USE_MSM: if specified, the QMSS descriptors are allocated in the MSM memory which is more efficient than the DDR3. If not used, descriptors are located in DDR. Of course the OpenEM initialization must be set accordingly.
- TI_EM_THREAD_MODEL: if specified the OpenEM cores are mapped to POSIX threads, otherwise they are mapped to processes with their own address space.
- TI_EM_HAS___THREAD: if specified, it indicates that both compiler and ld.so support __*thread* qualifier (TLS) which provides better performances for the thread model case. It must generally be set when setting the TI_EM_THREAD_MODEL.
- TI_EM_USE_RTPRIO: if specified, the created OpenEM cores are put in Linux RT scheduling high priority on ARM cores. This option must be used carefully because it can cause deadlocks of the complete Linux system if OpenEM application is not well suitable for that purpose or not fully debugged.
- TI_EM_XGE_LOOPBACK: if specified, the XGE chaining support is emulated using infrastructure-PktDMA. This flag must be removed if using real 10 Gigabit Ethernet chaining.

The best performances can be obtained on ARM with the TI_EM_USE_MSM, TI_EM_USE_RTPRIO, EM_CHECK_LEVEL=0, TI_EM_TRACE_LEVEL=0 flags.