

## EDMA3 FAQ

### ABSTRACT

This document is a collection of frequently asked questions (FAQs) on enhanced direct memory access (EDMA) on KeyStone™ I (KS1) and KeyStone II (KS2) devices, along with useful collateral and software reference links.

### Contents

1	Where are software and hardware documentation resources located? .....	2
2	What is the functional description of the EDMA hardware blocks? .....	3
3	What are EDMA3 channel types and how does the EDMA3 transfer happen? .....	3
4	What is parameter RAM (PaRAM) and how is the PaRAM set organized? .....	3
5	What are EDMA3 channel controller regions and how are shadow region registers interpreted with each EDMA3 shadow region associated with its memory map? .....	4
6	What are the software building blocks: EDMA3 LLD, EDMA CSL, and StarterWare®? .....	5
7	Where is EDMA3 software (LLD, CSL, and StarterWare™) located in the releases? .....	6
8	Where are EDMA3 ARM® Linux® drivers for KeyStone™ devices located? .....	6
9	What is a 1D, 2D, and 3D EDMA3 transfer and how are the transfers configured? .....	7
10	What is the difference between DMA and QDMA channels? .....	7
11	What APIs can be used in EDMA3 LLD, EDMA3 CSL, and StarterWare™? .....	7
12	How do you configure EDMA from Linux™ using the Linux™ driver API? .....	8
13	What type of interrupts does the EDMA3 channel controller generate on transfer completion? .....	8
14	What type of events can initiate a DMA or QDMA transfer using EDMA3CC? .....	9
15	How does the DMAQNUM map a channel to a specific event queue in the EDMA3 channel controller? .....	9
16	Why does an EDMA3 completion interrupt arrive early (intermediate completion) and data is still moved to the destination later? .....	9
17	How do you ensure the DMA Transfer Completion Interrupt and the Interrupt Pending Register (IPR and IPRH) bit position is appropriate to the TCC value set upon transfer completion? .....	10
18	What is the difference between A-sync and AB-sync transfer modes in EDMA3? .....	10
19	How do you evaluate the status of any pending completion requests from EDMA3CC, destination FIFO status, and error status on EDMA3TC? .....	10
20	What is the difference between constant addressing mode (SAM/DAM=1) and increment mode (SAM/DAM=0)? Are there any limitations on buffer alignments between these two modes? .....	11
21	What are shadow registers and how are they used? .....	11
22	What is the EDMA3 linking feature and how do linking transfers happen? .....	12
23	What are channel chaining capabilities of EDMA3? .....	12
24	In what way do EDMA3 chained events differ from the EDMA3 linking feature when multiple events occur through receiving a single event? .....	12
25	Can EDMA3 be used to transfer data to or from DSP L2 or L1D memory in a multicore device? .....	12
26	Where are the EDMA3 LLD examples located in the LLD release? .....	13
27	What EDMA3 CSL examples are available in the Processor SDK RTOS release? .....	14
28	Where are the instructions on how to build and run the EDMA3 StarterWare™ examples that are available in the StarterWare™ releases? .....	14
29	Where are the instructions on how to build the Processor SDK and individual components from source that are available in the Processor SDK 2.0.1 release which includes Linux EDMA3 driver? .....	14
30	References .....	14

## Trademarks

KeyStone, StarterWare, Sitara are trademarks of Texas Instruments.

ARM is a registered trademark of ARM Limited.

Linux is a registered trademark of Linus Torvalds.

All other trademarks are the property of their respective owners.

## 1 Where are software and hardware documentation resources located?

TI provides key software components (processor SDK, MCSDK) for both Linux® and TI-RTOS platforms and documentation from the [Tools & Software](#) page.

Users can also download the development tools, technical documents, training, and videos for 66AK2x and AM57x platforms. For the complete list of software, refer to the following software sections.

- [66AK2Ex](#)
- [66AK2Hx](#)
- [66AK2Lx](#)
- [AM5718](#)
- [AM5728](#)

Descriptions of platform hardware, schematics, block diagrams, bill of materials (BOM), reference guides, Gerber files, and more are available at the following reference design links.

- [66AK2E](#)
- [66AK2G](#)
- [66AK2L](#)
- [AM5718](#)
- [AM5728](#)

The key software components (processor SDK for TI-RTOS platform, MCSDK for Linux and TI-RTOS platforms) are available for C667x and C665x KeyStone I platforms. To get tools and software for C66x Multi-Core DSPs, refer to [C66x](#).

DSP platform hardware, schematics, block diagrams, BOM, reference guides, Gerber files, and more are available at the following link locations.

- [C6678](#)
- [C6657](#)

## 2 What is the functional description of the EDMA hardware blocks?

The primary purpose of the EDMA3 controller is to service data transfers between two memory-mapped endpoints on a device. EDMA3 module consists of EDMA3 channel controllers (EDMA3CC) and EDMA3 transfer controllers (EDMA3TC).

The EDMA3 channel controller serves as the user interface for the EDMA3 controller. The EDMA3CC includes set of parameter RAM (PaRAM), channel control registers, and interrupt control registers. The EDMA3CC prioritizes incoming software requests or events from peripherals and submits Transfer Requests (TR) to the EDMA3 Transfer Controller.

The EDMA3 transfer controllers are responsible for data movement. A Transfer Request Packet (TRP) submitted by the EDMA3CC contains the transfer context. Based on the TRP, the transfer controller issues a read or write command that includes source and destination addresses and other transfer parameters.

For more details of the main blocks of EDMA3CC and EDMA3TC, including functional block diagrams, refer to the EDMA3 architecture section of the [KeyStone Architecture Enhanced Direct Memory Access \(EDMA3\) Controller](#) user's guide.

## 3 What are EDMA3 channel types and how does the EDMA3 transfer happen?

The EDMA3CC includes two channel types: DMA channels and QDMA channels.

Each channel is associated with a given event queue controller, transfer controller, and a given PaRAM.

A trigger event is necessary to initiate a transfer. For DMA channels, a trigger event may be an external event, a manual write to the event set register, or a chained event. QDMA channels are autotriggered when a write is performed to the user-programmed trigger word in the PaRAM.

Once a trigger event is recognized, the channel that is associated with the event is queued in the appropriate EDMA3CC event queue.

Each event in the event queue is processed in a FIFO order. On reaching the head of the queue, the PaRAM associated with that channel is read to determine the transfer details. The transfer controller (TC) submission logic evaluates the validity of the transfer request (TR) and is responsible for submitting a valid TR to the appropriate EDMA3TC.

The EDMA3TC receives the request and is responsible for data movement as specified in the transfer request packet (TRP). Other necessary tasks, like buffering and ensuring transfers, are carried out optimally wherever applicable.

The EDMA3TC can be configured to generate an event either when the transfer is complete or when the channel controller (CC) starts the transfer. The event that is generated can be used to generate interrupt to a CPU or for chaining (see [Section 23](#)). Additionally, the EDMA3CC has an error detection logic that causes an error-interrupt generation on various error conditions (for example, missed events, exceeding event queue thresholds, and more).

## 4 What is parameter RAM (PaRAM) and how is the PaRAM set organized?

The set of PaRAMs gives the user the ability to preconfigure multiple transfer parameters during initialization to minimize application execution time. Each PaRAM is a 32-byte (or 8 32-bit words) structure that contains transfer context (source and destination addresses, count, indexes, and more). To facilitate transfers, each PaRAM is associated with a DMA or QDMA channel. Before channel transfer starts, the PaRAM that is associated with the channel is loaded into the TC.

Details on EDMA3 channel parameter description are provided in the PaRAM set figure and the EDMA3 channel parameter description table of [\[1\]](#).

## 5 What are EDMA3 channel controller regions and how are shadow region registers interpreted with each EDMA3 shadow region associated with its memory map?

The EDMA3 channel controller (EDMA3CC) divides its address space into multiple regions. Individual channel resources are assigned to a specific region, where each region is used by a different part of the execution so it is easy to control the distribution of EDMA resources. The EDMA3CC memory-mapped registers are divided in three main categories:

- Global registers
- Global region channel registers
- Shadow region channel registers

The global registers are located at a fixed location in the EDMA3CC memory map. These registers control EDMA3 resource mapping and provide debug visibility and error tracking information.

The channel registers (including DMA, QDMA, and interrupt registers) are accessible via the global channel region address range, or in the shadow n channel (see [Section 21](#)) region address ranges.

Refer the shadow region registers table and figure of [\[1\]](#) for shadow region registers. The shadow region registers figure provides the offset of the shadow registers in the EDMA3 controller block. The base addresses of each EDMA controller can be found in the memory map section of the user's guide of the appropriate device.

## 6 What are the software building blocks: EDMA3 LLD, EDMA CSL, and StarterWare®?

TI provides multiple levels of application programming interface (APIs) to drive the EDMA. The following list provides a description of each API.

- StarterWare™ is a basic library to manipulate EDMA that does not require any operating system.
- CSL is part of the TI-RTOS operating system and provide low-level EDMA drivers.
- EDMA LLD is a set of APIs that abstract details of the implementation and make it easy for the user to use EDMA.

In addition, devices that run LINUX on ARM have a set of LINUX drivers that control the EDMA.

### 6.1 EDMA3 LLD

The EDMA3 low-level driver (LLD) is a set of APIs that supports programming the EDMA3 peripheral. EDMA3 LLD is a component of the processor SDK RTOS. EDMA3 LLD has the following two modules:

- The driver module (drv) — contains APIs for configuration and run-time operation of the EDMA3.
- The resource manager (rm) module — contains APIs to manage the usage of various EDMA3 resources across multi-processor devices.

The EDMA3 User's Guide and API description is part of the document directory (`\edma3_lld_2_12_01_24\packages\ti\sd\edma3\drv\docs` for the drv and `\edma3_lld_2_12_01_24\packages\ti\sd\edma3\rm\docs` for the rm). Example projects that demonstrate how to use the EDMA3 LLD are part of the examples directory (`\edma3_lld_2_12_01_24\examples`).

### 6.2 CSL

The Chip Support Library (CSL) is part of Programmer Development Kit (PDK). CSL contains sets of low-level drivers to manipulate hardware components of the device, including EDMA3. The file `csl_edma3Aux.h` in directory `\pdk_c667x_2_0_2\packages\ti\csl` (release version may vary) contains the APIs that can be used by an application as well as other include files `edma3.h` and `edma.h`. The source code for the CSL EDMA3 is in directory `\pdk_c667x_2_0_2\packages\ti\csl\src\ip\edma\V0` (see V1 as well, release version varies). Some of the CSL EDMA3 APIs are inline functions as part of the `csl_edma3Aux.h`.

### 6.3 StarterWare

StarterWare is a free software development package that provides no-OS platform support for ARM® and DSP TI processors. StarterWare includes Device Abstraction Layer (DAL) libraries, peripheral programming, and board level example applications that demonstrate the capabilities of the peripherals on the TI processors. StarterWare can be used stand-alone or with an RTOS.

---

**NOTE:** There is no StarterWare support available for KeyStone I and Keystone II devices. The StarterWare support is only available for single-core DSP devices and is intended to support multiple processors of the TI SoC family, such as AM1808, OMAPL138, C6748, and AM335x.

---

## 7 Where is EDMA3 software (LLD, CSL, and StarterWare™) located in the releases?

### 7.1 EDMA3 LLD

EDMA3 LLD is part of Processor SDK RTOS perspective. Processor SDK can be downloaded from the following device locations.

- [Processor SDK for AM335x](#)
- [Processor SDK for AM437x](#)
- [Processor SDK for AM57xx](#)
- [Processor SDK RTOS for C665x](#)
- [Processor SDK RTOS for C667x](#)
- [Processor SDK RTOS for K2E](#)
- [Processor SDK RTOS for K2G](#)
- [Processor SDK RTOS for K2H/K2K](#)
- [Processor SDK RTOS for K2L](#)

The standalone EDMA3 LLD can be downloaded from the [EDMA3 Low-level Driver Product Download Pages](#).

### 7.2 EDMA3 CSL

CSL is part of the PDK package that is included in the Processor SDK RTOS release. The included files are in the `$(TI_PDK_INSTALL_DIR)\packages\ti\csl` directory, where `$(TI_PDK_INSTALL_DIR)` is the PDK directory of the Processor SDK RTOS. Source files are in `$(TI_PDK_INSTALL_DIR)\packages\ti\csl\src\ip\edma\V0`. EDMA examples can be found in `$(TI_PDK_INSTALL_DIR)\packages\ti\csl\example`.

### 7.3 StarterWare™

StarterWare releases for the devices that are supported can be downloaded from the [StarterWare Wiki](#).

## 8 Where are EDMA3 ARM® Linux® drivers for KeyStone™ devices located?

Linux EDMA3 drivers are part of TI Linux release and Real-Time (RT) Linux release for the KeyStone family devices and some of the Sitara™ family devices. Go to the [Scalable Linux and TI-RTOS solutions for TI processors](#) page to find the latest version of Linux releases.

The EDMA3 drivers are part of the Linux Kernel. The EDMA3 driver APIs are defined in the `dma.h` include file. This include file and other `dma` include files are located in `/processorSDKLINUX_XX_XX_XX_XX/board-support/linux-4.4.12+gitAUTOINC+3639bea54a-g3639bea54a/arch/arm/include/asm`, where `processorSDKLINUX_XX_XX_XX_XX` is the directory where the LINUX release was installed.

The translation between logical and physical addresses for devices with EDMA3 controllers that do not have unified memory management unit (MMU) is done using memory protection and extension (MPAX) registers and not MMU.

## 9 What is a 1D, 2D, and 3D EDMA3 transfer and how are the transfers configured?

The EDMA3 device can move 1D, 2D, or 3D arrays.

- The 1D transfer moves a 1D vector. The transfer moves A count (ACNT) bytes from the source address to the destination address.
- The 2D transfer moves matrices. The channel moves ACNT bytes to the destination, then it skips destination B index (DSTBIDX) bytes and source B index (SRCBIDX) in the destination, and then moves ACNT bytes again. The process is repeated B count (BCNT) times. The 2D configuration can be on the source address, destination address, or both addresses.
- The 3D transfer moves multiple matrices. This transfer moves a matrix with the parameters from the 2D transfer, then it skips destination C index (DSTCINX), source C index (SRCDINX), or both indexes and moves the next matrix. This process repeats itself C count (CCNT) times.

A collection of EDMA3 transfer example configurations are included in the Processor SDK release and can be found in `\edma3_llid_2_12_01_24\examples` (version number may vary).

## 10 What is the difference between DMA and QDMA channels?

QDMA channel and DMA channel transfers are configured by the PaRAMs. The difference between DMA and QDMA channels is the event and channel synchronization.

- QDMA events are autotriggered or link triggered. Autotriggering allows QDMA channels to be triggered by a CPU with a single write to the PaRAM. Link triggering allows a linked list of transfers to be executed using a single QDMA PaRAM set and multiple linked PaRAM sets.
- DMA channels are triggered by an event. Events can be generated by an internal component like a CPU, an EDMA channel linking another component, or by an external device by using an interface such as Serial Rapid I/O (SRIO) or Peripheral Component Interconnect Express (PCIe).

## 11 What APIs can be used in EDMA3 LLD, EDMA3 CSL, and StarterWare™?

### 11.1 EDMA3 LLD

The *EDMA3 Resource Manager User Guide* is in the `dma3_llid_X_XX_XX_XX\packages\ti\sdo\edma3\rm\docs` directory, where `dma3_llid_X_XX_XX_XX` is the directory where the EDMA3 was installed. The same directory has a CHM system of description of all APIs. See [9] for a free .chm file viewer and instructions on how to use the free download.

The *EDMA3 Driver APIs User Guide* is in the `dma3_llid_X_XX_XX_XX\packages\ti\sdo\edma3\drv\docs` directory, where `dma3_llid_X_XX_XX_XX` is the directory where the EDMA3 was installed. The same directory has a CHM system of description of all APIs.

### 11.2 EDMA3 CSL

A complete list of CSL EDMA3 API functions can be found in the `pdk_YYY_X_X_X\packages\ti\cs\docs\doxygen\html\group___c_s_l___e_d_m_a3___f_u_n_c_t_i_o_n.html` directory, where `pdk_YYY_X_X_X` is the directory where the PDK was installed.

### 11.3 StarterWare™

StarterWare documents, including a CHM system of description of all StarterWare APIs, can be found in the `\pdk_YY_X_X_X\packages\ti\starterware\docs` directory, where `pdk_YY_X_X_X` is the directory where the PDK was installed. StarterWare is only available for a subset of Sitara devices.

The EDMA StarterWare include file `edma.h` is in the `\pdk_YY_X_X_X\packages\ti\starterware\include` directory, and the source code file `edma.c` is in the `\pdk_YY_X_X_X\packages\ti\starterware\dal` directory.



## 12 How do you configure EDMA from Linux™ using the Linux™ driver API?

EDMA Linux files for KeyStone devices are available from the following git locations:

- <https://git.ti.com/keystone-linux/uio-module-driv>
- <https://git.ti.com/keystone-linux/uio-module-driv/trees/master>

An example for the device tree that includes the EDMA definition is available in the Linux release /processorSDKLINUX\_XX\_XX\_XX\_XX/board-support/linux-4.4.12+gitAUTOINC+3639bea54a-g3639bea54a/arch/arm/dts/keystone.dtsi, where processorSDKLINUX\_XX\_XX\_XX\_XX is the directory where the Linux release was installed. The following code was taken from the device tree.

```
udma0 {
compatible = "ti,keystone-udma";
};

uio_edma3: edma3 {
compatible = "ti,uio-module-driv";
mem = <0x02700000 0x000C0000>;
label = "edma3";
};
```

## 13 What type of interrupts does the EDMA3 channel controller generate on transfer completion?

There are two types of completion events that the EDMA3 generates: intermediate transfer completion and complete interrupt.

The EDMA3 channel controller block diagram of [1] shows that a completion event can come from the transfer request process module and from one of the transfer controllers.

The first completion event is sent when a request is sent to the transfer controller. This event is called an intermediate transfer completion and it is enabled by bit 21 of the Channel Options Parameter (OPT) word of the PaRAM (ITCINTEN) for interrupt and by bit 23 (ITCCHEN) for chaining. The complete interrupt comes from one of the transfer controllers and it is set when the transfer is complete.

The complete interrupt is enabled by bit 20 (TCINTE) and bit 22 (TCCHE) of the OPT word of PaRAM for generating interrupt and for chaining respectively. When either events are enabled, the appropriate bit of the interrupt pending register and interrupt pending register high (IPR and IPRH) of the appropriate shadow register or the global register set registers is latched. The location of the bit that is set is defined by the TCC bits (bits 12-17) of the OPT word for the associated PaRAM. Transfer completion interrupts that are latched to the IPR and IPRH are cleared by writing a 1 to the corresponding bit in the interrupt clear register (ICR and ICRH).



## 14 What type of events can initiate a DMA or QDMA transfer using EDMA3CC?

There are multiple ways to initiate a programmed data transfer using the EDMA3 channel controller. The following three sources can initiate a DMA event.

- **Event-Triggered Transfer Request:** When an event is asserted from a peripheral or device pins, it gets latched in the corresponding bit of the event register (ER.En = 1). If the corresponding event in the event enable register (EER) is enabled (EER.En = 1), then the EDMA3CC prioritizes and queues the event in the appropriate event queue. When the event reaches the head of the queue, it is evaluated for submission as a transfer request to the transfer controller.
- **Manually Triggered Transfer Request:** A manually-triggered DMA event is initiated by a write to the event set register (ESR) by the DSP/EDMA3. Writing a 1 to an event bit in the ESR results in the event being prioritized and queued in the appropriate event queue, regardless of the state of the EER.En bit. When the event reaches the head of the queue, it is evaluated for submission as a transfer request to the transfer controller.
- **Chain-Triggered Transfer Request:** A chain-triggered transfer event is triggered when the completion of one transfer automatically sets the event for another channel. When a chained completion code is detected, the value of which is dictated by the transfer completion code (TCC[5:0] in OPT of the PaRAM set associated with the channel), it results in the corresponding bit in the chained event register (CER.E[TCC] = 1).

The following two sources can initiate a QDMA event.

- **Auto-Triggered Transfer Request:** A bit corresponding to a QDMA channel is set in the QDMA event register (QER) when a DSP or EDMA3 write occurs to a PaRAM address that is defined as a QDMA channel trigger word which is programmed in the QDMA channel mapping register (QCHMAP n) for the particular QDMA channel and the same channel is enabled through the QDMA Event Enable Register (QEER.En = 1).
- **Link-Triggered Transfer Request:** A bit corresponding to a QDMA channel is set in the QER when EDMA3CC performs a link update on a PaRAM set address that is configured as a QDMA channel matches QCHMAPn settings and the corresponding channel is enabled through the QEER (QEER.En = 1).

## 15 How does the DMAQNUM map a channel to a specific event queue in the EDMA3 channel controller?

The DMA Channel Queue Number Register (DMAQNUMn) enables mapping of DMA channels to the queues in the EDMACC. Each queue is hardcoded to a transfer control; mapping of a channel to a queue N (N = 0, 1, 2, or 3) means mapping the channel to transfer controller N.

Each channel has three bits in the eight DMAQNUMn registers that contains the queue number (see the DMA channel queue n and bits in DMAQNUM n tables of [1]). For example, bits 12 through 14 in DMAQNUM number 2 sets the queue for channel 19.

When a particular DMA channel event N (En) and its corresponding bit fields in the DMAQNUM register holds the value of 0, the events for the channel are queued on DMA event queue number 0. Similarly, if it holds the value of 1, the corresponding events are queued on queue number 1, and so on.

## 16 Why does an EDMA3 completion interrupt arrive early (intermediate completion) and data is still moved to the destination later?

Transfer requests (TR) from EDMA CC are queued in the transfer controller (TC) and are serviced based on a priority mechanism. Early completion events can be generated when the TC starts the data move process. See the TCCMODE bit field of OPT and the channel options parameter table in [1]. The actual transfer continues after the early completion, or intermediate event was generated. A completion event will be generated when all of the data moves are finished.

**17 How do you ensure the DMA Transfer Completion Interrupt and the Interrupt Pending Register (IPR and IPRH) bit position is appropriate to the TCC value set upon transfer completion?**

The OPT is the first 32-bit word of the PaRAM. PaRAM bit 21 (Intermediate Transfer Completion Interrupt Enable [ITCINTEN]) and bit 20 (Transfer Completion Interrupt Enable [TCINTEN]) control the interrupt generation upon queuing of the transfer (ITCINTEN) or the actual completion of the transfer (TCINTEN). If one of these bits is set, an event is generated by the TC and is latched by a bit in the IPR and IPRH. The IPR and IPRH set of two 32-bit registers supports up to 64 channels. The bit location of the channel interrupt in the IPR and IPRH register is determined by the value of the TCC (Transfer Complete Code). The TCC is in bits 12-17 of the OPT register (6 bits). Thus, it can address up to 64 locations.

DMA3 controllers may use more than 64 channels (64 DMA channels and 8 QDMA channels). In that case, shadow registers must be used to accumulate more than 64 channels on IPR and IPRH set of registers.

**18 What is the difference between A-sync and AB-sync transfer modes in EDMA3?**

A-Sync and AB-Sync are used for 2D transfers and 3D transfers. There are two main differences between A-sync and AB-sync.

The first difference is the trigger requirement. In A-sync mode, each transfer of 1D row of data should be triggered separately. In AB-sync mode, one trigger starts the transfer of all 1D rows in the 2D matrix.

The second difference has to do with the skip index of the source and the destination of the B dimension (SRCBIDX and DSTBIDX). In a-sync mode, the skip index is calculated from the last byte that was transferred in the previous 1D row. In AB-sync mode, the skip index is calculated from the start of the previous 1D row transfer.

See the A-synchronized and AB-synchronized transfer sections of [1] for more information about A-sync and AB-sync.

**19 How do you evaluate the status of any pending completion requests from EDMA3CC, destination FIFO status, and error status on EDMA3TC?**

The EDMA3CC status register (CCSTAT) provides information on the number of completion requests submitted to the transmit control (COMPACTV bits) and the activity of each one of the transfer queues (bits QUEACTV0, QUEACTV1, QUEACTV2, QUEACTV3). Other indicators in the same register are the channel control active (ACTV, bit 4), write status active (WSTATV, bit 3), transfer request active (TRACTV, bit 2), QDMA event active (QEVTACTV, bit 1), and DMA event active (EVTACTV, bit 0). See the EDMA3CC status register table in [1].

The EDMA3TC channel status register (TCSTAT) provides information on the destination FIFO (bits 4-6), write status active (bit 2), source active state (bit 1), and program status (bit 0). See the EDMA3TC channel status register table in [1].

Error indication is recorded in the error register (ERRSTAT). See error register and error enable register sections of [1].

## 20 What is the difference between constant addressing mode (SAM/DAM=1) and increment mode (SAM/DAM=0)? Are there any limitations on buffer alignments between these two modes?

The constant source addressing mode (SAM) or the constant destination addressing mode (DAM) are used when the DMA moves data to or from address that does not change. A typical case is hardware FIFO where the read and the write are always to and from the same locations. Other examples include peripherals where the ingress or egress data is written into a fixed memory mapped register.

The SAM (bit 0 in the OPT word of the PaRAM) and DAM (bit 1 in the OPT word of the PaRAM) control the constant address mode for the source address and the destination address respectively. If the SAM or DAM bit is set to 1, the corresponding source or destination address stays constant throughout the transfer.

Note that the number of bytes that are read or written for SAM/DAM=1 is defined in the FWID field (FIFO WIDTH, bit 10-8 in the OPT word of the PaRAM). Constant addressing mode is detailed in the constant addressing mode transfers section of [1], which includes the following note.

---

**NOTE:** The constant addressing (CONST) mode has limited applicability. The EDMA3 must be configured for the CONST mode (SAM/DAM=1) only if the transfer source or destination (on-chip memory, off-chip memory controllers, slave peripherals) support the CONST mode. See the device-specific data manual and peripheral user's guide to verify if constant addressing mode is supported. If constant addressing mode is not supported, the similar logical transfer can be achieved using increment (INCR) mode (SAM/DAM=0) by appropriately programming the count and indices values.

---

## 21 What are shadow registers and how are they used?

System on Chip (SoC) devices that have multiple processing cores need more control of EDMA execution to implement resource managing.

The EDMA3 controller has eight sets of shadow registers. A set of shadow registers contains a set of almost all EDMA3 control registers. Each channel is associated with a set of shadow registers. When an EDMA3 channel transfers data, the registers that control the transfer are the registers in the shadow register set that is associated with the channel.

The shadow region registers table of [1] describes the registers in a shadow register set. Registers DMA region access enable (DRAE) and QDMA region access enable (QRAE) define which channels are associated with each shadow register. Shadow registers are also called region registers; the name region is used because shadow registers determine what memory regions can be accessed by the channel that is associated with a specific shadow register.

## 22 What is the EDMA3 linking feature and how do linking transfers happen?

Linking is the feature of loading a new PaRAM to a channel upon completion of the channel current transfer. Upon completion of a transfer, the transfer parameters are reloaded with new PaRAM parameter set addressed by the 16-bit link address field of the current parameter set.

Linking only occurs when the STATIC bit (bit 3 of OPT word of the PaRAM) is cleared to 0. That is, set is not static. Note that even though a new PaRAM is loaded into the channel, no new transfer starts until a there is a new trigger to the channel.

## 23 What are channel chaining capabilities of EDMA3?

The channel chaining capability for the EDMA3 allows the completion of an EDMA3 channel transfer to trigger another EDMA3 channel transfer. After one EDMA channel completes, the second channel starts without CPU intervention. This feature enables the transfer of complex patterns with minimal intervention of CPU. The chaining EDMA3 channels section of [1] details chaining.

## 24 In what way do EDMA3 chained events differ from the EDMA3 linking feature when multiple events occur through receiving a single event?

Linking works on PaRAM:

- Linking loads a new PaRAM to a channel after the transfer.
- Linking does not trigger the channel.
- Typical usage of linking is for ping-pong buffers, data buffers, or any fixed-structure multiple buffers.

Chaining works on channels:

- Chaining triggers a new channel when the current channel transfer completes
- Typical usage of chaining is for complex data transfers that do not require CPU intervention

It is possible to have linking and chaining for the same transfer, even to the same channel; a new PaRAM is loaded and the channel triggers itself. The linking transfers section of [1] details linking and the chaining EDMA3 channels details chaining.

## 25 Can EDMA3 be used to transfer data to or from DSP L2 or L1D memory in a multicore device?

Yes, EDMA can transfer data from any memory that has a global address to any memory that has a global address.

L1D and L2 for each DSP core have two addresses: a local address and a global address. The local address of L2 (for example) is the same for all of the cores, but the global address for each core is different. The EDMA cannot use the local address and must use the global address; it will not know which memory is the destination if the local address is used (for example, if the local address of L2 is used, the EDMA will not know if it is L2 of core 0 or L2 of core 1).

The memory-map of any device (see device user's guide) shows the global address of L1D or L2 for each one of the DSP cores in the device. To transfer data to and from L2 or L1D of a DSP core, the user should configure the EDMA with the appropriate global memory.



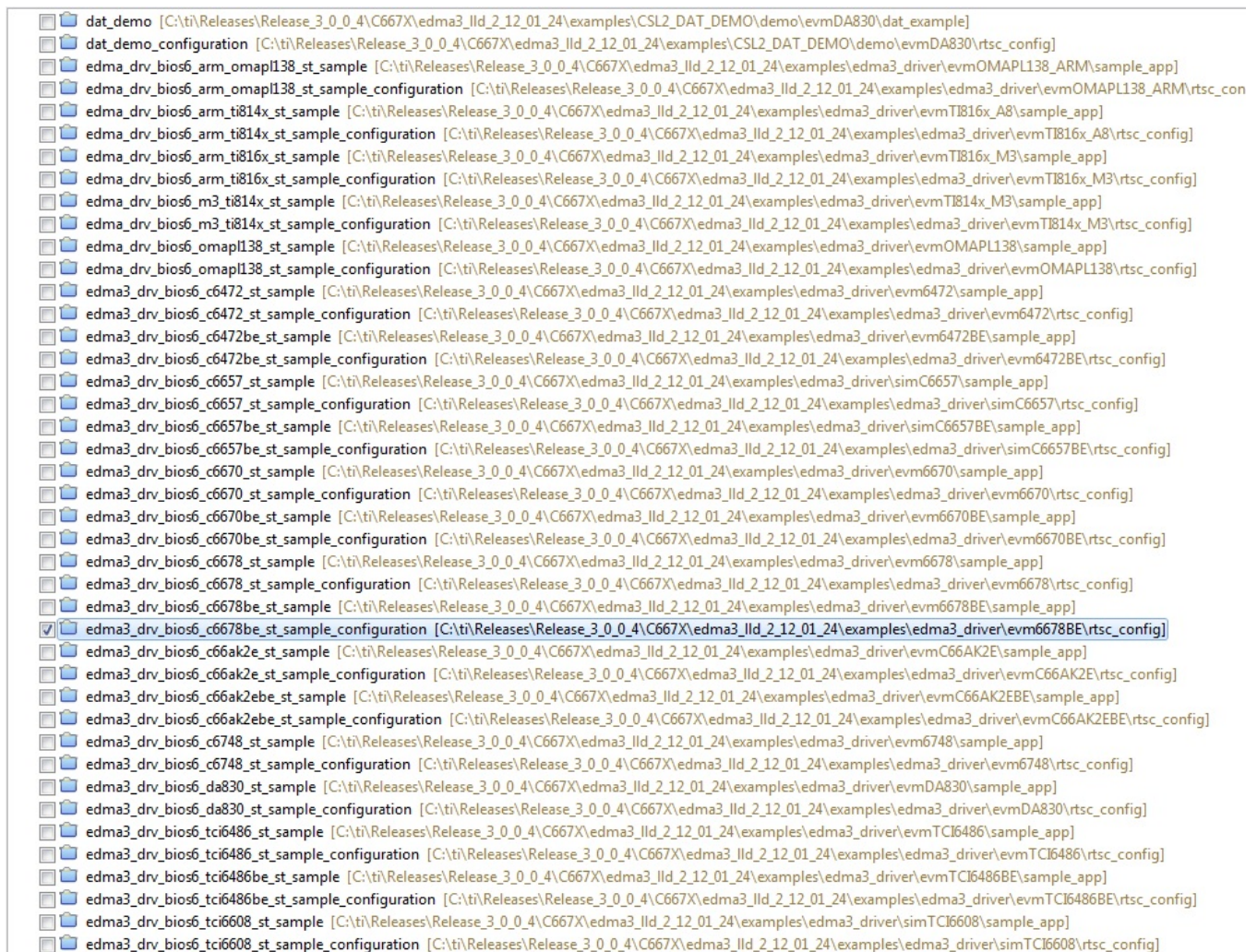
## 26 Where are the EDMA3 LLD examples located in the LLD release?

The Processor SDK RTOS release directory `edma3_lld_X_XX_XX_XX` has many example projects. Use the following instructions to access the EDMA3 LLD examples.

1. Click on the Project tab in the CCS window edit perspective.
2. Select the Import CCS Projects option.  
A dialogue box will open.
3. Navigate to the directory location of the EDMA3 in the Processor SDK RTOS release.

The window contains about 70 projects (depending on the device release). [Figure 1](#) shows the examples of EDMA3 for C66x Release 3.0.0.4 of Processor SDK RTOS:

**Figure 1. EDMA3 Example Projects**



4. Select one or more projects and click Finish.
5. Build the selected projects and follow the instructions in [\[2\]](#).

**27 What EDMA3 CSL examples are available in the Processor SDK RTOS release?**

EDMA3 CSL is part of the PDK module in the Processor SDK RTOS release. A test source code (and a make file in some of the devices) is in the `pdk_YY_X_XX_XX\packages\ti\cs\example\edma\edma_polled_mode_test` directory, where `pdk_YY_X_XX_XX` is the PDK module directory in the installed release. Using CCS to build and run the CSL EDMA3 test requires starting a new CCS project and manually building the project.

**28 Where are the instructions on how to build and run the EDMA3 StarterWare™ examples that are available in the StarterWare™ releases?**

A detailed procedure is provided in [3] for building the StarterWare libraries or applications on both Windows and Linux devices. In order to run EDMA example applications with UART, I<sup>2</sup>C, and SPI peripherals in StarterWare releases, follow the steps mentioned in [4], as well the individual Peripheral User Guide Wikis (with DMA mode of operation) shown in the following list.

- [StarterWare UART](#)
- [StarterWare I2C](#)
- [StarterWare SPI](#)

**29 Where are the instructions on how to build the Processor SDK and individual components from source that are available in the Processor SDK 2.0.1 release which includes Linux EDMA3 driver?**

Currently, the Processor SDK Linux perspective is available only for a Linux 64-bit computer. Instructions on how to download and install Linux releases for a variety of devices are provided in [5]. Instructions on how to set up the Linux build environment are in [6].

Instructions for building all modules and peripheral code are in [7]. Instructions on how to build the SDK and its examples can be found in [8].

**30 References**

1. [KeyStone Architecture Enhanced Direct Memory Access \(EDMA3\) Controller](#), (SPRUGS5)
2. [Processor SDK RTOS Software Developer Guide](#)
3. [StarterWare Getting Started 01.10.XX.XX](#)
4. [StarterWare 01.10.01.01 User Guide](#)
5. [Processor SDK Linux Installer](#)
6. [Processor SDK Linux Getting Started Guide](#)
7. [Processor SDK Linux Software Developer's Guide](#)
8. [Processor SDK Building The SDK](#)
9. [Compiled HTML Help](#)

## IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, enhancements, improvements and other changes to its semiconductor products and services per JESD46, latest issue, and to discontinue any product or service per JESD48, latest issue. Buyers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All semiconductor products (also referred to herein as "components") are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its components to the specifications applicable at the time of sale, in accordance with the warranty in TI's terms and conditions of sale of semiconductor products. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by applicable law, testing of all parameters of each component is not necessarily performed.

TI assumes no liability for applications assistance or the design of Buyers' products. Buyers are responsible for their products and applications using TI components. To minimize the risks associated with Buyers' products and applications, Buyers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right relating to any combination, machine, or process in which TI components or services are used. Information published by TI regarding third-party products or services does not constitute a license to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of significant portions of TI information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. TI is not responsible or liable for such altered documentation. Information of third parties may be subject to additional restrictions.

Resale of TI components or services with statements different from or beyond the parameters stated by TI for that component or service voids all express and any implied warranties for the associated TI component or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

Buyer acknowledges and agrees that it is solely responsible for compliance with all legal, regulatory and safety-related requirements concerning its products, and any use of TI components in its applications, notwithstanding any applications-related information or support that may be provided by TI. Buyer represents and agrees that it has all the necessary expertise to create and implement safeguards which anticipate dangerous consequences of failures, monitor failures and their consequences, lessen the likelihood of failures that might cause harm and take appropriate remedial actions. Buyer will fully indemnify TI and its representatives against any damages arising out of the use of any TI components in safety-critical applications.

In some cases, TI components may be promoted specifically to facilitate safety-related applications. With such components, TI's goal is to help enable customers to design and create their own end-product solutions that meet applicable functional safety standards and requirements. Nonetheless, such components are subject to these terms.

No TI components are authorized for use in FDA Class III (or similar life-critical medical equipment) unless authorized officers of the parties have executed a special agreement specifically governing such use.

Only those TI components which TI has specifically designated as military grade or "enhanced plastic" are designed and intended for use in military/aerospace applications or environments. Buyer acknowledges and agrees that any military or aerospace use of TI components which have **not** been so designated is solely at the Buyer's risk, and that Buyer is solely responsible for compliance with all legal and regulatory requirements in connection with such use.

TI has specifically designated certain components as meeting ISO/TS16949 requirements, mainly for automotive use. In any case of use of non-designated products, TI will not be responsible for any failure to meet ISO/TS16949.

### Products

Audio	<a href="http://www.ti.com/audio">www.ti.com/audio</a>
Amplifiers	<a href="http://amplifier.ti.com">amplifier.ti.com</a>
Data Converters	<a href="http://dataconverter.ti.com">dataconverter.ti.com</a>
DLP® Products	<a href="http://www.dlp.com">www.dlp.com</a>
DSP	<a href="http://dsp.ti.com">dsp.ti.com</a>
Clocks and Timers	<a href="http://www.ti.com/clocks">www.ti.com/clocks</a>
Interface	<a href="http://interface.ti.com">interface.ti.com</a>
Logic	<a href="http://logic.ti.com">logic.ti.com</a>
Power Mgmt	<a href="http://power.ti.com">power.ti.com</a>
Microcontrollers	<a href="http://microcontroller.ti.com">microcontroller.ti.com</a>
RFID	<a href="http://www.ti-rfid.com">www.ti-rfid.com</a>
OMAP Applications Processors	<a href="http://www.ti.com/omap">www.ti.com/omap</a>
Wireless Connectivity	<a href="http://www.ti.com/wirelessconnectivity">www.ti.com/wirelessconnectivity</a>

### Applications

Automotive and Transportation	<a href="http://www.ti.com/automotive">www.ti.com/automotive</a>
Communications and Telecom	<a href="http://www.ti.com/communications">www.ti.com/communications</a>
Computers and Peripherals	<a href="http://www.ti.com/computers">www.ti.com/computers</a>
Consumer Electronics	<a href="http://www.ti.com/consumer-apps">www.ti.com/consumer-apps</a>
Energy and Lighting	<a href="http://www.ti.com/energy">www.ti.com/energy</a>
Industrial	<a href="http://www.ti.com/industrial">www.ti.com/industrial</a>
Medical	<a href="http://www.ti.com/medical">www.ti.com/medical</a>
Security	<a href="http://www.ti.com/security">www.ti.com/security</a>
Space, Avionics and Defense	<a href="http://www.ti.com/space-avionics-defense">www.ti.com/space-avionics-defense</a>
Video and Imaging	<a href="http://www.ti.com/video">www.ti.com/video</a>

### TI E2E Community

[e2e.ti.com](http://e2e.ti.com)