# TMS320DM643x DMP
# Enhanced Direct Memory Access (EDMA3) Controller

# User's Guide


Literature Number: SPRU987A
March 2008

# Contents

# List of Figures

# List of Tables

# Read This First

## About This Manual

Describes the operation of the enhanced direct memory access (EDMA3) controller in the TMS320DM643x Digital Media Processor (DMP).

## Notational Conventions

This document uses the following conventions.

- Hexadecimal numbers are shown with the suffix h. For example, the following number is 40 hexadecimal (decimal 64): 40h.
- Registers in this document are shown in figures and described in tables.
  - Each register figure shows a rectangle divided into fields that represent the fields of the register. Each field is labeled with its bit name, its beginning and ending bit numbers above, and its read/write properties below. A legend explains the notation used for the properties.
  - Reserved bits in a register figure designate a bit that is used for future device expansion.

## Related Documentation From Texas Instruments

The following documents describe the TMS320DM643x Digital Media Processor (DMP). Copies of these documents are available on the Internet at http://www.ti.com. *Tip:* Enter the literature number in the search box provided at www.ti.com.

The current documentation that describes the DM643x DMP, related peripherals, and other technical collateral, is available in the C6000 DSP product folder at: http://www.ti.com/c6000.

**SPRU978 — *TMS320DM643x DMP DSP Subsystem Reference Guide.*** Describes the digital signal processor (DSP) subsystem in the TMS320DM643x Digital Media Processor (DMP).

**SPRU983 — *TMS320DM643x DMP Peripherals Overview Reference Guide.*** Provides an overview and briefly describes the peripherals available on the TMS320DM643x Digital Media Processor (DMP).

**SPRAA84 — *TMS320C64x to TMS320C64x+ CPU Migration Guide.*** Describes migrating from the Texas Instruments TMS320C64x digital signal processor (DSP) to the TMS320C64x+ DSP. The objective of this document is to indicate differences between the two cores. Functionality in the devices that is identical is not included.

**SPRU732 — *TMS320C64x/C64x+ DSP CPU and Instruction Set Reference Guide.*** Describes the CPU architecture, pipeline, instruction set, and interrupts for the TMS320C64x and TMS320C64x+ digital signal processors (DSPs) of the TMS320C6000 DSP family. The C64x/C64x+ DSP generation comprises fixed-point devices in the C6000 DSP platform. The C64x+ DSP is an enhancement of the C64x DSP with added functionality and an expanded instruction set.

**SPRU871 — *TMS320C64x+ DSP Megamodule Reference Guide.*** Describes the TMS320C64x+ digital signal processor (DSP) megamodule. Included is a discussion on the internal direct memory access (IDMA) controller, the interrupt controller, the power-down controller, memory protection, bandwidth management, and the memory and cache.

# Introduction

This document describes the features and operations of the enhanced direct memory access (EDMA3) controller in the TMS320DM643x Digital Media Processor (DMP).

The EDMA3 is a high-performance, multichannel, multithreaded DMA controller that allows you to program a wide variety of transfer geometries and transfer sequences.

Chapter 1 provides a brief overview, features, and terminology. Chapter 2 provides the architecture details and common operations of the EDMA3 channel controller (EDMA3CC) and the EDMA3 transfer controller (EDMA3TC). Chapter 3 contains examples and common usage scenarios. Chapter 4 describes the memory-mapped registers associated with the EDMA3 controller.

**Topic** **Page**

## 1.1 Overview

The enhanced direct memory access (EDMA3) controller's primary purpose is to service user-programmed data transfers between two memory-mapped slave endpoints on the device. Typical usage includes, but is not limited to:

- Servicing software driven paging transfers (for example, from external memory such as DDR2 to internal device memory such as DSP L2 SRAM
- Servicing event driven peripherals, such as a serial port
- Performing sorting or subframe extraction of various data structures
- Offloading data transfers from the main device CPU(s) or DSP(s) (See the device data manual for specific peripherals that are accessible via EDMA3. See the section on SCR connectivity in the device data manual for EDMA3 connectivity.)

The EDMA3 has a different architecture from the previous EDMA2 controller on the TMS320C621x/C671x DSPs and TMS320C64x DSPs.

The EDMA3 controller consists of two principal blocks:

- EDMA3 channel controller (EDMA3CC)
- EDMA3 transfer controller(s) (EDMA3TC)

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

The EDMA3 transfer controllers are slaves to the EDMA3 channel controller responsible for data movement. The transfer controller issues read/write commands to the source and destination addresses programmed for a given transfer. The operation is transparent to you.

## 1.2 Features

The EDMA3 channel controller (EDMA3CC) has following features:

- Fully orthogonal transfer description
  - 3 transfer dimensions
  - A-synchronized transfers: 1 dimension serviced per event
  - AB-synchronized transfers: 2 dimensions serviced per event
  - Independent indexes on source and destination
  - Chaining feature allows 3-D transfer based on single event
- Flexible transfer definition
  - Increment or constant addressing modes
  - Linking mechanism allows automatic PaRAM set update
  - Chaining allows multiple transfers to execute with one event
- Interrupt generation for:
  - Transfer completion
  - Error conditions
- Debug visibility
  - Queue watermarking/threshold
  - Error and status recording to facilitate debug
- 64 DMA channels
  - Event synchronization
  - Manual synchronization (CPU(s) write to event set register)
  - Chain synchronization (completion of one transfer triggers another transfer)

- 8 QDMA channels
  - QDMA channels are triggered automatically upon writing to a parameter RAM (PaRAM) set entry
  - Support for programmable QDMA channel to PaRAM mapping
- 128 parameter RAM (PaRAM) sets. Each PaRAM set can be used for a DMA channel, QDMA channel, or link set (remaining).
- 3 transfer controllers/event queues. The system-level priority of these queues is user programmable. (See the device data manual for the possible system priorities.)
- 16 event entries per event queue

The EDMA3 transfer controller (EDMA3TC) has following features:

- 3 transfer controllers (TC)
- 64-bit wide read and write ports per transfer controller (TC)
- Up to 4 in-flight transfer requests (TR)
- Programmable priority level
- Supports 2-dimensional transfers with independent indexes on source and destination (EDMA3CC manages the 3rd dimension)
- Support for increment or constant addressing mode transfers
- Interrupt and error support
- Little-endian or big-endian operations
- Memory-mapped register (MMR) bit fields are fixed position in 32-bit MMR, regardless of endianness

## 1.3 Terminology Used in This Document

The following is a brief explanation of some terms used in this document.

| Term | Meaning |
|---|---|
| A-synchronized transfer | A transfer type where 1 dimension is serviced per synchronization event. |
| AB-synchronized transfer | A transfer type where 2 dimensions are serviced per synchronization event. |
| Chaining | A trigger mechanism in which a transfer can be initiated at the completion of another transfer or subtransfer. |
| CPU(s) | The main processing engine or engines on a device. Typically a DSP or general-purpose processor. (See the device data manual to learn more about the CPU on your system.) |
| Device | TMS320DM643x DMP |
| DMA channel | One of the 64 channels that can be triggered by external, manual, and chained events. All DMA channels exist in the EDMA3CC. |
| Dummy set or Dummy PaRAM set | A PaRAM set for which at least one of the count fields is equal to 0 and at least one of the count fields is nonzero. A null PaRAM set has all the count set fields cleared. |
| Dummy transfer | A dummy set results in the EDMA3CC performing a dummy transfer. This is not an error condition. A null set results in an error condition. |
| EDMA3 channel controller (EDMA3CC) | The user-programmable portion of the EDMA3. The EDMA3CC contains the parameter RAM (PaRAM) , event processing logic, DMA/QDMA channels, event queues, etc. The EDMA3CC services events (external, manual, chained, QDMA) and is responsible for submitting transfer requests to the transfer controllers (EDMA3TC), which perform the actual transfer. |
| EDMA3 programmer | Any entity on the chip that has read/write access to the EDMA3 registers and can program an EDMA3 transfer. |
| EDMA3 transfer controller(s) (EDMA3TC) | Transfer controllers are the transfer engine for the EDMA3. Performs the read/writes as dictated by the transfer requests submitted by the EDMA3CC. |
| Enhanced direct memory access (EDMA3) controller | Consists of the EDMA3 channel controller (EDMA3CC) and EDMA3 transfer controller(s) (EDMA3TC). Is referred to as EDMA3 in this document. |

| Term | Meaning |
|---|---|
| Link parameter set | A PaRAM set that is used for linking. |
| Linking | The mechanism of reloading a PaRAM set with new transfer characteristics on completion of the current transfer. |
| Memory-mapped slave | All on-chip memories, off-chip memories, and slave peripherals. These typically rely on the EDMA3 (or other master peripheral) to perform transfers to and from them. |
| Master peripherals | All peripherals that are capable of initiating read and write transfers to the peripherals system and may not solely rely on the EDMA3 for their data transfers. |
| Null set or Null PaRAM set | A PaRAM set that has all count fields cleared (except for the link field). A dummy PaRAM set has at least one of the count fields nonzero. |
| Null transfer | A trigger event for a null PaRAM set results in the EDMA3CC performing a null transfer. This is an error condition. A dummy transfer is not an error condition. |
| QDMA channel | One of the 8 channels that can be triggered when writing to the trigger word (TRWORD) of a PaRAM set. All QDMA channels exist in the EDMA3CC. |
| Parameter RAM (PaRAM) | Programmable RAM that stores PaRAM sets used by DMA channels, QDMA channels, and linking. |
| Parameter RAM (PaRAM) set | A 32-byte EDMA3 channel transfer definition. Each parameter set consists of 8 words ( 4-bytes each), which store the context for a DMA/QDMA/link transfer. A PaRAM set includes source address, destination address, counts, indexes, options, etc. |
| Parameter RAM (PaRAM) set entry | One of the 4-byte components of the parameter set. |
| Slave end points | All on-chip memories, off-chip memories, and slave peripherals. These rely on the EDMA3 to perform transfers to and from them. |
| Transfer request (TR) | A command for data movement that is issued from the EDMA3CC to the EDMA3TC. A TR includes source and destination addresses, counts, indexes, options, etc. |
| Trigger event | Action that causes the EDMA3CC to service the PaRAM set and submit a transfer request to the EDMA3TC. Trigger events for DMA channels include manual triggered (CPU triggered), external event triggered, and chain triggered. Trigger events for QDMA channels include autotriggered and link triggered. |
| Trigger word | For QDMA channels, the trigger word specifies the PaRAM set entry that when written results in a QDMA trigger event. The trigger word is programmed via the QDMA channel *n* mapping register (QCHMAP*n*) and can point to any PaRAM set entry. |
| TR synchronization (sync) event | See Trigger event. |

# EDMA3 Architecture

This chapter discusses the architecture of the EDMA3 controller.

**Topic** **Page**

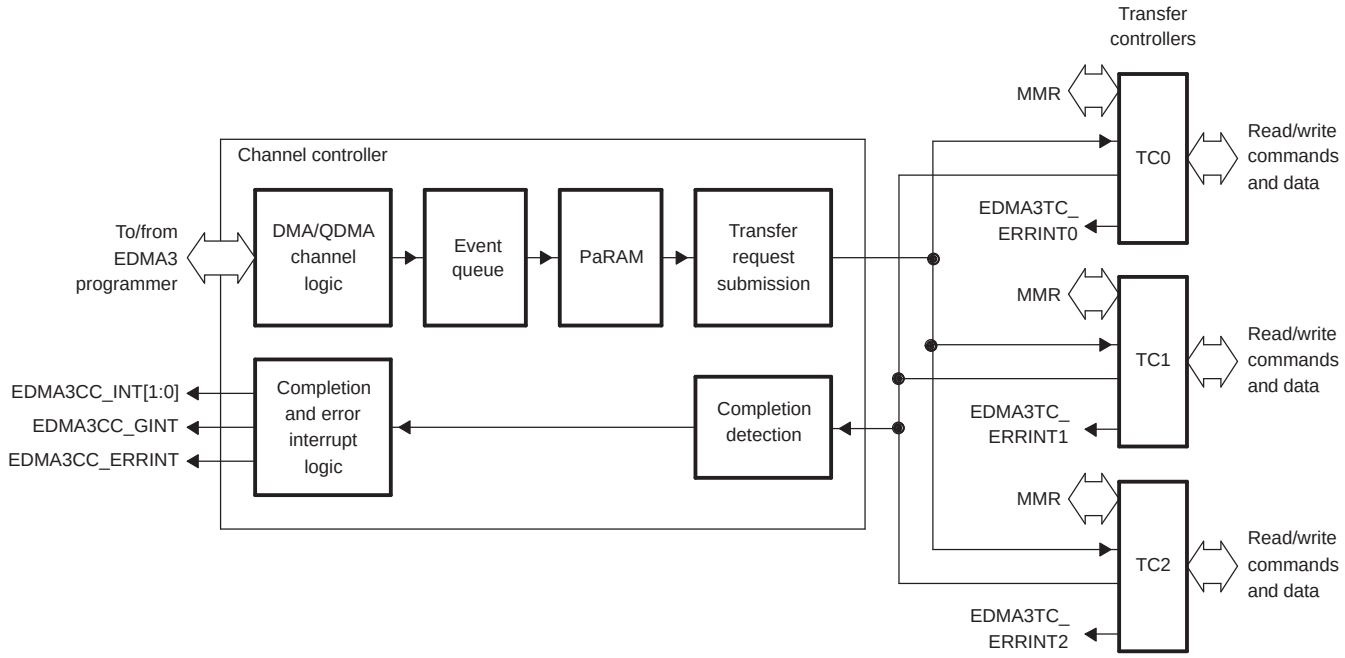
## 2.1 Functional Overview

This section provides a block diagram of the EDMA3 controller, EDMA3 channel controller (EDMA3CC), and EDMA3 transfer controller (EDMA3TC).

### 2.1.1 EDMA3 Controller Block Diagram

Figure 2-1 shows a block diagram for the EDMA3 controller.

**Figure 2-1. EDMA3 Controller Block Diagram**




### 2.1.2 EDMA3 Channel Controller (EDMA3CC)

Figure 2-2 shows a functional block diagram of the EDMA3 channel controller (EDMA3CC).

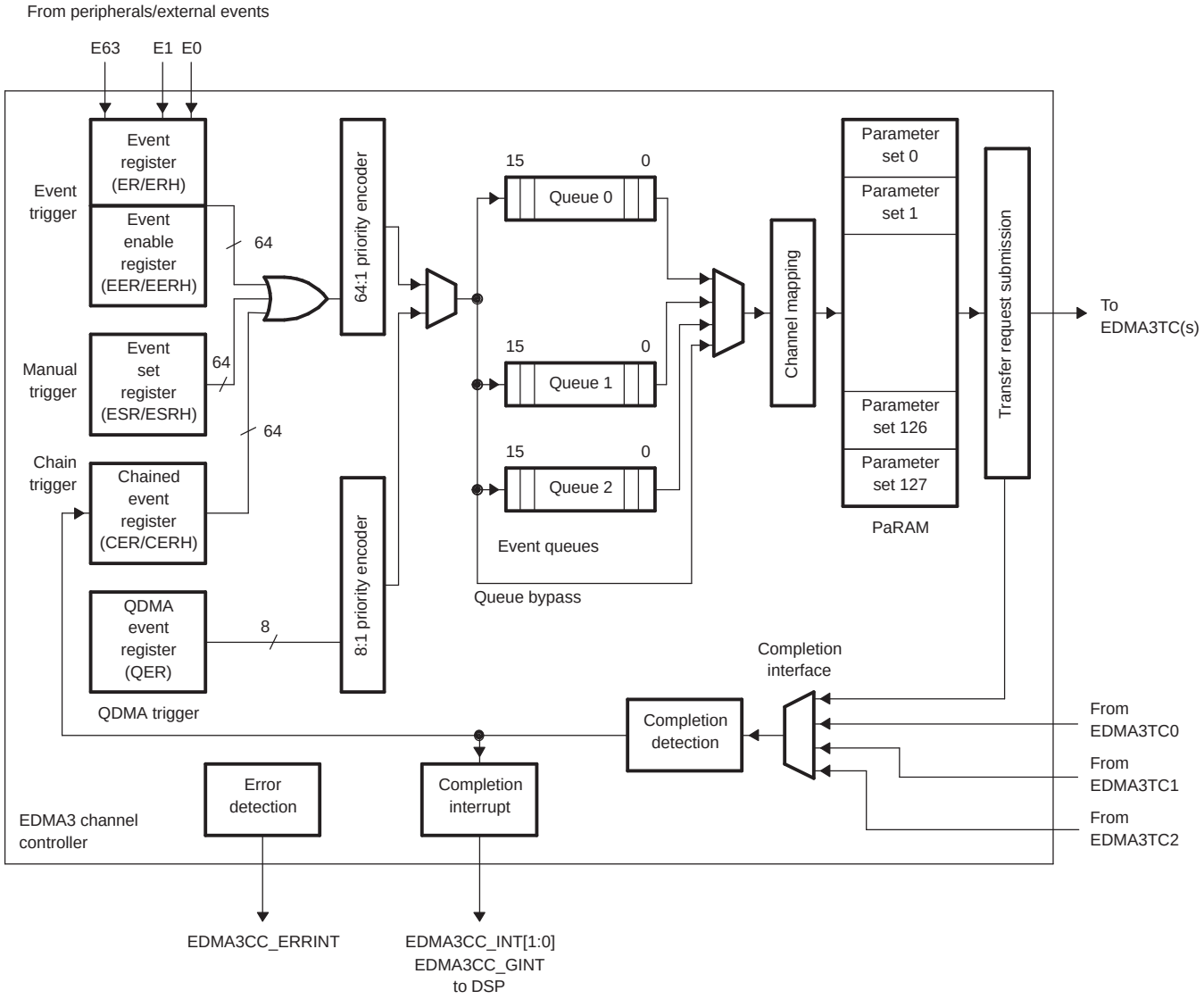
The main blocks of the EDMA3CC are:

- Parameter RAM (PaRAM): Maintains parameter set entries for channel and reload parameter sets. The PaRAM needs to be written with the transfer context for the desired channels and link parameter sets. EDMA3CC processes PaRAM sets based on a trigger event and submits a transfer request (TR) to the transfer controller.
- EDMA3 event and interrupt processing registers: Enable/disable events, enable/disable interrupt conditions, and clearing interrupts.
- Completion detection: The completion detect block detects completion of transfers by the EDMA3 transfer controller (EDMA3TC) and/or slave peripherals. Completion of transfers can optionally be used to chain trigger new transfers or to assert interrupts.
- Event queues: These form the interface between the event detection logic and the transfer request submission logic.

Other functions include:

- Region registers: Region registers allow DMA resources (DMA channels and interrupts) to be assigned to unique regions, which are owned by different EDMA3 programmers.
- Debug registers: Debug registers allow debug visibility by providing registers to read the queue status, controller status, and missed event status.

**Figure 2-2. EDMA3 Channel Controller (EDMA3CC) Block Diagram**




The EDMA3CC includes two channel types: DMA channels (64 channels) and QDMA channels (8 channels).

Each channel is associated with a given event queue/transfer controller and with a given PaRAM set. The main difference between a DMA channel and QDMA channel is how the transfers are triggered by the system. See Section 2.4.

A trigger event is needed to initiate a transfer. For DMA channels, a trigger event may be due to an external event, manual write to the event set register, or chained event. QDMA channels are autotriggered when a write is performed to the user-programmed trigger word. All such trigger events are logged into appropriate registers upon recognition. See DMA channel registers (Section 4.3.5) and QDMA registers (Section 4.3.7).

Once a trigger event is recognized, the event type/channel is queued in the appropriate EDMA3CC event queue. The assignment of each DMA/QDMA channel to event queue is programmable. Each queue is 16 deep, so up to 16 events may be queued (on a single queue) in the EDMA3CC at an instant in time. Additional pending events mapped to a full queue are queued when event queue space becomes available. See Section 2.10.

If events on different channels are detected simultaneously, the events are queued based on fixed priority arbitration scheme with the DMA channels being higher priority than the QDMA channels. Among the two groups of channels, the lowest-numbered channel is the highest priority.

Each event in the event queue is processed in the order it was queued. On reaching the head of the queue, the PaRAM associated with that channel is read to determine the transfer details. The TR submission logic evaluates the validity of the TR and is responsible for submitting a valid transfer request (TR) to the appropriate EDMA3TC (based on the event queue to EDMA3TC association, Q0 goes to TC0, Q1 goes to TC1, and Q2 goes to TC2). For more details, see Section 2.3.

The EDMA3TC receives the request and is responsible for data movement as specified in the transfer request packet (TRP) and other necessary tasks like buffering, ensuring transfers are carried out in an optimal fashion wherever possible. For more details on EDMA3TC, see Section 2.1.3.

You may have chosen to receive an interrupt or chain to another channel on completion of the current transfer in which case the EDMA3TC signals completion to the EDMA3CC completion detection logic when the transfer is done. You can alternately choose to trigger completion when a TR leaves the EDMA3CC boundary rather than wait for all the data transfers to complete. Based on the setting of the EDMA3CC interrupt registers, the completion interrupt generation logic is responsible for generating EDMA3CC completion interrupts to the CPU. For more details, see Section 2.5.

Additionally, the EDMA3CC also has an error detection logic, which causes error interrupt generation on various error conditions (like missed events, exceeding event queue thresholds, etc.). For more details on error interrupts, see Section 2.9.4.

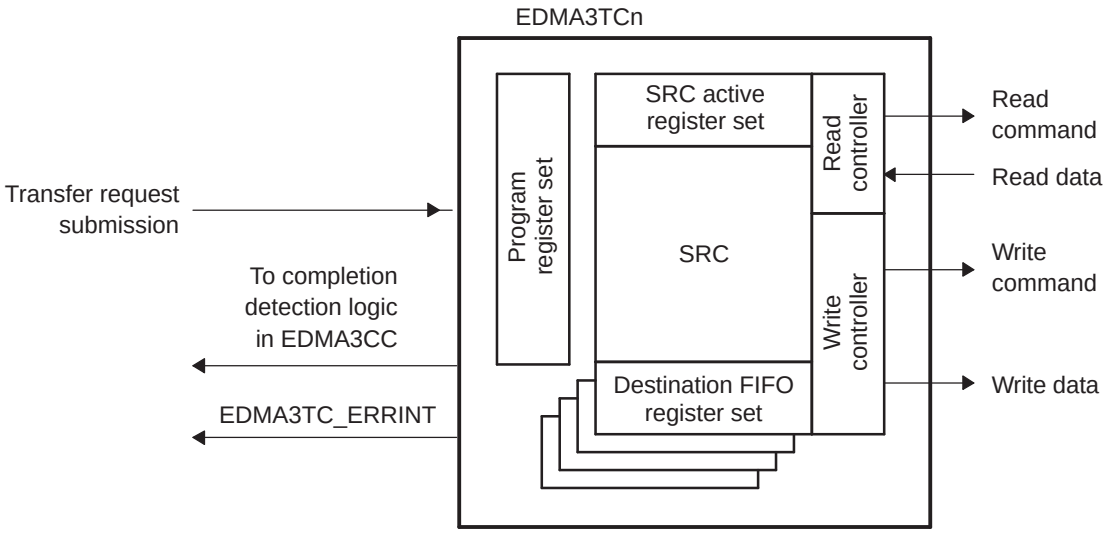
### 2.1.3 EDMA3 Transfer Controller (EDMA3TC)

Figure 2-3 shows a functional block diagram of the EDMA3 transfer controller (EDMA3TC).

The main blocks of the EDMA3TC are:

- DMA program register set: The DMA program register set stores the transfer requests received from the EDMA3 channel controller (EDMA3CC).
- DMA source active register set: The DMA source active register set stores the context for the DMA transfer request currently in progress in the read controller.
- Read controller: The read controller issues read commands to the source address.
- Destination FIFO register set: The destination (Dst) FIFO register set stores the context for the DMA transfer request(s) currently in progress in the write controller.
- Write controller: The write controller issues write commands/write data to the destination slave.
- Data FIFO: The data FIFO exists for holding temporary in-flight data.
- Completion interface: The completion interface sends completion codes to the EDMA3CC when a transfer completes, and is used for generating interrupts and chained events (see Section 2.5 for details on transfer completion reporting).

**Figure 2-3. EDMA3 Transfer Controller (EDMA3TC) Block Diagram**




When the EDMA3TC is idle and receives its first TR, the TR is received in the DMA program register set, where it transitions to the DMA source active set and the destination FIFO register set immediately. The second TR (if pending from EDMA3CC) is loaded into the DMA program set, ensuring it can start as soon as possible when the active transfer is completed. As soon as the current active set is exhausted, the TR is loaded from the DMA program register set into the DMA source active register set as well as to the appropriate entry in the destination FIFO register set.

The read controller issues read commands governed by the rules of command fragmentation and optimization. These are issued only when the data FIFO has space available for the data read. When sufficient data is in the data FIFO, the write controller starts issuing a write command again following the rules for command fragmentation and optimization. For details on command fragmentation and optimization, see Section 2.11.1.1.

Depending on the number of entries, the read controller can process up to 4 transfer requests ahead of the destination subject to the amount of free data FIFO.

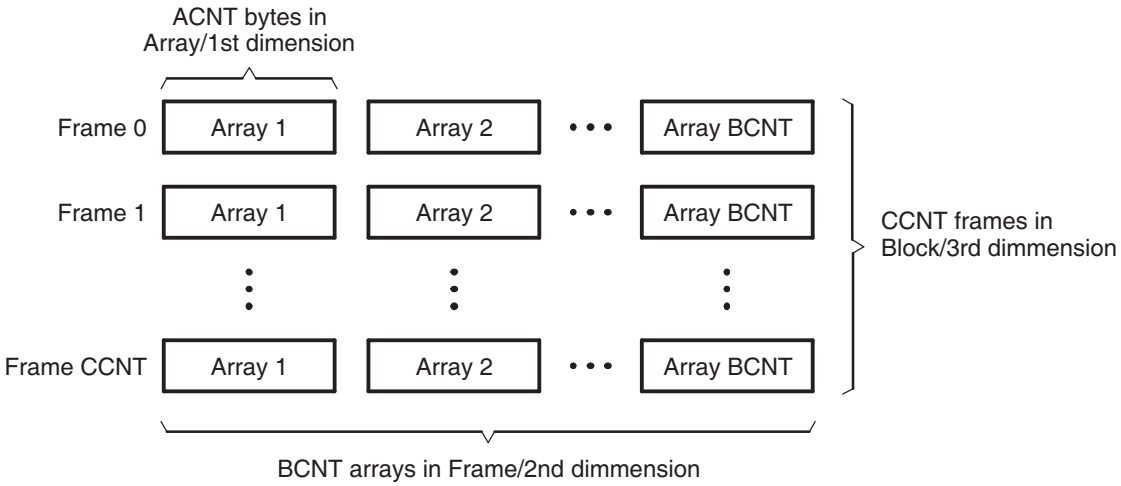
## 2.2 Types of EDMA3 Transfers

An EDMA3 transfer is always defined in terms of three dimensions. Figure 2-4 shows the three dimensions used by EDMA3 transfers. These three dimensions are defined as:

- 1st Dimension or Array (A): The 1st dimension in a transfer consists of ACNT contiguous bytes.
- 2nd Dimension or Frame (B): The 2nd dimension in a transfer consists of BCNT arrays of ACNT bytes. Each array transfer in the 2nd dimension is separated from each other by an index programmed using SRCBIDX or DSTBIDX.
- 3rd Dimension or Block (C): The 3rd dimension in a transfer consists of CCNT frames of BCNT arrays of ACNT bytes. Each transfer in the 3rd dimension is separated from the previous by an index programmed using SRCCIDX or DSTCIDX.

Note that the reference point for the index depends on the synchronization type. The amount of data transferred upon receipt of a trigger/synchronization event is controlled by the synchronization types (SYNCDIM bit in OPT). Of the three dimensions, only two synchronization types are supported: A-synchronized transfers and AB-synchronized transfers.

**Figure 2-4. Definition of ACNT, BCNT, and CCNT**



### 2.2.1 A-Synchronized Transfers

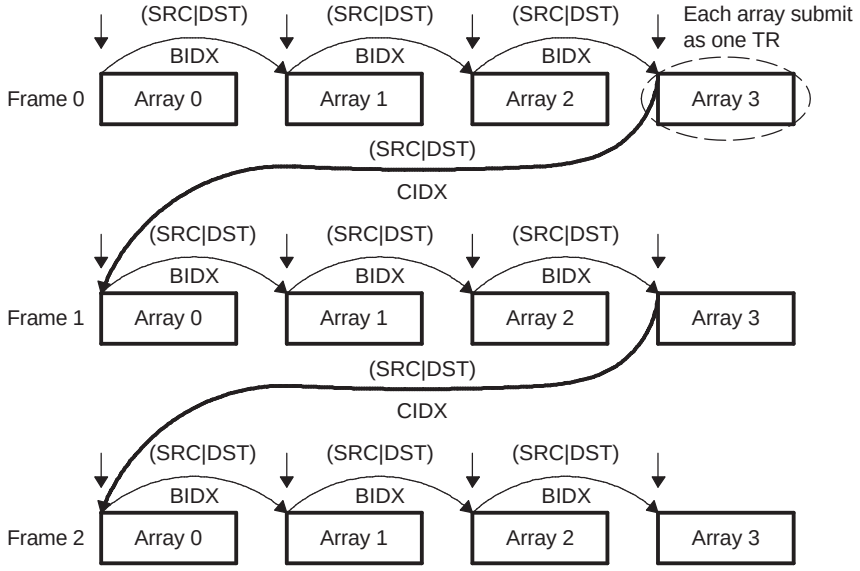
In an A-synchronized transfer, each EDMA3 sync event initiates the transfer of the 1st dimension of ACNT bytes, or one array of ACNT bytes. In other words, each event/TR packet conveys the transfer information for one array only. Thus, BCNT × CCNT events are needed to completely service a PaRAM set.

Arrays are always separated by SRCBIDX and DSTBIDX, as shown in Figure 2-5, where the start address of Array N is equal to the start address of Array N – 1 plus source (SRCBIDX) or destination (DSTBIDX).

Frames are always separated by SRCCIDX and DSTCIDX. For A-synchronized transfers, after the frame is exhausted, the address is updated by adding SRCCIDX/DSTCIDX to the beginning address of the last array in the frame. As in Figure 2-5, SRCCIDX/DSTCIDX is the difference between the start of Frame 0 Array 3 to the start of Frame 1 Array 0.

Figure 2-5 shows an A-synchronized transfer of 3 (CCNT) frames of 4 (BCNT) arrays of n (ACNT) bytes. In this example, a total of 12 sync events (BCNT × CCNT) exhaust a PaRAM set. See Section 2.3.6 for details on parameter set updates.

**Figure 2-5. A-Synchronized Transfers (ACNT = n, BCNT = 4, CCNT = 3)**



### 2.2.2 AB-Synchronized Transfers

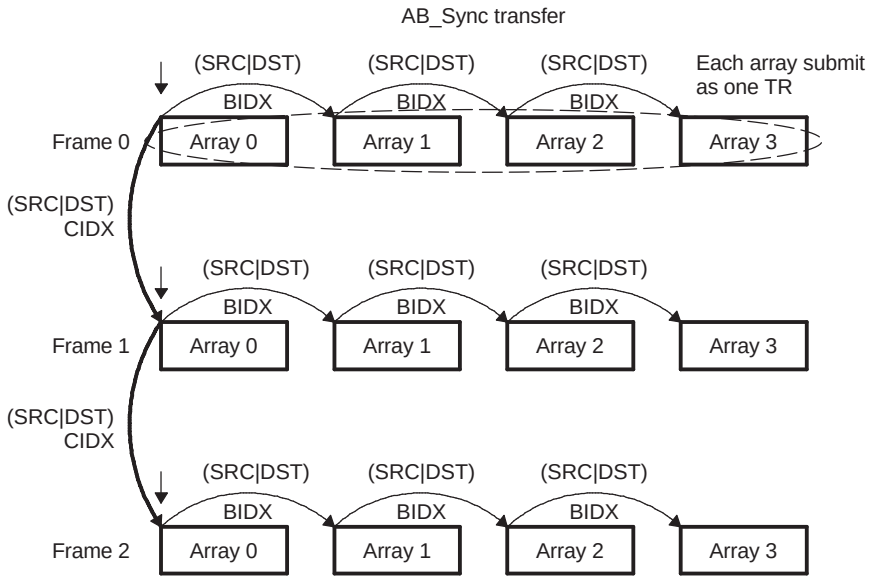
In a AB-synchronized transfer, each EDMA3 sync event initiates the transfer of 2 dimensions or one frame. In other words, each event/TR packet conveys information for one entire frame of BCNT arrays of ACNT bytes. Thus, CCNT events are needed to completely service a PaRAM set.

Arrays are always separated by SRCBIDX and DSTBIDX as shown in Figure 2-6. Frames are always separated by SRCCIDX and DSTCIDX.

Note that for AB-synchronized transfers, after a TR for the frame is submitted, the address update is to add SRCCIDX/DSTCIDX to the beginning address of the beginning array in the frame. This is different from A-synchronized transfers where the address is updated by adding SRCCIDX/DSTCIDX to the start address of the last array in the frame. See Section 2.3.6 for details on parameter set updates.

Figure 2-6 shows an AB-synchronized transfer of 3 (CCNT) frames of 4 (BCNT) arrays of *n* (ACNT) bytes. In this example, a total of 3 sync events (CCNT) exhaust a PaRAM set; that is, a total of 3 transfers of 4 arrays each completes the transfer.

**Figure 2-6. AB-Synchronized Transfers (ACNT = n, BCNT = 4, CCNT = 3)**




**NOTE:** ABC-synchronized transfers are not directly supported. But can be logically achieved by chaining between multiple AB-synchronized transfers.

## 2.3 Parameter RAM (PaRAM)

The EDMA3 controller is a RAM-based architecture. The transfer context (source/destination addresses, count, indexes, etc.) for DMA or QDMA channels is programmed in a parameter RAM table within EDMA3CC, referred to as PaRAM. The PaRAM table is segmented into multiple PaRAM sets. Each PaRAM set includes eight 4-byte PaRAM set entries (32-bytes total per PaRAM set), which includes typical DMA transfer parameters such as source address, destination address, transfer counts, indexes, options, etc. The PaRAM structure is shown in Table 2-1.

The PaRAM structure supports flexible ping-pong, circular buffering, channel chaining, and autoreloading (linking). The contents of the PaRAM include:

- 128 PaRAM sets
- 64-channels are direct mapped. Can be used as link or QDMA sets, if not used for DMA channels.
- 64-channels remain for link or QDMA sets

**NOTE:** By default, all DMA channels map to PaRAM set 0. These should be remapped before use, see Section 2.6.2.

**Table 2-1. EDMA3 Parameter RAM Contents**

| Address | Parameters |
|---|---|
| 01C0 4000h to 01C0 401Fh | Parameters for event 0 (8 words) |
| 01C0 4020h to 01C0 403Fh | Parameters for event 1 (8 words) |
| 01C0 4040h to 01C0 405Fh | Parameters for event 2 (8 words) |
| 01C0 4060h to 01C0 407Fh | Parameters for event 3 (8 words) |
| 01C0 4080h to 01C0 409Fh | Parameters for event 4 (8 words) |
| 01C0 40A0h to 01C0 40BFh | Parameters for event 5 (8 words) |
| 01C0 40C0h to 01C0 40DFh | Parameters for event 6 (8 words) |
| 01C0 40E0h to 01C0 40FFh | Parameters for event 7 (8 words) |
| 01C0 4100h to 01C0 411Fh | Parameters for event 8 (8 words) |
| 01C0 4120h to 01C0 413Fh | Parameters for event 9 (8 words) |
| 01C0 4140h to 01C0 415Fh | Parameters for event 10 (8 words) |
| 01C0 4160h to 01C0 417Fh | Parameters for event 11 (8 words) |
| 01C0 4180h to 01C0 419Fh | Parameters for event 12 (8 words) |
| 01C0 41A0h to 01C0 41BFh | Parameters for event 13 (8 words) |
| 01C0 41C0h to 01C0 41DFh | Parameters for event 14 (8 words) |
| 01C0 41E0h to 01C0 41FFh | Parameters for event 15 (8 words) |
| 01C0 4200h to 01C0 421Fh | Parameters for event 16 (8 words) |
| . . . | . . . |
| 01C0 47C0h to 01C0 47DFh | Parameters for event 62 (8 words) |
| 01C0 47E0h to 01C0 47FFh | Parameters for event 63 (8 words) |
| 01C0 4800h to 01C0 481Fh | 1st reload/link set (8 words)(1) |
| 01C0 4820h to 01C0 483Fh | 2nd reload/link set (8 words) (1) |
| . . . | . . . |
| 01C0 4FC0h to 01C0 4FDFh | 62nd reload/link set (8 words) (1) |
| 01C0 4FE0h to 01C0 4FFFh | 63rd reload/link set (8 words) (1) |

(1) DM643x devices have 8 QDMA channels that can be mapped to any parameter set number from 0 to 127.

### 2.3.1 PaRAM Set

Each parameter set of PaRAM is organized into eight 32-bit words or 32 bytes, as shown in Figure 2-7 and described in Table 2-2. Each PaRAM set consists of 16-bit and 32-bit parameters.

**Figure 2-7. PaRAM Set**

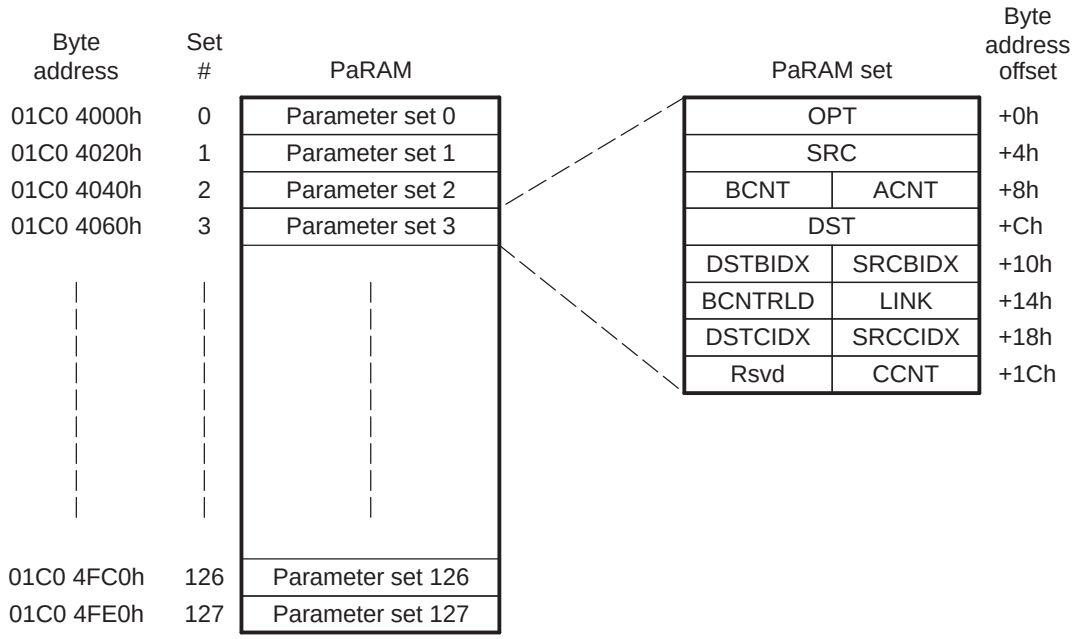


**Table 2-2. EDMA3 Channel Parameter Description**

| Offset Address (bytes) | Acronym | Parameter | Description |
|---|---|---|---|
| 0h | OPT | Channel Options | Transfer Configuration Options |
| 4h | SRC | Channel Source Address | The byte address from which data is transferred. |
| 8h[(1)] | ACNT | Count for 1st Dimension | Unsigned value specifying the number of contiguous bytes within an array (first dimension of the transfer). Valid values range from 1 to 65 535. |
| | BCNT | Count for 2nd Dimension | Unsigned value specifying the number of arrays in a frame, where an array is ACNT bytes. Valid values range from 1 to 65 535. |
| Ch | DST | Channel Destination Address | The byte address to which data is transferred. |
| 10h[(1)] | SRCBIDX | Source BCNT Index | Signed value specifying the byte address offset between source arrays within a frame (2nd dimension). Valid values range from –32 768 and 32 767. |
| | DSTBIDX | Destination BCNT Index | Signed value specifying the byte address offset between destination arrays within a frame (2nd dimension). Valid values range from –32 768 and 32 767. |
| 14h[(1)] | LINK | Link Address | The PaRAM address containing the PaRAM set to be linked (copied from) when the current PaRAM set is exhausted. A value of FFFFh specifies a null link. |
| | BCNTRLD | BCNT Reload | The count value used to reload BCNT when BCNT decrements to 0 (TR submitted for the last array in 2nd dimension). Only relevant in A-synchronized transfers. |
| 18h[(1)] | SRCCIDX | Source CCNT Index | Signed value specifying the byte address offset between frames within a block (3rd dimension). Valid values range from –32 768 and 32 767.<br>A-synchronized transfers: The byte address offset from the beginning of the last source array in a frame to the beginning of the first source array in the next frame.<br>AB-synchronized transfers: The byte address offset from the beginning of the first source array in a frame to the beginning of the first source array in the next frame. |
| | DSTCIDX | Destination CCNT index | Signed value specifying the byte address offset between frames within a block (3rd dimension). Valid values range from –32 768 and 32 767.<br>A-synchronized transfers: The byte address offset from the beginning of the last destination array in a frame to the beginning of the first destination array in the next frame.<br>AB-synchronized transfers: The byte address offset from the beginning of the first destination array in a frame to the beginning of the first destination array in the next frame. |
| 1Ch | CCNT | Count for 3rd Dimension | Unsigned value specifying the number of frames in a block, where a frame is BCNT arrays of ACNT bytes. Valid values range from 1 to 65 535. |
| | RSVD | Reserved | Reserved |

(1) It is recommended to access the parameter set entries as 32-bit words whenever possible.

## 2.3.2 EDMA3 Channel Parameter Set Fields

### 2.3.2.1 Channel Options Parameter (OPT)

The 32-bit channel options parameter (OPT) specifies the transfer configuration options. The channel options parameter (OPT) is described in Section 4.2.1.

### 2.3.2.2 Channel Source Address (SRC)

The 32-bit source address parameter specifies the starting byte address of the source. For SAM in increment mode, there are no alignment restrictions imposed by EDMA3. For SAM in constant addressing mode, you must program the source address to be aligned to a 256-bit aligned address (5 LSBs of address must be 0). The EDMA3TC will signal an error, if this rule is violated. See Section 2.11.2 for additional details.

### 2.3.2.3 Channel Destination Address (DST)

The 32-bit destination address parameter specifies the starting byte address of the destination. For DAM in increment mode, there are no alignment restrictions imposed by EDMA3. For DAM in constant addressing mode, you must program the destination address to be aligned to a 256-bit aligned address (5 LSBs of address must be 0). The EDMA3TC will signal an error, if this rule is violated. See Section 2.11.2 for additional details.

### 2.3.2.4 Count for 1st Dimension (ACNT)

ACNT represents the number of bytes within the 1st dimension of a transfer. ACNT is a 16-bit unsigned value with valid values between 0 and 65 535. Therefore, the maximum number of bytes in an array is 65 535 bytes (64K – 1 bytes). ACNT must be greater than or equal to 1 for a TR to be submitted to EDMA3TC. A transfer with ACNT equal to 0 is considered either a null or dummy transfer.

See Section 2.3.5 and Section 2.5.3 for details on dummy/null completion conditions.

### 2.3.2.5 Count for 2nd Dimension (BCNT)

BCNT is a 16-bit unsigned value that specifies the number of arrays of length ACNT. For normal operation, valid values for BCNT are between 1 and 65 535. Therefore, the maximum number of arrays in a frame is 65 535 (64K – 1 arrays). A transfer with BCNT equal to 0 is considered either a null or dummy transfer.

See Section 2.3.5 and Section 2.5.3 for details on dummy/null completion conditions.

### 2.3.2.6 Count for 3rd Dimension (CCNT)

CCNT is a 16-bit unsigned value that specifies the number of frames in a block. Valid values for CCNT are between 1 and 65 535. Therefore, the maximum number of frames in a block is 65 535 (64K – 1 frames). A transfer with CCNT equal to 0 is considered either a null or dummy transfer.

See Section 2.3.5 and Section 2.5.3 for details on dummy/null completion conditions.

### 2.3.2.7 BCNT Reload (BCNTRLD)

BCNTRLD is a 16-bit unsigned value used to reload the BCNT field once the last array in the 2nd dimension is transferred. This field is only used for A-synchronized transfers. In this case, the EDMA3CC decrements the BCNT value by 1 on each TR submission. When BCNT reaches 0, the EDMA3CC decrements CCNT and uses the BCNTRLD value to reinitialize the BCNT value.

For AB-synchronized transfers, the EDMA3CC submits the BCNT in the TR and the EDMA3TC decrements BCNT appropriately. For AB-synchronized transfers, BCNTRLD is not used.

#### 2.3.2.8 Source B Index (SRCBIDX)

SRCBIDX is a 16-bit signed value (2s complement) used for source address modification between each array in the 2nd dimension. Valid values for SRCBIDX are between –32 768 and 32 767. It provides a byte address offset from the beginning of the source array to the beginning of the next source array. It applies to both A-synchronized and AB-synchronized transfers. Some examples:

- SRCBIDX = 0000h (0): no address offset from the beginning of an array to the beginning of the next array. All arrays are fixed to the same beginning address.
- SRCBIDX = 0003h (+3): the address offset from the beginning of an array to the beginning of the next array in a frame is 3 bytes. For example, if the current array begins at address 1000h, the next array begins at 1003h.
- SRCBIDX = FFFFh (–1): the address offset from the beginning of an array to the beginning of the next array in a frame is –1 byte. For example, if the current array begins at address 5054h, the next array begins at 5053h.

#### 2.3.2.9 Destination B Index (DSTBIDX)

DSTBIDX is a 16-bit signed value (2s complement) used for destination address modification between each array in the 2nd dimension. Valid values for DSTBIDX are between –32 768 and 32 767. It provides a byte address offset from the beginning of the destination array to the beginning of the next destination array within the current frame. It applies to both A-synchronized and AB-synchronized transfers. See SRCBIDX (Section 2.3.2.8) for examples.

#### 2.3.2.10 Source C Index (SRCCIDX)

SRCCIDX is a 16-bit signed value (2s complement) used for source address modification in the 3rd dimension. Valid values for SRCCIDX are between –32 768 and 32 767. It provides a byte address offset from the beginning of the current array (pointed to by SRC address) to the beginning of the first source array in the next frame. It applies to both A-synchronized and AB-synchronized transfers. Note that when SRCCIDX is applied, the current array in an A-synchronized transfer is the last array in the frame (Figure 2-5), while the current array in an AB-synchronized transfer is the first array in the frame (Figure 2-6).

#### 2.3.2.11 Destination C Index (DSTCIDX)

DSTCIDX is a 16-bit signed value (2s complement) used for destination address modification in the 3rd dimension. Valid values are between –32 768 and 32 767. It provides a byte address offset from the beginning of the current array (pointed to by DST address) to the beginning of the first destination array TR in the next frame. It applies to both A-synchronized and AB-synchronized transfers. Note that when DSTCIDX is applied, the current array in an A-synchronized transfer is the last array in the frame (Figure 2-5), while the current array in a AB-synchronized transfer is the first array in the frame (Figure 2-6).

#### 2.3.2.12 Link Address (LINK)

The EDMA3CC provides a mechanism, called linking, to reload the current PaRAM set upon its natural termination (that is, after the count fields are decremented to 0) with a new PaRAM set. The 16-bit parameter LINK specifies the byte address offset in the PaRAM from which the EDMA3CC loads/reloads the next PaRAM set during linking.

You must program the link address to point to a valid aligned 32-byte PaRAM set. The 5 LSBs of the LINK field should be cleared to 0.

The EDMA3CC ignores the upper 2 bits of the LINK entry, allowing the programmer the flexibility of programming the link address as either an absolute/literal byte address or use the PaRAM-base-relative offset address. Therefore, if you make use of the literal address with a range from 4000h to 7FFFh, it will be treated as a PaRAM-base-relative value of 0000h to 3FFFh.

You should make sure to program the LINK field correctly, so that link update is requested from a PaRAM address that falls in the range of the available PaRAM addresses on the device.

A LINK value of FFFFh is referred to as a NULL link that should cause the EDMA3CC to perform an internal write of 0 to all entries of the current PaRAM set, except for the LINK field that is set to FFFFh. Also, see Section 2.5 for details on terminating a transfer.

### 2.3.3 Null PaRAM Set

A null PaRAM set is defined as a PaRAM set where all count fields (ACNT, BCNT, and CCNT) are cleared to 0. If a PaRAM set associated with a channel is a NULL set, then when serviced by the EDMA3CC, the bit corresponding to the channel is set in the associated event missed register (EMR, EMRH, or QEMR). This bit remains set in the associated secondary event register (SER, SERH, or QSER). *This implies that any future events on the same channel are ignored by the EDMA3CC and you are required to clear the bit in SER, SERH, or QSER for the channel.* This is considered an error condition, since events are not expected on a channel that is configured as a null transfer. See Section 4.3.5.8 and Section 4.3.2.1 for more information on the SER and EMR registers, respectively.

### 2.3.4 Dummy PaRAM Set

A dummy PaRAM set is defined as a PaRAM set where at least one of the count fields (ACNT, BCNT, or CCNT) is cleared to 0 and at least one of the count fields is nonzero.

If a PaRAM set associated with a channel is a dummy set, then when serviced by the EDMA3CC, it will not set the bit corresponding to the channel (DMA/QDMA) in the event missed register (EMR, EMRH, or QEMR) and the secondary event register (SER, SERH, or QSER) bit gets cleared similar to a normal transfer. Future events on that channel are serviced. A dummy transfer is a legal transfer of 0 bytes. See Section 4.3.5.8 and Section 4.3.2.1 for more information on the SER and EMR registers, respectively.

### 2.3.5 Dummy Versus Null Transfer Comparison

There are some differences in the way the EDMA3CC logic treats a dummy versus a null transfer request. A null transfer request is an error condition, but a dummy transfer is a legal transfer of 0 bytes. A null transfer causes an error bit (E*n*) in EMR to get set and the E*n* bit in SER remains set, essentially preventing any further transfers on that channel without clearing the associated error registers.

Table 2-3 summarizes the conditions and effects of null and dummy transfer requests.

**Table 2-3. Dummy and Null Transfer Request**

| Feature | Null TR | Dummy TR |
|---|---|---|
| EMR/EMRH/QEMR is set | Yes | No |
| SER/SERH/QSER remains set | Yes | No |
| Link update (STATIC = 0 in OPT) | Yes | Yes |
| QER is set | Yes | Yes |
| IPR/IPRH CER/CERH is set using early completion | Yes | Yes |

### 2.3.6 Parameter Set Updates

When a TR is submitted for a given DMA/QDMA channel and its corresponding PaRAM set, the EDMA3CC is responsible for updating the PaRAM set in anticipation of the next trigger event. For nonfinal events, this includes address and count updates; for final events, this includes the link update.

The specific PaRAM set entries that are updated depend on the channel's synchronization type (A-synchronized or B-synchronized) and the current state of the PaRAM set. A B-update refers to the decrementing of BCNT in the case of A-synchronized transfers after the submission of successive TRs. A C-update refers to the decrementing of CCNT in the case of A-synchronized transfers after BCNT TRs for ACNT byte transfers have submitted. For AB-synchronized transfers, a C-update refers to the decrementing of CCNT after submission of every transfer request.

See Table 2-4 for details and conditions on the parameter updates. A link update occurs when the PaRAM set is exhausted, as described in Section 2.3.7.

After the TR is read from the PaRAM (and is in process of being submitted to EDMA3TC), the following fields are updated if needed:

- A-synchronized: BCNT, CCNT, SRC, DST
- AB-synchronized: CCNT, SRC, DST

The following fields are not updated (except for during linking, where all fields are overwritten by the link PaRAM set):

- A-synchronized: ACNT, BCNTRLD, SRCBIDX, DSTBIDX, SRCCIDX, DSTCIDX, OPT, LINK
- AB-synchronized: ACNT, BCNT, BCNTRLD, SRCBIDX, DSTBIDX, SRCCIDX, DSTCIDX, OPT, LINK

Note that PaRAM updates only pertain to the information that is needed to properly submit the next transfer request to the EDMA3TC. Updates that occur while data is moved within a transfer request are tracked within the transfer controller, and is detailed in Section 2.11. For A-synchronized transfers, the EDMA3CC always submits a TRP for ACNT bytes (BCNT = 1 and CCNT = 1). For AB-synchronized transfers, the EDMA3CC always submits a TRP for ACNT bytes of BCNT arrays (CCNT = 1). The EDMA3TC is responsible for updating source and destination addresses within the array based on ACNT and FWID (in OPT). For AB-synchronized transfers, the EDMA3TC is also responsible to update source and destination addresses between arrays based on SRCBIDX and DSTBIDX.

Table 2-4 shows the details of parameter updates that occur within EDMA3CC for A-synchronized and AB-synchronized transfers.

**Table 2-4. Parameter Updates in EDMA3CC (for Non-Null, Non-Dummy PaRAM Set)**

| | A-Synchronized Transfer | | | AB-Synchronized Transfer | | |
|---|---|---|---|---|---|---|
| | **B-Update** | **C-Update** | **Link Update** | **B-Update** | **C-Update** | **Link Update** |
| **Condition:** | **BCNT > 1** | **BCNT == 1 && CCNT > 1** | **BCNT == 1 && CCNT == 1** | **N/A** | **CCNT > 1** | **CCNT == 1** |
| SRC | += SRCBIDX | += SRCCIDX | = Link.SRC | in EDMA3TC | += SRCCIDX | = Link.SRC |
| DST | += DSTBIDX | += DSTCIDX | = Link.DST | in EDMA3TC | += DSTCIDX | = Link.DST |
| ACNT | None | None | = Link.ACNT | None | None | = Link.ACNT |
| BCNT | –= 1 | = BCNTRLD | = Link.BCNT | in EDMA3TC | N/A | = Link.BCNT |
| CCNT | None | –= 1 | = Link.CCNT | in EDMA3TC | –=1 | = Link.CCNT |
| SRCBIDX | None | None | = Link.SRCBIDX | in EDMA3TC | None | = Link.SRCBIDX |
| DSTBIDX | None | None | = Link.DSTBIDX | None | None | = Link.DSTBIDX |
| SRCCIDX | None | None | = Link.SRCBIDX | in EDMA3TC | None | = Link.SRCBIDX |
| DSTCIDX | None | None | = Link.DSTBIDX | None | None | = Link.DSTBIDX |
| LINK | None | None | = Link.LINK | None | None | = Link.LINK |
| BCNTRLD | None | None | = Link.BCNTRLD | None | None | = Link.BCNTRLD |
| OPT[1] | None | None | = LINK.OPT | None | None | = LINK.OPT |

(1) In all cases, no updates occur if OPT.STATIC == 1 for the current PaRAM set.

**NOTE:** The EDMA3CC includes no special hardware to detect when an indexed address update calculation overflows/underflows. The address update will wrap across boundaries as programmed by the user. You should ensure that no transfer is allowed to cross internal port boundaries between peripherals. A single TR must target a single source/destination slave endpoint.

### 2.3.7 Linking Transfers

The EDMA3CC provides a mechanism known as linking, which allows the entire PaRAM set to be reloaded from a location within the PaRAM memory map (for both DMA and QDMA channels). Linking is especially useful for maintaining ping-pong buffers, circular buffering, and repetitive/continuous transfers all with no CPU intervention. Upon completion of a transfer, the current transfer parameters are reloaded with the parameter set pointed to by the 16-bit link address field (of the current parameter set). Linking only occurs when the STATIC bit in OPT is cleared to 0.

**NOTE:** A transfer (DMA or QDMA) should always be linked to another useful transfer. If it is required to terminate a transfer, the transfer should be linked to a NULL set.

The link update occurs after the current PaRAM set event parameters have been exhausted. An event's parameters are exhausted when the EDMA3 channel controller has submitted all the transfers associated with the PaRAM set.

A link update occurs for null and dummy transfers depending on the state of the STATIC bit in OPT and the LINK field. In both cases (null or dummy), if the value of LINK is FFFFh then a null PaRAM set (with all 0s and LINK set to FFFFh) is written to the current PaRAM set. Similarly, if LINK is set to a value other than FFFFh then the appropriate PaRAM location pointed to by LINK is copied to the current PaRAM set.

Once the channel completion conditions are met for an event, the transfer parameters located at the link address are loaded into the current DMA or QDMA channel's associated parameter set. The EDMA3CC reads the entire PaRAM set (8 words) from the PaRAM set specified by LINK and writes all 8 words to the PaRAM set associated with the current channel. Figure 2-8 shows an example of a linked transfer.

Any PaRAM set in the PaRAM can be used as a link/reload parameter set. The PaRAM sets associated with peripheral synchronization events (see Section 2.6) should only be used for linking if the corresponding events are disabled.

If a PaRAM set location is mapped to a QDMA channel (by QCHMAP*n*), then copying the link PaRAM set onto the current QDMA channel PaRAM set is recognized as a trigger event and is latched in QER since a write to the trigger word was performed. This feature can be used to create a linked list of transfers using a single QDMA channel and multiple PaRAM sets.

Link-to-self transfers replicate the behavior of autoinitialization, which facilitates the use of circular buffering and repetitive transfers. After an EDMA3 channel exhausts its current PaRAM set, it reloads all the parameter set entries from another PaRAM set, which is initialized with values identical to the original PaRAM set. Figure 2-9 shows an example of a linked-to-self transfer. In Figure 2-9, parameter set 127 has the LINK field address pointing to the address of parameter set 127 (4FE0h), that is, linked-to-self.

**NOTE:** If the STATIC bit in OPT is set for a PaRAM set, then link updates are not performed.

#### 2.3.7.1 Constant Addressing Mode Transfers/Alignment Issues

If either SAM or DAM is set to 1 (constant addressing mode), then the source or destination address must be aligned to a 256-bit aligned address, respectively, and the corresponding BIDX should be an even multiple of 32 bytes (256 bit). The EDMA3CC does not recognize errors here but the EDMA3TC asserts an error, if this is not true. See Section 2.11.2.

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

### 2.3.7.2 Element Size

The EDMA3 controller does not use the concept of element-size and element-indexing. Instead, all transfers are defined in terms of all three dimensions: ACNT, BCNT, and CCNT. An element-indexed transfer is logically achieved by programming ACNT to the size of the element and BCNT to the number of elements that need to be transferred. For example, if you have 16-bit audio data and 256 audio samples that needed to be transferred to a serial port, this can be done by programming the ACNT = 2 (2 bytes) and BCNT = 256.

**Figure 2-8. Linked Transfer**

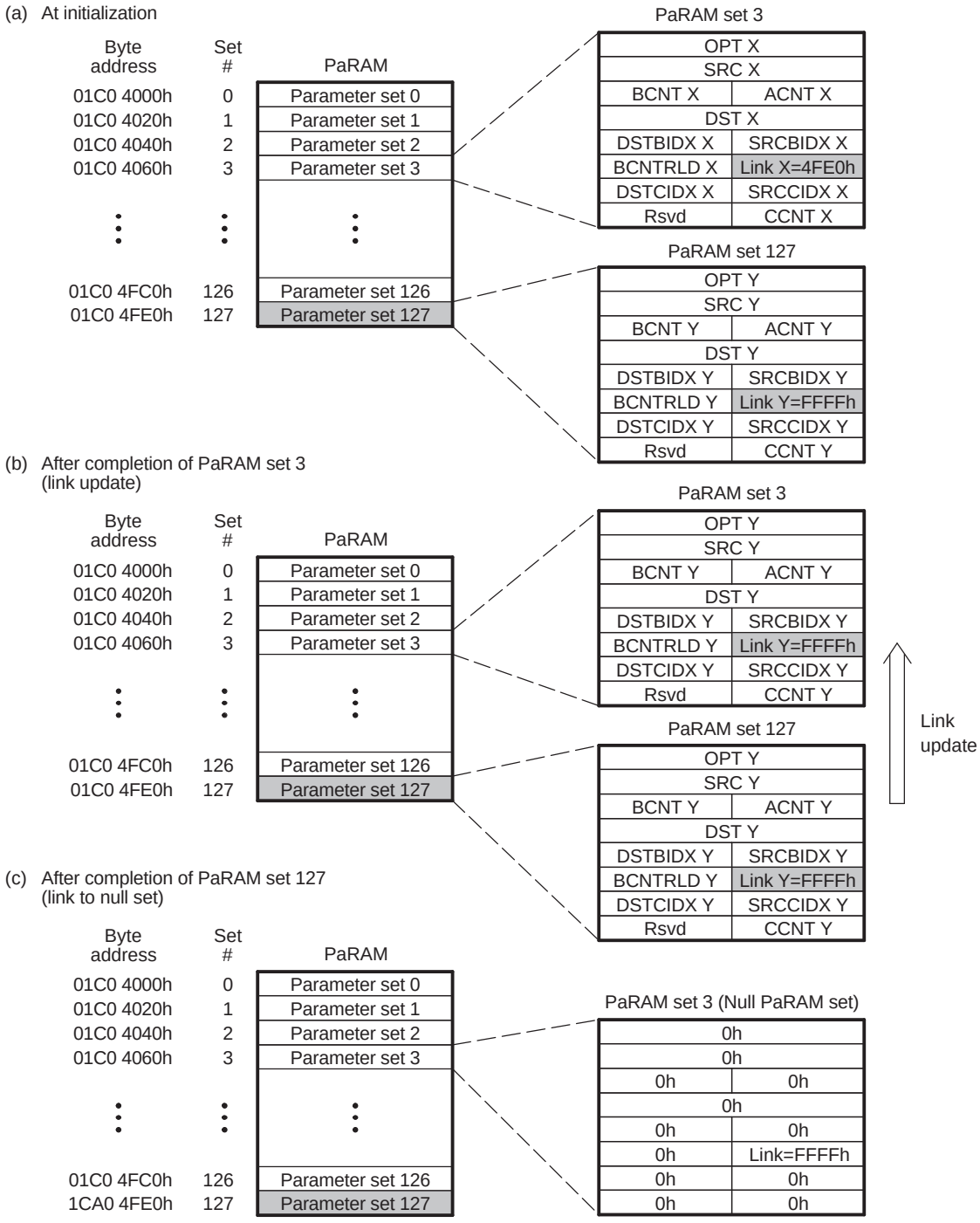
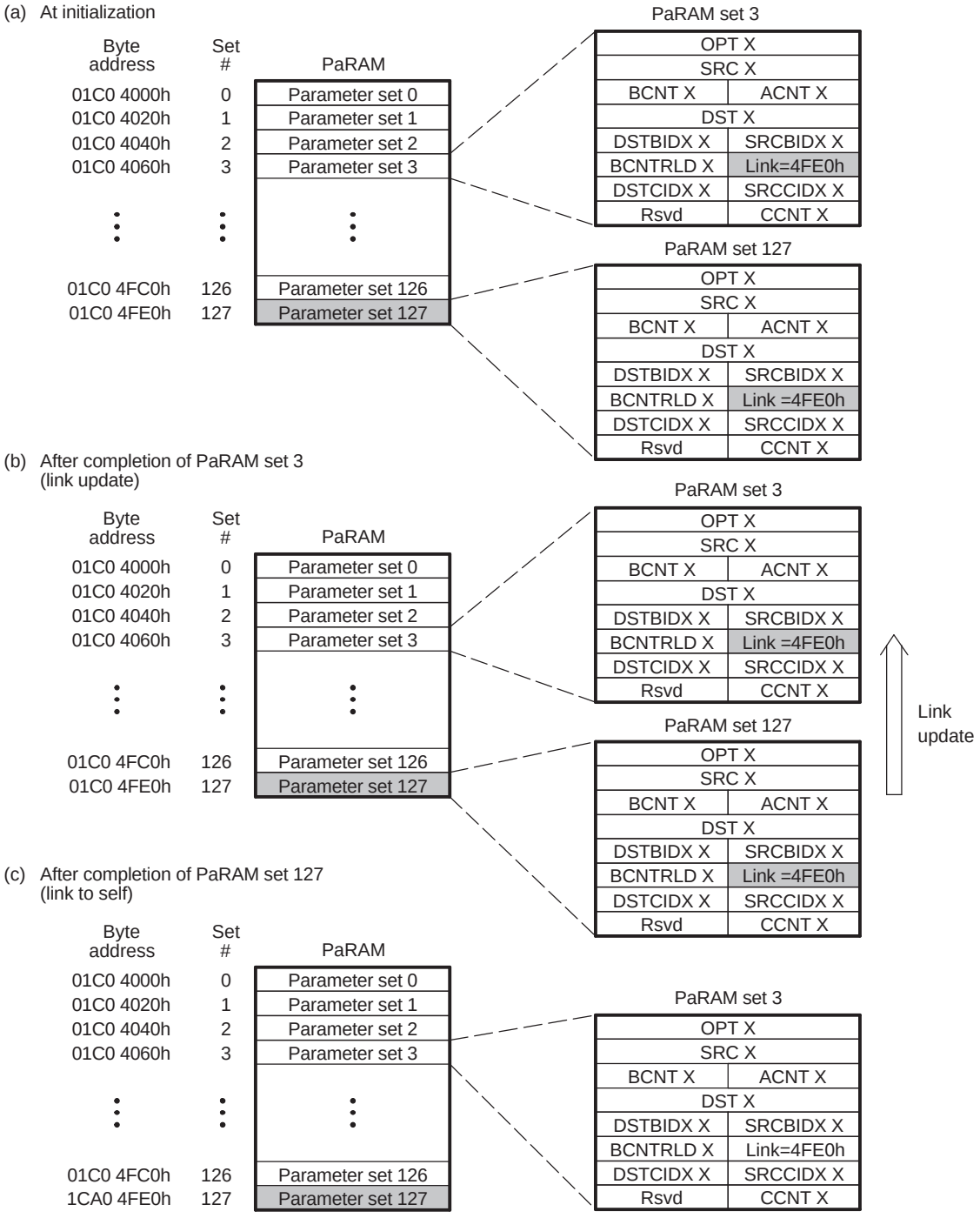


**Figure 2-9. Link-to-Self Transfer**



## 2.4 Initiating a DMA Transfer

There are multiple ways to initiate a programmed data transfer using the EDMA3 channel controller. Transfers on DMA channels are initiated by three sources:

- **Event-triggered transfer request** (this is the more typical usage of EDMA3): Allows for a peripheral, system, or externally-generated event to trigger a transfer request.
- **Manually-triggered transfer request:** The CPU manually triggers a transfer by writing a 1 to the corresponding bit in the event set register (ESR/ESRH).
- **Chain-triggered transfer request:** A transfer is triggered on the completion of another transfer or subtransfer.

Transfers on QDMA channels are initiated by two sources:

- **Autotriggered transfer request:** A transfer is triggered when the programmed trigger word is written to.
- **Link-triggered transfer requests:** When linking occurs, the transfer is triggered when the trigger word is written to.

### *2.4.1 DMA Channel*

#### 2.4.1.1 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.E*n* = 1). If the corresponding event in the event enable register (EER) is enabled (EER.E*n* = 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.

If the PaRAM set is valid (not a NULL set), then a transfer request packet (TRP) is submitted to the EDMA3TC and the E*n* bit in ER is cleared. At this point, a new event can be safely received by the EDMA3CC.

If the PaRAM set associated with the channel is a NULL set (see Section 2.3.3), then no transfer request (TR) is submitted and the corresponding E*n* bit in ER is cleared and simultaneously the corresponding channel bit is set in the event miss register (EMR.E*n* = 1) to indicate that the event was discarded due to a null TR being serviced. Good programming practices should include cleaning the event missed error before retriggering the DMA channel.

When an event is received, the corresponding event bit in the event register is set (ER.E*n* = 1), regardless of the state of EER.E*n*. If the event is disabled when an external event is received (ER.E*n* = 1 and EER.E*n* = 0), the ER.E*n* bit remains set. If the event is subsequently enabled (EER.E*n* = 1), then the pending event is processed by the EDMA3CC and the TR is processed/submitted, after which the ER.E*n* bit is cleared.

If an event is being processed (prioritized or is in the event queue) and another sync event is received for the same channel prior to the original being cleared (ER.E*n* != 0), then the second event is registered as a missed event in the corresponding bit of the event missed register (EMR.E*n* = 1).

Table 2-5 gives an example of the synchronization events associated with each of the programmable DMA channels in the TMS320DM6437 DMP. See the device-specific data manual to determine the event to channel mapping.

**Table 2-5. EDMA3 Channel Synchronization Events for TMS320DM6437 DMP**

| EDMA3 Channel | Event Name | Event Description |
|---|---|---|
| 0-1 | — | Reserved |
| 2 | XEVT0 | McBSP0 Transmit Event |
| 3 | REVT0 | McBSP0 Receive Event |
| 4 | XEVT1 | McBSP1 Transmit Event |
| 5 | REVT1 | McBSP1 Receive Event |
| 6 | HISTEVT | VPSS Histogram Event |

**Table 2-5. EDMA3 Channel Synchronization Events for TMS320DM6437 DMP (continued)**

| EDMA3 Channel | Event Name | Event Description |
|---|---|---|
| 7 | H3AEVT | VPSS H3A Event |
| 8 | PRVUEVT | VPSS Previewer Event |
| 9 | RSZEVT | VPSS Resizer Event |
| 10 | AXEVTE0 | McASP0 Transmit Event Even |
| 11 | AXEVTO0 | McASP0 Transmit Event Odd |
| 12 | AXEVT0 | McASP0 Transmit Event |
| 13 | AREVTE0 | McASP0 Receive Event Even |
| 14 | AREVTO0 | McASP0 Receive Event Odd |
| 15 | AREVT0 | McASP0 Receive Event |
| 16-21 | — | Reserved |
| 22 | URXEVT0 | UART 0 Receive Event |
| 23 | UTXEVT0 | UART 0 Transmit Event |
| 24 | URXEVT1 | UART 1 Receive Event |
| 25 | UTXEVT1 | UART 1 Transmit Event |
| 26-27 | — | Reserved |
| 28 | ICREVT | I2C Receive Event |
| 29 | ICXEVT | I2C Transmit Event |
| 30-31 | — | Reserved |
| 32 | GPINT0 | GPIO 0 Interrupt |
| 33 | GPINT1 | GPIO 1 Interrupt |
| 34 | GPINT2 | GPIO 2 Interrupt |
| 35 | GPINT3 | GPIO 3 Interrupt |
| 36 | GPINT4 | GPIO 4 Interrupt |
| 37 | GPINT5 | GPIO 5 Interrupt |
| 38 | GPINT6 | GPIO 6 Interrupt |
| 39 | GPINT7 | GPIO 7 Interrupt |
| 40 | GPBNKINT0 | GPIO Bank 0 Interrupt |
| 41 | GPBNKINT1 | GPIO Bank 1 Interrupt |
| 42 | GPBNKINT2 | GPIO Bank 2 Interrupt |
| 43 | GPBNKINT3 | GPIO Bank 3 Interrupt |
| 44 | GPBNKINT5 | GPIO Bank 4 Interrupt |
| 45 | GPBNKINT4 | GPIO Bank 5 Interrupt |
| 46 | GPBNKINT6 | GPIO Bank 6 Interrupt |
| 47 | — | Reserved |
| 48 | TEVTL0 | Timer 0 Event Low Interrupt |
| 49 | TEVTH0 | Timer 0 Event High Interrupt |
| 50 | TEVTL1 | Timer 1 Event Low Interrupt |
| 51 | TEVTH1 | Timer 1 Event High Interrupt |
| 52 | PWM0 | PWM 0 Event |
| 53 | PWM1 | PWM 1 Event |
| 54 | PWM2 | PWM 2 Event |
| 55-63 | — | Reserved |

#### 2.4.1.2 Manually-Triggered Transfer Request

A DMA transfer is initiated by a write to the event set register (ESR) by the CPU (or any EDMA programmer). Writing a 1 to an event bit in the ESR results in the event being prioritized/queued in the appropriate event queue, regardless of the state of the EER.E*n* bit. When the event reaches the head of the queue, it is evaluated for submission as a transfer request to the transfer controller.

As in the event-triggered transfers, if the PaRAM set associated with the channel is valid (it is not a null set) then the TR is submitted to the associated EDMA3TC and the channel can be triggered again.

If the PaRAM set associated with the channel is a NULL set (see Section 2.3.3), then no transfer request (TR) is submitted and the corresponding E*n* bit in ER is cleared and simultaneously the corresponding channel bit is set in the event miss register (EMR.E*n* = 1) to indicate that the event was discarded due to a null TR being serviced. Good programming practices should include clearing the event missed error before retriggering the DMA channel.

If an event is being processed (prioritized or is in the event queue) and the same channel is manually set by a write to the corresponding channel bit of the event set register (ESR.E*n* = 1) prior to the original being cleared (ESR.E*n* = 0), then the second event is registered as a missed event in the corresponding bit of the event missed register (EMR.E*n* = 1).

#### 2.4.1.3 Chain-Triggered Transfer Request

Chaining is a mechanism by which 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) to be set (CER.E[TCC] = 1).

Once a bit is set in CER, 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.

As in the event-triggered transfers, if the PaRAM set associated with the channel is valid (it is not a null set) then the TR is submitted to the associated EDMA3TC and the channel can be triggered again.

If the PaRAM set associated with the channel is a NULL set (see Section 2.3.3), then no transfer request (TR) is submitted and the corresponding E*n* bit in CER is cleared and simultaneously the corresponding channel bit is set in the event miss register (EMR.E*n* = 1) to indicate that the event was discarded due to a null TR being serviced. In this case, the error condition must be cleared by you before the DMA channel can be retriggered. Good programming practices might include clearing the event missed error before retriggering the DMA channel.

If a chaining event is being processed (prioritized or queued) and another chained event is received for the same channel prior to the original being cleared (CER.E*n* != 0), then the second chained event is registered as a missed event in the corresponding channel bit of the event missed register (EMR.E*n* = 1).

**NOTE:** Chained event registers, event registers, and event set registers operate independently. An event (E*n*) can be triggered by any of the trigger sources (event-triggered, manually-triggered, or chain-triggered).

### 2.4.2 QDMA Channels

#### 2.4.2.1 Autotriggered and Link-Triggered Transfer Request

QDMA-based transfer requests are issued when a QDMA event gets latched in the QDMA event register (QER.E*n* = 1). A bit corresponding to a QDMA channel is set in the QDMA event register (QER) when the following occurs:

- A CPU (or any EDMA3 programmer) write occurs to a PaRAM address that is defined as a QDMA channel trigger word (programmed in the QDMA channel *n* mapping register (QCHMAP*n*)) for the particular QDMA channel and the QDMA channel is enabled via the QDMA event enable register (QEER.E*n* = 1).
- EDMA3CC performs a link update on a PaRAM set address that is configured as a QDMA channel (matches QCHMAP*n* settings) and the corresponding channel is enabled via the QDMA event enable register (QEER.E*n* = 1).

Once a bit is set in QER, 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.

As in the event-triggered transfers, if the PaRAM set associated with the channel is valid (it is not a null set) then the TR is submitted to the associated EDMA3TC and the channel can be triggered again.

If a bit is already set in QER (QER.E*n* = 1) and a second QDMA event for the same QDMA channel occurs prior to the original being cleared, the second QDMA event gets captured in the QDMA event miss register (QEMR.E*n* = 1).

### 2.4.3 Comparison Between DMA and QDMA Channels

The primary difference between DMA and QDMA channels is the event/channel synchronization. QDMA events are either autotriggered or link triggered. Autotriggering allows QDMA channels to be triggered by CPU(s) with a minimum number of linear writes to PaRAM. Link triggering allows a linked list of transfers to be executed, using a single QDMA PaRAM set and multiple link PaRAM sets.

A QDMA transfer is triggered when a CPU (or other EDMA3 programmer) writes to the trigger word of the QDMA channel parameter set (autotriggered) or when the EDMA3CC performs a link update on a PaRAM set that has been mapped to a QDMA channel (link triggered). Note that for CPU triggered (manually triggered) DMA channels, in addition to writing to the PaRAM set, it is required to write to the event set register (ESR) to kick-off the transfer.

QDMA channels are typically for cases where a single event will accomplish a complete transfer since the CPU (or EDMA3 programmer) must reprogram some portion of the QDMA PaRAM set in order to retrigger the channel. In other words, QDMA transfers are programmed with BCNT = CCNT = 1 for A-synchronized transfers, and CCNT = 1 for AB-synchronized transfers.

Additionally, since linking is also supported (if STATIC = 0 in OPT) for QDMA transfers, it allows you to initiate a linked list of QDMAs, so when EDMA3CC copies over a link PaRAM set (including the write to the trigger word), the current PaRAM set mapped to the QDMA channel will automatically be recognized as a valid QDMA event and initiate another set of transfers as specified by the linked set.

## 2.5 Completion of a DMA Transfer

A parameter set for a given channel is complete when the required number of transfer requests is submitted (based on receiving the number of synchronization events). The expected number of TRs for a non-null/non-dummy transfer is shown in Table 2-6 for both synchronization types along with state of the PaRAM set prior to the final TR being submitted. When the counts (BCNT and/or CCNT) are this value, the next TR results in a:

- Final chaining or interrupt codes to be sent by the transfer controllers (instead of intermediate).
- Link updates (linking to either null or another valid link set).

**Table 2-6. Expected Number of Transfers for Non-Null Transfer**

| Sync Mode | Counts at time 0 | Total # Transfers | Counts prior to final TR |
|---|---|---|---|
| A-synchronized | ACNT<br>BCNT<br>CCNT | (BCNT × CCNT ) TRs of ACNT bytes each | BCNT == 1 && CCNT == 1 |
| AB-synchronized | ACNT<br>BCNT<br>CCNT | CCNT TRs for ACNT × BCNT bytes each | CCNT == 1 |

You must program the PaRAM OPT field with a specific transfer completion code (TCC) along with the other OPT fields (TCCHEN, TCINTEN, ITCCHEN, and ITCINTEN bits) to indicate whether the completion code is to be used for generating a chained event or/and for generating an interrupt upon completion of a transfer.

The specific TCC value (6-bit binary value) programmed dictates which of the 64-bits in the chain event register (CER[TCC]) and/or interrupt pending register (IPR[TCC]) is set.

See Section 2.9 for details on interrupts and Section 2.8 for details on chaining.

You can also selectively program whether the transfer controller sends back completion codes on completion of the final transfer request (TR) of a parameter set (TCCHEN or TCINTEN), for all but the final transfer request (TR) of a parameter set (ITCCHEN or ITCINTEN), or for all TRs of a parameter set (both). See Section 2.8 for details on chaining (intermediate/final chaining) and Section 2.9 for details on intermediate/final interrupt completion.

A completion detection interface exists between the EDMA3 channel controller and transfer controller(s). This interface sends back information from the transfer controller to the channel controller to indicate that a specific transfer is completed.

All DMA/QDMA PaRAM sets must also specify a link address value. For repetitive transfers such as ping-pong buffers, the link address value should point to another predefined PaRAM set. Alternatively, a nonrepetitive transfer should set the link address value to the null link value. The null link value is defined as FFFFh. See Section 2.3.7 for more details.

**NOTE:** Any incoming events that are mapped to a null PaRAM set results in an error condition. The error condition should be cleared before the corresponding channel is used again. See Section 2.3.5.

There are three ways the EDMA3CC gets updated/informed about a transfer completion: normal completion, early completion, and dummy/null completion. This applies to both chained events and completion interrupt generation.

### 2.5.1 Normal Completion

In normal completion mode (TCCMODE = 0 in OPT), the transfer or sub-transfer is considered to be complete when the EDMA3 channel controller receives the completion codes from the EDMA3 transfer controller. In this mode, the completion code to the channel controller is posted by the transfer controller after it receives a signal from the destination peripheral. Normal completion is typically used to generate an interrupt to inform the CPU that a set of data is ready for processing.

### 2.5.2 Early Completion

In early completion mode (TCCMODE = 1 in OPT), the transfer is considered to be complete when the EDMA3 channel controller submits the transfer request (TR) to the EDMA3 transfer controller. In this mode, the channel controller generates the completion code internally. Early completion is typically useful for chaining, as it allows subsequent transfers to be chained-triggered while the previous transfer is still in progress within the transfer controller, maximizing the overall throughput of the set of the transfers.

### 2.5.3 Dummy or Null Completion

This is a variation of early completion. Dummy or null completion is associated with a dummy set (Section 2.3.4) or null set (Section 2.3.3). In both cases, the EDMA3 channel controller does not submit the associated transfer request to the EDMA3 transfer controller(s). However, if the set (dummy/null) has the OPT field programmed to return completion code (intermediate/final interrupt/chaining completion), then it will set the appropriate bits in the interrupt pending registers (IPR/IPRH) or chained event register (CER/CERH). The internal early completion path is used by the channel controller to return the completion codes internally (that is, EDMA3CC generates the completion code).

## 2.6 Event, Channel, and PaRAM Mapping

Several of the 64 DMA channels are tied to a specific hardware event, thus allowing transfers to be triggered by events from device peripherals or external hardware. A DMA channel typically requests a data transfer when it receives its event (apart from manually-triggered, chain-triggered, and other transfers). The amount of data transferred per synchronization event depends on the channel's configuration (ACNT, BCNT, CCNT, etc.) and the synchronization type (A-synchronized or AB-synchronized).

The association of an event to a channel is fixed. Each of the DMA channels has one specific event associated with it. Table 2-5 provides the synchronization events associated with each of the programmable DMA channels.

If in an application, a channel does not make use of the associated synchronization event or does not have an associated synchronization event (unused), that channel can be used for manually-triggered or chained-triggered transfers, for linking/reloading, or as a QDMA channel.

### 2.6.1 DMA Channel to PaRAM Mapping

The mapping between the DMA channel numbers and the PaRAM sets is a fixed, one-to-one mapping (see Table 2-7). In other words, channel (event) 0 is mapped to PaRAM set 0, channel (event 1) is mapped to PaRAM set 1, etc. So, for example, in order to program a transfer for event number 3, DMA channel 3 is associated with PaRAM set number 3 and you need to program this PaRAM set for configuring transfers associated with event number 3.

**Table 2-7. EDMA3 DMA Channel to PaRAM Mapping**

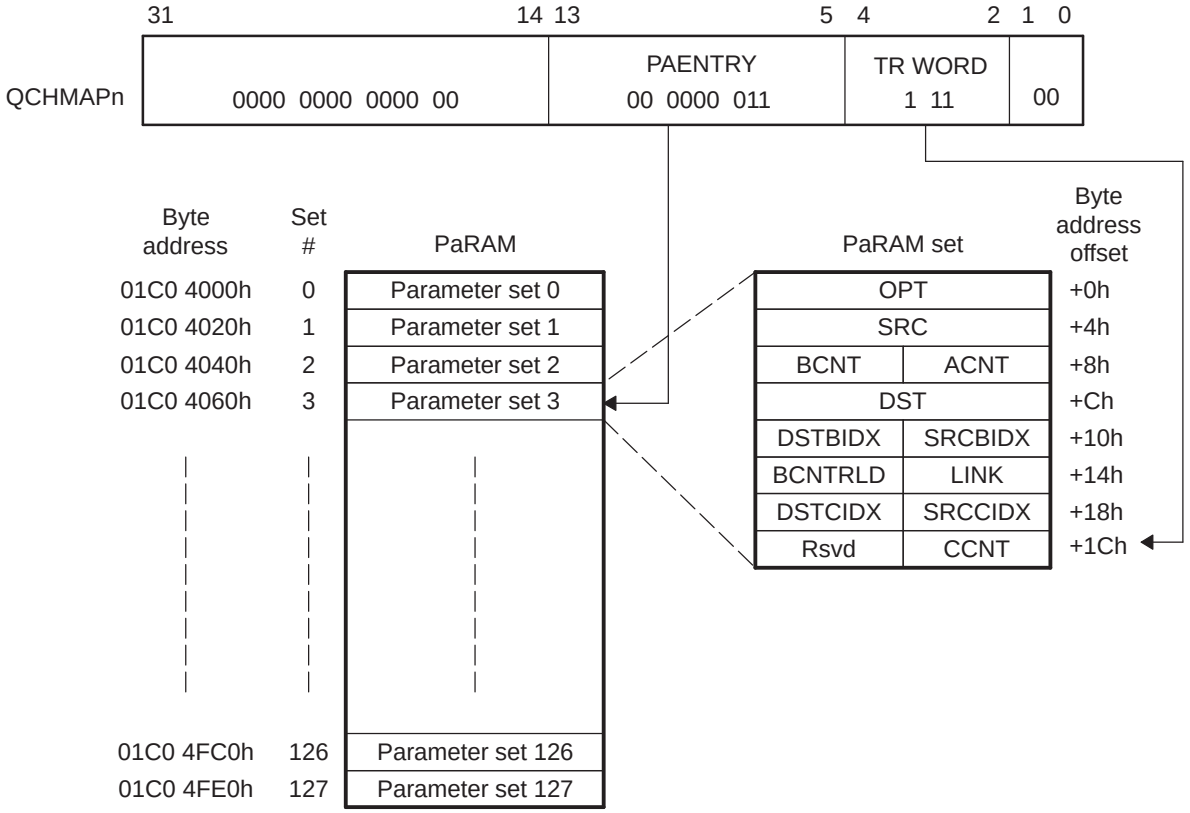
| PaRAM Set Number | Address | Mapping |
|---|---|---|
| PaRAM Set 0 | 01C0 4000h to 01C0 401Fh | DMA Channel 0/Reload/QDMA |
| PaRAM Set 1 | 01C0 4020h to 01C0 403Fh | DMA Channel 1/Reload/QDMA |
| PaRAM Set 2 | 01C0 4040h to 01C0 405Fh | DMA Channel 2/Reload/QDMA |
| PaRAM Set 3 | 01C0 4060h to 01C0 407Fh | DMA Channel 3/Reload/QDMA |
| PaRAM Set 4 | 01C0 4080h to 01C0 409Fh | DMA Channel 4/Reload/QDMA |
| PaRAM Set 5 | 01C0 40A0h to 01C0 40BFh | DMA Channel 5/Reload/QDMA |
| PaRAM Set 6 | 01C0 40C0h to 01C0 40DFh | DMA Channel 6/Reload/QDMA |
| PaRAM Set 7 | 01C0 40E0h to 01C0 40FFh | DMA Channel 7/Reload/QDMA |
| PaRAM Set 8 | 01C0 4100h to 01C0 411Fh | DMA Channel 8/Reload/QDMA |
| PaRAM Set 9 | 01C0 4120h to 01C0 413Fh | DMA Channel 9/Reload/QDMA |
| PaRAM Set 10 | 01C0 4140h to 01C0 415Fh | DMA Channel 10/Reload/QDMA |
| PaRAM Set 11 | 01C0 4160h to 01C0 417Fh | DMA Channel 11/Reload/QDMA |
| PaRAM Set 12 | 01C0 4180h to 01C0 419Fh | DMA Channel 12/Reload/QDMA |
| PaRAM Set 13 | 01C0 41A0h to 01C0 41BFh | DMA Channel 13/Reload/QDMA |
| PaRAM Set 14 | 01C0 41C0h to 01C0 41DFh | DMA Channel 14/Reload/QDMA |
| PaRAM Set 15 | 01C0 41E0h to 01C0 41FFh | DMA Channel 15/Reload/QDMA |
| PaRAM Set 16 | 01C0 4200h to 01C0 421Fh | DMA Channel 16/Reload/QDMA |
| ... | ... | ... |
| PaRAM Set 62 | 01C0 47C0h to 01C0 47DFh | DMA Channel 62/Reload/QDMA |
| PaRAM Set 63 | 01C0 47E0h to 01C0 47FFh | DMA Channel 63/Reload/QDMA |
| PaRAM Set 64 | 01C0 4800h to 01C0 481Fh | Reload/QDMA |
| PaRAM Set 65 | 01C0 4820h to 01C0 483Fh | Reload/QDMA |
| ... | ... | ... |
| PaRAM Set 126 | 01C0 4FC0h to 01C0 4FDFh | Reload/QDMA |
| PaRAM Set 127 | 01C0 4FE0h to 01C0 4FFFh | Reload/QDMA |

### 2.6.2 QDMA Channel to PaRAM Mapping

The mapping between the QDMA channels and the PaRAM sets is programmable .The QDMA channel *n* mapping register (QCHMAP*n*) in the EDMA3CC provides programmability for the QDMA channels to be mapped to any of the PaRAM sets in the PaRAM memory map. Figure 2-10 illustrates the use of QCHMAP.

Additionally, QCHMAP allows you to program the trigger word in the PaRAM set for the QDMA channel. A trigger word is one of the 8 words in the PaRAM set. For a QDMA transfer to occur, a valid TR synchronization event for EDMA3CC is a write to the trigger word in the PaRAM set pointed to by QCHMAP for a particular QDMA channel. By default, QDMA channels are mapped to PaRAM set 0. Care must be taken to appropriately remap PaRAM set 0 before it is used.

**Figure 2-10. QDMA Channel to PaRAM Mapping**




## 2.7 EDMA3 Channel Controller Regions

The EDMA3 channel controller (EDMA3CC) divides its address space into four regions. Individual channel resources are assigned to a specific region, where each region is typically assigned to a specific EDMA programmer. Application software is required to use the appropriate region.

### *2.7.1 Region Overview*

The EDMA3CC memory-mapped registers are divided in three main categories:

1. Global registers
2. Global region channel registers
3. Shadow region channel registers

The global registers are located at a single/fixed location in the EDMA3CC memory map. These registers control EDMA3 resource mapping and provide debug visibility and error tracking information. See the device-specific data manual for the EDMA3CC memory map.

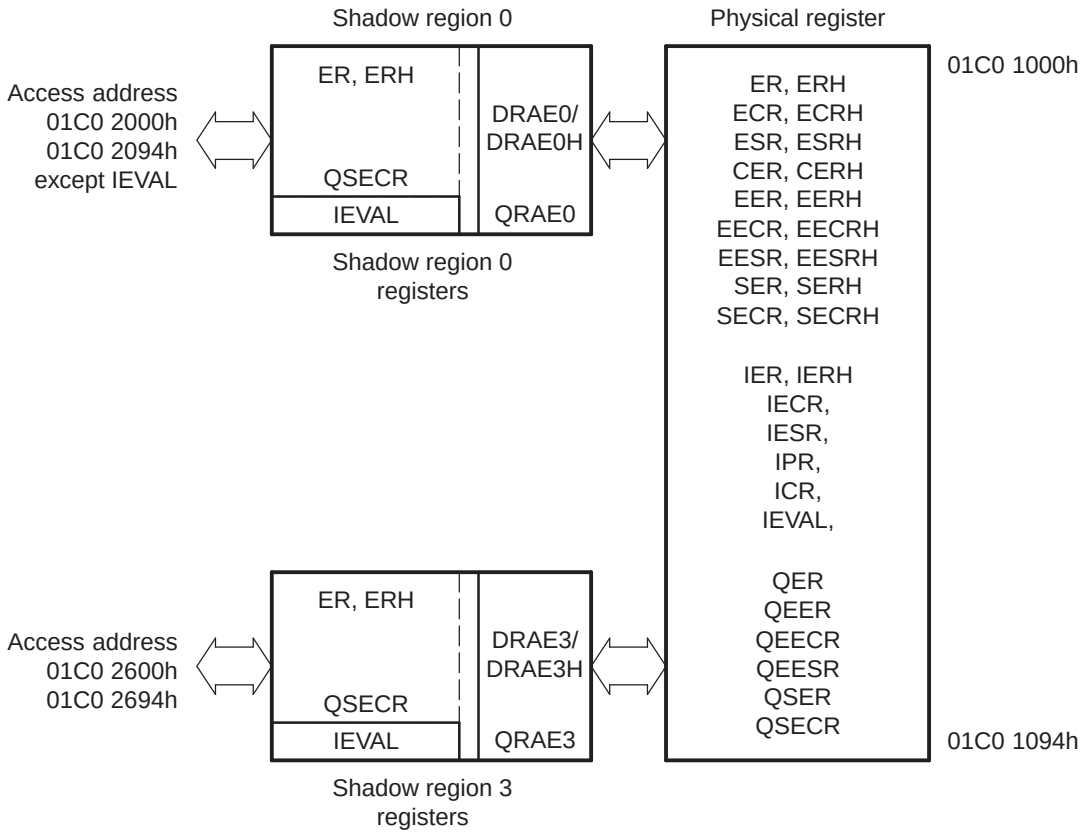
The channel registers (including DMA, QDMA, and interrupt registers) are accessible via the global channel region address range, or in the shadow *n* channel region address range(s). For example, the event enable register (EER) is visible at the global address of 01C0 1020h or region addresses of 01C0 2020h for region 0 and 01C0 2220h for region 1.

The underlying control register bits that are accessible via the shadow region address space (except for IEVAL*n*) are controlled by the DMA region access enable registers (DRAE*m*) and QDMA region access enable registers (QRAE*n*). Table 2-8 lists the registers in the shadow region memory-map. (See EDMA3CC memory-map figure for the complete global and shadow region memory-maps.) Figure 2-11 illustrates the conceptual view of the regions.

**Table 2-8. Shadow Region Registers**

| DRAE*m* | DRAEH*m* | QRAE*n* |
|---|---|---|
| ER | ERH | QER |
| ECR | ECRH | QEER |
| ESR | ESRH | QEECR |
| CER | CERH | QEESR |
| EER | EERH | |
| EECR | EECRH | |
| EESR | EESRH | |
| SER | SERH | |
| SECR | SECRH | |
| IER | IERH | |
| IECR | IECRH | |
| IESR | IESRH | |
| IPR | IPRH | |
| ICR | ICRH | |
| **Register not affected by DRAE\DRAEH** | | |
| IEVAL | | |

**Figure 2-11. Shadow Region Registers**



### 2.7.2 Channel Controller Regions

**NOTE:** Only shadow region 0 and shadow region 1 are useful. The remaining shadow regions and associated registers should not be used.

There are four EDMA3 shadow regions (and associated memory-maps). Associated with each shadow region are a set of registers defining which channels and interrupt completion codes belong to that region. These registers are user-programmed per region to assign ownership of the DMA/QDMA channels to a region.

- DRAE*m* and DRAEH*m*: One register pair exists for each of the shadow regions. The number of bits in each register pair matches the number of DMA channels (64 DMA channels). These registers need to be programmed to assign ownership of DMA channels and interrupt (or TCC codes) to the respective region. Accesses to DMA and interrupt registers via the shadow region address view are filtered through the DRAE/DRAEH pair. A value of 1 in the corresponding DRAE(H) bit implies that the corresponding DMA/interrupt channel is accessible; a value of 0 in the corresponding DRAE(H) bit forces writes to be discarded and returns a value of 0 for reads.
- QRAE*n*: One register exists for every region. The number of bits in each register matches the number of QDMA channels (8 QDMA channels). These registers must be programmed to assign ownership of QDMA channels to the respective region. To enable a channel in a shadow region using shadow region 0 QEER, the respective bit in QRAE must be set or writing into QEESR will not have the desired effect.

It is typical for an application to have a unique assignment of QDMA/DMA channels (and, therefore, a given bit position) to a given region.

The use of shadow regions allows for restricted access to EDMA3 resources (DMA channels, QDMA channels, TCC, interrupts) by tasks in a system by setting or clearing bits in the DRAE/QRAE registers. If exclusive access to any given channel/TCC code is required for a region, then only that region's DRAE/QRAE should have the associated bit set.

#### *Example 2-1. Resource Pool Division Across Two Regions*

This example illustrates a judicious resource pool division across two regions, assuming region 0 must be allocated 16 DMA channels (0-15) and 1 QDMA channel (0) and 32 TCC codes (0-15 and 48-63). Region 1 needs to be allocated 16 DMA channels (16-32) and the remaining 3 QDMA channels (1-3) and TCC codes (16-47). DRAE should be equal to the OR of the bits that are required for the DMA channels and the TCC codes:

```
Region 0: DRAEH, DRAE = 0xFFFF0000, 0x0000FFFF QRAE = 0x0000001 Region 1: DRAEH, DRAE = 0x0000FFFF,
0xFFFF0000 QRAE = 0x000000E
```

## 2.8 Chaining EDMA3 Channels

The channel chaining capability for the EDMA3 allows the completion of an EDMA3 channel transfer to trigger another EDMA3 channel transfer. The purpose is to allow you the ability to chain several events through one event occurrence.

Chaining is different from linking (Section 2.3.7). The EDMA3 link feature reloads the current channel parameter set with the linked parameter set. The EDMA3 chaining feature does not modify or update any channel parameter set; it provides a synchronization event to the chained channel (see Section 2.4.1.3 for chain-triggered transfer requests).

Chaining is achieved at either final transfer completion or intermediate transfer completion, or both, of the current channel. Consider a channel *m* (DMA/QDMA) required to chain to channel *n*. Channel number *n* (0-63) needs to be programmed into the TCC field of channel *m* channel options parameter (OPT) set.

- If final transfer completion chaining (TCCHEN = 1 and ITCCHEN = 0 in channel *m* OPT) is enabled, the chain-triggered event occurs after the *last* transfer request of channel *m* is submitted (early completion) or completed (normal completion).
- If intermediate transfer completion chaining (ITCCHEN = 1 and ITCCHEN = 0 in channel *m* OPT) is enabled, the chain-triggered event occurs after every *intermediate* transfer request of channel *m* is submitted (early completion) or completed (normal completion).
- If both final and intermediate transfer completion chaining (TCCHEN = 1 and ITCCHEN = 1 in channel *m* OPT) are enabled, the chain-trigger event occurs after *every* transfer request of channel *m* is submitted (early completion) or completed (normal completion).

Table 2-9 shows the number of chain event triggers occurring in different synchronized scenarios. Consider channel 31 programmed with ACNT = 3, BCNT = 4, CCNT = 5, and TCC = 30.

**Table 2-9. Chain Event Triggers**

| | (Number of chained event triggers on channel 30) | |
|---|---|---|
| **Options** | **A-Synchronized** | **AB-Synchronized** |
| TCCHEN = 1, ITCCHEN = 0 | 1 (Last TR) | 1 (Last TR) |
| TCCHEN = 0, ITCCHEN = 1 | 19 (All but the last TR) | 4 (All but the last TR) |
| TCCHEN = 1, ITCCHEN = 1 | 20 (All TRs) | 5 (All TRs) |

## 2.9 EDMA3 Interrupts

The EDMA3 interrupts are divided into 2 categories:

- Transfer completion interrupts
- Error interrupts

The transfer completion interrupts are listed in Table 2-10 and the error interrupts are listed in Table 2-11.

**Table 2-10. EDMA3 Transfer Completion Interrupt Numbers**

| Name | Description | DSPINTC |
|---|---|---|
| EDMA3CC_GINT | EDMA3CC Global Transfer Completion Interrupt | 34 |
| EDMA3CC_INT0 | EDMA3CC Transfer Completion Interrupt Shadow Region 0 | 35 |
| EDMA3CC_INT1 | EDMA3CC Transfer Completion Interrupt Shadow Region 1 | 36 |

**Table 2-11. EDMA3 Error Interrupt Numbers**

| Name | Description | DSPINTC |
|---|---|---|
| EDMA3CC_ERRINT | EDMA3CC Error Interrupt | 37 |
| EDMA3TC_ERRINT0 | TC0 Error Interrupt | 38 |
| EDMA3TC_ERRINT1 | TC1 Error Interrupt | 39 |
| EDMA3TC_ERRINT2 | TC2 Error Interrupt | 40 |

### *2.9.1 Transfer Completion Interrupts*

The EDMA3CC is responsible for generating transfer completion interrupts to the CPU. The EDMA3 generates a single completion interrupt per shadow region and one for the global region on behalf of all DMA/QDMA channels. Various control registers and bit fields facilitate EDMA3 interrupt generation. For a given DMA/QDMA channel, the software architecture should either use the global region completion interrupt or the shadow region completion interrupt, but not both.

The transfer completion code (TCC) value is directly mapped to the bits of the interrupt pending register (IPR/IPRH), as shown in Table 2-12. For example, if TCC = 10 0001b, IPRH[1] is set after transfer completion, and results in an interrupt generation to the CPU if the completion interrupt is enabled for the CPU. See Section 2.9.1.1 for details on enabling EDMA3 transfer completion interrupts.

When a completion code is returned (as a result of early or normal completion), the corresponding bit in IPR/IPRH is set. For the completion code to be returned, the PaRAM set associated with the transfer must enable the transfer completion interrupt (final/intermediate) in the channel options parameter (OPT).

The transfer completion code (TCC) can be programmed to any value for a DMA/QDMA channel. There does not need to be a direct relation between the channel number and the transfer completion code value. This allows multiple channels having the same transfer completion code value to cause a CPU to execute the same interrupt service routine (ISR) for different channels.

If the channel is used in the context of a shadow region and you intend for the shadow region interrupt to be asserted, then ensure that the bit corresponding to the TCC code is enabled in IER/IERH and in the corresponding shadow regions's DMA region access registers (DRAE/DRAEH).

**Table 2-12. Transfer Complete Code (TCC) to EDMA3CC Interrupt Mapping**

| TCC Bits in OPT (TCINTEN/ITCINTEN = 1) | IPR Bit Set | TCC Bits in OPT (TCINTEN/ITCINTEN = 1) | IPRH Bit Set (1) |
|---|---|---|---|
| 00 0000b | IPR0 | 10 0000b | IPR32/IPRH0 |
| 00 0001b | IPR1 | 10 0001b | IPR33/IPRH1 |
| 00 0010b | IPR2 | 10 0010b | IPR34/IPRH2 |
| 00 0011b | IPR3 | 10 0011b | IPR35/IPRH3 |
| 00 0100b | IPR4 | 10 0100b | IPR36/IPRH4 |
| … | … | … | … |
| … | … | … | … |
| 01 1110b | IPR30 | 11 1110b | IPR62/IPRH30 |
| 01 1111b | IPR31 | 11 1111b | IPR63/IPRH31 |

(1) Bit fields IPR[32-63] correspond to bits 0 to 31 in IPRH, respectively.

You can enable interrupt generation at either final transfer completion or intermediate transfer completion, or both. Consider channel *m* as an example.

- If the final transfer interrupt (TCINTEN = 1 and ITCINTEN = 0 in OPT) is enabled, the interrupt occurs after the *last* transfer request of channel *m* is either submitted or completed (depending on early or normal completion).
- If the intermediate transfer interrupt (TCINTEN = 0 and ITCINTEN = 1 in OPT) is enabled, the interrupt occurs after every *intermediate* transfer request of channel *m* is either submitted or completed (depending on early or normal completion).
- If both final and intermediate transfer completion interrupts (TCINTEN = 1 and ITCINTEN = 1 in OPT) are enabled, the interrupt occurs after *every* transfer request of channel *m* is submitted or completed (depending on early or normal completion).

Table 2-13 shows the number of interrupts occurring in different synchronized scenarios. Consider channel 31 programmed with ACNT = 3, BCNT = 4, CCNT = 5, and TCC = 30.

**Table 2-13. Number of Interrupts**

| Options | A-Synchronized | AB-Synchronized |
|---|---|---|
| TCINTEN = 1, ITCINTEN = 0 | 1 (Last TR) | 1 (Last TR) |
| TCINTEN = 0, ITCINTEN = 1 | 19 (All but the last TR) | 4 (All but the last TR) |
| TCINTEN = 1, ITCINTEN = 1 | 20 (All TRs) | 5 (All TRs) |

#### 2.9.1.1 Enabling Transfer Completion Interrupts

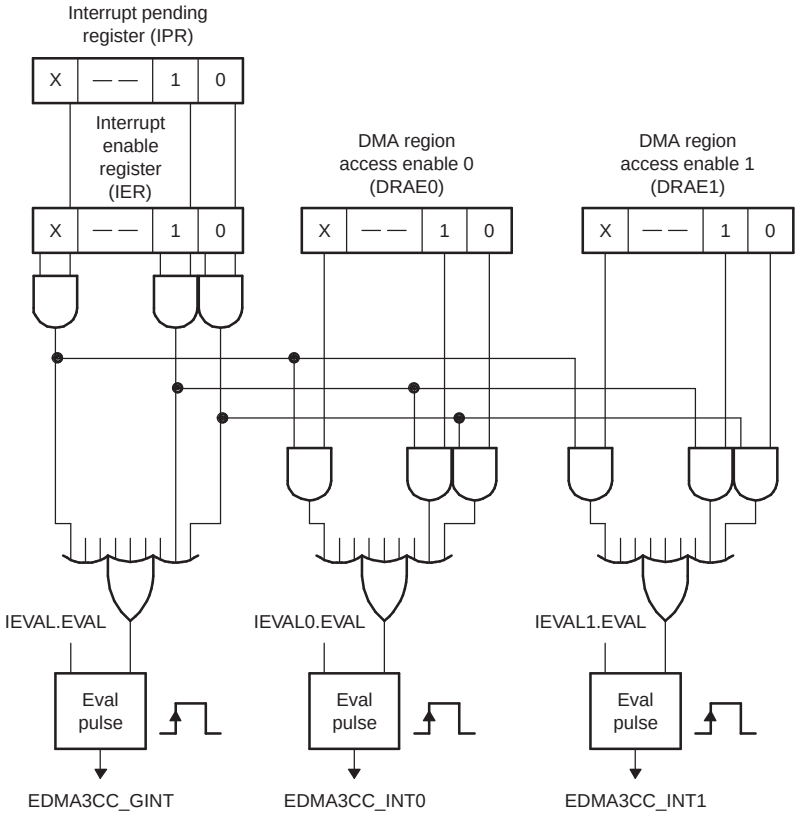
For the EDMA3 channel controller to assert a transfer completion to the external world, the interrupts have to be enabled in the EDMA3CC. This is in addition to setting up the TCINTEN and ITCINTEN bits in OPT of the associated PaRAM set.

The EDMA3 channel controller has interrupt enable registers (IER/IERH) and each bit location in IER/IERH serves as a primary enable for the corresponding interrupt pending registers (IPR/IPRH).

All the interrupt registers (IER, IESR, IECR, and IPR) are either manipulated from the global DMA channel region or by way of the DMA channel shadow regions. The shadow regions provide a view to the same set of physical registers that are in the global region.

The EDMA3 channel controller has a hierarchical completion interrupt scheme that makes use of a single set of interrupt pending registers (IPR/IPRH) and single set of interrupt enable registers (IER/IERH). A second level of interrupt masking is provided by the programmable DMA region access enable registers (DRAE/DRAEH). The global region interrupt output is gated based on the enable mask that is provided by IER/IERH. See Figure 2-12.

**Figure 2-12. Interrupt Diagram**




For the EDMA3CC to generate the transfer completion interrupts that are associated with each shadow region, the following conditions must be true:

- EDMA3CC_INT0: (IPR.E0 & IER.E0 & DRAE0.E0) | (IPR.E1 & IER.E1 & DRAE0.E1) | …|(IPRH.E63 & IERH.E63 & DRAHE0.E63)
- EDMA3CC_INT1: (IPR.E0 & IER.E0 & DRAE1.E0) | (IPR.E1 & IER.E1 & DRAE1.E1) | …| (IPRH.E63 & IERH.E63 & DRAHE1.E63)

**NOTE:** The DRAE/DRAEH for all regions are expected to be set up at system initialization and to remain static for an extended period of time. The interrupt enable registers should be used for dynamic enable/disable of individual interrupts.

Because there is no relation between the TCC value and the DMA/QDMA channel, it is possible, for example, for DMA channel 0 to have the OPT.TCC = 63 in its associated PaRAM set. This would mean that if a transfer completion interrupt is enabled (OPT.TCINTEN or OPT.ITCINTEN is set), then based on the TCC value, IPRH.E63 is set up on completion. For proper channel operations and interrupt generation using the shadow region map, you must program the DRAE/DRAEH that is associated with the shadow region to have read/write access to both bit 0 (corresponding to channel 0) and bit 63 (corresponding to IPRH bit that is set upon completion).

#### 2.9.1.2 Clearing Transfer Completion Interrupts

Transfer completion interrupts that are latched to the interrupt pending registers (IPR/IPRH) are cleared by writing a 1 to the corresponding bit in the interrupt pending clear register (ICR/ICRH). For example, a write of 1 to ICR.E0 clears a pending interrupt in IPR.E0.

If an incoming transfer completion code (TCC) gets latched to a bit in IPR/IPRH, then additional bits that get set due to a subsequent transfer completion will not result in asserting the EDMA3CC completion interrupt. In order for the completion interrupt to be pulsed, the required transition is from a state where no enabled interrupts are set to a state where at least one enabled interrupt is set.

### *2.9.2 EDMA3 Interrupt Servicing*

On completion of a transfer (early or normal completion), the EDMA3 channel controller sets the appropriate bit in the interrupt pending registers (IPR/IPRH) as specified by the transfer completion codes. If the completion interrupts are appropriately enabled, then the CPU enters the interrupt service routine (ISR) when the completion interrupt is asserted. Since there is a single completion interrupt for all DMA/QDMA channels.

After servicing the interrupt, the ISR should clear the corresponding bit in IPR/IPRH; therefore, enabling recognition of future interrupts. Only when all IPR/IPRH bits are cleared, the EDMA3CC will assert additional completion interrupts.

It is possible that when one interrupt is serviced; many other transfer completions result in additional bits being set in IPR/IPRH, thereby resulting in additional interrupts. It is likely that each of these bits in IPR/IPRH would need different types of service; therefore, the ISR must check all pending interrupts and continue until all the posted interrupts are appropriately serviced.

Following are examples (pseudo code) for a CPU interrupt service routine for an EDMA3CC completion interrupt.

The ISR routine in Example 2-2 is more exhaustive and incurs a higher latency.

***Example 2-2. Interrupt Servicing***

The pseudo code:

1. Read the interrupt pending register (IPR/IPRH).
2. Perform the operations needed.
3. Write to the interrupt pending clear register (ICR/ICRH) to clear the corresponding IPR/IPRH bit.
4. Read IPR/IPRH again:
   (a) If IPR/IPRH is not equal to 0, repeat from step 2 (implies occurrence of new event between step 2 to step 4).
   (b) If IPR/IPRH is equal to 0, this should assure you that all enabled interrupts are inactive.

**NOTE:** It is possible that during step 4, an event occurs while the IPR/IPRH bits are read to be 0 and the application is still in the interrupt service routine. If this happens, a new interrupt is recorded in the device interrupt controller and a new interrupt is generated as soon as the application exits the interrupt service routine.

Example 2-3 is less rigorous, with less burden on the software in polling for set interrupt bits, but can occasionally cause a race condition, as mentioned above.

***Example 2-3. Interrupt Servicing***

If it is desired to leave any enabled and pending (possibly lower priority) interrupts, it is required to force the interrupt logic to reassert the interrupt pulse by setting the EVAL bit in the interrupt evaluation register (IEVAL).

The pseudo code:

1. Enter ISR.
2. Read IPR/IPRH.
3. For the condition set in IPR/IPRH that you desire to service:
   (a) Service interrupt as required by application.
   (b) Clear bit for serviced conditions (others may still be set, and other transfers may have resulted in returning the TCC to EDMA3CC after step 2).
4. Read IPR/IPRH prior to exiting ISR:
   (a) If IPR/IPRH is equal to 0, then exit ISR.
   (b) If IPR/IPRH is not equal to 0, then set IEVAL so that upon exit of ISR, a new interrupt is triggered if any enabled interrupts are still pending.

The EVAL bit must not be set when IPR and IPRH are read to be 0, to avoid generation of extra interrupt pulses.

**NOTE:** Since the DMA region access registers (DRAE/DRAEH) are required to enable the transfer completion region interrupts, it is assumed that there will be a unique and nonoverlapping (in most cases) assignment of the channels and interrupts among the different shadow regions. This allows the interrupt registers (IER, IESR, IECR, IPR, and ICR) in the different shadow regions to functionally operate in an independent manner and nonoverlapping. The above examples for the interrupt service routine is based on this assumption.

### 2.9.3 Interrupt Evaluation Operations

The EDMA3CC has interrupt evaluate registers (IEVAL) in each shadow region. These registers are the only registers in the DMA channel shadow region memory map that are not affected by the settings for the DMA region access enable registers (DRAE/DRAEH). A write of 1 to the EVAL bit in these registers associated with a particular shadow region results in pulsing the associated region interrupt, if any enabled interrupt (via IER/IERH) is still pending (IPR/IPRH). This register can be used in order to assure that the interrupts are not missed by the CPU (or the EDMA3 master associated with the shadow region) if the software architecture chooses not to use all interrupts. See Example 2-3 for the use of IEVAL in the EDMA3 interrupt service routine (ISR).

Similarly an error evaluate register (EEVAL) exists in the global region. A write of 1 to the EVAL bit in EEVAL causes the pulsing of the error interrupt if any pending errors are in EMR/EMRH, QEMR, or CCERR. See Section 2.9.4 for additional details on error interrupts.

**NOTE:** While using IEVAL for shadow region completion interrupts, you should make sure that the IEVAL operated upon is from that particular shadow region memory map.

### 2.9.4 Error Interrupts

The EDMA3CC error registers provide the capability to differentiate error conditions (event missed, threshold exceed, etc.). Additionally, if the error bits are set in these registers, it results in asserting the EDMA3CC error interrupt. If EDMA3CC error interrupt is enabled in the device interrupt controller, then it allows the CPU to handle the error conditions.

The EDMA3CC has a single error interrupt (EDMA3CC_ERRINT) that gets asserted for all EDMA3CC error conditions. There are four conditions that cause the error interrupt to be pulsed:

- DMA missed events: for all 64 DMA channels. These get latched in the event missed registers (EMR/EMRH).
- QDMA missed events: for all QDMA channels. These get latched in the QDMA event missed register (QEMR).
- Threshold exceed: for all event queues. These get latched in EDMA3CC error register (CCERR).
- TCC error: for outstanding transfer requests expected to return completion code (TCCHEN or TCINTEN bit in OPT is set to 1) exceeding the maximum limit of 63. This also gets latched in the EDMA3CC error register (CCERR).

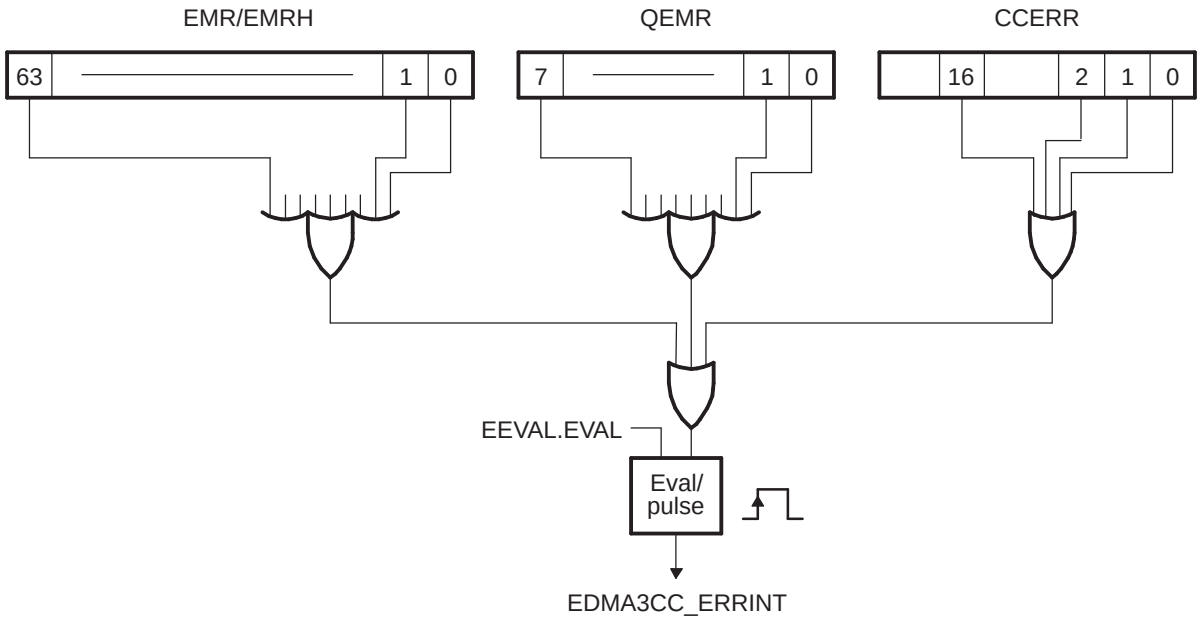
Figure 2-13 illustrates the EDMA3CC error interrupt generation operation.

If any of the bits are set in the error registers due to any error condition, the EDMA3CC_ERRINT always is asserted, as there are no enables for masking these error events. Similar to transfer completion interrupts (EDMA3CC_INT), the error interrupt also is pulsed only when the error interrupt condition transitions from a state where no errors are set to a state where at least one error bit is set. If additional error events are latched prior to the original error bits being cleared, the EDMA3CC does not generate additional interrupt pulses.

To reduce the burden on the software, similar to the interrupt evaluate register (IEVAL), there is an error evaluate register (EEVAL) that allows reevaluation of pending set error events/bits. This can be used so that the CPU(s) does not miss any error events.

**NOTE:** It is a good practice to have the error interrupt enabled in the device interrupt controller and associate an interrupt service routine with it to address the various error conditions appropriately. This puts less burden on software (polling for error status) and additionally provides a good debug mechanism for unexpected error conditions.

**Figure 2-13. Error Interrupt Operation**



## 2.10 Event Queue(s)

Event queues are a part of the EDMA3 channel controller. Event queues form the interface between the event detection logic in the EDMA3CC and the transfer request (TR) submission logic of the EDMA3CC. Each queue is 16 entries deep, that is, a maximum of 16 queued events per event queue. If there are more than 16 events, then the events that cannot find a place in the event queue remain set in the associated event register.

There are three event queues (Q0, Q1, and Q2) for the DM643x devices. Events in Q0 result in submission of its associated transfer requests (TRs) to TC0. Similarly, transfer requests associated with events in Q1 are submitted to TC1 and transfer requests associated with events in Q2 are submitted to TC2.

An event that wins prioritization against other DMA and/or QDMA pending events is placed at the end of the appropriate event queue. Each event queue is serviced in a FIFO (first in–first out) order. Once the event reaches the head of its queue and the corresponding transfer controller is ready to receive another TR, the event is dequeued and the PaRAM set corresponding to the dequeued event is processed and submitted as a transfer request packet (TRP) to the associated EDMA3 transfer controller.

A lower numbered queue has a higher dequeuing priority then a higher numbered queue. For example, Q0 has higher priority than Q1, if Q0 and Q1 both have at least one event entry and if both TC0 and TC1 can accept transfer requests, then the event in Q0 is dequeued first and its associated PaRAM set is processed and submitted as a transfer request (TR) to TC0.

See Section 2.10.4 for system-level performance considerations. All the event entries in all the event queues are software readable (not writeable) by accessing the event queue entry registers (Q0E0 to Q2E15). Each event entry register characterizes the queued event in terms of the type of event (manual, event, chained or autotriggered) and the event number. See Section 4.3.4.1 for a description of the bit fields in the queue event entry registers.

### *2.10.1 DMA/QDMA Channel to Event Queue Mapping*

Each of the 64 DMA channels and 8 QDMA channels are independently programmed to map to a specific queue using the DMA queue number register *n* (DMAQNUM*n*) and the QDMA channel queue number register (QDMANUM). The mapping of DMA/QDMA channels is critical to achieving the desired performance level for the EDMA and most importantly in meeting real-time deadlines. See Section 2.10.4.

**NOTE:** If an event is ready to be queued and both the event queue and the EDMA3 transfer controller associated to the event queue are empty, then the event bypasses the event queue, and goes to the PaRAM processing logic and eventually to the transfer request submission logic for submission to the EDMA3TC. In this case, the event is not logged in the event queue status registers.

### 2.10.2 Queue RAM Debug Visibility

Each event queue has 16 entries. These 16 entries are managed in a circular FIFO manner. All event queue entries for all event queues are software readable by the event queue entry register (Q*x*E*x*). Additionally, for each queue there is a queue status register (QSTAT*n*).

These registers provide user visibility and may be helpful while debugging real-time issues (typically post-mortem), involving multiple events and event sources. The event queue entry register (Q*x*E*x*) uniquely identifies the specific event type (event-triggered, manually-triggered, chain-triggered, and QDMA events) along with the event number (for DMA/QDMA channels) that are in the queue or have been de-queued (passed through the queue). QSTAT*n* includes fields for the start pointer (STRTPTR) that provides the offset to the head entry of an event. It also includes a NUMVAL field that provides the total number of valid entries residing in the event queue at a given instance of time. The STRTPTR field may be used to index appropriately into the 16 event entries. The NUMVAL number of entries starting from STRTPTR are indicative of events still queued in the respective queue. The remaining entries may be read to determine which events have already been de-queued and submitted to the associated transfer controller.

### 2.10.3 Queue Resource Tracking

The EDMA3CC event queue includes watermarking/threshold logic that allows you to keep track of maximum usage of all event queues. This is useful for debugging real-time deadline violations that may result from head-of-line blocking on a given EDMA3 event queue.

You can program the maximum number of events that can queue up in an event queue by programming the threshold value (between 0 to 15) in the queue watermark threshold A register (QWMTHRA). The maximum queue usage is recorded actively in the watermark (WM) field of the queue status register (QSTAT*n*) that keeps getting updated based on a comparison of number of valid entries, which is also visible in the NUMVAL bit in QSTAT*n* and the maximum number of entries (WM bit in QSTAT*n*).

If the queue usage is exceeded, this status is visible in the EDMA3CC registers: the QTHRXCD*n* bit in the channel controller error register (CCERR) and the THRXCD bit in QSTAT*n*, where *n* stands for the event queue number. Any bits that are set in CCERR also generate an EDMA3CC error interrupt.

### 2.10.4 Performance Considerations

The main switched central resource (SCR) (see the data manual) arbitrates bus requests from all the masters (DSP, master peripherals, VPSS, and the EDMA3 transfer controllers) to the shared slave resources (peripherals and memories).

The priorities of transfer requests (read and write commands) from the EDMA3 transfer controllers with respect to other masters within the system are programmed using the queue priority register (QUEPRI). QUEPRI programs the priority of the event queues (or indirectly, TC0-TC2, since Queue*n* transfer requests are submitted to TC*n*).

Therefore, the priority of unloading queues has a secondary affect compared to the priority of the transfers as they are executed by the EDMA3TC (dictated by the priority set using QUEPRI).

## 2.11 EDMA3 Transfer Controller (EDMA3TC)

The EDMA3 channel controller is the user-interface of the EDMA3 and the EDMA3 transfer controller (EDMA3TC) is the data movement engine of the EDMA3. The EDMA3CC submits transfer requests (TR) to the EDMA3TC and the EDMA3TC performs the data transfers dictated by the TR; thus, the EDMA3TC is a slave to the EDMA3CC.

### *2.11.1 Architecture Details*

#### 2.11.1.1 Command Fragmentation

The TC read and write controllers in conjunction with the source and destination register sets are responsible for issuing optimally-sized reads and writes to the slave endpoints. An optimally-sized command is defined by the transfer controller default burst size (DBS). See Section 2.11.4 for the DBS value of each EDMA3TC.

The EDMA3TC attempts to issue the largest possible command size as limited by the DBS value or the ACNT/BCNT value of the TR. EDMA3TC obeys the following rules:

- The read/write controllers always issue commands less than or equal to the DBS value.
- The first command of a 1D transfer is always issued so that subsequent commands align to the DBS value.

Example 2-4 shows the command fragmentation for a DBS of 32 bytes. In summary, if the ACNT value is larger than the DBS value, then the EDMA3TC breaks the ACNT array into DBS-sized commands to the source/destination addresses. Each BCNT number of arrays are then serviced in succession.

***Example 2-4. Command Fragmentation (DBS = 32)***

The pseudo code:

1. ACNT = 8, BCNT = 8, SRCBIDX = 8, DSTBIDX = 10, SRCADDR = 64, DSTADDR = 191
   Read Controller: This is optimized from a 2D-transfer to a 1D-transfer such that the read side is equivalent to ACNT = 64, BCNT = 1.
   Cmd0 = 32 byte, Cmd0 = 32 byte
   Write Controller: Since DSTBIDX != ACNT, it is not optimized.
   Cmd0 = 8 byte, Cmd1 = 8 byte, Cmd2 = 8 byte, Cmd3 = 8 byte, Cmd4 = 8 byte, Cmd5 = 8 byte, Cmd6 = 8 byte, Cmd7 = 8 byte.
2. ACNT=64, BCNT = 1, SRCADDR = 31, DSTADDR = 513
   Read Controller: Read address is not aligned.
   Cmd0 = 1 byte, (now the SRCADDR is aligned to 32 for the next command)
   Cmd1 = 32 bytes
   Cmd2 = 31 bytes
   Write Controller: The write address is also not aligned.
   Cmd0 = 31 bytes, (now the DSTADDR is aligned to 32 for the next command)
   Cmd1 = 32 bytes
   Cmd2 = 1 byte

For BCNT arrays of ACNT bytes (that is, a 2D transfer), if the ACNT value is less than or equal to the DBS value, then the TR may be optimized into a 1D transfer in order to maximize efficiency. The optimization takes place if the EDMA3TC recognizes that the 2D transfer is organized as a single dimension (SAM/DAM = 0, increment), BIDX = ACNT, the ACNT value is a power of 2, and the BCNT value is less than or equal to 1023.

Table 2-14 summarizes the conditions in which the optimizations are performed.

**Table 2-14. Read/Write Command Optimization Rules**

| ACNT ≤ DBS | ACNT is power of 2 | BIDX = ACNT | BCNT ≤ 1023 | SAM/DAM = 0 (Increment) | Description |
|---|---|---|---|---|---|
| Yes | Yes | Yes | Yes | Yes | Optimized |
| No | x | x | x | x | Not Optimized |
| x | No | x | x | x | Not Optimized |
| x | x | No | x | x | Not Optimized |
| x | x | x | No | x | Not Optimized |
| x | x | x | x | No | Not Optimized |

#### 2.11.1.2 TR Pipelining

TR pipelining refers to the ability of the source active set to get ahead of the destination active set. Essentially, the reads for a given TR may already be in progress while the writes of a previous TR may not have completed.

The number of outstanding TRs is limited by the number of destination FIFO register entries. A single TR must be assured to target a single source peripheral endpoint.

TR pipelining is useful for maintaining throughput on back-to-back small TRs. It eliminates the read overhead because reads start in the background of a previous TR writes.

#### 2.11.1.3 Performance Tuning

By default, reads are as issued as fast as possible. In some cases, the reads issued by the EDMA3TC could fill the available command buffering for a slave, delaying other (potentially higher priority) masters from successfully submitting commands to that slave. The rate at which read commands are issued by the EDMA3TC is controlled by the read command rate register (RDRATE). RDRATE defines the number of cycles that the EDMA3TC read controller waits before issuing subsequent commands for a given TR, thus minimizing the chance of the EDMA3TC consuming all available slave resources. The RDRATE value should be set to a relatively small value if the transfer controller is targeted for high priority transfers and to a higher value if the transfer controller is targeted for low priority transfers.

In contrast, the Write Interface does not have any performance turning knobs because writes always have an interval between commands as write commands are submitted along with the associated write data.

### *2.11.2 Error Generation*

Errors are generated if enabled under three conditions:

- EDMA3TC detection of an error signaled by the source or destination address.
- Attempt to read or write to an invalid address in the configuration memory map.
- Detection of a constant addressing mode TR violating the constant addressing mode transfer rules (the source/destination addresses and source/destination indexes must be aligned to 32 bytes).

Either or all error types may be disabled. If an error bit is set and enabled, the error interrupt for the concerned transfer controller is pulsed.

### 2.11.3 Debug Features

The DMA program register set, DMA source active register set, and the destination FIFO register set are used to derive a brief history of TRs serviced through the transfer controller.

Additionally, the EDMA3TC status register (TCSTAT) has dedicated bit fields to indicate the ongoing activity within different parts of the transfer controller:

- The SRCACTV bit indicates whether the source active set is active.
- The DSTACTV bit indicates the number of TRs resident in the destination register active set at a given instance.
- The PROGBUSY bit indicates whether a valid TR is present in the DMA program set.

If the TRs are in progression, caution must be used and you must realize that there is a chance that the values read from the EDMA3TC status registers will be inconsistent since the EDMA3TC may change the values of these registers due to ongoing activities.

It is recommended that you ensure no additional submission of TRs to the EDMA3TC in order to facilitate ease of debug.

#### 2.11.3.1 Destination FIFO Register Pointer

The destination FIFO register pointer is implemented as a circular buffer with the start pointer being DFSTRTPTR and a buffer depth of usually 2 or 4. The EDMA3TC maintains two important status details in TCSTAT that may be used during advanced debugging, if necessary. The DFSTRTPTR is a start pointer, that is, the index to the head of the destination FIFO register. The DSTACTV is a counter for the number of valid (occupied) entries. These registers may be used to get a brief history of transfers.

Examples of some register field values and their interpretation:

- DFSTRTPTR = 0 and DSTACTV = 0 implies that no TRs are stored in the destination FIFO register.
- DFSTRTPTR = 1 and DSTACTV = 2h implies that two TRs are present. The first pending TR is read from the destination FIFO register entry 1 and the second pending TR is read from the destination FIFO register entry 2.
- DFSTRTPTR = 3h and DSTACTV = 2h implies that two TRs are present. The first pending TR is read from the destination FIFO register entry 3 and the second pending TR is read from the destination FIFO register entry 0.

### 2.11.4 EDMA3TC Configuration

The EDMA transfer controller default burst size configuration register (EDMATCCFG) in the System module configures the default burst size for the EDMA transfer controllers (EDMA3TC0, EDMA3TC1, and EDMA3TC2). Refer to the device-specific data manual for more information on this register. Table 2-15 provides the configuration of the individual EDMA3 transfer controllers present on the device. The DBS for each transfer controller is configurable by EDMATCCFG to 16, 32, or 64 bytes. The default values for DBS have been chosen for optimal performance in most intended-use conditions; therefore, if you decide to use nondefault values for DBS, you should evaluate the impact on performance. It is expected that the DBS value for a transfer controller is static and should be based on the application requirement. It is not recommended that the DBS value be changed on-the-fly.

**Table 2-15. EDMA3 Transfer Controller Configurations**

| Name | TC0 | TC1 | TC2 |
|---|---|---|---|
| FIFOSIZE | 128 bytes | 256 bytes | 128 bytes |
| BUSWIDTH | 8 bytes | 8 bytes | 8 bytes |
| DSTREGDEPTH | 4 entries | 4 entries | 4 entries |
| DBS (default) | 16 bytes | 32 bytes | 64 bytes |

## 2.12 Event Dataflow

This section summarizes the data flow of a single event, from the time the event is latched to the channel controller to the time the transfer completion code is returned. The following steps list the sequence of EDMA3CC activity:

1. Event is asserted from an external source (peripheral or external interrupt). This also is similar for a manually-triggered, chained-triggered, or QDMA-triggered event. The event is latched into the ER.E*n*/ERH.E*n* (or CER.E*n*/CERH.E*n*, ESR.E*n* /ESRH.E*n*, QER.E*n*) bit.
2. Once an event is prioritized and queued into the appropriate event queue, the SER.E*n*\SERH.E*n* (or QSER.E*n*) bit is set to inform the event prioritization/processing logic to disregard this event since it is already in the queue. Alternatively, if the transfer controller and the event queue are empty, then the event bypasses the queue.
3. The EDMA3CC processing and the submission logic evaluates the appropriate PaRAM set and determines whether it is a non-null and non-dummy transfer request (TR).
4. The EDMA3CC clears the ER.E*n*/ERH.E*n* (or CER.E*n*/CERH.E*n*, ESR.E*n*/ESRH.E*n*, QER.E*n*) bit and the SER.E*n*/SERH.E*n* bit as soon as it determines the TR is non-null. In the case of a null set, the SER.E*n*/SERH.E*n* bit remains set. It submits the non-null/non-dummy TR to the associated transfer controller. If the TR was programmed for early completion, the EDMA3CC immediately sets the interrupt pending register (IPR.I[TCC]/IPRH.I[TCC]-32).
5. If the TR was programmed for normal completion, the EDMA3CC sets the interrupt pending register (IPR.I[TCC]/IPRH.I[TCC]) when the EDMA3TC informs the EDMA3CC about completion of the transfer (returns transfer completion codes).
6. The EDMA3CC programs the associated EDMA3TC*n*'s Program Register Set with the TR.
7. The TR is then passed to the Source Active set and the Dst FIFO Register Set, if both the register sets are available.
8. The Read Controller processes the TR by issuing read commands to the source slave endpoint. The Read Data lands in the Data FIFO of the EDMA3TC*n*.
9. As soon as sufficient data is available, the Write Controller begins processing the TR by issuing write commands to the destination slave endpoint.
10. This continues until the TR completes and the EDMA3TC*n* then signals completion status to the EDMA3CC.

## 2.13 EDMA3 Prioritization

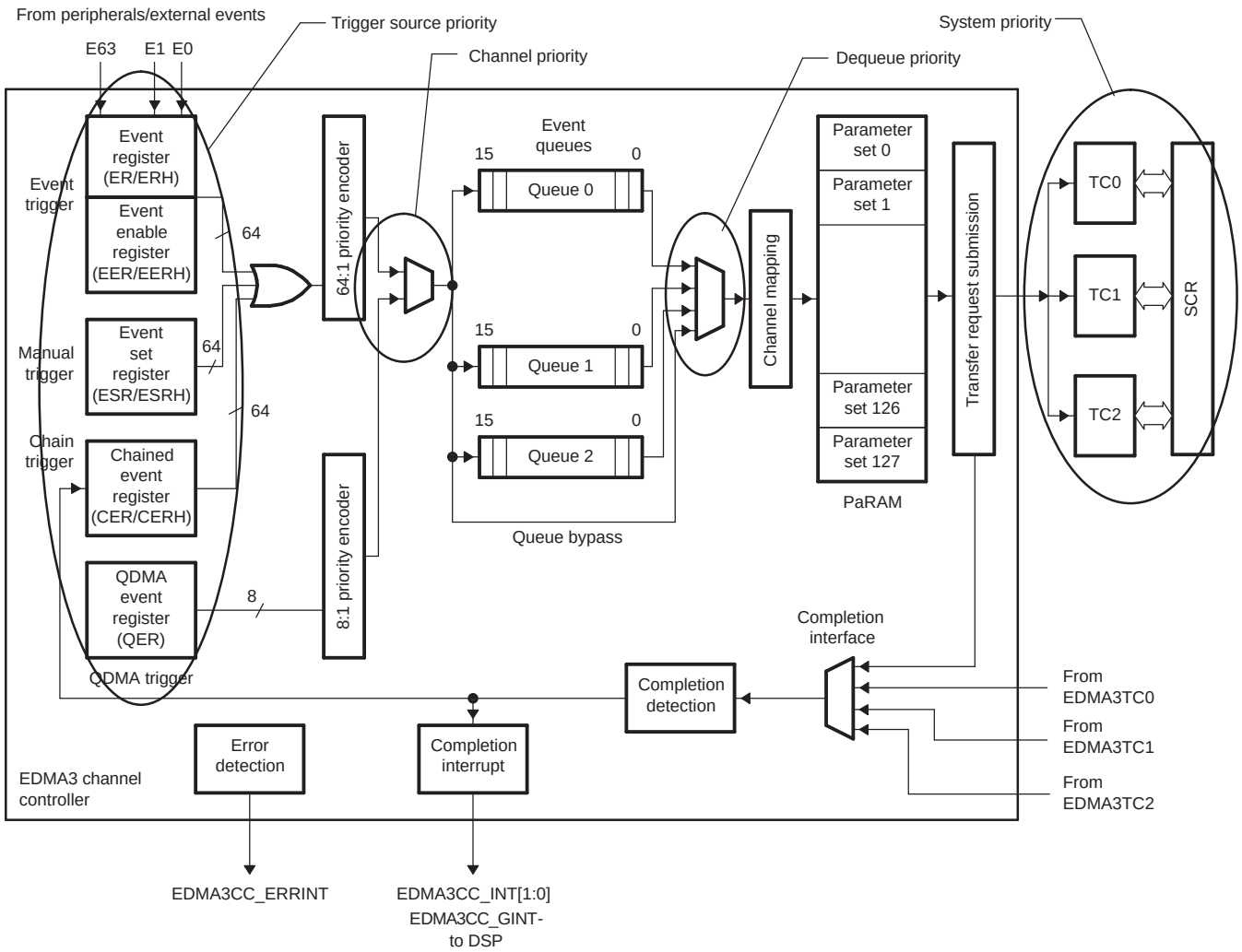
The EDMA3 controller has many implementation rules to deal with concurrent events/channels, transfers, etc. The following subsections detail various arbitration details whenever there might be occurrence of concurrent activity. Figure 2-14 shows the different places EDMA3 priorities come into play.

### *2.13.1 Channel Priority*

The DMA event registers (ER and ERH) capture up to 64 events; likewise, the QDMA event register (QER) captures QDMA events for all QDMA channels; therefore, it is possible for events to occur simultaneously on the DMA/QDMA event inputs. For events arriving simultaneously, the event associated with the lowest channel number is prioritized for submission to the event queues (for DMA events, channel 0 has the highest priority and channel 63 has the lowest priority; similarly, for QDMA events, channel 0 has the highest priority and channel 7 has the lowest priority). This mechanism only sorts simultaneous events for submission to the event queues.

If a DMA and QDMA event occurs simultaneously, the DMA event always has prioritization against the QDMA event for submission to the event queues.

**Figure 2-14. EDMA3 Prioritization**



### 2.13.2 Trigger Source Priority

If a DMA channel is associated with more than one trigger source (event trigger, manual trigger, and chain trigger), and if multiple events are set simultaneously for the same channel (ER.E*n* = 1, ESR.E*n* = 1, CER.E*n* = 1), then the EDMA3CC always services these events in the following priority order: event trigger (via ER) is higher priority than chain trigger (via CER) and chain trigger is higher priority than manual trigger (via ESR).

This implies that if for channel 0, both ER.E0 = 1 and CER.E0 = 1 at the same time, then the ER.E0 event is always queued before the CER.E0 event.

### 2.13.3 Dequeue Priority

The priority of the associated transfer request (TR) is further mitigated by which event queue is being used for event submission (dictated by DMAQNUM*n* and QDMAQNUM). For submission of a TR to the transfer controller, events need to be dequeued from the event queues. A lower numbered queue has a higher dequeuing priority then a higher numbered queue. For example, Q0 has the highest dequeue priority and Q2 has the lowest dequeue priority. In other words, if there are events in Q0, Q1, and Q2 and all the respective transfer controllers (TC0, TC1, and TC2) are ready to receive the next TR from the EDMA3CC, then the transfer requests associated with events in Q0 will get submitted to TC0 prior to any transfer requests associated with events in Q1 getting submitted to TC1, and events in Q1 will get submitted to TC1 prior to any transfer requests associated with events in Q2 getting submitted to TC2.

**NOTE:** At any given time, if there are outstanding events in multiple queues, when the transfer controller associated with the lower numbered (higher priority) queue is busy processing earlier transfer requests and the transfer controller associated with the higher numbered (lower priority) queue is idle, then the event in the higher numbered (lower priority) queue will dequeue first.

### 2.13.4 System (Transfer Controller) Priority

Each transfer controller has a programmed system priority (programmed via the QUEPRI) that is implemented when multiple masters in the system are vying for the same endpoint. The priority of the associated transfer request (TR) is further mitigated by system priority setting of the transfer controller. This priority is necessary when several masters are submitting requests to the main switched central resource (SCR), which in turn has to arbitrate the requests from these masters.

**NOTE:** The default priority for all TCs is the same, 0 or highest priority relative to other masters (like EMAC, VPSS, etc.). It is recommended that this priority be changed based on system level considerations, such as real-time deadlines for all masters including the priority of the transfer controllers with respect to each other. (The priority configuration registers for other masters are either present within the memory-map of the master or implemented as a chip level register, see the device-specific data manual).

## 2.14 EDMA3 Operating Frequency (Clock Control)

The EDMA3 channel controller and transfer controller are clocked from PLL1. The EDMA3 system runs at DSP frequency divided by 3. On DM643x devices, this is 153 MHZ in normal mode (DSP operating at 459 MHZ) or 198 MHZ in turbo mode (594 MHZ).

## 2.15 Reset Considerations

A hardware reset resets the EDMA3 (EDMA3CC and EDMA3TC) and the EDMA3 configuration registers. The PaRAM memory contents are undefined after device reset and you should not rely on parameters to be reset to a known state. The PaRAM set must be initialized to a desired value before it is used.

## 2.16 Power Management

The EDMA3 (EDMA3CC and EDMA3TC) can be placed in reduced-power modes to conserve power during periods of low activity. The power management of the peripheral is controlled by the device Power and Sleep Controller (PSC). The PSC acts as a master controller for power management for all peripherals on the device. For detailed information on power management procedures using the PSC, see the *TMS320DM643x DMP DSP Subsystem Reference Guide* (SPRU978).

The EDMA3 controller can be idled on receiving a clock stop request from the PSC. The requests to EDMA3CC and EDMA3TC are separate. In general, you should verify that there are no pending activities in the EDMA3 controller before issuing a clock stop request via PSC.

The EDMA3CC checks for the following conditions:

- No pending DMA/QDMA events
- No outstanding events in the event queues
- Transfer request processing logic is not active
- No completion requests outstanding (early or normal completion)
- No configuration bus requests in progress

The first four conditions are software readable by the channel controller status register (CCSTAT) in the EDMA3CC.

Similarly, from the EDMA3TC perspective, you should check that there are no outstanding TRs that are getting processed and essentially the read/write controller is not busy processing a TR. The activity of EDMA3TC logic is read in TCSTAT for each EDMA3TC..

It is generally recommended to first disable the EDMA3CC and then the EDMA3TC(s) to put the EDMA3 controller in reduced-power modes.

Additionally, when EDMA3 is involved in servicing a peripheral and it is required to power-down both the peripheral and the EDMA, the recommended sequence is to first disable the peripheral, then disable the DMA channel associated with the peripheral (clearing the EER/EERH bit for the channel), then disable the EDMA3CC, and finally disable the EDMA3TC(s).

## 2.17 Emulation Considerations

During debug when using the emulator, the CPU(s) may be halted on an execute packet boundary for single-stepping, benchmarking, profiling, or other debug purposes. During an emulation halt, the EDMA3 channel controller and transfer controller operations continue. Events continue to be latched and processed and transfer requests continue to be submitted and serviced.

Since EDMA3 is involved in servicing multiple master and slave peripherals, it is not feasible to have an independent behavior of the EDMA3 for emulation halts. EDMA3 functionality would be coupled with the peripherals it is servicing, which might have different behavior during emulation halts. For example, if a multichannel buffered serial port (McBSP) is halted during an emulation access (FREE = 0 and SOFT = 0 or 1 in the McBSP registers), the McBSP stops generating the McBSP receive or transmit events (REVT or XEVT) to the EDMA. From the point of view of the McBSP, the EDMA3 is suspended, but other peripherals (for example, a timer) still assert events and will be serviced by the EDMA.

# EDMA3 Transfer Examples

The EDMA3 channel controller performs a variety of transfers depending on the parameter configuration. The following sections provides a description and PaRAM configuration for some typical use case scenarios.

**Topic** **Page**

## 3.1 Block Move Example

The most basic transfer performed by the EDMA3 is a block move. During device operation it is often necessary to transfer a block of data from one location to another, usually between on-chip and off-chip memory.

In this example, a section of data is to be copied from external memory to internal L2 SRAM. A data block of 256 bytes residing at address 8000 0000h (external memory ) needs to be transferred to internal address 1080 0000h (L2), as shown in Figure 3-1. Figure 3-2 shows the parameters for this transfer.

The source address for the transfer is set to the start of the data block in external memory, and the destination address is set to the start of the data block in L2. If the data block is less than 64K bytes, the PaRAM configuration in Figure 3-2 holds true with the synchronization type set to A-synchronized and indexes cleared to 0. If the amount of data is greater than 64K bytes, BCNT and the B-indexes need to be set appropriately with the synchronization type set to AB-synchronized. The STATIC bit in OPT is set to prevent linking.

This transfer example may also be set up using QDMA. For successive transfer submissions, of a similar nature, the number of cycles used to submit the transfer are fewer depending on the number of changing transfer parameters. You may program the QDMA trigger word to be the highest numbered offset in the PaRAM set that undergoes change.

**Figure 3-1. Block Move Example**

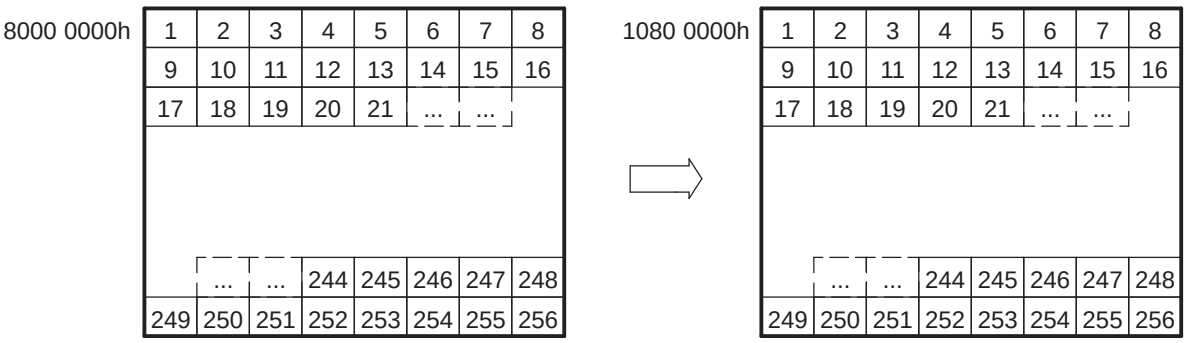


**Figure 3-2. Block Move Example PaRAM Configuration**

*(a) EDMA Parameters*

| Parameter Contents | | Parameter | |
|---|---|---|---|
| 0010 0008h | | Channel Options Parameter (OPT) | |
| 8000 0000h | | Channel Source Address (SRC) | |
| 0001h | 0100h | Count for 2nd Dimension (BCNT) | Count for 1st Dimension (ACNT) |
| 1080 0000h | | Channel Destination Address (DST) | |
| 0000h | 0000h | Destination BCNT Index (DSTBIDX) | Source BCNT Index (SRCBIDX) |
| 0000h | FFFFh | BCNT Reload (BCNTRLD) | Link Address (LINK) |
| 0000h | 0000h | Destination CCNT Index (DSTCIDX) | Source CCNT Index (SRCCIDX) |
| 0000h | 0001h | Reserved | Count for 3rd Dimension (CCNT) |

*(b) Channel Options Parameter (OPT) Content*

| 31 | 30 28 | 27 24 | 23 | 22 | 21 | 20 | 19 18 | 17 16 |
|---|---|---|---|---|---|---|---|---|
| 0 | 000 | 0000 | 0 | 0 | 0 | 1 | 00 | 00 |
| PRIV | Reserved | PRIVID | ITCCHEN | TCCHEN | ITCINTEN | TCINTEN | Reserved | TCC |

| 15 12 | 11 | 10 8 | 7 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| 0000 | 0 | 000 | 0000 | 1 | 0 | 0 | 0 |
| TCC | TCCMOD | FWID | Reserved | STATIC | SYNCDIM | DAM | SAM |

## 3.2 Subframe Extraction Example

The EDMA3 can efficiently extract a small frame of data from a larger frame of data. By performing a 2D-to-1D transfer, the EDMA3 retrieves a portion of data for the CPU to process. In this example, a 640 × 480-pixel frame of video data is stored in external memory, DDR2. Each pixel is represented by a 16-bit halfword. The CPU extracts a 16 × 12-pixel subframe of the image for processing. To facilitate more efficient processing time by the CPU, the EDMA3 places the subframe in internal L2 SRAM. Figure 3-3 shows the transfer of a subframe from external memory to L2. Figure 3-4 shows the parameters for this transfer.

The same PaRAM set options are used for QDMA channels, as well as DMA channels. The STATIC bit in OPT is set to 1 to prevent linking. For successive transfers, only changed parameters need to be programmed before triggering the channel.

**Figure 3-3. Subframe Extraction Example**

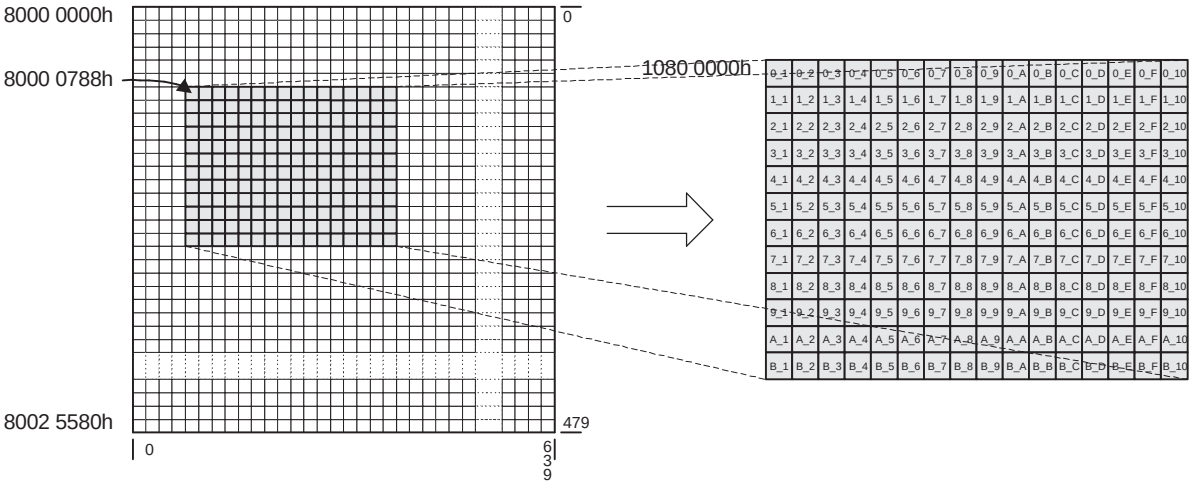


**Figure 3-4. Subframe Extraction Example PaRAM Configuration**

*(a) EDMA Parameters*

| Parameter Contents | | Parameter | |
|---|---|---|---|
| 0010 000Ch | | Channel Options Parameter (OPT) | |
| 8000 0788h | | Channel Source Address (SRC) | |
| 000Ch | 0020h | Count for 2nd Dimension (BCNT) | Count for 1st Dimension (ACNT) |
| 1080 0000h | | Channel Destination Address (DST) | |
| 0020h | 0500h | Destination BCNT Index (DSTBIDX) | Source BCNT Index (SRCBIDX) |
| 0000h | FFFFh | BCNT Reload (BCNTRLD) | Link Address (LINK) |
| 0000h | 0000h | Destination CCNT Index (DSTCIDX) | Source CCNT Index (SRCCIDX) |
| 0000h | 0001h | Reserved | Count for 3rd Dimension (CCNT) |

*(b) Channel Options Parameter (OPT) Content*

| 31 | 30 28 | 27 24 | 23 | 22 | 21 | 20 | 19 18 | 17 16 |
|---|---|---|---|---|---|---|---|---|
| 0 | 000 | 0000 | 0 | 0 | 0 | 1 | 00 | 00 |
| PRIV | Reserved | PRIVID | ITCCHEN | TCCHEN | ITCINTEN | TCINTEN | Reserved | TCC |

| 15 12 | 11 | 10 8 | 7 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| 0000 | 0 | 000 | 0000 | 1 | 1 | 0 | 0 |
| TCC | TCCMOD | FWID | Reserved | STATIC | SYNCDIM | DAM | SAM |

## 3.3 Data Sorting Example

Many applications require the use of multiple data arrays; it is often desirable to have the arrays arranged such that the first elements of each array are adjacent, the second elements are adjacent, and so on. Often this is not how the data is presented to the device. Either data is transferred via a peripheral with the data arrays arriving one after the other or the arrays are located in memory with each array occupying a portion of contiguous memory spaces. For these instances, the EDMA3 can reorganize the data into the desired format. Figure 3-5 shows the data sorting.

In order to determine the parameter entry values, the following need to be considered:

- ACNT – Program this to be the size in bytes of an element.
- BCNT – Program this to be the number of elements in a frame.
- CCNT – Program this to be the number of frames.
- SRCBIDX – Program this to be the size of the element or ACNT.
- DSTBIDX = CCNT × ACNT
- SRCCDX = ACNT × BCNT
- DSTCIDX = ACNT

The synchronization type needs to be AB-synchronized and the STATIC bit is 0 to allow updates to the parameter set. It is advised to use normal DMA channels for sorting.

It is not possible to sort this with a single trigger event. Instead, the channel can be programmed to be chained to itself. After BCNT elements get sorted, intermediate chaining could be used to trigger the channel again causing the transfer of the next BCNT elements and so on. Figure 3-6 shows the parameter set programming for this transfer, assuming channel 0 and an element size of 4 bytes.

**Figure 3-5. Data Sorting Example**

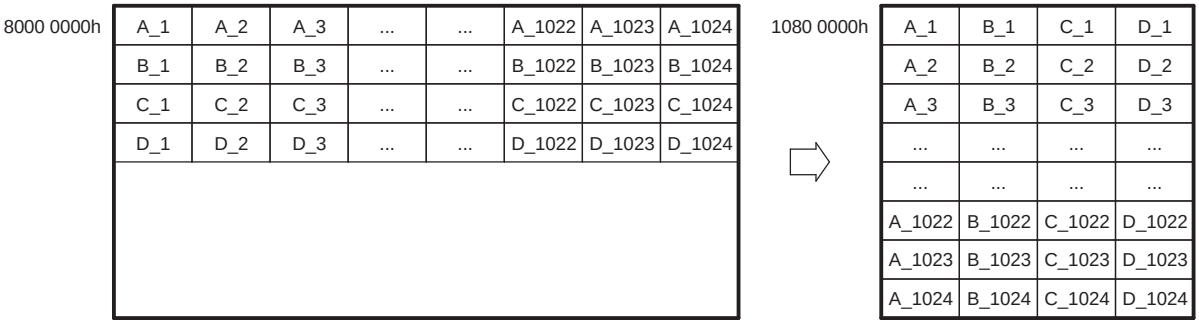



**Figure 3-6. Data Sorting Example PaRAM Configuration**

*(a) EDMA Parameters*

| Parameter Contents | | Parameter | |
|---|---|---|---|
| 0090 0004h | | Channel Options Parameter (OPT) | |
| 8000 0000h | | Channel Source Address (SRC) | |
| 0400h | 0004h | Count for 2nd Dimension (BCNT) | Count for 1st Dimension (ACNT) |
| 1080 0000h | | Channel Destination Address (DST) | |
| 0010h | 0001h | Destination BCNT Index (DSTBIDX) | Source BCNT Index (SRCBIDX) |
| 0000h | FFFFh | BCNT Reload (BCNTRLD) | Link Address (LINK) |
| 0001h | 1000h | Destination CCNT Index (DSTCIDX) | Source CCNT Index (SRCCIDX) |
| 0000h | 0004h | Reserved | Count for 3rd Dimension (CCNT) |

*(b) Channel Options Parameter (OPT) Content*

| 31 | 30 28 | 27 24 | 23 | 22 | 21 | 20 | 19 18 | 17 16 |
|---|---|---|---|---|---|---|---|---|
| 0 | 000 | 0000 | 1 | 0 | 0 | 1 | 00 | 00 |
| PRIV | Reserved | PRIVID | ITCCHEN | TCCHEN | ITCINTEN | TCINTEN | Reserved | TCC |

| 15 12 | 11 | 10 8 | 7 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| 0000 | 0 | 000 | 0000 | 0 | 1 | 0 | 0 |
| TCC | TCCMOD | FWID | Reserved | STATIC | SYNCDIM | DAM | SAM |

## 3.4 Peripheral Servicing Example

The EDMA3 channel controller also services peripherals in the background of CPU operation, without requiring any CPU intervention. Through proper initialization of the DMA channels, they can be configured to continuously service on-chip and off-chip peripherals throughout the device operation. Each event available to the EDMA3 has its own dedicated channel, and all channels operate simultaneously. The only requirements are to use the proper channel for a particular transfer and to enable the channel event in the event enable register (EER). When programming a DMA channel to service a peripheral, it is necessary to know how data is to be presented to the DSP. Data is always provided with some kind of synchronization event as either one element per event (nonbursting) or multiple elements per event (bursting).

### *3.4.1 Nonbursting Peripherals*

Nonbursting peripherals include the on-chip multichannel buffered serial port (McBSP) and many external devices, such as codecs. Regardless of the peripheral, the DMA channel configuration is the same.

The McBSP transmit and receive data streams are treated independently by the EDMA3. The transmit and receive data streams can have completely different counts, data sizes, and formats. Figure 3-7 shows servicing incoming McBSP data.

To transfer the incoming data stream to its proper location in L2 memory, the DMA channel must be set up for a 1D-to-1D transfer with A-synchronization. Since an event (REVT) is generated for every word as it arrives, it is necessary to have the EDMA3 issue the transfer request for each element individually. Figure 3-8 shows the parameters for this transfer. The source address of the DMA channel is set to the data receive register (DRR) address for the McBSP, and the destination address is set to the start of the data block in L2. Since the address of DRR is fixed, the source B index is cleared to 0 (no modification) and the destination B index is set to 01b (increment).

Based on the premise that serial data is typically a high priority, the DMA channel should be programmed to be on queue 0.

**Figure 3-7. Servicing Incoming McBSP Data Example**

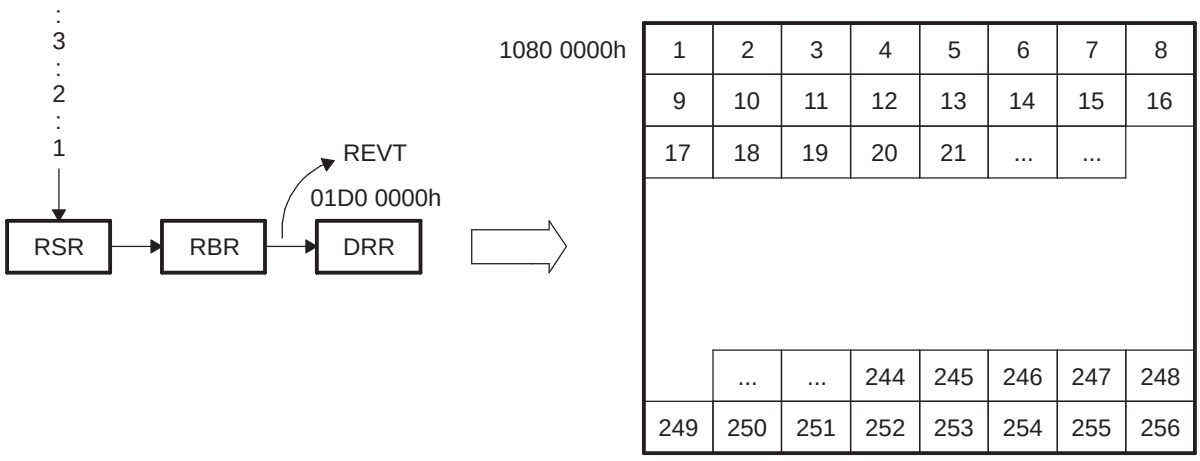


**Figure 3-8. Servicing Incoming McBSP Data Example PaRAM**

*(a) EDMA Parameters*

| Parameter Contents | | Parameter | |
|---|---|---|---|
| 0010 0000h | | Channel Options Parameter (OPT) | |
| 01D0 0000h | | Channel Source Address (SRC) | |
| 0100h | 0001h | Count for 2nd Dimension (BCNT) | Count for 1st Dimension (ACNT) |
| 1080 0000h | | Channel Destination Address (DST) | |
| 0001h | 0000h | Destination BCNT Index (DSTBIDX) | Source BCNT Index (SRCBIDX) |
| 0000h | FFFFh | BCNT Reload (BCNTRLD) | Link Address (LINK) |
| 0000h | 0000h | Destination CCNT Index (DSTCIDX) | Source CCNT Index (SRCCIDX) |
| 0000h | 0004h | Reserved | Count for 3rd Dimension (CCNT) |

*(b) Channel Options Parameter (OPT) Content*

| 31 | 30 28 | 27 24 | 23 | 22 | 21 | 20 | 19 18 | 17 16 |
|---|---|---|---|---|---|---|---|---|
| 0 | 000 | 0000 | 0 | 0 | 0 | 1 | 00 | 00 |
| PRIV | Reserved | PRIVID | ITCCHEN | TCCHEN | ITCINTEN | TCINTEN | Reserved | TCC |

| 15 12 | 11 | 10 8 | 7 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| 0000 | 0 | 000 | 0000 | 0 | 0 | 0 | 0 |
| TCC | TCCMOD | FWID | Reserved | STATIC | SYNCDIM | DAM | SAM |

### 3.4.2 Bursting Peripherals

Higher bandwidth applications require that multiple data elements be presented to the CPU for every synchronization event. This frame of data can either be from multiple sources that are working simultaneously or from a single high-throughput peripheral that streams data to/from the CPU.

In this example, a port is receiving a video frame from a camera and presenting it to the CPU one array at a time. The video image is 640 × 480 pixels, with each pixel represented by a 16-bit element. The image is to be stored in external memory. Figure 3-9 shows this example.

To transfer data from an external peripheral to an external buffer one array at a time based on EVT*n* , channel *n* must be configured. Due to the nature of the data (a video frame made up of arrays of pixels) the destination is essentially a 2D entity. Figure 3-10 shows the parameters to service the incoming data with a 1D-to-2D transfer using AB-synchronization. The source address is set to the location of the video framer peripheral, and the destination address is set to the start of the data buffer. Since the input address is static, the SRCBIDX is 0 (no modification to the source address). The destination is made up of arrays of contiguous, linear elements; therefore, the DSTBIDX is set to pixel size, 2 bytes. ANCT is equal to the pixel size, 2 bytes. BCNT is set to the number of pixels in an array, 640. CCNT is equal to the total number of arrays in the block, 480. SRCCIDX is 0 since the source address undergoes no increment. The DSTCIDX is equal to the difference between the starting addresses of each array. Since a pixel is 16 bits (2 bytes), DSTCIDX is equal to 640 × 2.

**Figure 3-9. Servicing Peripheral Burst Example**

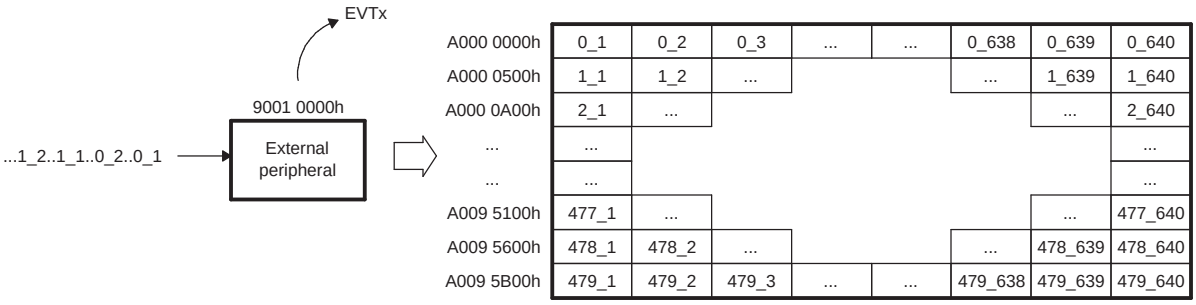


**Figure 3-10. Servicing Peripheral Burst Example PaRAM**

*(a) EDMA Parameters*

| Parameter Contents | | Parameter | |
|---|---|---|---|
| 0010 0004h | | Channel Options Parameter (OPT) | |
| Channel Source Address | | Channel Source Address (SRC) | |
| 0280h | 0002h | Count for 2nd Dimension (BCNT) | Count for 1st Dimension (ACNT) |
| 8000 0000h | | Channel Destination Address (DST) | |
| 0002h | 0000h | Destination BCNT Index (DSTBIDX) | Source BCNT Index (SRCBIDX) |
| 0000h | FFFFh | BCNT Reload (BCNTRLD) | Link Address (LINK) |
| 0500h | 0000h | Destination CCNT Index (DSTCIDX) | Source CCNT Index (SRCCIDX) |
| 0000h | 01E0h | Reserved | Count for 3rd Dimension (CCNT) |

*(b) Channel Options Parameter (OPT) Content*

| 31 | 30 28 | 27 24 | 23 | 22 | 21 | 20 | 19 18 | 17 16 |
|---|---|---|---|---|---|---|---|---|
| 0 | 000 | 0000 | 0 | 0 | 0 | 1 | 00 | 00 |
| PRIV | Reserved | PRIVID | ITCCHEN | TCCHEN | ITCINTEN | TCINTEN | Reserved | TCC |

| 15 12 | 11 | 10 8 | 7 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| 0000 | 0 | 000 | 0000 | 0 | 1 | 0 | 0 |
| TCC | TCCMOD | FWID | Reserved | STATIC | SYNCDIM | DAM | SAM |

### 3.4.3 Continuous Operation

Configuring a DMA channel to receive a single frame of data is useful, and is applicable to some systems. A majority of the time, however, data is going to be continuously transmitted and received throughout the entire operation of the CPU. In this case, it is necessary to implement some form of linking such that the DMA channels continuously reload the necessary parameter sets. In this example, the multichannel buffered serial port (McBSP) is configured to transmit and receive data on a array. To simplify the example, only two channels are active for both transmit and receive data streams. Each channel receives packets of 128 elements. The packets are transferred from the serial port to L2 memory and from L2 memory to the serial port, as shown in Figure 3-11.

The McBSP generates REVT for every element received and generates XEVT for every element transmitted. To service the data streams, the DMA channels associated with the McBSP must be set up for 1D-to-1D transfers with A-synchronization.

Figure 3-12 shows the parameters for the parameter entries for the channel for these transfers. In order to service the McBSP continuously throughout CPU operation, the channels must be linked to a duplicate PaRAM set in the PaRAM. After all frames have been transferred, the DMA channels reload and continue. Figure 3-13 shows the reload parameters for the channel.

#### 3.4.3.1 Receive Channel

DMA channel 3 services the incoming data stream of the McBSP. The source address is set to that of the data receiver register (DRR), and the destination address is set to the first element of the data block. Since there are two data channels being serviced, A and B, they are to be located separately within the L2 SRAM.

In order to facilitate continuous operation, a copy of the PaRAM set for the channel is placed in PaRAM set 64. The LINK option is set and the link address is provided in the PaRAM set. Upon exhausting the channel 3 parameter set, the parameters located at the link address are loaded into the channel 3 parameter set and operation continues. This function continues throughout device operation until halted by the CPU.

### 3.4.3.2 Transmit Channel

DMA channel 2 services the outgoing data stream of the McBSP. In this case the destination address needs no update, hence, the parameter set changes accordingly. Linking is also used to allow continuous operation by the DMA channel, with duplicate PaRAM set entries at PaRAM set 65.

**Figure 3-11. Servicing Continuous McBSP Data Example**

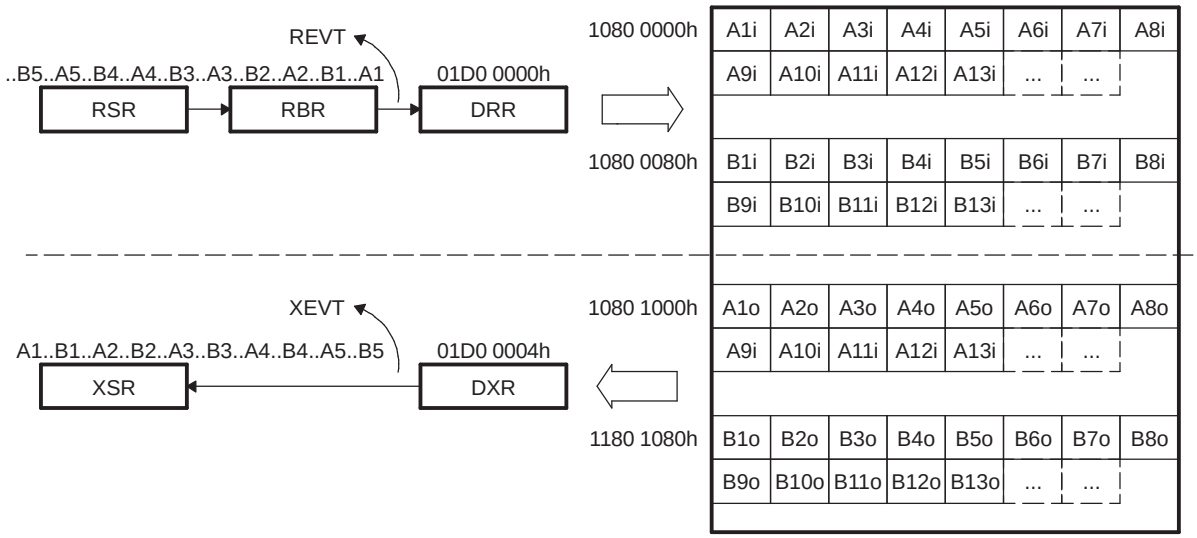


**Figure 3-12. Servicing Continuous McBSP Data Example PaRAM**

*(a) EDMA Parameters for Receive Channel (PaRAM Set 3) being Linked to PaRAM Set 64*

| Parameter Contents | | Parameter | |
|---|---|---|---|
| 0010 0000h | | Channel Options Parameter (OPT) | |
| 01D0 0000h | | Channel Source Address (SRC) | |
| 0080h | 0001h | Count for 2nd Dimension (BCNT) | Count for 1st Dimension (ACNT) |
| 1080 0000h | | Channel Destination Address (DST) | |
| 0001h | 0000h | Destination BCNT Index (DSTBIDX) | Source BCNT Index (SRCBIDX) |
| 0080h | 4800h | BCNT Reload (BCNTRLD) | Link Address (LINK) |
| 0000h | 0000h | Destination CCNT Index (DSTCIDX) | Source CCNT Index (SRCCIDX) |
| 0000h | FFFFh | Reserved | Count for 3rd Dimension (CCNT) |

*(b) Channel Options Parameter (OPT) Content for Receive Channel (PaRAM Set 3)*

| 31 | 30 28 | 27 24 | 23 | 22 | 21 | 20 | 19 18 | 17 16 |
|---|---|---|---|---|---|---|---|---|
| 0 | 000 | 0000 | 0 | 0 | 0 | 1 | 00 | 00 |
| PRIV | Reserved | PRIVID | ITCCHEN | TCCHEN | ITCINTEN | TCINTEN | Reserved | TCC |

| 15 12 | 11 | 10 8 | 7 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| 0000 | 0 | 000 | 0000 | 0 | 0 | 0 | 0 |
| TCC | TCCMOD | FWID | Reserved | STATIC | SYNCDIM | DAM | SAM |

*(c) EDMA Parameters for Transmit Channel (PaRAM Set 2) being Linked to PaRAM Set 65*

| Parameter Contents | | Parameter | |
|---|---|---|---|
| 0010 1000h | | Channel Options Parameter (OPT) | |
| 1080 1000h | | Channel Source Address (SRC) | |
| 0080h | 0001h | Count for 2nd Dimension (BCNT) | Count for 1st Dimension (ACNT) |
| 01D0 0004h | | Channel Destination Address (DST) | |
| 0000h | 0001h | Destination BCNT Index (DSTBIDX) | Source BCNT Index (SRCBIDX) |
| 0080h | 4820h | BCNT Reload (BCNTRLD) | Link Address (LINK) |
| 0000h | 0000h | Destination CCNT Index (DSTCIDX) | Source CCNT Index (SRCCIDX) |
| 0000h | FFFFh | Reserved | Count for 3rd Dimension (CCNT) |

*(d) Channel Options Parameter (OPT) Content for Transmit Channel (PaRAM Set 2)*

| 31 | 30 28 | 27 24 | 23 | 22 | 21 | 20 | 19 18 | 17 16 |
|---|---|---|---|---|---|---|---|---|
| 0 | 000 | 0000 | 0 | 0 | 0 | 1 | 00 | 00 |
| PRIV | Reserved | PRIVID | ITCCHEN | TCCHEN | ITCINTEN | TCINTEN | Reserved | TCC |

| 15 12 | 11 | 10 8 | 7 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| 0001 | 0 | 000 | 0000 | 0 | 0 | 0 | 0 |
| TCC | TCCMOD | FWID | Reserved | STATIC | SYNCDIM | DAM | SAM |

**Figure 3-13. Servicing Continuous McBSP Data Example Reload PaRAM**

*(a) EDMA Reload Parameters (PaRAM Set 64) for Receive Channel*

| Parameter Contents | | Parameter | |
|---|---|---|---|
| 0010 0000h | | Channel Options Parameter (OPT) | |
| 01D0 0000h | | Channel Source Address (SRC) | |
| 0080h | 0001h | Count for 2nd Dimension (BCNT) | Count for 1st Dimension (ACNT) |
| 1080 0000h | | Channel Destination Address (DST) | |
| 0001h | 0000h | Destination BCNT Index (DSTBIDX) | Source BCNT Index (SRCBIDX) |
| 0080h | 4800h | BCNT Reload (BCNTRLD) | Link Address (LINK) |
| 0000h | 0000h | Destination CCNT Index (DSTCIDX) | Source CCNT Index (SRCCIDX) |
| 0000h | FFFFh | Reserved | Count for 3rd Dimension (CCNT) |

*(b) Channel Options Parameter (OPT) Content for Receive Channel (PaRAM Set 64)*

| 31 | 30 28 | 27 24 | 23 | 22 | 21 | 20 | 19 18 | 17 16 |
|---|---|---|---|---|---|---|---|---|
| 0 | 000 | 0000 | 0 | 0 | 0 | 1 | 00 | 00 |
| PRIV | Reserved | PRIVID | ITCCHEN | TCCHEN | ITCINTEN | TCINTEN | Reserved | TCC |

| 15 12 | 11 | 10 8 | 7 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| 0000 | 0 | 000 | 0000 | 0 | 0 | 0 | 0 |
| TCC | TCCMOD | FWID | Reserved | STATIC | SYNCDIM | DAM | SAM |

*(c) EDMA Reload Parameters (PaRAM Set 65) for Transmit Channel*

| Parameter Contents | | Parameter | |
|---|---|---|---|
| 0010 1000h | | Channel Options Parameter (OPT) | |
| 1080 1000h | | Channel Source Address (SRC) | |
| 0080h | 0001h | Count for 2nd Dimension (BCNT) | Count for 1st Dimension (ACNT) |
| 01D0 0004h | | Channel Destination Address (DST) | |
| 0000h | 0001h | Destination BCNT Index (DSTBIDX) | Source BCNT Index (SRCBIDX) |
| 0080h | 4820h | BCNT Reload (BCNTRLD) | Link Address (LINK) |
| 0000h | 0000h | Destination CCNT Index (DSTCIDX) | Source CCNT Index (SRCCIDX) |
| 0000h | FFFFh | Reserved | Count for 3rd Dimension (CCNT) |

*(d) Channel Options Parameter (OPT) Content for Transmit Channel (PaRAM Set 65)*

| 31 | 30 28 | 27 24 | 23 | 22 | 21 | 20 | 19 18 | 17 16 |
|---|---|---|---|---|---|---|---|---|
| 0 | 000 | 0000 | 0 | 0 | 0 | 1 | 00 | 00 |
| PRIV | Reserved | PRIVID | ITCCHEN | TCCHEN | ITCINTEN | TCINTEN | Reserved | TCC |

| 15 12 | 11 | 10 8 | 7 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| 0001 | 0 | 000 | 0000 | 0 | 0 | 0 | 0 |
| TCC | TCCMOD | FWID | Reserved | STATIC | SYNCDIM | DAM | SAM |

### 3.4.4 Ping-Pong Buffering

Although the previous configuration allows the EDMA3 to service a peripheral continuously, it presents a number of restrictions to the CPU. Since the input and output buffers are continuously being filled/emptied, the CPU must match the pace of the EDMA3 very closely in order to process the data. The EDMA3 receive data must always be placed in memory before the CPU accesses it, and the CPU must provide the output data before the EDMA3 transfers it. Though not impossible, this is an unnecessary challenge. It is particularly difficult in a 2-level cache scheme.

Ping-pong buffering is a simple technique that allows the CPU activity to be distanced from the EDMA3 activity. This means that there are multiple (usually two) sets of data buffers for all incoming and outgoing data streams. While the EDMA3 transfers the data into and out of the ping buffers, the CPU manipulates the data in the pong buffers. When both CPU and EDMA3 activity completes, they switch. The EDMA3 then writes over the old input data and transfers the new output data. Figure 3-14 shows the ping-pong scheme for this example.

To change the continuous operation example, such that a ping-pong buffering scheme is used, the DMA channels need only a moderate change. Instead of one parameter set, there are two; one for transferring data to/from the ping buffers and one for transferring data to/from the pong buffers. As soon as one transfer completes, the channel loads the PaRAM set for the other and the data transfers continue. Figure 3-15 shows the DMA channel configuration required.

Each channel has two parameter sets, ping and pong. The DMA channel is initially loaded with the ping parameters (Figure 3-15). The link address for the ping set is set to the PaRAM offset of the pong parameter set (Figure 3-16). The link address for the pong set is set to the PaRAM offset of the ping parameter set (Figure 3-17). The channel options, count values, and index values are all identical between the ping and pong parameters for each channel. The only differences are the link address provided and the address of the data buffer.

#### 3.4.4.1 Synchronization with the CPU

In order to utilize the ping-pong buffering technique, the system must signal the CPU when to begin to access the new data set. After the CPU finishes processing an input buffer (ping), it waits for the EDMA3 to complete before switching to the alternate (pong) buffer. In this example, both channels provide their channel numbers as their report word and set the TCINTEN bit to 1 to generate an interrupt after completion. When channel 3 fills an input buffer, the E3 bit in the interrupt pending register (IPR) is set to 1; when channel 2 empties an output buffer, the E2 bit in IPR is set to 1. The CPU must manually clear these bits. With the channel parameters set, the CPU polls IPR to determine when to switch. The EDMA3 and CPU could alternatively be configured such that the channel completion interrupts the CPU. By doing this, the CPU could service a background task while waiting for the EDMA3 to complete.

**Figure 3-14. Ping-Pong Buffering for McBSP Data Example**

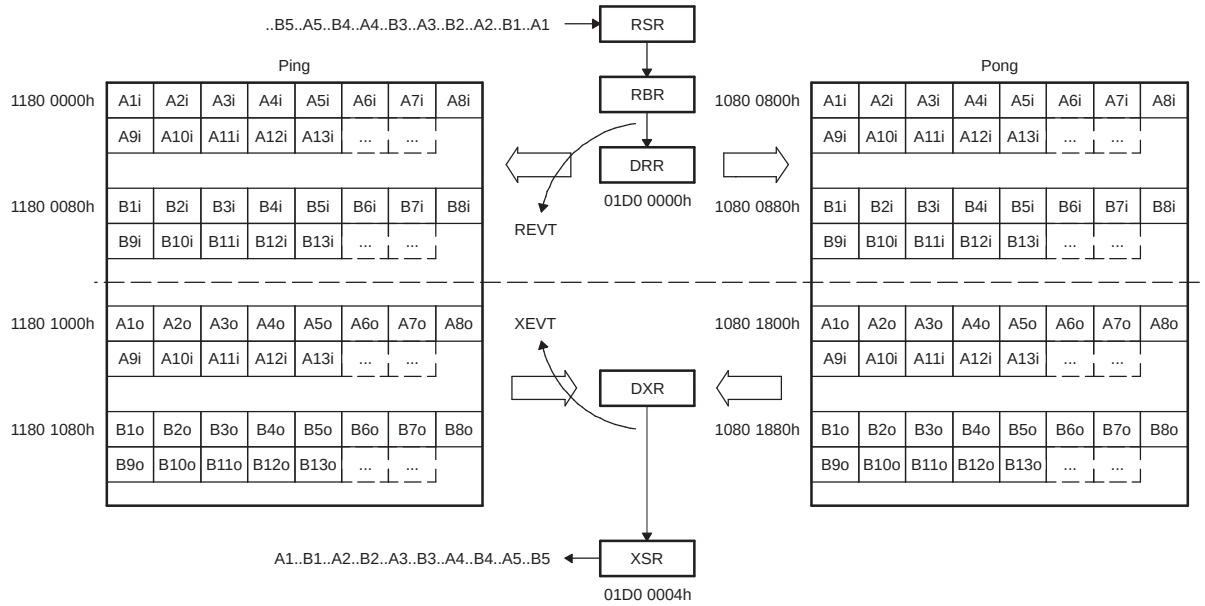



**Figure 3-15. Ping-Pong Buffering for McBSP Example PaRAM**

*(a) EDMA Parameters for Channel 3 (Using PaRAM Set 3 Linked to Pong Set 64)*

| Parameter Contents | | Parameter | |
|---|---|---|---|
| 0010 3000h | | Channel Options Parameter (OPT) | |
| 01D0 0000h | | Channel Source Address (SRC) | |
| 0080h | 0001h | Count for 2nd Dimension (BCNT) | Count for 1st Dimension (ACNT) |
| 1080 0000h | | Channel Destination Address (DST) | |
| 0001h | 0000h | Destination BCNT Index (DSTBIDX) | Source BCNT Index (SRCBIDX) |
| 0080h | 4800h | BCNT Reload (BCNTRLD) | Link Address (LINK) |
| 0000h | 0000h | Destination CCNT Index (DSTCIDX) | Source CCNT Index (SRCCIDX) |
| 0000h | 0001h | Reserved | Count for 3rd Dimension (CCNT) |

*(b) Channel Options Parameter (OPT) Content for Channel 3*

| 31 | 30 28 | 27 24 | 23 | 22 | 21 | 20 | 19 18 | 17 16 |
|---|---|---|---|---|---|---|---|---|
| 0 | 000 | 0000 | 0 | 0 | 0 | 1 | 00 | 00 |
| PRIV | Reserved | PRIVID | ITCCHEN | TCCHEN | ITCINTEN | TCINTEN | Reserved | TCC |

| 15 12 | 11 | 10 8 | 7 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| 0011 | 0 | 000 | 0000 | 0 | 0 | 0 | 0 |
| TCC | TCCMOD | FWID | Reserved | STATIC | SYNCDIM | DAM | SAM |

*(c) EDMA Parameters for Channel 2 (Using PaRAM Set 2 Linked to Pong Set 65)*

| Parameter Contents | | Parameter | |
|---|---|---|---|
| 0010 2000h | | Channel Options Parameter (OPT) | |
| 1080 1000h | | Channel Source Address (SRC) | |
| 0080h | 0001h | Count for 2nd Dimension (BCNT) | Count for 1st Dimension (ACNT) |
| 01D0 0004h | | Channel Destination Address (DST) | |
| 0000h | 0001h | Destination BCNT Index (DSTBIDX) | Source BCNT Index (SRCBIDX) |
| 0080h | 4840h | BCNT Reload (BCNTRLD) | Link Address (LINK) |
| 0000h | 0000h | Destination CCNT Index (DSTCIDX) | Source CCNT Index (SRCCIDX) |
| 0000h | 0001h | Reserved | Count for 3rd Dimension (CCNT) |

*(d) Channel Options Parameter (OPT) Content for Channel 2*

| 31 | 30 28 | 27 24 | 23 | 22 | 21 | 20 | 19 18 | 17 16 |
|---|---|---|---|---|---|---|---|---|
| 0 | 000 | 0000 | 0 | 0 | 0 | 1 | 00 | 00 |
| PRIV | Reserved | PRIVID | ITCCHEN | TCCHEN | ITCINTEN | TCINTEN | Reserved | TCC |

| 15 12 | 11 | 10 8 | 7 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| 0010 | 0 | 000 | 0000 | 0 | 0 | 0 | 0 |
| TCC | TCCMOD | FWID | Reserved | STATIC | SYNCDIM | DAM | SAM |

**Figure 3-16. Ping-Pong Buffering for McBSP Example Pong PaRAM**

*(a) EDMA Pong Parameters for Channel 3 at Set 64 Linked to Set 65*

| Parameter Contents | | Parameter | |
|---|---|---|---|
| 0010 D000h | | Channel Options Parameter (OPT) | |
| 01D0 0000h | | Channel Source Address (SRC) | |
| 0080h | 0001h | Count for 2nd Dimension (BCNT) | Count for 1st Dimension (ACNT) |
| 1080 0800h | | Channel Destination Address (DST) | |
| 0001h | 0000h | Destination BCNT Index (DSTBIDX) | Source BCNT Index (SRCBIDX) |
| 0080h | 4820h | BCNT Reload (BCNTRLD) | Link Address (LINK) |
| 0000h | 0000h | Destination CCNT Index (DSTCIDX) | Source CCNT Index (SRCCIDX) |
| 0000h | 0001h | Reserved | Count for 3rd Dimension (CCNT) |

*(b) EDMA Pong Parameters for Channel 2 at Set 66 Linked to Set 67*

| Parameter Contents | | Parameter | |
|---|---|---|---|
| 0010 C000h | | Channel Options Parameter (OPT) | |
| 1080 1800h | | Channel Source Address (SRC) | |
| 0080h | 0001h | Count for 2nd Dimension (BCNT) | Count for 1st Dimension (ACNT) |
| 01D0 0004h | | Channel Destination Address (DST) | |
| 0000h | 0001h | Destination BCNT Index (DSTBIDX) | Source BCNT Index (SRCBIDX) |
| 0080h | 4860h | BCNT Reload (BCNTRLD) | Link Address (LINK) |
| 0000h | 0000h | Destination CCNT Index (DSTCIDX) | Source CCNT Index (SRCCIDX) |
| 0000h | 0001h | Reserved | Count for 3rd Dimension (CCNT) |

**Figure 3-17. Ping-Pong Buffering for McBSP Example Ping PaRAM**

*(a) EDMA Ping Parameters for Channel 3 at Set 65 Linked to Set 64*

| Parameter Contents | | Parameter | |
|---|---|---|---|
| 0010 D000h | | Channel Options Parameter (OPT) | |
| 01D0 0000h | | Channel Source Address (SRC) | |
| 0080h | 0001h | Count for 2nd Dimension (BCNT) | Count for 1st Dimension (ACNT) |
| 1080 0000h | | Channel Destination Address (DST) | |
| 0001h | 0000h | Destination BCNT Index (DSTBIDX) | Source BCNT Index (SRCBIDX) |
| 0080h | 4800h | BCNT Reload (BCNTRLD) | Link Address (LINK) |
| 0000h | 0000h | Destination CCNT Index (DSTCIDX) | Source CCNT Index (SRCCIDX) |
| 0000h | 0001h | Reserved | Count for 3rd Dimension (CCNT) |

*(b) EDMA Ping Parameters for Channel 2 at Set 67 Linked to Set 66*

| Parameter Contents | | Parameter | |
|---|---|---|---|
| 0010 C000h | | Channel Options Parameter (OPT) | |
| 1080 1000h | | Channel Source Address (SRC) | |
| 0080h | 0001h | Count for 2nd Dimension (BCNT) | Count for 1st Dimension (ACNT) |
| 01D0 0004h | | Channel Destination Address (DST) | |
| 0000h | 0001h | Destination BCNT Index (DSTBIDX) | Source BCNT Index (SRCBIDX) |
| 0080h | 4840h | BCNT Reload (BCNTRLD) | Link Address (LINK) |
| 0000h | 0000h | Destination CCNT Index (DSTCIDX) | Source CCNT Index (SRCCIDX) |
| 0000h | 0001h | Reserved | Count for 3rd Dimension (CCNT) |

### *3.4.5 Transfer Chaining Examples*

The following examples explain the intermediate transfer complete chaining function.

#### 3.4.5.1 Servicing Input/Output FIFOs with a Single Event

Many systems require the use of a pair of external FIFOs that must be serviced at the same rate. One FIFO buffers data input, and the other buffers data output. The EDMA3 channels that service these FIFOs can be set up for AB-synchronized transfers. While each FIFO is serviced with a different set of parameters, both can be signaled from a single event. For example, an external interrupt pin can be tied to the status flags of one of the FIFOs. When this event arrives, the EDMA3 needs to perform servicing for both the input and output streams. Without the intermediate transfer complete chaining feature this would require two events, and thus two external interrupt pins. The intermediate transfer complete chaining feature allows the use of a single external event (for example, a GPIO event). Figure 3-18 shows the EDMA3 setup and illustration for this example.

A GPIO event (in this case, GPINT0) triggers an array transfer. Upon completion of each intermediate array transfer of channel 32, intermediate transfer complete chaining sets the E63 bit (specified by TCC of 63) in the chained event register high (CERH) and provides a synchronization event to channel 63. Upon completion of the last array transfer of channel 32, transfer complete chaining—not intermediate transfer complete chaining—sets the E63 bit in CERH (specified by TCCMODE:TCC) and provides a synchronization event to channel 63. The completion of channel 63 sets the I63 bit (specified by TCCMODE:TCC) in the interrupt pending register high (IPRH), which can generate an interrupt to the CPU, if the I63 bit in the interrupt enable register high (IERH) is set to 1.

#### 3.4.5.2 Breaking Up Large Transfers with Intermediate Chaining

Another feature of intermediate transfer chaining (ITCCHEN) is for breaking up large transfers. A large transfer may lock out other transfers of the same priority level for the duration of the transfer. For example, a large transfer on queue 0 from the internal memory to the external memory using the EMIF may starve other EDMA3 transfers on the same queue. In addition, this large high-priority transfer may prevent the EMIF for a long duration to service other lower priority transfers. When a large transfer is considered to be high priority, it should be split into multiple smaller transfers. Figure 3-19 shows the EDMA3 setup and illustration of an example single large block transfer.

The intermediate transfer chaining enable (ITCCHEN) provides a method to break up a large transfer into smaller transfers. For example, to move a single large block of memory (16K bytes), the EDMA3 performs an A-synchronized transfer. The element count is set to a reasonable value, where reasonable derives from the amount of time it would take to move this smaller amount of data. Assume 1 Kbyte is a reasonable small transfer in this example. The EDMA3 is set up to transfer 16 arrays of 1 Kbyte elements, for a total of 16K byte elements. The TCC field in the channel options parameter (OPT) is set to the same value as the channel number and ITCCHEN are set. In this example, DMA channel 25 is used and TCC is also set to 25. The TCINTEN may also be set to trigger interrupt 25 when the last 1 Kbyte array is transferred. The CPU starts the EDMA3 transfer by writing to the appropriate bit of the event set register (ESR.E25). The EDMA3 transfers the first 1 Kbyte array. Upon completion of the first array, intermediate transfer complete code chaining generates a synchronization event to channel 25, a value specified by the TCC field. This intermediate transfer completion chaining event causes DMA channel 25 to transfer the next 1 Kbyte array. This process continues until the transfer parameters are exhausted, at which point the EDMA3 has completed the 16K byte transfer. This method breaks up a large transfer into smaller packets, thus providing natural time slices in the transfer such that other events may be processed. Figure 3-20 shows the EDMA3 setup and illustration of the broken up smaller packet transfers.

**Figure 3-18. Intermediate Transfer Completion Chaining Example**

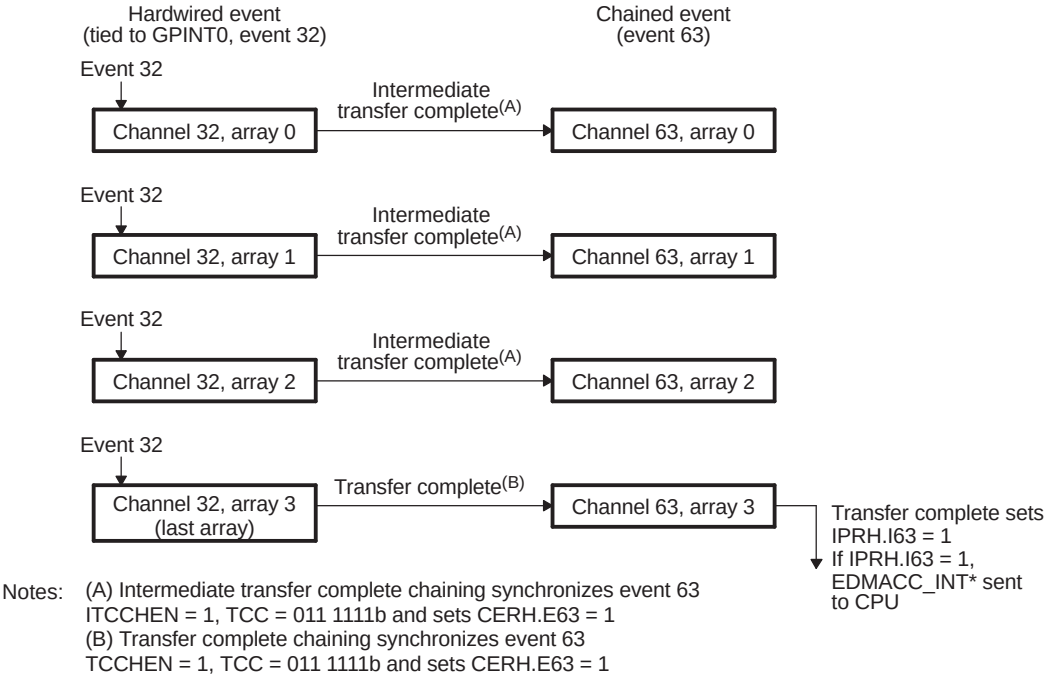



**Figure 3-19. Single Large Block Transfer Example**

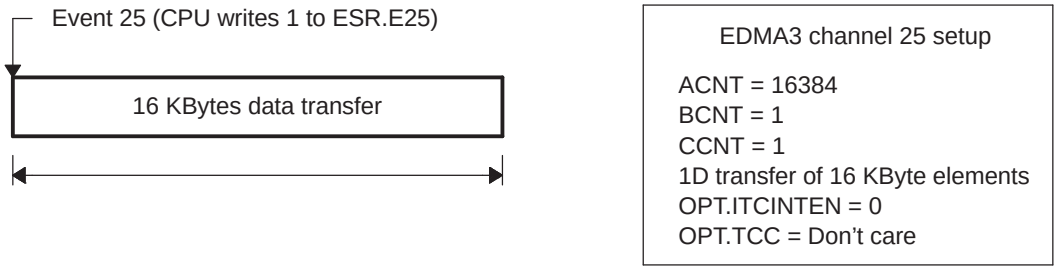
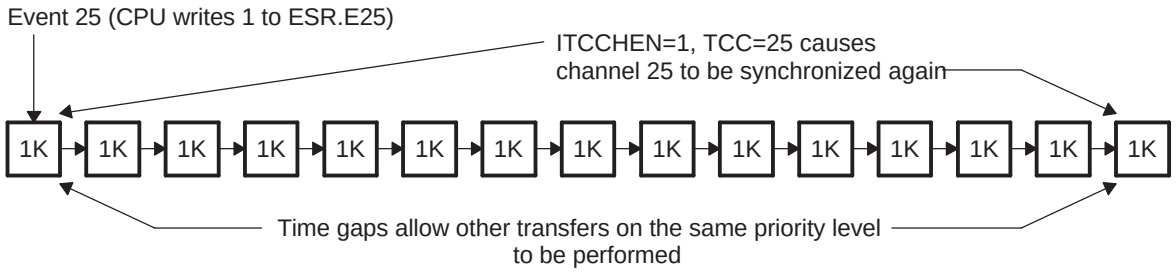


**Figure 3-20. Smaller Packet Data Transfers Example**




EDMA channel 25 setup

ACNT = 1024
BCNT = 16
CCNT = 1
OPT.SYNCDIM = A SYNC
OPT.ITCCHEN = 1
OPT.TCINTEN = 1
OPT.TCC = 25

# Registers

This chapter discusses the registers of the EDMA3 controller.

**Topic** **Page**

## 4.1 Register Memory Maps

See your device-specific data manual for the register memory maps.

## 4.2 Parameter RAM (PaRAM) Entries

Table 4-1 lists the parameter RAM (PaRAM) entries for the EDMA3 channel controller (EDMA3CC). See the device-specific data manual for the memory address of these registers.

**Table 4-1. EDMA3 Channel Controller (EDMA3CC) Parameter RAM (PaRAM) Entries**

| Offset | Acronym | Parameter | Section |
|---|---|---|---|
| 0h | OPT | Channel Options | Section 4.2.1 |
| 4h | SRC | Channel Source Address | Section 4.2.2 |
| 8h | A_B_CNT | A Count/B Count | Section 4.2.3 |
| Ch | DST | Channel Destination Address | Section 4.2.4 |
| 10h | SRC_DST_BIDX | Source B Index/Destination B Index | Section 4.2.5 |
| 14h | LINK_BCNTRLD | Link Address/B Count Reload | Section 4.2.6 |
| 18h | SRC_DST_CIDX | Source C Index/Destination C Index | Section 4.2.7 |
| 1Ch | CCNT | C Count | Section 4.2.8 |

### *4.2.1 Channel Options Parameter (OPT)*

The channel options parameter (OPT) is shown in Figure 4-1 and described in Table 4-2.

**Figure 4-1. Channel Options Parameter (OPT)**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | PRIVID | | | | ITCCHEN | TCCHEN | ITCINTEN | TCINTEN | Reserved | | TCC | |
| R-0 | | | | R-0 | | | | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R-0 | R/W-0 | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| TCC | | | | TCCMOD | FWID | | | Reserved | | | | STATIC | SYNCDIM | DAM | SAM |
| R/W-0 | | | | R/W-0 | R/W-0 | | | R-0 | | | | R/W-0 | R/W-0 | R/W-0 | R/W-0 |

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

**Table 4-2. Channel Options Parameters (OPT) Field Descriptions**

| Bit | Field | Value | Description |
|---|---|---|---|
| 31-28 | Reserved | 0 | Reserved |
| 27-24 | PRIVID | 0-Fh | Privilege identification for the external host/CPU/DMA that programmed this PaRAM set. This value is set with the EDMA3 master's privilege identification value when any part of the PaRAM set is written. |
| 23 | ITCCHEN | | Intermediate transfer completion chaining enable. |
| | | 0 | Intermediate transfer complete chaining is disabled. |
| | | 1 | Intermediate transfer complete chaining is enabled. |
| | | | When enabled, the chained event register (CER/CERH) bit is set on every intermediate chained transfer completion (upon completion of every intermediate TR in the PaRAM set, except the final TR in the PaRAM set). The bit (position) set in CER or CERH is the TCC value specified. |
| 22 | TCCHEN | | Transfer complete chaining enable. |
| | | 0 | Transfer complete chaining is disabled. |
| | | 1 | Transfer complete chaining is enabled. |
| | | | When enabled, the chained event register (CER/CERH) bit is set on final chained transfer completion (upon completion of the final TR in the PaRAM set). The bit (position) set in CER or CERH is the TCC value specified. |

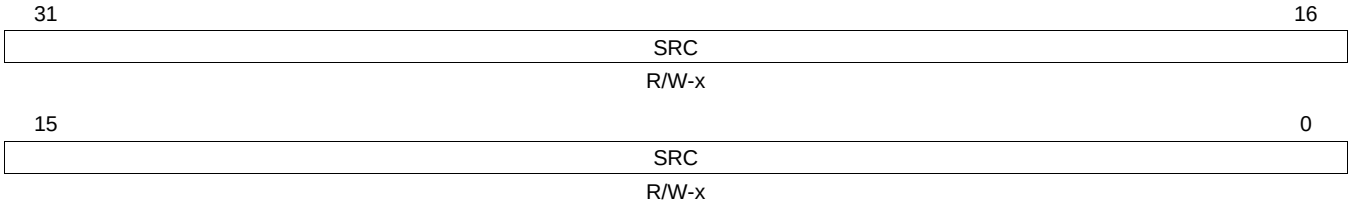
**Table 4-2. Channel Options Parameters (OPT) Field Descriptions (continued)**

| Bit | Field | Value | Description |
|---|---|---|---|
| 21 | ITCINTEN | | Intermediate transfer completion interrupt enable. |
| | | 0 | Intermediate transfer complete interrupt is disabled. |
| | | 1 | Intermediate transfer complete interrupt is enabled. |
| | | | When enabled, the interrupt pending register (IPR/IPRH) bit is set on every intermediate transfer completion (upon completion of every intermediate TR in the PaRAM set, except the final TR in the PaRAM set). The bit (position) set in IPR or IPRH is the TCC value specified. In order to generate a completion interrupt to the CPU, the corresponding IER[TCC]/IERH[TCC] bit must be set to 1. |
| 20 | TCINTEN | | Transfer complete interrupt enable. |
| | | 0 | Transfer complete interrupt is disabled. |
| | | 1 | Transfer complete interrupt is enabled. |
| | | | When enabled, the interrupt pending register (IPR/IPRH) bit is set on transfer completion (upon completion of the final TR in the PaRAM set). The bit (position) set in IPR or IPRH is the TCC value specified. In order to generate a completion interrupt to the CPU, the corresponding IER[TCC]/IERH[TCC] bit must be set to 1. |
| 19 | Reserved | 0 | Reserved. Always write 0 to this bit. |
| 18 | Reserved | 0 | Reserved |
| 17-12 | TCC | 0-3Fh | Transfer complete code. This 6-bit code is used to set the relevant bit in chaining enable register (CER[TCC]/CERH[TCC]) for chaining or in interrupt pending register (IPR[TCC]/IPRH[TCC]) for interrupts. |
| 11 | TCCMODE | | Transfer complete code mode. Indicates the point at which a transfer is considered completed for chaining and interrupt generation. |
| | | 0 | Normal completion: A transfer is considered completed after the data has been transferred. |
| | | 1 | Early completion: A transfer is considered completed after the EDMA3CC submits a TR to the EDMA3TC. TC may still be transferring data when interrupt/chain is triggered. |
| 10-8 | FWID | 0-7h | FIFO Width. Applies if either SAM or DAM is set to constant addressing mode. |
| | | 0 | FIFO width is 8-bit. |
| | | 1h | FIFO width is 16-bit. |
| | | 2h | FIFO width is 32-bit. |
| | | 3h | FIFO width is 64-bit. |
| | | 4h | FIFO width is 128-bit. |
| | | 5h | FIFO width is 256-bit. |
| | | 6h-7h | Reserved |
| 7-4 | Reserved | 0 | Reserved |
| 3 | STATIC | | Static PaRAM set. |
| | | 0 | PaRAM set is not static. PaRAM set is updated or linked after TR is submitted. A value of 0 should be used for DMA channels and for nonfinal transfers in a linked list of QDMA transfers. |
| | | 1 | PaRAM set is static. PaRAM set is not updated or linked after TR is submitted. A value of 1 should be used for isolated QDMA transfers or for the final transfer in a linked list of QDMA transfers. |
| 2 | SYNCDIM | | Transfer synchronization dimension. |
| | | 0 | A-synchronized. Each event triggers the transfer of a single array of ACNT bytes. |
| | | 1 | AB-synchronized. Each event triggers the transfer of BCNT arrays of ACNT bytes. |
| 1 | DAM | | Destination address mode. |
| | | 0 | Increment (INCR) mode. Destination addressing within an array increments. Destination is not a FIFO. |
| | | 1 | Constant addressing (CONST) mode. Destination addressing within an array wraps around upon reaching FIFO width. |
| 0 | SAM | | Source address mode. |
| | | 0 | Increment (INCR) mode. Source addressing within an array increments. Source is not a FIFO. |
| | | 1 | Constant addressing (CONST) mode. Source addressing within an array wraps around upon reaching FIFO width. |

### 4.2.2 Channel Source Address Parameter (SRC)

The channel source address parameter (SRC) specifies the starting byte address of the source. The SRC is shown in Figure 4-2 and described in Table 4-3.

**Figure 4-2. Channel Source Address Parameter (SRC)**




LEGEND: R/W = Read/Write; -*n* = value after reset; -x = value is indeterminate after reset

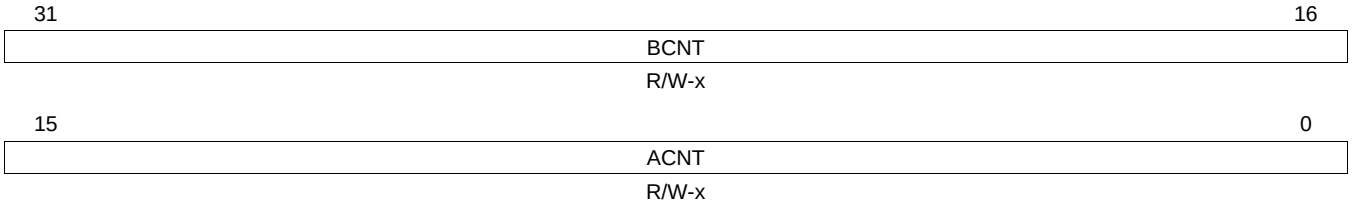
**Table 4-3. Channel Source Address Parameter (SRC) Field Descriptions**

| Bit | Field | Value | Description |
|---|---|---|---|
| 31-0 | SRC | 0-FFFF FFFFh | Source address. Specifies the starting byte address of the source. |

### 4.2.3 A Count/B Count Parameter (A_B_CNT)

The A count/B count parameter (A_B_CNT) specifies the number of bytes within the 1st dimension of a transfer and the number of arrays of length ACNT. The A_B_CNT is shown in Figure 4-3 and described in Table 4-4.

**Figure 4-3. A Count/B Count Parameter (A_B_CNT)**




LEGEND: R/W = Read/Write; -*n* = value after reset; -x = value is indeterminate after reset

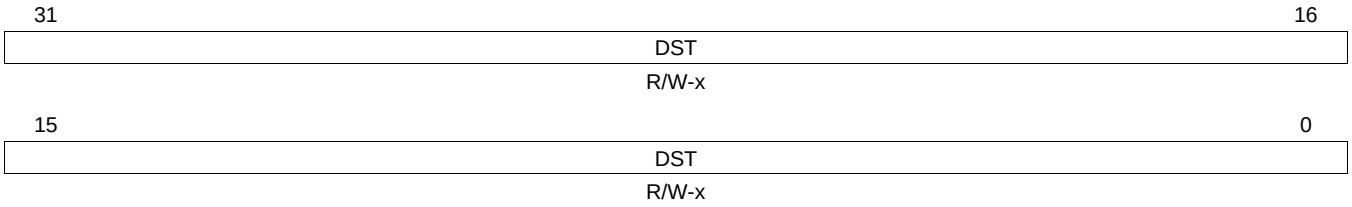
**Table 4-4. A Count/B Count Parameter (A_B_CNT) Field Descriptions**

| Bit | Field | Value | Description |
|---|---|---|---|
| 31-16 | BCNT | 0-FFFFh | B count. Unsigned value specifying the number of arrays in a frame, where an array is ACNT bytes. Valid values range from 1 to 65 535. |
| 15-0 | ACNT | 0-FFFFh | A count for 1st Dimension. Unsigned value specifying the number of contiguous bytes within an array (first dimension of the transfer). Valid values range from 1 to 65 535. |

### 4.2.4 Channel Destination Address Parameter (DST)

The channel destination address parameter (DST) specifies the starting byte address of the source. The DST is shown in Figure 4-4 and described in Table 4-5.

**Figure 4-4. Channel Destination Address Parameter (DST)**




LEGEND: R/W = Read/Write; -*n* = value after reset; -x = value is indeterminate after reset

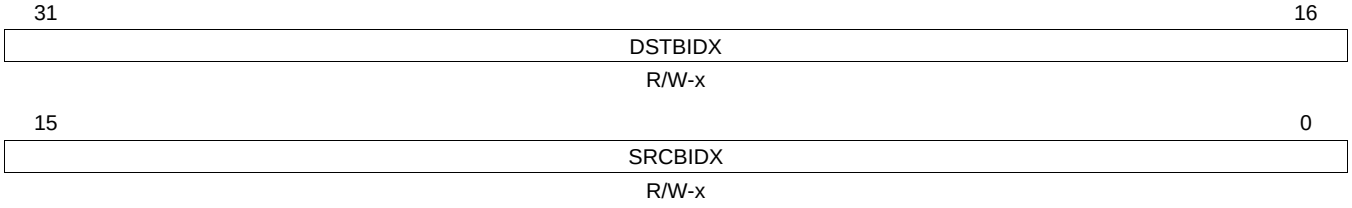
**Table 4-5. Channel Destination Address Parameter (DST) Field Descriptions**

| Bit | Field | Value | Description |
|---|---|---|---|
| 31-0 | DST | 0-FFFF FFFFh | Destination address. Specifies the starting byte address of the destination where data is transferred. |

### 4.2.5 Source B Index/Destination B Index Parameter (SRC_DST_BIDX)

The source B index/destination B index parameter (SRC_DST_BIDX) specifies the value (2s complement) used for source address modification between each array in the 2nd dimension and the value (2s complement) used for destination address modification between each array in the 2nd dimension. The SRC_DST_BIDX is shown in Figure 4-5 and described in Table 4-6.

**Figure 4-5. Source B Index/Destination B Index Parameter (SRC_DST_BIDX)**




LEGEND: R/W = Read/Write; -*n* = value after reset; -x = value is indeterminate after reset

**Table 4-6. Source B Index/Destination B Index Parameter (SRC_DST_BIDX) Field Descriptions**

| Bit | Field | Value | Description |
|---|---|---|---|
| 31-16 | DSTBIDX | 0-FFFFh | Destination B index. Signed value specifying the byte address offset between destination arrays within a frame (2nd dimension). Valid values range from –32 768 and 32 767. |
| 15-0 | SRCBIDX | 0-FFFFh | Source B index. Signed value specifying the byte address offset between source arrays within a frame (2nd dimension). Valid values range from –32 768 and 32 767. |

### 4.2.6 Link Address/B Count Reload Parameter (LINK_BCNTRLD)

The link address/B count reload parameter (LINK_BCNTRLD) specifies the byte address offset in the PaRAM from which the EDMA3CC loads/reloads the next PaRAM set during linking and the value used to reload the BCNT field in the A count/B count parameter (A_B_CNT) once the last array in the 2nd dimension is transferred. The LINK_BCNTRLD is shown in Figure 4-6 and described in Table 4-7.

**Figure 4-6. Link Address/B Count Reload Parameter (LINK_BCNTRLD)**

| 31 | 16 |
|---|---|
| BCNTRLD | |
| R/W-x | |

| 15 | 0 |
|---|---|
| LINK | |
| R/W-x | |

LEGEND: R/W = Read/Write; -*n* = value after reset; -x = value is indeterminate after reset

**Table 4-7. Link Address/B Count Reload Parameter (LINK_BCNTRLD) Field Descriptions**

| Bit | Field | Value | Description |
|---|---|---|---|
| 31-16 | BCNTRLD | 0-FFFFh | B count reload. The count value used to reload BCNT in the A count/B count parameter (A_B_CNT) when BCNT decrements to 0 (TR submitted for the last array in 2nd dimension). Only relevant in A-synchronized transfers. |
| 15-0 | LINK | 0-FFFFh | Link address. The PaRAM address containing the PaRAM set to be linked (copied from) when the current PaRAM set is exhausted. You must program the link address to point to a valid aligned 32-byte PaRAM set. The 5 LSBs of the LINK field should be cleared to 0. A value of FFFFh specifies a null link. |

### 4.2.7 Source C Index/Destination C Index Parameter (SRC_DST_CIDX)

The source C index/destination C index parameter (SRC_DST_CIDX) specifies the value (2s complement) used for source address modification between each array in the 3rd dimension and the value (2s complement) used for destination address modification between each array in the 3rd dimension. The SRC_DST_CIDX is shown in Figure 4-7 and described in Table 4-8.

**Figure 4-7. Source C Index/Destination C Index Parameter (SRC_DST_CIDX)**

| 31 | 16 |
|---|---|
| DSTCIDX | |
| R/W-x | |

| 15 | 0 |
|---|---|
| SRCCIDX | |
| R/W-x | |

LEGEND: R/W = Read/Write; -*n* = value after reset; -x = value is indeterminate after reset

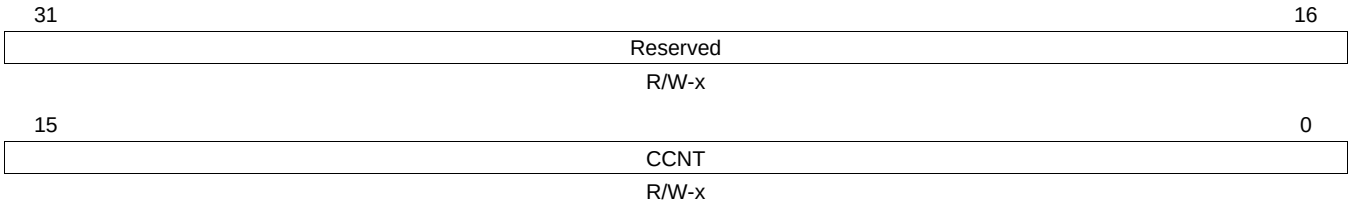
**Table 4-8. Source C Index/Destination C Index Parameter (SRC_DST_CIDX) Field Descriptions**

| Bit | Field | Value | Description |
|---|---|---|---|
| 31-16 | DSTCIDX | 0-FFFFh | Destination C index. Signed value specifying the byte address offset between frames within a block (3rd dimension). Valid values range from –32 768 and 32 767. |
| 15-0 | SRCCIDX | 0-FFFFh | Source C index. Signed value specifying the byte address offset between frames within a block (3rd dimension). Valid values range from –32 768 and 32 767. |

### 4.2.8 C Count Parameter (CCNT)

The C count parameter (CCNT) specifies the number of frames in a block. The CCNT is shown in Figure 4-8 and described in Table 4-9.

**Figure 4-8. C Count Parameter (CCNT)**




LEGEND: R/W = Read/Write; -*n* = value after reset; -x = value is indeterminate after reset

**Table 4-9. C Count Parameter (CCNT) Field Descriptions**

| Bit | Field | Value | Description |
|---|---|---|---|
| 31-16 | Reserved | 0 | Reserved |
| 15-0 | CCNT | 0-FFFFh | C counter. Unsigned value specifying the number of frames in a block, where a frame is BCNT arrays of ACNT bytes. Valid values range from 1 to 65 535. |

## 4.3 EDMA3 Channel Controller (EDMA3CC) Registers

Table 4-10 lists the memory-mapped registers for the EDMA3 channel controller (EDMA3CC). See the device-specific data manual for the memory address of these registers and for the shadow region addresses. All other register offset addresses not listed in Table 4-10 should be considered as reserved locations and the register contents should not be modified.

**Table 4-10. EDMA3 Channel Controller (EDMA3CC) Registers**

| Offset | Acronym | Register Description | Section |
|---|---|---|---|
| 00h | PID | Peripheral Identification Register | Section 4.3.1.1 |
| 04h | CCCFG | EDMA3CC Configuration Register | Section 4.3.1.2 |
| | | **Global Registers** | |
| 0200h | QCHMAP0 | QDMA Channel 0 Mapping Register | Section 4.3.1.3 |
| 0204h | QCHMAP1 | QDMA Channel 1 Mapping Register | Section 4.3.1.3 |
| 0208h | QCHMAP2 | QDMA Channel 2 Mapping Register | Section 4.3.1.3 |
| 020Ch | QCHMAP3 | QDMA Channel 3 Mapping Register | Section 4.3.1.3 |
| 0210h | QCHMAP4 | QDMA Channel 4 Mapping Register | Section 4.3.1.3 |
| 0214h | QCHMAP5 | QDMA Channel 5 Mapping Register | Section 4.3.1.3 |
| 0218h | QCHMAP6 | QDMA Channel 6 Mapping Register | Section 4.3.1.3 |
| 021Ch | QCHMAP7 | QDMA Channel 7 Mapping Register | Section 4.3.1.3 |
| 0240h | DMAQNUM0 | DMA Channel Queue Number Register 0 | Section 4.3.1.4 |
| 0244h | DMAQNUM1 | DMA Channel Queue Number Register 1 | Section 4.3.1.4 |
| 0248h | DMAQNUM2 | DMA Channel Queue Number Register 2 | Section 4.3.1.4 |
| 024Ch | DMAQNUM3 | DMA Channel Queue Number Register 3 | Section 4.3.1.4 |
| 0250h | DMAQNUM4 | DMA Channel Queue Number Register 4 | Section 4.3.1.4 |
| 0254h | DMAQNUM5 | DMA Channel Queue Number Register 5 | Section 4.3.1.4 |
| 0258h | DMAQNUM6 | DMA Channel Queue Number Register 6 | Section 4.3.1.4 |
| 025Ch | DMAQNUM7 | DMA Channel Queue Number Register 7 | Section 4.3.1.4 |
| 0260h | QDMAQNUM | QDMA Channel Queue Number Register | Section 4.3.1.5 |
| 0284h | QUEPRI | Queue Priority Register | Section 4.3.1.6 |
| 0300h | EMR | Event Missed Register | Section 4.3.2.1 |
| 0304h | EMRH | Event Missed Register High | Section 4.3.2.1 |
| 0308h | EMCR | Event Missed Clear Register | Section 4.3.2.2 |
| 030Ch | EMCRH | Event Missed Clear Register High | Section 4.3.2.2 |
| 0310h | QEMR | QDMA Event Missed Register | Section 4.3.2.3 |
| 0314h | QEMCR | QDMA Event Missed Clear Register | Section 4.3.2.4 |
| 0318h | CCERR | EDMA3CC Error Register | Section 4.3.2.5 |
| 031Ch | CCERRCLR | EDMA3CC Error Clear Register | Section 4.3.2.6 |
| 0320h | EEVAL | Error Evaluate Register | Section 4.3.2.7 |
| 0340h | DRAE0 | DMA Region Access Enable Register for Region 0 | Section 4.3.3.1 |
| 0344h | DRAEH0 | DMA Region Access Enable Register High for Region 0 | Section 4.3.3.1 |
| 0348h | DRAE1 | DMA Region Access Enable Register for Region 1 | Section 4.3.3.1 |
| 034Ch | DRAEH1 | DMA Region Access Enable Register High for Region 1 | Section 4.3.3.1 |
| 0350h | DRAE2 | DMA Region Access Enable Register for Region 2 | Section 4.3.3.1 |
| 0354h | DRAEH2 | DMA Region Access Enable Register High for Region 2 | Section 4.3.3.1 |
| 0358h | DRAE3 | DMA Region Access Enable Register for Region 3 | Section 4.3.3.1 |
| 035Ch | DRAEH3 | DMA Region Access Enable Register High for Region 3 | Section 4.3.3.1 |
| 0380h | QRAE0 | QDMA Region Access Enable Register for Region 0 | Section 4.3.3.2 |
| 0384h | QRAE1 | QDMA Region Access Enable Register for Region 1 | Section 4.3.3.2 |
| 0388h | QRAE2 | QDMA Region Access Enable Register for Region 2 | Section 4.3.3.2 |
| 038Ch | QRAE3 | QDMA Region Access Enable Register for Region 3 | Section 4.3.3.2 |

**Table 4-10. EDMA3 Channel Controller (EDMA3CC) Registers (continued)**

| Offset | Acronym | Register Description | Section |
|---|---|---|---|
| 0400h-043Ch | Q0E0-Q0E15 | Event Queue Entry Registers Q0E0-Q0E15 | Section 4.3.4.1 |
| 0440h-047Ch | Q1E0-Q1E15 | Event Queue Entry Registers Q1E0-Q1E15 | Section 4.3.4.1 |
| 0480h-04BCh | Q2E0-Q2E15 | Event Queue Entry Registers Q2E0-Q2E15 | Section 4.3.4.1 |
| 0600h | QSTAT0 | Queue 0 Status Register | Section 4.3.4.2 |
| 0604h | QSTAT1 | Queue 1 Status Register | Section 4.3.4.2 |
| 0608h | QSTAT2 | Queue 2 Status Register | Section 4.3.4.2 |
| 0620h | QWMTHRA | Queue Watermark Threshold A Register | Section 4.3.4.3 |
| 0640h | CCSTAT | EDMA3CC Status Register | Section 4.3.4.4 |
| | | **Global Channel Registers** | |
| 1000h | ER | Event Register | Section 4.3.5.1 |
| 1004h | ERH | Event Register High | Section 4.3.5.1 |
| 1008h | ECR | Event Clear Register | Section 4.3.5.2 |
| 100Ch | ECRH | Event Clear Register High | Section 4.3.5.2 |
| 1010h | ESR | Event Set Register | Section 4.3.5.3 |
| 1014h | ESRH | Event Set Register High | Section 4.3.5.3 |
| 1018h | CER | Chained Event Register | Section 4.3.5.4 |
| 101Ch | CERH | Chained Event Register High | Section 4.3.5.4 |
| 1020h | EER | Event Enable Register | Section 4.3.5.5 |
| 1024h | EERH | Event Enable Register High | Section 4.3.5.5 |
| 1028h | EECR | Event Enable Clear Register | Section 4.3.5.6 |
| 102Ch | EECRH | Event Enable Clear Register High | Section 4.3.5.6 |
| 1030h | EESR | Event Enable Set Register | Section 4.3.5.7 |
| 1034h | EESRH | Event Enable Set Register High | Section 4.3.5.7 |
| 1038h | SER | Secondary Event Register | Section 4.3.5.8 |
| 103Ch | SERH | Secondary Event Register High | Section 4.3.5.8 |
| 1040h | SECR | Secondary Event Clear Register | Section 4.3.5.9 |
| 1044h | SECRH | Secondary Event Clear Register High | Section 4.3.5.9 |
| 1050h | IER | Interrupt Enable Register | Section 4.3.6.1 |
| 1054h | IERH | Interrupt Enable Register High | Section 4.3.6.1 |
| 1058h | IECR | Interrupt Enable Clear Register | Section 4.3.6.2 |
| 105Ch | IECRH | Interrupt Enable Clear Register High | Section 4.3.6.2 |
| 1060h | IESR | Interrupt Enable Set Register | Section 4.3.6.3 |
| 1064h | IESRH | Interrupt Enable Set Register High | Section 4.3.6.3 |
| 1068h | IPR | Interrupt Pending Register | Section 4.3.6.4 |
| 106Ch | IPRH | Interrupt Pending Register High | Section 4.3.6.4 |
| 1070h | ICR | Interrupt Clear Register | Section 4.3.6.5 |
| 1074h | ICRH | Interrupt Clear Register High | Section 4.3.6.5 |
| 1078h | IEVAL | Interrupt Evaluate Register | Section 4.3.6.6 |
| 1080h | QER | QDMA Event Register | Section 4.3.7.1 |
| 1084h | QEER | QDMA Event Enable Register | Section 4.3.7.2 |
| 1088h | QEECR | QDMA Event Enable Clear Register | Section 4.3.7.3 |
| 108Ch | QEESR | QDMA Event Enable Set Register | Section 4.3.7.4 |
| 1090h | QSER | QDMA Secondary Event Register | Section 4.3.7.5 |
| 1094h | QSECR | QDMA Secondary Event Clear Register | Section 4.3.7.6 |

**Table 4-10. EDMA3 Channel Controller (EDMA3CC) Registers (continued)**

| Offset | Acronym | Register Description | Section |
|---|---|---|---|
| | | **Shadow Region 0 Channel Registers** | |
| 2000h | ER | Event Register | |
| 2004h | ERH | Event Register High | |
| 2008h | ECR | Event Clear Register | |
| 200Ch | ECRH | Event Clear Register High | |
| 2010h | ESR | Event Set Register | |
| 2014h | ESRH | Event Set Register High | |
| 2018h | CER | Chained Event Register | |
| 201Ch | CERH | Chained Event Register High | |
| 2020h | EER | Event Enable Register | |
| 2024h | EERH | Event Enable Register High | |
| 2028h | EECR | Event Enable Clear Register | |
| 202Ch | EECRH | Event Enable Clear Register High | |
| 2030h | EESR | Event Enable Set Register | |
| 2034h | EESRH | Event Enable Set Register High | |
| 2038h | SER | Secondary Event Register | |
| 203Ch | SERH | Secondary Event Register High | |
| 2040h | SECR | Secondary Event Clear Register | |
| 2044h | SECRH | Secondary Event Clear Register High | |
| 2050h | IER | Interrupt Enable Register | |
| 2054h | IERH | Interrupt Enable Register High | |
| 2058h | IECR | Interrupt Enable Clear Register | |
| 205Ch | IECRH | Interrupt Enable Clear Register High | |
| 2060h | IESR | Interrupt Enable Set Register | |
| 2064h | IESRH | Interrupt Enable Set Register High | |
| 2068h | IPR | Interrupt Pending Register | |
| 206Ch | IPRH | Interrupt Pending Register High | |
| 2070h | ICR | Interrupt Clear Register | |
| 2074h | ICRH | Interrupt Clear Register High | |
| 2078h | IEVAL | Interrupt Evaluate Register | |
| 2080h | QER | QDMA Event Register | |
| 2084h | QEER | QDMA Event Enable Register | |
| 2088h | QEECR | QDMA Event Enable Clear Register | |
| 208Ch | QEESR | QDMA Event Enable Set Register | |
| 2090h | QSER | QDMA Secondary Event Register | |
| 2094h | QSECR | QDMA Secondary Event Clear Register | |

**Table 4-10. EDMA3 Channel Controller (EDMA3CC) Registers (continued)**

| Offset | Acronym | Register Description | Section |
|---|---|---|---|
| | | **Shadow Region 1 Channel Registers** | |
| 2200h | ER | Event Register | |
| 2204h | ERH | Event Register High | |
| 2208h | ECR | Event Clear Register | |
| 220Ch | ECRH | Event Clear Register High | |
| 2210h | ESR | Event Set Register | |
| 2214h | ESRH | Event Set Register High | |
| 2218h | CER | Chained Event Register | |
| 221Ch | CERH | Chained Event Register High | |
| 2220h | EER | Event Enable Register | |
| 2224h | EERH | Event Enable Register High | |
| 2228h | EECR | Event Enable Clear Register | |
| 222Ch | EECRH | Event Enable Clear Register High | |
| 2230h | EESR | Event Enable Set Register | |
| 2234h | EESRH | Event Enable Set Register High | |
| 2238h | SER | Secondary Event Register | |
| 223Ch | SERH | Secondary Event Register High | |
| 2240h | SECR | Secondary Event Clear Register | |
| 2244h | SECRH | Secondary Event Clear Register High | |
| 2250h | IER | Interrupt Enable Register | |
| 2254h | IERH | Interrupt Enable Register High | |
| 2258h | IECR | Interrupt Enable Clear Register | |
| 225Ch | IECRH | Interrupt Enable Clear Register High | |
| 2260h | IESR | Interrupt Enable Set Register | |
| 2264h | IESRH | Interrupt Enable Set Register High | |
| 2268h | IPR | Interrupt Pending Register | |
| 226Ch | IPRH | Interrupt Pending Register High | |
| 2270h | ICR | Interrupt Clear Register | |
| 2274h | ICRH | Interrupt Clear Register High | |
| 2278h | IEVAL | Interrupt Evaluate Register | |
| 2280h | QER | QDMA Event Register | |
| 2284h | QEER | QDMA Event Enable Register | |
| 2288h | QEECR | QDMA Event Enable Clear Register | |
| 228Ch | QEESR | QDMA Event Enable Set Register | |
| 2290h | QSER | QDMA Secondary Event Register | |
| 2294h | QSECR | QDMA Secondary Event Clear Register | |
| 4000h-4FFFh | — | Parameter RAM (PaRAM) | |

## 4.3.1 Global Registers

### 4.3.1.1 Peripheral Identification Register (PID)

The peripheral identification register (PID) uniquely identifies the EDMA3CC and the specific revision of the EDMA3CC. The PID is shown in Figure 4-9 and described in Table 4-11.

**Figure 4-9. Peripheral ID Register (PID)**

| 31 | 16 |
|---|---|
| PID | |
| R-4001h | |

| 15 | 0 |
|---|---|
| PID | |
| R-5300h | |

LEGEND: R = Read only; -*n* = value after reset

**Table 4-11. Peripheral ID Register (PID) Field Descriptions**

| Bit | Field | Value | Description |
|---|---|---|---|
| 31-0 | PID | 4001 5300h | Peripheral identifier.<br>Uniquely identifies the EDMA3CC and the specific revision of the EDMA3CC. |

### 4.3.1.2 EDMA3CC Configuration Register (CCCFG)

The EDMA3CC configuration register (CCCFG) provides the features/resources for the EDMA3CC in a particular device. The CCCFG is shown in Figure 4-10 and described in Table 4-12.

**Figure 4-10. EDMA3CC Configuration Register (CCCFG)**

| 31 | 26 | 25 | 24 |
|---|---|---|---|
| Reserved | | MP_EXIST | CHMAP_EXIST |
| R-x | | R-0 | R-0 |

| 23 | 22 | 21 | 20 | 19 | 18 | 16 |
|---|---|---|---|---|---|---|
| Reserved | | NUM_REGN | | Reserved | NUM_EVQUE | |
| R-0 | | R-2h | | R-x | R-2h | |

| 15 | 14 | 12 | 11 | 10 | 8 |
|---|---|---|---|---|---|
| Reserved | NUM_PAENTRY | | Reserved | NUM_INTCH | |
| R-x | R-3h | | R-x | R-4h | |

| 7 | 6 | 4 | 3 | 2 | 0 |
|---|---|---|---|---|---|
| Reserved | NUM_QDMACH | | Reserved | NUM_DMACH | |
| R-x | R-4h | | R-x | R-5h | |

LEGEND: R = Read only; -*n* = value after reset; -x = value is indeterminate after reset

**Table 4-12. EDMA3CC Configuration Register (CCCFG) Field Descriptions**

| Bit | Field | Value | Description |
|---|---|---|---|
| 31-26 | Reserved | 0-3Fh | Reserved |
| 25 | MP_EXIST | | Memory protection existence. |
| | | 0 | No memory protection. |
| | | 1 | Reserved |
| 24 | CHMAP_EXIST | | Channel mapping existence |
| | | 0 | No channel mapping. This implies that there is fixed association for a channel number to a parameter entry number or, in other words, PaRAM entry *n* corresponds to channel *n*. |
| | | 1 | Reserved |
| 23-22 | Reserved | 0 | Reserved |
| 21-20 | NUM_REGN | 0-3h | Number of shadow regions. |
| | | 0-1 | Reserved |
| | | 2h | 4 regions |
| | | 3h | Reserved |
| 19 | Reserved | 0 | Reserved |
| 18-16 | NUM_EVQUE | 0-7h | Number of queues/number of TCs. |
| | | 0-1h | Reserved |
| | | 2h | 3 EDMA3TC/Event Queues |
| | | 3h-7h | Reserved |
| 15 | Reserved | 0 | Reserved |
| 14-12 | NUM_PAENTRY | 0-7h | Number of PaRAM sets. |
| | | 0-2h | Reserved |
| | | 3h | 128 sets |
| | | 4h-7h | Reserved |
| 11 | Reserved | 0 | Reserved |
| 10-8 | NUM_INTCH | 0-7h | Number of interrupt channels. |
| | | 0-3h | Reserved |
| | | 4h | 64 interrupt channels |
| | | 5h-7h | Reserved |
| 7 | Reserved | 0 | Reserved |
| 6-4 | NUM_QDMACH | 0-7h | Number of QDMA channels. |
| | | 0-3h | Reserved |
| | | 4h | 8 QDMA channels |
| | | 5h-7h | Reserved |
| 3 | Reserved | 0 | Reserved |
| 2-0 | NUM_DMACH | 0-7h | Number of DMA channels. |
| | | 0-4h | Reserved |
| | | 5h | 64 DMA channels |
| | | 6h-7h | Reserved |

### 4.3.1.3 QDMA Channel *n* Mapping Register (QCHMAP*n*)

Each QDMA channel in EDMA3CC can be associated with any PaRAM set available on the device. Furthermore, the specific trigger word (0-7) of the PaRAM set can be programmed. The PaRAM set association and trigger word for every QDMA channel register is configurable using the QDMA channel *n* mapping register (QCHMAP*n*). The QCHMAP*n* is shown in Figure 4-11 and described in Table 4-13.

**NOTE:** At reset the QDMA channel mapping registers for all QDMA channels point to the PaRAM set 0. Prior to using any QDMA channel, QCHMAP*n* should be programmed appropriately to point to a different PaRAM set.

**Figure 4-11. QDMA Channel *n* Mapping Register (QCHMAP*n*)**

| 31 | 16 |
|---|---|
| Reserved | |
| R-0 | |

| 15 | 14 | 13 | 5 | 4 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| Reserved | | PAENTRY | | TRWORD | | Reserved | |
| R-0 | | R/W-0 | | R/W-0 | | R-0 | |

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

**Table 4-13. QDMA Channel *n* Mapping Register (QCHMAP*n*) Field Descriptions**

| Bit | Field | Value | Description |
|---|---|---|---|
| 31-14 | Reserved | 0 | Reserved |
| 13-5 | PAENTRY | 0-1FFh | PAENTRY points to the PaRAM set number for QDMA channel *n*. |
| | | 0-7Fh | PaRAM set number 0 through 127 |
| | | 80h-1FFh | Reserved |
| 4-2 | TRWORD | 0-7h | Points to the specific PaRAM entry or the trigger word in the PaRAM set pointed to by PAENTRY. A write to the trigger word results in a QDMA event being recognized. |
| 1-0 | Reserved | 0 | Reserved |

### 4.3.1.4 DMA Channel Queue Number Register *n* (DMAQNUM*n*)

The DMA channel queue number register *n* (DMAQNUM*n*) allows programmability of each of the 64 DMA channels in the EDMA3CC to submit its associated synchronization event to any event queue in the EDMA3CC. At reset, all channels point to event queue 0. The DMAQNUM*n* is shown in Figure 4-12 and described in Table 4-14. Table 4-15 shows the channels and their corresponding bits in DMAQNUM*n*.

**NOTE:** Since the event queues in EDMA3CC have a fixed association to the transfer controllers, that is, Q0 TRs are submitted to TC0, Q1 TRs are submitted to TC1, and Q2 TRs are submitted to TC2, by programming DMAQNUM*n* for a particular DMA channel also dictates which transfer controller is utilized for the data movement (or which EDMA3TC receives the TR request).

**Figure 4-12. DMA Channel Queue Number Register *n* (DMAQNUM*n*)**

| 31 | 30 28 | 27 | 26 24 | 23 | 22 20 | 19 | 18 16 |
|---|---|---|---|---|---|---|---|
| Rsvd | E*n* | Rsvd | E*n* | Rsvd | E*n* | Rsvd | E*n* |
| R-0 | R/W-0 | R-0 | R/W-0 | R-0 | R/W-0 | R-0 | R/W-0 |

| 15 | 14 12 | 11 | 10 8 | 7 | 6 4 | 3 | 2 0 |
|---|---|---|---|---|---|---|---|
| Rsvd | E*n* | Rsvd | E*n* | Rsvd | E*n* | Rsvd | E*n* |
| R-0 | R/W-0 | R-0 | R/W-0 | R-0 | R/W-0 | R-0 | R/W-0 |

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

**Table 4-14. DMA Channel Queue Number Register *n* (DMAQNUM*n*) Field Descriptions**

| Bit | Field | Value | Description |
|---|---|---|---|
| 31-0 | E*n* | 0-7h | DMA queue number. Contains the event queue number to be used for the corresponding DMA channel. Programming DMAQNUM*n* for an event queue number to a value more then the number of queues available in the EDMA3CC results in undefined behavior. |
| | | 0 | Event *n* is queued on Q0. |
| | | 1h | Event *n* is queued on Q1. |
| | | 2h | Event *n* is queued on Q2. |
| | | 3h-7h | Reserved |

**Table 4-15. Bits in DMAQNUM*n***

| | DMAQNUM*n* | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **E*n* bit** | **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** |
| 0-2 | E0 | E8 | E16 | E24 | E32 | E40 | E48 | E56 |
| 4-6 | E1 | E9 | E17 | E25 | E33 | E41 | E49 | E57 |
| 8-10 | E2 | E10 | E18 | E26 | E34 | E42 | E50 | E58 |
| 12-14 | E3 | E11 | E19 | E27 | E35 | E43 | E51 | E59 |
| 16-18 | E4 | E12 | E20 | E28 | E36 | E44 | E52 | E60 |
| 20-22 | E5 | E13 | E21 | E29 | E37 | E45 | E53 | E61 |
| 24-26 | E6 | E14 | E22 | E30 | E38 | E46 | E54 | E62 |
| 28-30 | E7 | E15 | E23 | E31 | E39 | E47 | E55 | E63 |

### 4.3.1.5 QDMA Channel Queue Number Register (QDMAQNUM)

The QDMA channel queue number register (QDMAQNUM) is used to program all the QDMA channels in the EDMA3CC to submit the associated QDMA event to any of the event queues in the EDMA3CC. The QDMAQNUM is shown in Figure 4-13 and described in Table 4-16.

**Figure 4-13. QDMA Channel Queue Number Register (QDMAQNUM)**

| 31 | 30 28 | 27 | 26 24 | 23 | 22 20 | 19 | 18 16 |
|---|---|---|---|---|---|---|---|
| Rsvd | E7 | Rsvd | E6 | Rsvd | E5 | Rsvd | E4 |
| R-0 | R/W-0 | R-0 | R/W-0 | R-0 | R/W-0 | R-0 | R/W-0 |

| 15 | 14 12 | 11 | 10 8 | 7 | 6 4 | 3 | 2 0 |
|---|---|---|---|---|---|---|---|
| Rsvd | E3 | Rsvd | E2 | Rsvd | E1 | Rsvd | E0 |
| R-0 | R/W-0 | R-0 | R/W-0 | R-0 | R/W-0 | R-0 | R/W-0 |

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

**Table 4-16. QDMA Channel Queue Number Register (QDMAQNUM) Field Descriptions**

| Bit | Field | Value | Description |
|---|---|---|---|
| 31-0 | E*n* | 0-7h | QDMA queue number. Contains the event queue number to be used for the corresponding QDMA channel. |
| | | 0 | Event *n* is queued on Q0. |
| | | 1h | Event *n* is queued on Q1. |
| | | 2h | Event *n* is queued on Q2. |
| | | 3h-7h | Reserved |

#### 4.3.1.6 Queue Priority Register (QUEPRI)

The queue priority register (QUEPRI) allows you to change the priority of the individual queues and the priority of the transfer request (TR) associated with the events queued in the queue. Since the queue to EDMA3TC mapping is fixed, programming QUEPRI essentially governs the priority of the associated transfer controller(s) read/write commands with respect to the other bus masters in the device. You can modify the EDMA3TC priority to obtain the desired system performance. The QUEPRI is shown in Figure 4-14 and described in Table 4-17.

**Figure 4-14. Queue Priority Register (QUEPRI)**

| 31 | 16 |
|---|---|
| Reserved | |
| R-0 | |

| 15 | 11 | 10 | 8 | 7 | 6 | 4 | 3 | 2 | 0 |
|---|---|---|---|---|---|---|---|---|---|
| Reserved | | PRIQ2 | | Rsvd | PRIQ1 | | Rsvd | PRIQ0 | |
| R-0 | | R/W-0 | | R-0 | R/W-0 | | R-0 | R/W-0 | |

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

**Table 4-17. Queue Priority Register (QUEPRI) Field Descriptions**

| Bit | Field | Value | Description |
|---|---|---|---|
| 31-11 | Reserved | 0 | Reserved |
| 10-8 | PRIQ2 | 0-7h | Priority level for queue 2. Dictates the priority level used by TC2 relative to other masters in the device. A value of 0 means highest priority and a value of 7 means lowest priority. |
| 7 | Reserved | 0 | Reserved |
| 6-4 | PRIQ1 | 0-7h | Priority level for queue 1. Dictates the priority level used by TC1 relative to other masters in the device. A value of 0 means highest priority and a value of 7 means lowest priority. |
| 3 | Reserved | 0 | Reserved |
| 2-0 | PRIQ0 | 0-7h | Priority level for queue 0. Dictates the priority level used by TC0 relative to other masters in the device. A value of 0 means highest priority and a value of 7 means lowest priority. |

### *4.3.2 Error Registers*

The EDMA3CC contains a set of registers that provide information on missed DMA and/or QDMA events, and instances when event queue thresholds are exceeded. If any of the bits in these registers is set, it results in the EDMA3CC generating an error interrupt.

#### 4.3.2.1 Event Missed Registers (EMR/EMRH)

For a particular DMA channel, if a second event is received prior to the first event getting cleared/serviced, the bit corresponding to that channel is set/asserted in the event missed registers (EMR/EMRH). All trigger types are treated individually, that is, manual triggered (ESR/ESRH), chain triggered (CER/CERH), and event triggered (ER/ERH) are all treated separately. The EMR/EMRH bits for a channel are also set if an event on that channel encounters a NULL entry (or a NULL TR is serviced). If any EMR/EMRH bit is set (and all errors, including bits in other error registers (QEMR, CCERR) were previously cleared), the EDMA3CC generates an error interrupt. See Section 2.9.4 for details on EDMA3CC error interrupt generation.

The EMR is shown in Figure 4-15 and described in Table 4-18. The EMRH is shown in Figure 4-16 and described in Table 4-19.

**Figure 4-15. Event Missed Register (EMR)**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| E31 | E30 | E29 | E28 | E27 | E26 | E25 | E24 | E23 | E22 | E21 | E20 | E19 | E18 | E17 | E16 |
| R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| E15 | E14 | E13 | E12 | E11 | E10 | E9 | E8 | E7 | E6 | E5 | E4 | E3 | E2 | E1 | E0 |
| R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 |

LEGEND: R = Read only; -*n* = value after reset

**Table 4-18. Event Missed Register (EMR) Field Descriptions**

| Bit | Field | Value | Description |
|---|---|---|---|
| 31-0 | E*n* | | Channel 0-31 event missed. E*n* is cleared by writing a 1 to the corresponding bit in the event missed clear register (EMCR). |
| | | 0 | No missed event. |
| | | 1 | Missed event occurred. |

**Figure 4-16. Event Missed Register High (EMRH)**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| E63 | E62 | E61 | E60 | E59 | E58 | E57 | E56 | E55 | E54 | E53 | E52 | E51 | E50 | E49 | E48 |
| R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| E47 | E46 | E45 | E44 | E43 | E42 | E41 | E40 | E39 | E38 | E37 | E36 | E35 | E34 | E33 | E32 |
| R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 |

LEGEND: R = Read only; -*n* = value after reset

**Table 4-19. Event Missed Register High (EMRH) Field Descriptions**

| Bit | Field | Value | Description |
|---|---|---|---|
| 31-0 | E*n* | | Channel 32–63 event missed. E*n* is cleared by writing a 1 to the corresponding bit in the event missed clear register high (EMCRH). |
| | | 0 | No missed event. |
| | | 1 | Missed event occurred. |

### 4.3.2.2 Event Missed Clear Registers (EMCR/EMCRH)

Once a missed event is posted in the event missed registers (EMR/EMRH), the bit remains set and you need to clear the set bit(s). This is done by way of CPU writes to the event missed clear registers (EMCR/EMCRH). Writing a 1 to any of the bits clears the corresponding missed event (bit) in EMR/EMRH; writing a 0 has no effect.

The EMCR is shown in Figure 4-17 and described in Table 4-20. The EMCRH is shown in Figure 4-18 and described in Table 4-21.

**Figure 4-17. Event Missed Clear Register (EMCR)**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| E31 | E30 | E29 | E28 | E27 | E26 | E25 | E24 | E23 | E22 | E21 | E20 | E19 | E18 | E17 | E16 |
| W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| E15 | E14 | E13 | E12 | E11 | E10 | E9 | E8 | E7 | E6 | E5 | E4 | E3 | E2 | E1 | E0 |
| W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 |

LEGEND: W = Write only; -*n* = value after reset

**Table 4-20. Event Missed Clear Register (EMCR) Field Descriptions**

| Bit | Field | Value | Description |
|---|---|---|---|
| 31-0 | E*n* | | Event missed 0-31 clear. All error bits must be cleared before additional error interrupts will be asserted by the EDMA3CC. |
| | | 0 | No effect. |
| | | 1 | Corresponding missed event bit in the event missed register (EMR) is cleared (E*n* = 0). |

**Figure 4-18. Event Missed Clear Register High (EMCRH)**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| E63 | E62 | E61 | E60 | E59 | E58 | E57 | E56 | E55 | E54 | E53 | E52 | E51 | E50 | E49 | E48 |
| W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| E47 | E46 | E45 | E44 | E43 | E42 | E41 | E40 | E39 | E38 | E37 | E36 | E35 | E34 | E33 | E32 |
| W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 |

LEGEND: W = Write only; -*n* = value after reset

**Table 4-21. Event Missed Clear Register High (EMCRH) Field Descriptions**

| Bit | Field | Value | Description |
|---|---|---|---|
| 31-0 | E*n* | | Event missed 32–63 clear. All error bits must be cleared before additional error interrupts will be asserted by the EDMA3CC. |
| | | 0 | No effect. |
| | | 1 | Corresponding missed event bit in the event missed register high (EMRH) is cleared (E*n* = 0). |

### 4.3.2.3 QDMA Event Missed Register (QEMR)

For a particular QDMA channel, if two QDMA events are detected without the first event getting cleared/serviced, the bit corresponding to that channel is set/asserted in the QDMA event missed register (QEMR). The QEMR bits for a channel are also set if a QDMA event on the channel encounters a NULL entry (or a NULL TR is serviced). If any QEMR bit is set (and all errors, including bits in other error registers (EMR/EMRH, CCERR) were previously cleared), the EDMA3CC generates an error interrupt. See Section 2.9.4 for details on EDMA3CC error interrupt generation.

The QEMR is shown in Figure 4-19 and described in Table 4-22.

**Figure 4-19. QDMA Event Missed Register (QEMR)**

| 31 | 16 |
|---|---|
| Reserved | |
| R-0 | |

| 15 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|
| Reserved | | E7 | E6 | E5 | E4 | E3 | E2 | E1 | E0 |
| R-0 | | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 |

LEGEND: R = Read only; -*n* = value after reset

**Table 4-22. QDMA Event Missed Register (QEMR) Field Descriptions**

| Bit | Field | Value | Description |
|---|---|---|---|
| 31-8 | Reserved | 0 | Reserved |
| 7-0 | E*n* | | Channel 0-7 QDMA event missed. E*n* is cleared by writing a 1 to the corresponding bit in the QDMA event missed clear register (QEMCR). |
| | | 0 | No missed event. |
| | | 1 | Missed event occurred. |

### 4.3.2.4 QDMA Event Missed Clear Register (QEMCR)

Once a missed event is posted in the QDMA event missed registers (QEMR), the bit remains set and you need to clear the set bit(s). This is done by way of CPU writes to the QDMA event missed clear registers (QEMCR). Writing a 1 to any of the bits clears the corresponding missed event (bit) in QEMR; writing a 0 has no effect.

The QEMCR is shown in Figure 4-20 and described in Table 4-23.

**Figure 4-20. QDMA Event Missed Clear Register (QEMCR)**

| 31 | 16 |
|---|---|
| Reserved | |
| R-0 | |

| 15 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|
| Reserved | | E7 | E6 | E5 | E4 | E3 | E2 | E1 | E0 |
| R-0 | | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 |

LEGEND: W = Write only; -*n* = value after reset

**Table 4-23. QDMA Event Missed Clear Register (QEMCR) Field Descriptions**

| Bit | Field | Value | Description |
|---|---|---|---|
| 31-8 | Reserved | 0 | Reserved |
| 7-0 | E*n* | | QDMA event missed clear. All error bits must be cleared before additional error interrupts will be asserted by the EDMA3CC. |
| | | 0 | No effect. |
| | | 1 | Corresponding missed event bit in the QDMA event missed register (QEMR) is cleared (E*n* = 0). |

### 4.3.2.5 EDMA3CC Error Register (CCERR)

The EDMA3CC error register (CCERR) indicates whether or not at any instant of time the number of events queued up in any of the event queues exceeds or equals the threshold/watermark value that is set in the queue watermark threshold register (QWMTHRA). Additionally, CCERR also indicates if when the number of outstanding TRs that have been programmed to return transfer completion code (TRs which have the TCINTEN or TCCHEN bit in OPT set to 1) to the EDMA3CC has exceeded the maximum allowed value of 63. If any bit in CCERR is set (and all errors, including bits in other error registers (EMR/EMRH, QEMR) were previously cleared), the EDMA3CC generates an error interrupt. See Section 2.9.4 for details on EDMA3CC error interrupt generation. Once the error bits are set in CCERR, they can only be cleared by writing to the corresponding bits in the EDMA3CC error clear register (CCERRCLR).

The CCERR is shown in Figure 4-21 and described in Table 4-24.

**Figure 4-21. EDMA3CC Error Register (CCERR)**

| 31 | 17 | 16 |
|---|---|---|
| Reserved | | TCCERR |
| R-0 | | R-0 |

| 15 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|
| Reserved | | QTHRXCD2 | QTHRXCD1 | QTHRXCD0 |
| R-0 | | R-0 | R-0 | R-0 |

LEGEND: R = Read only; -*n* = value after reset

**Table 4-24. EDMA3CC Error Register (CCERR) Field Descriptions**

| Bit | Field | Value | Description |
|---|---|---|---|
| 31-17 | Reserved | 0 | Reserved |
| 16 | TCCERR | | Transfer completion code error. TCCERR is cleared by writing a 1 to the corresponding bit in the EDMA3CC error clear register (CCERRCLR). |
| | | 0 | Total number of allowed TCCs outstanding has not been reached. |
| | | 1 | Total number of allowed TCCs has been reached. |
| 15-3 | Reserved | 0 | Reserved |
| 2 | QTHRXCD2 | | Queue threshold error for queue 2. QTHRXCD2 is cleared by writing a 1 to the corresponding bit in the EDMA3CC error clear register (CCERRCLR). |
| | | 0 | Watermark/threshold has not been exceeded. |
| | | 1 | Watermark/threshold has been exceeded. |
| 1 | QTHRXCD1 | | Queue threshold error for queue 1. QTHRXCD1 is cleared by writing a 1 to the corresponding bit in the EDMA3CC error clear register (CCERRCLR). |
| | | 0 | Watermark/threshold has not been exceeded. |
| | | 1 | Watermark/threshold has been exceeded. |
| 0 | QTHRXCD0 | | Queue threshold error for queue 0. QTHRXCD0 is cleared by writing a 1 to the corresponding bit in the EDMA3CC error clear register (CCERRCLR). |
| | | 0 | Watermark/threshold has not been exceeded. |
| | | 1 | Watermark/threshold has been exceeded. |

### 4.3.2.6 EDMA3CC Error Clear Register (CCERRCLR)

The EDMA3CC error clear register (CCERRCLR) is used to clear any error bits that are set in the EDMA3CC error register (CCERR). In addition, CCERRCLR also clears the values of some bit fields in the queue status registers (QSTAT*n*) associated with a particular event queue. Writing a 1 to any of the bits clears the corresponding bit in CCERR; writing a 0 has no effect.

The CCERRCLR is shown in Figure 4-22 and described in Table 4-25.

**Figure 4-22. EDMA3CC Error Clear Register (CCERRCLR)**

| 31 | | | | 17 | 16 |
|---|---|---|---|---|---|
| Reserved | | | | | TCCERR |
| W-0 | | | | | W-0 |
| 15 | | 3 | 2 | 1 | 0 |
| Reserved | | | QTHRXCD2 | QTHRXCD1 | QTHRXCD0 |
| W-0 | | | W-0 | W-0 | W-0 |

LEGEND: W= Write only; -*n* = value after reset

**Table 4-25. EDMA3CC Error Clear Register (CCERRCLR) Field Descriptions**

| Bit | Field | Value | Description |
|---|---|---|---|
| 31-17 | Reserved | 0 | Reserved |
| 16 | TCCERR | | Transfer completion code error clear. |
| | | 0 | No effect. |
| | | 1 | Clears the TCCERR bit in the EDMA3CC error register (CCERR). |
| 15-3 | Reserved | 0 | Reserved |
| 2 | QTHRXCD2 | | Queue threshold error clear for queue 2. |
| | | 0 | No effect. |
| | | 1 | Clears the QTHRXCD2 bit in the EDMA3CC error register (CCERR) and the WM and THRXCD bits in the queue status register 2 (QSTAT2). |
| 1 | QTHRXCD1 | | Queue threshold error clear for queue 1. |
| | | 0 | No effect. |
| | | 1 | Clears the QTHRXCD1 bit in the EDMA3CC error register (CCERR) and the WM and THRXCD bits in the queue status register 1 (QSTAT1). |
| 0 | QTHRXCD0 | | Queue threshold error clear for queue 0. |
| | | 0 | No effect. |
| | | 1 | Clears the QTHRXCD0 bit in the EDMA3CC error register (CCERR) and the WM and THRXCD bits in the queue status register 0 (QSTAT0). |

### 4.3.2.7 Error Evaluate Register (EEVAL)

The EDMA3CC error interrupt is asserted whenever an error bit is set in any of the error registers (EMR/EMRH, QEMR, and CCERR). For subsequent error bits that get set, the EDMA3CC error interrupt is reasserted only when transitioning from an “all the error bits cleared” to “at least one error bit is set”. Alternatively, a CPU write of 1 to the EVAL bit in the error evaluate register (EEVAL) results in reasserting the EDMA3CC error interrupt, if there are any outstanding error bits set due to subsequent error conditions. Writes of 0 have no effect.

The EEVAL is shown in Figure 4-23 and described in Table 4-26.

**Figure 4-23. Error Evaluate Register (EEVAL)**

| 31 | 16 |
|---|---|
| Reserved | |
| R-0 | |

| 15 | 2 | 1 | 0 |
|---|---|---|---|
| Reserved | | Rsvd | EVAL |
| R-0 | | R/W-0 | W-0 |

LEGEND: R/W = Read/Write; R = Read only; W = Write only; -*n* = value after reset

**Table 4-26. Error Evaluate Register (EEVAL) Field Descriptions**

| Bit | Field | Value | Description |
|---|---|---|---|
| 31-2 | Reserved | 0 | Reserved |
| 1 | Reserved | 0 | Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior. |
| 0 | EVAL | | Error interrupt evaluate. |
| | | 0 | No effect. |
| | | 1 | EDMA3CC error interrupt will be pulsed if any errors have not been cleared in any of the error registers (EMR/EMRH, QEMR, or CCERR). |

### 4.3.3 Region Access Enable Registers

The region access enable register group consists of the DMA access enable registers (DRAE*m* and DRAEH*m*) and the QDMA access enable registers (QRAE*m*). Where *m* is the number of shadow regions in the EDMA3CC memory map for a device. You can configure these registers to assign ownership of DMA/QDMA channels to a particular shadow region.

#### 4.3.3.1 DMA Region Access Enable for Region *m* (DRAE*m*)

The DMA region access enable register for shadow region *m* (DRAE*m*/DRAEH*m*) is programmed to allow or disallow read/write accesses on a bit-by-bit bases for all DMA registers in the shadow region *m* view of the DMA channel registers. See the EDMA3CC register memory map for a list of all the DMA channel and interrupt registers mapped in the shadow region view. Additionally, the DRAE*m*/DRAEH*m* configuration determines completion of which DMA channels will result in assertion of the shadow region *m* DMA completion interrupt (see Section 2.9).

The DRAE*m* is shown in Figure 4-24 and described in Table 4-27. The DRAEH*m* is shown in Figure 4-25 and described in Table 4-27.

**Figure 4-24. DMA Region Access Enable Register for Region *m* (DRAE*m*)**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| E31 | E30 | E29 | E28 | E27 | E26 | E25 | E24 | E23 | E22 | E21 | E20 | E19 | E18 | E17 | E16 |
| R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| E15 | E14 | E13 | E12 | E11 | E10 | E9 | E8 | E7 | E6 | E5 | E4 | E3 | E2 | E1 | E0 |
| R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |

LEGEND: R/W = Read/Write; -*n* = value after reset

**Figure 4-25. DMA Region Access Enable High Register for Region *m* (DRAEH*m*)**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| E63 | E62 | E61 | E60 | E59 | E58 | E57 | E56 | E55 | E54 | E53 | E52 | E51 | E50 | E49 | E48 |
| R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| E47 | E46 | E45 | E44 | E43 | E42 | E41 | E40 | E39 | E38 | E37 | E36 | E35 | E34 | E33 | E32 |
| R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |

LEGEND: R/W = Read/Write; -*n* = value after reset

**Table 4-27. DMA Region Access Enable Registers for Region *m* (DRAE*m*/DRAEH*m*) Field Descriptions**

| Bit | Field | Value | Description |
|---|---|---|---|
| 31-0 | E*n* | | DMA region access enable for bit *n*/channel *n* in region *m*. |
| | | 0 | Accesses via region *m* address space to bit *n* in any DMA channel register are not allowed. Reads return 0 on bit *n* and writes do not modify the state of bit *n*. Enabled interrupt bits for bit *n* do not contribute to the generation of a transfer completion interrupt for shadow region *m*. |
| | | 1 | Accesses via region *m* address space to bit *n* in any DMA channel register are allowed. Reads return the value from bit *n* and writes modify the state of bit *n*. Enabled interrupt bits for bit *n* contribute to the generation of a transfer completion interrupt for shadow region *m*. |

### 4.3.3.2 QDMA Region Access Enable Registers (QRAE*m*)

The QDMA region access enable register for shadow region *m* (QRAE*m*) is programmed to allow or disallow read/write accesses on a bit-by-bit bases for all QDMA registers in the shadow region *m* view of the QDMA registers. This includes all 8-bit QDMA registers.

The QRAE*m* is shown in Figure 4-26 and described in Table 4-28.

**Figure 4-26. QDMA Region Access Enable for Region *m* (QRAE*m*)**

| 31 | | | | | | | | 16 |
|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | |
| R-0 | | | | | | | | |
| 15 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | | E7 | E6 | E5 | E4 | E3 | E2 | E1 | E0 |
| R-0 | | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

**Table 4-28. QDMA Region Access Enable for Region *m* (QRAE*m*) Field Descriptions**

| Bit | Field | Value | Description |
|---|---|---|---|
| 31-8 | Reserved | 0 | Reserved |
| 7-0 | E*n* | | QDMA region access enable for bit *n*/QDMA channel *n* in region *m*. |
| | | 0 | Accesses via region *m* address space to bit *n* in any QDMA channel register are not allowed. Reads return 0 on bit *n* and writes do not modify the state of bit *n*. |
| | | 1 | Accesses via region *m* address space to bit *n* in any QDMA channel register are allowed. Reads return the value from bit *n* and writes modify the state of bit *n*. |

### 4.3.4 Status/Debug Visibility Registers

The following set of registers provide visibility into the event queues and a TR lifecycle. These are useful for system debug as they provide in-depth visibility for the events queued up in the event queue and also provide information on what parts of the EDMA3CC logic are active once the event has been received by the EDMA3CC.

#### 4.3.4.1 Event Queue Entry Registers (QxEy)

The event queue entry registers (Q*x*E*y*) exist for all 16 queue entries (the maximum allowed queue entries) for all event queues (Q0, Q1, and Q2) in the EDMA3CC: Q0E0 to Q0E15, Q1E0 to Q1E15, and Q2E0 to Q2E15. Each register details the event number (ENUM) and the event type (ETYPE). For example, if the value in Q1E4 is read as 0000 004Fh, this means the 4th entry in queue 1 is a manually-triggered event on DMA channel 15.

The Q*x*E*y* is shown in Figure 4-27 and described in Table 4-29.

**Figure 4-27. Event Queue Entry Registers (Q*x*E*y*)**

| 31 | 16 |
|---|---|
| Reserved | |
| R-0 | |

| 15 | 8 | 7 | 6 | 5 | 0 |
|---|---|---|---|---|---|
| Reserved | | ETYPE | | ENUM | |
| R-0 | | R-x | | R-x | |

LEGEND: R = Read only; -*n* = value after reset; -x = value is indeterminate after reset

**Table 4-29. Event Queue Entry Registers (Q*x*E*y*) Field Descriptions**

| Bit | Field | Value | Description |
|---|---|---|---|
| 31-8 | Reserved | 0 | Reserved |
| 7-6 | ETYPE | 0-3h | Event entry *y* in queue *x*. Specifies the specific event type for the given entry in the event queue. |
| | | 0 | Event triggered via ER |
| | | 1h | Manual triggered via ESR |
| | | 2h | Chain triggered via CER |
| | | 3h | Autotriggered via QER |
| 5-0 | ENUM | 0-3Fh | Event entry *y* in queue *x*. Event number: |
| | | 0-7h | QDMA channel number (0 to 7) |
| | | 0-3Fh | DMA channel/event number (0 to 63) |

### 4.3.4.2 Queue *n* Status Registers (QSTAT*n*)

The queue *n* status register (QSTAT*n*) is shown in Figure 4-28 and described in Table 4-30.

**Figure 4-28. Queue *n* Status Register (QSTAT*n*)**

| 31 | 25 | 24 | 23 | 21 | 20 | 16 |
|---|---|---|---|---|---|---|
| Reserved | | THRXCD | Reserved | | WM | |
| R-0 | | R-0 | R-0 | | R-0 | |

| 15 | 13 | 12 | 8 | 7 | 4 | 3 | 0 |
|---|---|---|---|---|---|---|---|
| Reserved | | NUMVAL | | Reserved | | STRTPTR | |
| R-0 | | R-0 | | R-0 | | R-0 | |

LEGEND: R = Read only; -*n* = value after reset

**Table 4-30. Queue *n* Status Register (QSTAT*n*) Field Descriptions**

| Bit | Field | Value | Description |
|---|---|---|---|
| 31-25 | Reserved | 0 | Reserved |
| 24 | THRXCD | | Threshold exceeded. THRXCD is cleared by writing a 1 to the corresponding QTHRXCD*n* bit in the EDMA3CC error clear register (CCERRCLR). |
| | | 0 | Threshold specified by the Q*n* bit in the queue watermark threshold A register (QWMTHRA) has not been exceeded. |
| | | 1 | Threshold specified by the Q*n* bit in the queue watermark threshold A register (QWMTHRA) has been exceeded. |
| 23-21 | Reserved | 0 | Reserved |
| 20-16 | WM | 0-1Fh | Watermark for maximum queue usage. Watermark tracks the most entries that have been in queue *n* since reset or since the last time that the watermark (WM) bit was cleared. WM is cleared by writing a 1 to the corresponding QTHRXCD*n* bit in the EDMA3CC error clear register (CCERRCLR). |
| | | 0-10h | Legal values are 0 (empty) to 10h (full). |
| | | 11h-1Fh | Reserved |
| 15-13 | Reserved | 0 | Reserved |
| 12-8 | NUMVAL | 0-1Fh | Number of valid entries in queue *n*. The total number of entries residing in the queue manager FIFO at a given instant. Always enabled. |
| | | 0-10h | Legal values are 0 (empty) to 10h (full). |
| | | 11h-1Fh | Reserved |
| 7-4 | Reserved | 0 | Reserved |
| 3-0 | STRTPTR | 0-Fh | Start pointer. The offset to the head entry of queue *n*, in units of entries. Always enabled. Legal values are 0 (0th entry) to Fh (15th entry). |

### 4.3.4.3 Queue Watermark Threshold A Register (QWMTHRA)

The queue watermark threshold A register (QWMTHRA) is shown in Figure 4-29 and described in Table 4-31.

**Figure 4-29. Queue Watermark Threshold A Register (QWMTHRA)**

| 31 | 21 | 20 | 16 |
|---|---|---|---|
| Reserved | | Q2 | |
| R-0 | | R/W-10h | |

| 15 | 13 | 12 | 8 | 7 | 5 | 4 | 0 |
|---|---|---|---|---|---|---|---|
| Reserved | | Q1 | | Reserved | | Q0 | |
| R-0 | | R/W-10h | | R-0 | | R/W-10h | |

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

**Table 4-31. Queue Watermark Threshold A Register (QWMTHRA) Field Descriptions**

| Bit | Field | Value | Description |
|---|---|---|---|
| 31-21 | Reserved | 0 | Reserved |
| 20-16 | Q2 | 0-1Fh | Queue threshold for queue 2 value. The QTHRXCD2 bit in the EDMA3CC error register (CCERR) and the THRXCD bit in the queue status register 2 (QSTAT2) are set when the number of events in queue 2 at an instant in time (visible via the NUMVAL bit in QSTAT2) equals or exceeds the value specified by Q2. |
| | | 0-10h | The default is 16 (maximum allowed). |
| | | 11h | Disables the threshold errors. |
| | | 12h-1Fh | Reserved |
| 15-13 | Reserved | 0 | Reserved |
| 12-8 | Q1 | 0-1Fh | Queue threshold for queue 1 value. The QTHRXCD1 bit in the EDMA3CC error register (CCERR) and the THRXCD bit in the queue status register 1 (QSTAT1) are set when the number of events in queue 1 at an instant in time (visible via the NUMVAL bit in QSTAT1) equals or exceeds the value specified by Q1. |
| | | 0-10h | The default is 16 (maximum allowed). |
| | | 11h | Disables the threshold errors. |
| | | 12h-1Fh | Reserved |
| 7-5 | Reserved | 0 | Reserved |
| 4-0 | Q0 | 0-1Fh | Queue threshold for queue 0 value. The QTHRXCD0 bit in the EDMA3CC error register (CCERR) and the THRXCD bit in the queue status register 0 (QSTAT0) are set when the number of events in queue 0 at an instant in time (visible via the NUMVAL bit in QSTAT0) equals or exceeds the value specified by Q0. |
| | | 0-10h | The default is 16 (maximum allowed). |
| | | 11h | Disables the threshold errors. |
| | | 12h-1Fh | Reserved |

### 4.3.4.4 EDMA3CC Status Register (CCSTAT)

The EDMA3CC status register (CCSTAT) has a number of status bits that reflect which parts of the EDMA3CC logic is active at any given instant of time. The CCSTAT is shown in Figure 4-30 and described in Table 4-32.

**Figure 4-30. EDMA3CC Status Register (CCSTAT)**

| 31 | 24 |
|---|---|
| Reserved | |
| R-0 | |

| 23 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|
| Reserved | | QUEACTV2 | QUEACTV1 | QUEACTV0 |
| R-0 | | R-0 | R-0 | R-0 |

| 15 | 14 | 13 | 8 |
|---|---|---|---|
| Reserved | | COMPACTV | |
| R-0 | | R-0 | |

| 7 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|
| Reserved | | ACTV | WSTATACTV | TRACTV | QEVTACTV | EVTACTV |
| R-0 | | R-0 | R-0 | R-0 | R-0 | R-0 |

LEGEND: R = Read only; -*n* = value after reset

**Table 4-32. EDMA3CC Status Register (CCSTAT) Field Descriptions**

| Bit | Field | Value | Description |
|---|---|---|---|
| 31-19 | Reserved | 0 | Reserved |
| 18 | QUEACTV2 | | Queue 2 active. |
| | | 0 | No events are queued in queue 2. |
| | | 1 | At least one TR is queued in queue 2. |
| 17 | QUEACTV1 | | Queue 1 active. |
| | | 0 | No events are queued in queue 1. |
| | | 1 | At least one TR is queued in queue 1. |
| 16 | QUEACTV0 | | Queue 0 active. |
| | | 0 | No events are queued in queue 0. |
| | | 1 | At least one TR is queued in queue 0. |
| 15-14 | Reserved | 0 | Reserved |
| 13-8 | COMPACTV | 0-3Fh | Completion request active. The COMPACTV field reflects the count for the number of completion requests submitted to the transfer controllers. This count increments every time a TR is submitted and is programmed to report completion (the TCINTEN or TCCCHEN bits in OPT in the parameter entry associated with the TR are set to 1). The counter decrements for every valid TCC received back from the transfer controllers. If at any time the count reaches a value of 63, the EDMA3CC will not service any new TRs until the count is less then 63 (or return a transfer completion code from a transfer controller, which would decrement the count). |
| | | 0 | No completion requests outstanding. |
| | | 1h-3Fh | Total of 1 completion request to 63 completion requests are outstanding. |
| 7-5 | Reserved | 0 | Reserved |
| 4 | ACTV | | Channel controller active. Channel controller active is a logical-OR of each of the *ACTV bits. The ACTV bit remains high through the life of a TR. |
| | | 0 | Channel is idle. |
| | | 1 | Channel is busy. |
| 3 | WSTATACTV | | Write status interface active. |
| | | 0 | Write status req is idle and write status fifo is idle. |
| | | 1 | Either the write status request is active or additional write status responses are pending in the write status fifo. |

**Table 4-32. EDMA3CC Status Register (CCSTAT) Field Descriptions (continued)**

| Bit | Field | Value | Description |
|---|---|---|---|
| 2 | TRACTV | | Transfer request active. |
| | | 0 | Transfer request processing/submission logic is inactive. |
| | | 1 | Transfer request processing/submission logic is active. |
| 1 | QEVTACTV | | QDMA event active. |
| | | 0 | No enabled QDMA events are active within the EDMA3CC. |
| | | 1 | At least one enabled QDMA event (QER) is active within the EDMA3CC. |
| 0 | EVTACTV | | DMA event active. |
| | | 0 | No enabled DMA events are active within the EDMA3CC. |
| | | 1 | At least one enabled DMA event (ER and EER, ESR, CER) is active within the EDMA3CC. |

### *4.3.5 DMA Channel Registers*

The following sets of registers pertain to the 64 DMA channels. The 64 DMA channels consist of a set of registers (with exception of DMAQNUM*n*) that each have 64 bits and the bit position of each register matches the DMA channel number. Each register is named with the format *reg_name* that corresponds to DMA channels 0 through 31 and *reg_name_High* that corresponds to DMA channels 32 through 64.

For example, the event register (ER) corresponds to DMA channel 0 through 31 and the event register high register (ERH) corresponds to DMA channel 32 through 63. The register is typically called the event register.

The DMA channel registers are accessible via read/writes to the global address range. They are also accessible via read/writes to the shadow address range. The read/write ability to the registers in the shadow region are controlled by the DMA region access registers (DRAE*m*/DRAEH*m*). The registers are described in Section 4.3.3.1 and the details for shadow region/global region usage is explained in Section 2.7.

#### 4.3.5.1 Event Registers (ER, ERH)

All external events are captured in the event register (ER/ERH). The events are latched even when the events are not enabled. If the event bit corresponding to the latched event is enabled (EER.E*n*/EERH.E*n* = 1), then the event is evaluated by the EDMA3CC logic for an associated transfer request submission to the transfer controllers. The event register bits are automatically cleared (ER.E*n*/ERH.E*n* = 0) once the corresponding events are prioritized and serviced. If ER.E*n*/ERH.E*n* are already set and another event is received on the same channel/event, then the corresponding event is latched in the event miss register (EMR.E*n*/EMRH.E*n*), provided that the event was enabled (EER.E*n*/EERH.E*n* = 1).

Event *n* can be cleared by the CPU writing a 1 to corresponding event bit in the event clear register (ECR/ECRH). The setting of an event is a higher priority relative to clear operations (via hardware or software). If set and clear conditions occur concurrently, the set condition wins. If the event was previously set, then EMR/EMRH would be set since an event is lost. If the event was previously clear, then the event remains set and is prioritized for submission to the event queues.

Table 2-5 provides the type of synchronization events and the EDMA3CC channels associated to each of these external events.

The ER is shown in Figure 4-31 and described in Table 4-33. The ERH is shown in Figure 4-32 and described in Table 4-34.

**Figure 4-31. Event Register (ER)**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| E31 | E30 | E29 | E28 | E27 | E26 | E25 | E24 | E23 | E22 | E21 | E20 | 19 | E18 | E17 | E16 |
| R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| E15 | E14 | E13 | E12 | E11 | E10 | E9 | E8 | E7 | E6 | E5 | E4 | E3 | E2 | E1 | E0 |
| R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 |

LEGEND: R = Read only; -*n* = value after reset

**Table 4-33. Event Register (ER) Field Descriptions**

| Bit | Field | Value | Description |
|---|---|---|---|
| 31-0 | E*n* | | Event 0-31. Events 0-31 are captured by the EDMA3CC and are latched into ER. The events are set (E*n* = 1) even when events are disabled (E*n* = 0 in the event enable register, EER). |
| | | 0 | EDMA3CC event is not asserted. |
| | | 1 | EDMA3CC event is asserted. Corresponding DMA event is prioritized versus other pending DMA/QDMA events for submission to the EDMA3TC. |

**Figure 4-32. Event Register High (ERH)**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| E63 | E62 | E61 | E60 | E59 | E58 | E57 | E56 | E55 | E54 | E53 | E52 | E51 | E50 | E49 | E48 |
| R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| E47 | E46 | E45 | E44 | E43 | E42 | E41 | E40 | E39 | E38 | E37 | E36 | E35 | E34 | E33 | E32 |
| R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 |

LEGEND: R = Read only; -*n* = value after reset

**Table 4-34. Event Register High (ERH) Field Descriptions**

| Bit | Field | Value | Description |
|---|---|---|---|
| 31-0 | E*n* | | Event 32-63. Events 32-63 are captured by the EDMA3CC and are latched into ERH. The events are set (E*n* = 1) even when events are disabled (E*n* = 0 in the event enable register high, EERH). |
| | | 0 | EDMA3CC event is not asserted. |
| | | 1 | EDMA3CC event is asserted. Corresponding DMA event is prioritized versus other pending DMA/QDMA events for submission to the EDMA3TC. |

### 4.3.5.2 Event Clear Registers (ECR, ECRH)

Once an event has been posted in the event registers (ER/ERH), the event is cleared in two ways. If the event is enabled in the event enable register (EER/EERH) and the EDMA3CC submits a transfer request for the event to the EDMA3TC, it clears the corresponding event bit in the event register. If the event is disabled in the event enable register (EER/EERH), the CPU can clear the event by way of the event clear registers (ECR/ECRH).

Writing a 1 to any of the bits clears the corresponding event; writing a 0 has no effect. Once an event bit is set in the event register, it remains set until EDMA3CC submits a transfer request for that event or the CPU clears the event by setting the corresponding bit in ECR/ECRH.

The ECR is shown in Figure 4-33 and described in Table 4-35. The ECRH is shown in Figure 4-34 and described in Table 4-36.

**Figure 4-33. Event Clear Register (ECR)**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| E31 | E30 | E29 | E28 | E27 | E26 | E25 | E24 | E23 | E22 | E21 | E20 | E19 | E18 | E17 | E16 |
| W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| E15 | E14 | E13 | E12 | E11 | E10 | E9 | E8 | E7 | E6 | E5 | E4 | E3 | E2 | E1 | E0 |
| W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 |

LEGEND: W = Write only; -*n* = value after reset

**Table 4-35. Event Clear Register (ECR) Field Descriptions**

| Bit | Field | Value | Description |
|---|---|---|---|
| 31-0 | E*n* | | Event clear for event 0-31. Any of the event bits in ECR is set to 1 to clear the event (E*n*) in the event register (ER). A write of 0 has no effect. |
| | | 0 | No effect. |
| | | 1 | EDMA3CC event is cleared in the event register (ER). |

**Figure 4-34. Event Clear Register High (ECRH)**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| E63 | E62 | E61 | E60 | E59 | E58 | E57 | E56 | E55 | E54 | E53 | E52 | E51 | E50 | E49 | E48 |
| W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| E47 | E46 | E45 | E44 | E43 | E42 | E41 | E40 | E39 | E38 | E37 | E36 | E35 | E34 | E33 | E32 |
| W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 |

LEGEND: W = Write only; -*n* = value after reset

**Table 4-36. Event Clear Register High (ECRH) Field Descriptions**

| Bit | Field | Value | Description |
|---|---|---|---|
| 31-0 | E*n* | | Event clear for event 32-63. Any of the event bits in ECRH is set to 1 to clear the event (E*n*) in the event register high (ERH). A write of 0 has no effect. |
| | | 0 | No effect. |
| | | 1 | EDMA3CC event is cleared in the event register high (ERH). |

### 4.3.5.3 Event Set Registers (ESR, ESRH)

The event set registers (ESR/ESRH) allow the CPU (or EDMA programmers) to manually set events to initiate DMA transfer requests. CPU writes of 1 to any event set register (E*n*) bits set the corresponding bits in the registers. The set event is evaluated by the EDMA3CC logic for an associated transfer request submission to the transfer controllers. Writing a 0 has no effect.

The event set registers operate independent of the event registers (ER/ERH), and a write of 1 is always considered a valid event regardless of whether the event is enabled (the corresponding event bits are set or cleared in EER.E*n*/EERH.E*n*).

Once the event is set in the event set registers, it cannot be cleared by CPU writes, in other words, the event clear registers (ECR/ECRH) have no effect on the state of ESR/ESRH. The bits will only be cleared once the transfer request corresponding to the event has been submitted to the transfer controller. The setting of an event is a higher priority relative to clear operations (via hardware). If set and clear conditions occur concurrently, the set condition wins. If the event was previously set, then EMR/EMRH would be set since an event is lost. If the event was previously clear, then the event remains set and is prioritized for submission to the event queues.

Manually-triggered transfers via writes to ESR/ESRH allow the CPU to submit DMA requests in the system, these are relevant for memory-to-memory transfer scenarios. If the ESR.E*n*/ESRH.E*n* bit is already set and another CPU write of 1 is attempted to the same bit, then the corresponding event is latched in the event missed registers (EMR.E*n*/EMRH.E*n* = 1).

The ESR is shown in Figure 4-35 and described in Table 4-37. The ESRH is shown in Figure 4-36 and described in Table 4-38.

**Figure 4-35. Event Set Register (ESR)**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| E31 | E30 | E29 | E28 | E27 | E26 | E25 | E24 | E23 | E22 | E21 | E20 | E19 | E18 | E17 | E16 |
| R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| E15 | E14 | E13 | E12 | E11 | E10 | E9 | E8 | E7 | E6 | E5 | E4 | E3 | E2 | E1 | E0 |
| R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |

LEGEND: R/W = Read/Write; -*n* = value after reset

**Table 4-37. Event Set Register (ESR) Field Descriptions**

| Bit | Field | Value | Description |
|---|---|---|---|
| 31-0 | E*n* | | Event set for event 0-31. |
| | | 0 | No effect. |
| | | 1 | Corresponding DMA event is prioritized versus other pending DMA/QDMA events for submission to the EDMA3TC. |

**Figure 4-36. Event Set Register High (ESRH)**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| E63 | E62 | E61 | E60 | E59 | E58 | E57 | E56 | E55 | E54 | E53 | E52 | E51 | E50 | E49 | E48 |
| R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| E47 | E46 | E45 | E44 | E43 | E42 | E41 | E40 | E39 | E38 | E37 | E36 | E35 | E34 | E33 | E32 |
| R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |

LEGEND: R/W = Read/Write; -*n* = value after reset

**Table 4-38. Event Set Register High (ESRH) Field Descriptions**

| Bit | Field | Value | Description |
|---|---|---|---|
| 31-0 | E*n* | | Event set for event 32-63. |
| | | 0 | No effect . |
| | | 1 | Corresponding DMA event is prioritized versus other pending DMA/QDMA events for submission to the EDMA3TC. |

### 4.3.5.4 Chained Event Registers (CER, CERH)

When the OPTIONS parameter for a PaRAM entry is programmed to returned a chained completion code (ITCCHEN = 1 and/or TCCHEN = 1), then the value dictated by the TCC[5:0] (also programmed in OPT) forces the corresponding event bit to be set in the chained event registers (CER/CERH). The set chained event is evaluated by the EDMA3CC logic for an associated transfer request submission to the transfer controllers. This results in a chained-triggered transfer.

The chained event registers do not have any enables. The generation of a chained event is essentially enabled by the PaRAM entry that has been configured for intermediate and/or final chaining on transfer completion. The E*n* bit is set (regardless of the state of EER.E*n*/EERH.E*n*) when a chained completion code is returned from one of the transfer controllers or is generated by the EDMA3CC via the early completion path. The bits in the chained event register are cleared when the corresponding events are prioritized and serviced.

If the E*n* bit is already set and another chaining completion code is return for the same event, then the corresponding event is latched in the event missed registers (EMR.E*n*/EMRH.E*n* = 1). The setting of an event is a higher priority relative to clear operations (via hardware). If set and clear conditions occur concurrently, the set condition wins. If the event was previously set, then EMR/EMRH would be set since an event is lost. If the event was previously clear, then the event remains set and is prioritized for submission to the event queues.

The CER is shown in Figure 4-37 and described in Table 4-39. The CERH is shown in Figure 4-38 and described in Table 4-40.

**Figure 4-37. Chained Event Register (CER)**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| E31 | E30 | E29 | E28 | E27 | E26 | E25 | E24 | E23 | E22 | E21 | E20 | E19 | E18 | E17 | E16 |
| R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| E15 | E14 | E13 | E12 | E11 | E10 | E9 | E8 | E7 | E6 | E5 | E4 | E3 | E2 | E1 | E0 |
| R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 |

LEGEND: R = Read only; -*n* = value after reset

**Table 4-39. Chained Event Register (CER) Field Descriptions**

| Bit | Field | Value | Description |
|---|---|---|---|
| 31-0 | E*n* | | Chained event for event 0-31. |
| | | 0 | No effect. |
| | | 1 | Corresponding DMA event is prioritized versus other pending DMA/QDMA events for submission to the EDMA3TC. |

**Figure 4-38. Chained Event Register High (CERH)**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| E63 | E62 | E61 | E60 | E59 | E58 | E57 | E56 | E55 | E54 | E53 | E52 | E51 | E50 | E49 | E48 |
| R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| E47 | E46 | E45 | E44 | E43 | E42 | E41 | E40 | E39 | E38 | E37 | E36 | E35 | E34 | E33 | E32 |
| R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 |

LEGEND: R = Read only; -*n* = value after reset

**Table 4-40. Chained Event Register High (CERH) Field Descriptions**

| Bit | Field | Value | Description |
|---|---|---|---|
| 31-0 | E*n* | | Chained event set for event 32-63. |
| | | 0 | No effect. |
| | | 1 | Corresponding DMA event is prioritized versus other pending DMA/QDMA events for submission to the EDMA3TC. |

#### 4.3.5.5 Event Enable Registers (EER, EERH)

The EDMA3CC provides the option of selectively enabling/disabling each event in the event registers (ER/ERH) by using the event enable registers (EER/EERH). If an event bit in EER/EERH is set to 1 (using the event enable set registers, EESR/EESRH), it will enable that corresponding event. Alternatively, if an event bit in EER/EERH is cleared (using the event enable clear registers, EECR/EECRH), it will disable the corresponding event.

The event registers latch all events that are captured by EDMA3CC, even if the events are disabled (although EDMA3CC does not process it). Enabling an event with a pending event already set in the event registers enables the EDMA3CC to process the already set event like any other new event. The EER/EERH settings do not have any effect on chained events (CER.E*n*/CERH.E*n* = 1) and manually set events (ESR.E*n*/ESRH.E*n* = 1).

The EER is shown in Figure 4-39 and described in Table 4-41. Th EERH is shown in Figure 4-40 and described in Table 4-42.

**Figure 4-39. Event Enable Register (EER)**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| E31 | E30 | E29 | E28 | E27 | E26 | E25 | E24 | E23 | E22 | E21 | E20 | E19 | E18 | E17 | E16 |
| R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| E15 | E14 | E13 | E12 | E11 | E10 | E9 | E8 | E7 | E6 | E5 | E4 | E3 | E2 | E1 | E0 |
| R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 |

LEGEND: R = Read only; -*n* = value after reset

**Table 4-41. Event Enable Register (EER) Field Descriptions**

| Bit | Field | Value | Description |
|---|---|---|---|
| 31-0 | E*n* | | Event enable for events 0-31. |
| | | 0 | Event is not enabled. An external event latched in the event register (ER) is not evaluated by the EDMA3CC. |
| | | 1 | Event is enabled. An external event latched in the event register (ER) is evaluated by the EDMA3CC. |

**Figure 4-40. Event Enable Register High (EERH)**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| E63 | E62 | E61 | E60 | E59 | E58 | E57 | E56 | E55 | E54 | E53 | E52 | E51 | E50 | E49 | E48 |
| R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| E47 | E46 | E45 | E44 | E43 | E42 | E41 | E40 | E39 | E38 | E37 | E36 | E35 | E34 | E33 | E32 |
| R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 |

LEGEND: R = Read only; -*n* = value after reset

**Table 4-42. Event Enable Register High (EERH) Field Descriptions**

| Bit | Field | Value | Description |
|---|---|---|---|
| 31-0 | E*n* | | Event enable for events 32-63. |
| | | 0 | Event is not enabled. An external event latched in the event register high (ERH) is not evaluated by the EDMA3CC. |
| | | 1 | Event is enabled. An external event latched in the event register high (ERH) is evaluated by the EDMA3CC. |

### 4.3.5.6 Event Enable Clear Register (EECR, EECRH)

The event enable registers (EER/EERH) cannot be modified by directly writing to them. The intent is to ease the software burden for the case where multiple tasks are attempting to simultaneously modify these registers. The event enable clear registers (EECR/EECRH) are used to disable events. Writes of 1 to the bits in EECR/EECRH clear the corresponding event bits in EER/EERH; writes of 0 have no effect.

The EECR is shown in Figure 4-41 and described in Table 4-43. The EECRH is shown in Figure 4-42 and described in Table 4-44.

**Figure 4-41. Event Enable Clear Register (EECR)**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| E31 | E30 | E29 | E28 | E27 | E26 | E25 | E24 | E23 | E22 | E21 | E20 | E19 | E18 | E17 | E16 |
| W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| E15 | E14 | E13 | E12 | E11 | E10 | E9 | E8 | E7 | E6 | E5 | E4 | E3 | E2 | E1 | E0 |
| W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 |

LEGEND: W = Write only; -*n* = value after reset

**Table 4-43. Event Enable Clear Register (EECR) Field Descriptions**

| Bit | Field | Value | Description |
|---|---|---|---|
| 31-0 | E*n* | | Event enable clear for events 0-31. |
| | | 0 | No effect. |
| | | 1 | Event is disabled. Corresponding bit in the event enable register (EER) is cleared (E*n* = 0). |

**Figure 4-42. Event Enable Clear Register High (EECRH)**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| E63 | E62 | E61 | E60 | E59 | E58 | E57 | E56 | E55 | E54 | E53 | E52 | E51 | E50 | E49 | E48 |
| W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| E47 | E46 | E45 | E44 | E43 | E42 | E41 | E40 | E39 | E38 | E37 | E36 | E35 | E34 | E33 | E32 |
| W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 |

LEGEND: W = Write only; -*n* = value after reset

**Table 4-44. Event Enable Clear Register High (EECRH) Field Descriptions**

| Bit | Field | Value | Description |
|---|---|---|---|
| 31-0 | E*n* | | Event enable clear for events 32-63. |
| | | 0 | No effect. |
| | | 1 | Event is disabled. Corresponding bit in the event enable register high (EERH) is cleared (E*n* = 0). |

### 4.3.5.7 Event Enable Set Registers (EESR, EESRH)

The event enable registers (EER/EERH) cannot be modified by directly writing to them. The intent is to ease the software burden for the case where multiple tasks are attempting to simultaneously modify these registers. The event enable set registers (EESR/EESRH) are used to enable events. Writes of 1 to the bits in EESR/EESRH set the corresponding event bits in EER/EERH; writes of 0 have no effect.

The EESR is shown in Figure 4-43 and described in Table 4-45. The EESRH is shown in Figure 4-44 and described in Table 4-46.

**Figure 4-43. Event Enable Set Register (EESR)**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| E31 | E30 | E29 | E28 | E27 | E26 | E25 | E24 | E23 | E22 | E21 | E20 | E19 | E18 | E17 | E16 |
| W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| E15 | E14 | E13 | E12 | E11 | E10 | E9 | E8 | E7 | E6 | E5 | E4 | E3 | E2 | E1 | E0 |
| W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 |

LEGEND: W = Write only; -*n* = value after reset

**Table 4-45. Event Enable Set Register (EESR) Field Descriptions**

| Bit | Field | Value | Description |
|---|---|---|---|
| 31-0 | E*n* | | Event enable set for events 0-31. |
| | | 0 | No effect. |
| | | 1 | Event is enabled. Corresponding bit in the event enable register (EER) is set (E*n* = 1). |

**Figure 4-44. Event Enable Set Register High (EESRH)**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| E63 | E62 | E61 | E60 | E59 | E58 | E57 | E56 | E55 | E54 | E53 | E52 | E51 | E50 | E49 | E48 |
| W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| E47 | E46 | E45 | E44 | E43 | E42 | E41 | E40 | E39 | E38 | E37 | E36 | E35 | E34 | E33 | E32 |
| W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 |

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

**Table 4-46. Event Enable Set Register High (EESRH) Field Descriptions**

| Bit | Field | Value | Description |
|---|---|---|---|
| 31-0 | E*n* | | Event enable set for events 32-63. |
| | | 0 | No effect. |
| | | 1 | Event is enabled. Corresponding bit in the event enable register high (EERH) is set (E*n* = 1). |

### 4.3.5.8 Secondary Event Registers (SER, SERH)

The secondary event registers (SER/SERH) provide information on the state of a DMA channel or event (0 through 63). If the EDMA3CC receives a TR synchronization due to a manual-trigger, event-trigger, or chained-trigger source (ESR.E*n*/ESRH.E*n* = 1, ER.E*n*/ERH.E*n* = 1, or CER.E*n*/CERH.E*n* = 1), which results in the setting of a corresponding event bit in SER/SERH (SER.E*n*/SERH.E*n* = 1), it implies that the corresponding DMA event is in the queue.

Once a bit corresponding to an event is set in SER/SERH, the EDMA3CC does not prioritize additional events on the same DMA channel. Depending on the condition that leads to the setting of the SER bits, either the EDMA3CC hardware or the software (using SECR/SECRH) needs to clear the SER/SERH bits for the EDMA3CC to evaluate subsequent events and perform subsequent transfers on the same channel. Based on whether the associated TR is valid, or it is a null or dummy TR, the implications on the state of SER/SERH and the required user action in order to submit another DMA transfer might be different.

The SER is shown in Figure 4-45 and described in Table 4-47. The SERH is shown in Figure 4-46 and described in Table 4-48.

**Figure 4-45. Secondary Event Register (SER)**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| E31 | E30 | E29 | E28 | E27 | E26 | E25 | E24 | E23 | E22 | E21 | E20 | E19 | E18 | E17 | E16 |
| R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| E15 | E14 | E13 | E12 | E11 | E10 | E9 | E8 | E7 | E6 | E5 | E4 | E3 | E2 | E1 | E0 |
| R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 |

LEGEND: R = Read only; -*n* = value after reset

**Table 4-47. Secondary Event Register (SER) Field Descriptions**

| Bit | Field | Value | Description |
|---|---|---|---|
| 31-0 | E*n* | | Secondary event register. The secondary event register is used to provide information on the state of an event. |
| | | 0 | Event is not currently stored in the event queue. |
| | | 1 | Event is currently stored in the event queue. Event arbiter will not prioritize additional events. |

**Figure 4-46. Secondary Event Register High (SERH)**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| E63 | E62 | E61 | E60 | E59 | E58 | E57 | E56 | E55 | E54 | E53 | E52 | E51 | E50 | E49 | E48 |
| R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| E47 | E46 | E45 | E44 | E43 | E42 | E41 | E40 | E39 | E38 | E37 | E36 | E35 | E34 | E33 | E32 |
| R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 |

LEGEND: R = Read only; -*n* = value after reset

**Table 4-48. Secondary Event Register High (SERH) Field Descriptions**

| Bit | Field | Value | Description |
|---|---|---|---|
| 31-0 | E*n* | | Secondary event register. The secondary event register is used to provide information on the state of an event. |
| | | 0 | Event is not currently stored in the event queue. |
| | | 1 | Event is currently stored in the event queue. Event submission/prioritization logic will not prioritize additional events. |

### 4.3.5.9 Secondary Event Clear Registers (SECR, SECRH)

The secondary event clear registers (SECR/SECRH) clear the status of the secondary event registers (SER/SERH). CPU writes of 1 clear the corresponding set bits in SER/SERH. Writes of 0 have no effect.

The SECR is shown in Figure 4-47 and described in Table 4-49. The SECRH is shown in Figure 4-48 and described in Table 4-50.

**Figure 4-47. Secondary Event Clear Register (SECR)**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| E31 | E30 | E29 | E28 | E27 | E26 | E25 | E24 | E23 | E22 | E21 | E20 | E19 | E18 | E17 | E16 |
| W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| E15 | E14 | E13 | E12 | E11 | E10 | E9 | E8 | E7 | E6 | E5 | E4 | E3 | E2 | E1 | E0 |
| W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 |

LEGEND: W = Write only; -*n* = value after reset

**Table 4-49. Secondary Event Clear Register (SECR) Field Descriptions**

| Bit | Field | Value | Description |
|---|---|---|---|
| 31-0 | E*n* | | Secondary event clear register |
| | | 0 | No effect. |
| | | 1 | Corresponding bit in the secondary event register (SER) is cleared (E*n* = 0). |

**Figure 4-48. Secondary Event Clear Register High (SECRH)**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| E63 | E62 | E61 | E60 | E59 | E58 | E57 | E56 | E55 | E54 | E53 | E52 | E51 | E50 | E49 | E48 |
| W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| E47 | E46 | E45 | E44 | E43 | E42 | E41 | E40 | E39 | E38 | E37 | E36 | E35 | E34 | E33 | E32 |
| W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 |

LEGEND: W = Write only; -*n* = value after reset

**Table 4-50. Secondary Event Clear Register High (SECRH) Field Descriptions**

| Bit | Field | Value | Description |
|---|---|---|---|
| 31-0 | E*n* | | Secondary event clear register. |
| | | 0 | No effect. |
| | | 1 | Corresponding bit in the secondary event registers high (SERH) is cleared (E*n* = 0). |

### 4.3.6 Interrupt Registers

All DMA/QDMA channels can be set to assert an EDMA3CC completion interrupt to the CPU on transfer completion, by appropriately configuring the PaRAM entry associated with the channels. The following set of registers is used for the transfer completion interrupt reporting/generating by the EDMA3CC. See Section 2.9 for more details on EDMA3CC completion interrupt generation.

#### 4.3.6.1 Interrupt Enable Registers (IER, IERH)

Interrupt enable registers (IER/IERH) are used to enable/disable the transfer completion interrupt generation by the EDMA3CC for all DMA/QDMA channels. The IER/IERH cannot be written to directly. To set any interrupt bit in IER/IERH, a 1 must be written to the corresponding interrupt bit in the interrupt enable set registers (IESR/IESRH). Similarly, to clear any interrupt bit in IER/IERH, a 1 must be written to the corresponding interrupt bit in the interrupt enable clear registers (IECR/IECRH).

The IER is shown in Figure 4-49 and described in Table 4-51. The IERH is shown in Figure 4-50 and described in Table 4-52.

**Figure 4-49. Interrupt Enable Register (IER)**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| I31 | I30 | I29 | I28 | I27I | I26 | I25 | I24 | I23 | I22 | I21 | I20 | I19 | I18 | I17 | I16 |
| R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| I15 | I14 | I13 | I12 | I11 | I10 | I9 | I8 | I7 | I6 | I5 | I4 | I3 | I2 | I1 | I0 |
| R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 |

LEGEND: R = Read only; -*n* = value after reset

**Table 4-51. Interrupt Enable Register (IER) Field Descriptions**

| Bit | Field | Value | Description |
|---|---|---|---|
| 31-0 | E*n* | | Interrupt enable for channels 0-31. |
| | | 0 | Interrupt is not enabled. |
| | | 1 | Interrupt is enabled. |

**Figure 4-50. Interrupt Enable Register High (IERH)**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| I63 | I62 | I61 | I60 | I59 | I58 | I57 | I56 | I55 | I54 | I53 | I52 | I51 | I50 | I49 | I48 |
| R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| I47 | I46 | I45 | I44 | I43 | I42 | I41 | I40 | I39 | I38 | I37 | I36 | I35 | I34 | I33 | I32 |
| R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 |

LEGEND: R = Read only; -*n* = value after reset

**Table 4-52. Interrupt Enable Register High (IERH) Field Descriptions**

| Bit | Field | Value | Description |
|---|---|---|---|
| 31-0 | E*n* | | Interrupt enable for channels 32-63. |
| | | 0 | Interrupt is not enabled. |
| | | 1 | Interrupt is enabled. |

### 4.3.6.2 Interrupt Enable Clear Register (IECR, IECRH)

The interrupt enable clear registers (IECR/IECRH) are used to clear interrupts. Writes of 1 to the bits in IECR/IECRH clear the corresponding interrupt bits in the interrupt enable registers (IER/IERH); writes of 0 have no effect.

The IECR is shown in Figure 4-51 and described in Table 4-53. The IECRH is shown in Figure 4-52 and described in Table 4-54.

**Figure 4-51. Interrupt Enable Clear Register (IECR)**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| I31 | I30 | I29 | I28 | I27 | I26 | I25 | I24 | I23 | I22 | I21 | I20 | I19 | I18 | I17 | 16 |
| W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| I15 | I14 | I13 | I12 | I11 | I10 | I9 | I8 | I7 | I6 | I5 | I4 | I3 | I2 | I1 | I0 |
| W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 |

LEGEND: W = Write only; -*n* = value after reset

**Table 4-53. Interrupt Enable Clear Register (IECR) Field Descriptions**

| Bit | Field | Value | Description |
|---|---|---|---|
| 31-0 | E*n* | | Interrupt enable clear for channels 0-31. |
| | | 0 | No effect |
| | | 1 | Corresponding bit in the interrupt enable register (IER) is cleared (I*n* = 0). |

**Figure 4-52. Interrupt Enable Clear Register High (IECRH)**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| I63 | I62 | I61 | I60 | I59 | I58 | I57 | I56 | I55 | I54 | I53 | I52 | I51 | I50 | I49 | I48 |
| W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| I47 | I46 | I45 | I44 | I43 | I42 | I41 | I40 | I39 | I38 | I37 | I36 | I35 | I34 | I33 | I32 |
| W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 |

LEGEND: W = Write only; -*n* = value after reset

**Table 4-54. Interrupt Enable Clear Register High (IECRH) Field Descriptions**

| Bit | Field | Value | Description |
|---|---|---|---|
| 31-0 | E*n* | | Interrupt enable clear for channels 32-63. |
| | | 0 | No effect. |
| | | 1 | Corresponding bit in the interrupt enable register high (IERH) is cleared (I*n* = 0). |

### 4.3.6.3 Interrupt Enable Set Registers (IESR, IESRH)

The interrupt enable set registers (IESR/IESRH) are used to enable interrupts. Writes of 1 to the bits in IESR/IESRH set the corresponding interrupt bits in the interrupt enable registers (IER/IERH); writes of 0 have no effect.

The IESR is shown in Figure 4-53 and described in Table 4-55. The IESRH is shown in Figure 4-54 and described in Table 4-56.

**Figure 4-53. Interrupt Enable Set Register (IESR)**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| I31 | I30 | I29 | I28 | I27 | I26 | I25 | I24 | I23 | I22 | I21 | I20 | I19 | II8 | I17 | I16 |
| W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| I15 | I14 | I13 | I12 | I11 | I10 | I9 | I8 | I7 | I6 | I5 | I4 | I3 | I2 | I1 | I0 |
| W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 |

LEGEND: W = Write only; -*n* = value after reset

**Table 4-55. Interrupt Enable Set Register (IESR) Field Descriptions**

| Bit | Field | Value | Description |
|---|---|---|---|
| 31-0 | E*n* | | Interrupt enable set for channels 0-31. |
| | | 0 | No effect. |
| | | 1 | Corresponding bit in the interrupt enable register (IER) is set (I*n* = 1). |

**Figure 4-54. Interrupt Enable Set Register High (IESRH)**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| I63 | I62 | I61 | I60 | I59 | I58 | I57 | I56 | I55 | I54 | I53 | I52 | I51 | I50 | I49 | I48 |
| W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| I47 | I46 | I45 | I44 | I43 | I42 | I41 | I40 | I39 | I38 | I37 | I36 | I35 | I34 | I33 | I32 |
| W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 |

LEGEND: W = Write only; -*n* = value after reset

**Table 4-56. Interrupt Enable Set Register High (IESRH) Field Descriptions**

| Bit | Field | Value | Description |
|---|---|---|---|
| 31-0 | E*n* | | Interrupt enable set for channels 32-63. |
| | | 0 | No effect. |
| | | 1 | Corresponding bit in the interrupt enable register high (IERH) is set (I*n* = 1). |

### 4.3.6.4 Interrupt Pending Register (IPR, IPRH)

If the TCINTEN and/or ITCINTEN bit in the channel option parameter (OPT) is set to 1 in the PaRAM entry associated with the channel (DMA or QDMA), then the EDMA3TC (for normal completion) or the EDMA3CC (for early completion) returns a completion code on transfer or intermediate transfer completion. The value of the returned completion code is equal to the TCC bit in OPT for the PaRAM entry associated with the channel.

When an interrupt transfer completion code with TCC = *n* is detected by the EDMA3CC, then the corresponding bit is set in the interrupt pending register (IPR.I*n*, if *n* = 0 to 31; IPRH.I*n,* if *n* = 32 to 63). Note that once a bit is set in the interrupt pending registers, it remains set; it is your responsibility to clear these bits. The bits set in IPR/IPRH are cleared by writing a 1 to the corresponding bits in the interrupt clear registers (ICR/ICRH).

The IPR is shown in Figure 4-55 and described in Table 4-57. The IPRH is shown in Figure 4-56 and described in Table 4-58.

**Figure 4-55. Interrupt Pending Register (IPR)**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| I31 | I30 | I29 | I28 | I27 | I26 | I25 | I24 | I23 | I22 | I21 | I20 | I19 | I18 | I17 | I16 |
| R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| I15 | I14 | I13 | I12 | I11 | I10 | I9 | I8 | I7 | I6 | I5 | I4 | I3 | I2 | I1 | I0 |
| R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 |

LEGEND: R = Read only; -*n* = value after reset

**Table 4-57. Interrupt Pending Register (IPR) Field Descriptions**

| Bit | Field | Value | Description |
|---|---|---|---|
| 31-0 | I*n* | | Interrupt pending for TCC = 0-31. |
| | | 0 | Interrupt transfer completion code is not detected or was cleared. |
| | | 1 | Interrupt transfer completion code is detected (I*n* = 1, *n* = EDMA3TC[5:0]). |

**Figure 4-56. Interrupt Pending Register High (IPRH)**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| I63 | I62 | I61 | I60 | I59 | I58 | I57 | I56 | I55 | I54 | I53 | I52 | I51 | I50 | I49 | I48 |
| R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| I47 | I46 | I45 | I44 | I43 | I42 | I41 | I40 | I39 | I38 | I37 | I36 | I35 | I34 | I33 | I32 |
| R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 |

LEGEND: R = Read only; -*n* = value after reset

**Table 4-58. Interrupt Pending Register High (IPRH) Field Descriptions**

| Bit | Field | Value | Description |
|---|---|---|---|
| 31-0 | I*n* | | Interrupt pending for TCC = 32-63. |
| | | 0 | Interrupt transfer completion code is not detected or was cleared. |
| | | 1 | Interrupt transfer completion code is detected (I*n* = 1, *n* = EDMA3TC[5:0]). |

### 4.3.6.5 Interrupt Clear Registers (ICR, ICRH)

The bits in the interrupt pending registers (IPR/IPRH) are cleared by writing a 1 to the corresponding bits in the interrupt clear registers(ICR/ICRH). Writes of 0 have no effect. All set bits in IPR/IPRH must be cleared to allow EDMA3CC to assert additional transfer completion interrupts.

The ICR is shown in Figure 4-57 and described in Table 4-59. The ICRH is shown in Figure 4-58 and described in Table 4-60.

**Figure 4-57. Interrupt Clear Register (ICR)**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| I31 | I30 | I29 | I28 | I27 | I26 | I25 | I24 | I23 | I22 | I21 | I20 | I19 | I18 | I17 | I16 |
| W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| I15 | I14 | I13 | I12 | I11 | I10 | I9 | I8 | I7 | I6 | I5 | I4 | I3 | I2 | I1 | I0 |
| W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 |

LEGEND: W = Write only; -*n* = value after reset

**Table 4-59. Interrupt Clear Register (ICR) Field Descriptions**

| Bit | Field | Value | Description |
|---|---|---|---|
| 31-0 | I*n* | | Interrupt clear register for TCC = 0-31. |
| | | 0 | No effect. |
| | | 1 | Corresponding bit in the interrupt pending register (IPR) is cleared (I*n* = 0). |

**Figure 4-58. Interrupt Clear Register High (ICRH)**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| I63 | I62 | I61 | I60 | I59 | I58 | I57 | I56 | I55 | I54 | I53 | I52 | I51 | I50 | I49 | I48 |
| W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| I47 | I46 | I45 | I44 | I43 | I42 | I41 | I40 | I39 | I38 | I37 | I36 | I35 | I34 | I33 | I32 |
| W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 |

LEGEND: W = Write only; -*n* = value after reset

**Table 4-60. Interrupt Clear Register High (ICRH) Field Descriptions**

| Bit | Field | Value | Description |
|---|---|---|---|
| 31-0 | I*n* | | Interrupt clear register for TCC = 32-63. |
| | | 0 | No effect. |
| | | 1 | Corresponding bit in the interrupt pending register high (IPRH) is cleared (I*n* = 0). |

### 4.3.6.6 Interrupt Evaluate Register (IEVAL)

The interrupt evaluate register (IEVAL) is the only register that physically exists in both the global region and the shadow regions. In other words, the read/write accessibility for the shadow region IEVAL is not affected by the DMA/QDMA region access registers (DRAE*m*/DRAEH*m*, QRAE*n*/QRAEH*n*). IEVAL is needed for robust ISR operations to ensure that interrupts are not missed by the CPU.

The IEVAL is shown in Figure 4-59 and described in Table 4-61.

**Figure 4-59. Interrupt Evaluate Register (IEVAL)**

| 31 | 16 |
|---|---|
| Reserved | |
| R-0 | |

| 15 | 2 | 1 | 0 |
|---|---|---|---|
| Reserved | | Rsvd | EVAL |
| R-0 | | W-0 | W-0 |

LEGEND: R = Read only; W = Write only; -*n* = value after reset

**Table 4-61. Interrupt Evaluate Register (IEVAL) Field Descriptions**

| Bit | Field | Value | Description |
|---|---|---|---|
| 31-2 | Reserved | 0 | Reserved |
| 1 | Reserved | 0 | Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior. |
| 0 | EVAL | | Interrupt evaluate. |
| | | 0 | No effect. |
| | | 1 | Causes EDMA3CC completion interrupt to be pulsed, if any enabled (IER*n*/IERH*n* = 1) interrupts are still pending (IPR*n*/IPRH*n* = 1).<br>The EDMA3CC completion region interrupt that is pulsed depends on which IEVAL is being exercised. For example, writing to the EVAL bit in IEVAL0 pulses the region 0 completion interrupt, but writing to the EVAL bit in IEVAL1 pulses the region 1 completion interrupt. |

### 4.3.7 QDMA Registers

The following sets of registers control the QDMA channels in the EDMA3CC. The QDMA channels (with the exception of the QDMA queue number register) consist of a set of registers, each of which have a bit location. Each bit position corresponds to a QDMA channel number. The QDMA channel registers are accessible via read/writes to the global address range. They are also accessible via read/writes to the shadow address range. The read/write accessibility in the shadow region address region is controlled by the QDMA region access registers (QRAE*n*/QRAEH*n*). Section 2.7 details shadow region/global region usage.

#### 4.3.7.1 QDMA Event Register (QER)

The QDMA event register (QER) channel *n* bit is set (E*n* = 1) when the CPU or any EDMA programmer (including EDMA3) performs a write to the trigger word (using the QDMA channel *n* mapping register (QCHMAP*n*)) in the PaRAM entry associated with QDMA channel *n* (which is also programmed using QCHMAP*n*). The E*n* bit is also set when the EDMA3CC performs a link update on a PaRAM address that matches the QCHMAP*n* settings. The QDMA event is latched only if the QDMA event enable register (QEER) channel *n* bit is also enabled (QEER.E*n* = 1). Once a bit is set in QER, then the corresponding QDMA event (auto-trigger) is evaluated by the EDMA3CC logic for an associated transfer request submission to the transfer controllers.

The setting of an event is a higher priority relative to clear operations (via hardware). If set and clear conditions occur concurrently, the set condition wins. If the event was previously set, then the QDMA event missed register (QEMR) would be set because an event is lost. If the event was previously clear, then the event remains set and is prioritized for submission to the event queues.

The set bits in QER are only cleared when the transfer request associated with the corresponding channels has been processed by the EDMA3CC and submitted to the transfer controller. If the E*n* bit is already set and a QDMA event for the same QDMA channel occurs prior to the original being cleared, then the second missed event is latched in QEMR (E*n* =1).

The QER is shown in Figure 4-60 and described in Table 4-62.

**Figure 4-60. QDMA Event Register (QER)**

| 31 | 16 |
|---|---|
| Reserved | |
| R-0 | |

| 15 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|
| Reserved | | E7 | E6 | E5 | E4 | E3 | E2 | E1 | E0 |
| R-0 | | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 |

LEGEND: R = Read only; -*n* = value after reset

**Table 4-62. QDMA Event Register (QER) Field Descriptions**

| Bit | Field | Value | Description |
|---|---|---|---|
| 31-8 | Reserved | 0 | Reserved |
| 7-0 | E*n* | | QDMA event for channels 0-7. |
| | | 0 | No effect. |
| | | 1 | Corresponding QDMA event is prioritized versus other pending DMA/QDMA events for submission to the EDMA3TC. |

### 4.3.7.2 QDMA Event Enable Register (QEER)

The EDMA3CC provides the option of selectively enabling/disabling each channel in the QDMA event register (QER) by using the QDMA event enable register (QEER). If any of the event bits in QEER is set to 1 (using the QDMA event enable set register, QEESR), it will enable that corresponding event. Alternatively, if any event bit in QEER is cleared (using the QDMA event enable clear register, QEECR), it will disable the corresponding QDMA channel. The QDMA event register will not latch any event for a QDMA channel, if it is not enabled via QEER.

The QEER is shown in Figure 4-61 and described in Table 4-63.

**Figure 4-61. QDMA Event Enable Register (QEER)**

| 31 | 16 |
|---|---|
| Reserved | |
| R-0 | |

| 15 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|
| Reserved | | E7 | E6 | E5 | E4 | E3 | E2 | E1 | E0 |
| R-0 | | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 |

LEGEND: R = Read only; -*n* = value after reset

**Table 4-63. QDMA Event Enable Register (QEER) Field Descriptions**

| Bit | Field | Value | Description |
|---|---|---|---|
| 31-8 | Reserved | 0 | Reserved |
| 7-0 | E*n* | | QDMA event enable for channels 0-7. |
| | | 0 | QDMA channel *n* is not enabled. QDMA event will not be recognized and will not latch in the QDMA event register (QER). |
| | | 1 | QDMA channel *n* is enabled. QDMA events will be recognized and will get latched in the QDMA event register (QER). |

### 4.3.7.3 QDMA Event Enable Clear Register (QEECR)

The QDMA event enable register (QEER) cannot be modified by directly writing to the register, in order to ease the software burden when multiple tasks are attempting to simultaneously modify these registers. The QDMA event enable clear register (QEECR) is used to disable events. Writes of 1 to the bits in QEECR clear the corresponding QDMA channel bits in QEER; writes of 0 have no effect.

The QEECR is shown in Figure 4-62 and described in Table 4-64.

**Figure 4-62. QDMA Event Enable Clear Register (QEECR)**

| 31 | 16 |
|---|---|
| Reserved | |
| R-0 | |

| 15 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|
| Reserved | | E7 | E6 | E5 | E4 | E3 | E2 | E1 | E0 |
| R-0 | | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 |

LEGEND: R = Read only; W = Write only; -*n* = value after reset

**Table 4-64. QDMA Event Enable Clear Register (QEECR) Field Descriptions**

| Bit | Field | Value | Description |
|---|---|---|---|
| 31-8 | Reserved | 0 | Reserved |
| 7-0 | E*n* | | QDMA event enable clear for channels 0-7. |
| | | 0 | No effect. |
| | | 1 | QDMA event is disabled. Corresponding bit in the QDMA event enable register (QEER) is cleared (E*n* = 0). |

### 4.3.7.4 QDMA Event Enable Set Register (QEESR)

The QDMA event enable register (QEER) cannot be modified by directly writing to the register, in order to ease the software burden when multiple tasks are attempting to simultaneously modify these registers. The QDMA event enable set register (QEESR) is used to enable events. Writes of 1 to the bits in QEESR set the corresponding QDMA channel bits in QEER; writes of 0 have no effect.

The QEESR is shown in Figure 4-63 and described in Table 4-65.

**Figure 4-63. QDMA Event Enable Set Register (QEESR)**

| 31 | 16 |
|---|---|
| Reserved | |
| R-0 | |

| 15 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|
| Reserved | | E7 | E6 | E5 | E4 | E3 | E2 | E1 | E0 |
| R-0 | | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 |

LEGEND: R = Read only; W = Write only; -*n* = value after reset

**Table 4-65. QDMA Event Enable Set Register (QEESR) Field Descriptions**

| Bit | Field | Value | Description |
|---|---|---|---|
| 31-8 | Reserved | 0 | Reserved |
| 7-0 | E*n* | | QDMA event enable set for channels 0-7. |
| | | 0 | No effect. |
| | | 1 | QDMA event is enabled. Corresponding bit in the QDMA event enable register (QEER) is set (E*n* = 1). |

### 4.3.7.5 QDMA Secondary Event Register (QSER)

The QDMA secondary event register (QSER) provides information on the state of a QDMA event. If at any time a bit corresponding to a QDMA channel is set in QSER, that implies that the corresponding QDMA event is in the queue. Once a bit corresponding to a QDMA channel is set in QSER, the EDMA3CC does not prioritize additional events on the same QDMA channel. Depending on the condition that lead to the setting of the QSER bits, either the EDMA3CC hardware or the software (using QSECR) needs to clear the QSER bits for the EDMA3CC to evaluate subsequent QDMA events on the channel. Based on whether the associated TR is valid, or it is a null or dummy TR, the implications on the state of QSER and the required user action in order to submit another QDMA transfer might be different.

The QSER is shown in Figure 4-64 and described in Table 4-66.

**Figure 4-64. QDMA Secondary Event Register (QSER)**

| 31 | 16 |
|---|---|
| Reserved | |
| R-0 | |

| 15 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|
| Reserved | | E7 | E6 | E5 | E4 | E3 | E2 | E1 | E0 |
| R-0 | | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 |

LEGEND: R = Read only; -*n* = value after reset

**Table 4-66. QDMA Secondary Event Register (QSER) Field Descriptions**

| Bit | Field | Value | Description |
|---|---|---|---|
| 31-8 | Reserved | 0 | Reserved |
| 7-0 | E*n* | | QDMA secondary event register for channels 0-7. |
| | | 0 | QDMA event is not currently stored in the event queue. |
| | | 1 | QDMA event is currently stored in event queue. EDMA3CC will not prioritize additional events. |

### 4.3.7.6 QDMA Secondary Event Clear Register (QSECR)

The QDMA secondary event clear register (QSECR) clears the status of the QDMA secondary event register (QSER) and the QDMA event register (QER). CPU writes of 1 clear the corresponding set bits in QSER and QER. Writes of 0 have no effect. Note that this differs from the secondary event clear register (SECR) operation, which only clears the secondary event register (SER) bits and does not affect the event registers.

The QSECR is shown in Figure 4-65 and described in Table 4-67.

**Figure 4-65. QDMA Secondary Event Clear Register (QSECR)**

| 31 | 16 |
|---|---|
| Reserved | |
| R-0 | |

| 15 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|
| Reserved | | E7 | E6 | E5 | E4 | E3 | E2 | E1 | E0 |
| R-0 | | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 | W-0 |

LEGEND: R = Read only; W = Write only; -*n* = value after reset

**Table 4-67. QDMA Secondary Event Clear Register (QSECR) Field Descriptions**

| Bit | Field | Value | Description |
|---|---|---|---|
| 31-8 | Reserved | 0 | Reserved |
| 7-0 | E*n* | | QDMA secondary event clear register for channels 0-7. |
| | | 0 | No effect. |
| | | 1 | Corresponding bit in the QDMA secondary event register (QSER) and the QDMA event register (QER) is cleared (E*n* = 0). |

## 4.4 EDMA3 Transfer Controller (EDMA3TC) Registers

Table 4-68 lists the memory-mapped registers for the EDMA3 transfer controller (EDMA3TC). See the device-specific data manual for the memory address of these registers. All other register offset addresses not listed in Table 4-68 should be considered as reserved locations and the register contents should not be modified.

**Table 4-68. EDMA3 Transfer Controller (EDMA3TC) Registers**

| Offset | Acronym | Register Description | Section |
|---|---|---|---|
| 00h | PID | Peripheral Identification Register | Section 4.4.1 |
| 04h | TCCFG | EDMA3TC Configuration Register | Section 4.4.2 |
| 0100h | TCSTAT | EDMA3TC Channel Status Register | Section 4.4.3 |
| 0120h | ERRSTAT | Error Status Register | Section 4.4.4.1 |
| 0124h | ERREN | Error Enable Register | Section 4.4.4.2 |
| 0128h | ERRCLR | Error Clear Register | Section 4.4.4.3 |
| 012Ch | ERRDET | Error Details Register | Section 4.4.4.4 |
| 0130h | ERRCMD | Error Interrupt Command Register | Section 4.4.4.5 |
| 0140h | RDRATE | Read Command Rate Register | Section 4.4.5 |
| 0240h | SAOPT | Source Active Options Register | Section 4.4.6.1 |
| 0244h | SASRC | Source Active Source Address Register | Section 4.4.6.2 |
| 0248h | SACNT | Source Active Count Register | Section 4.4.6.3 |
| 024Ch | SADST | Source Active Destination Address Register | Section 4.4.6.4 |
| 0250h | SABIDX | Source Active B-Index Register | Section 4.4.6.5 |
| 0254h | SAMPPRXY | Source Active Memory Protection Proxy Register | Section 4.4.6.6 |
| 0258h | SACNTRLD | Source Active Count Reload Register | Section 4.4.6.7 |
| 025Ch | SASRCBREF | Source Active Source Address B-Reference Register | Section 4.4.6.8 |
| 0260h | SADSTBREF | Source Active Destination Address B-Reference Register | Section 4.4.6.9 |
| 0280h | DFCNTRLD | Destination FIFO Set Count Reload Register | Section 4.4.6.10 |
| 0284h | DFSRCBREF | Destination FIFO Set Source Address B-Reference Register | Section 4.4.6.11 |
| 0288h | DFDSTBREF | Destination FIFO Set Destination Address B-Reference Register | Section 4.4.6.12 |
| 0300h | DFOPT0 | Destination FIFO Options Register 0 | Section 4.4.6.13 |
| 0304h | DFSRC0 | Destination FIFO Source Address Register 0 | Section 4.4.6.14 |
| 0308h | DFCNT0 | Destination FIFO Count Register 0 | Section 4.4.6.15 |
| 030Ch | DFDST0 | Destination FIFO Destination Address Register 0 | Section 4.4.6.16 |
| 0310h | DFBIDX0 | Destination FIFO B-Index Register 0 | Section 4.4.6.17 |
| 0314h | DFMPPRXY0 | Destination FIFO Memory Protection Proxy Register 0 | Section 4.4.6.18 |
| 0340h | DFOPT1 | Destination FIFO Options Register 1 | Section 4.4.6.13 |
| 0344h | DFSRC1 | Destination FIFO Source Address Register 1 | Section 4.4.6.14 |
| 0348h | DFCNT1 | Destination FIFO Count Register 1 | Section 4.4.6.15 |
| 034Ch | DFDST1 | Destination FIFO Destination Address Register 1 | Section 4.4.6.16 |
| 0350h | DFBIDX1 | Destination FIFO B-Index Register 1 | Section 4.4.6.17 |
| 0354h | DFMPPRXY1 | Destination FIFO Memory Protection Proxy Register 1 | Section 4.4.6.18 |
| 0380h | DFOPT2 | Destination FIFO Options Register 2 | Section 4.4.6.13 |
| 0384h | DFSRC2 | Destination FIFO Source Address Register 2 | Section 4.4.6.14 |
| 0388h | DFCNT2 | Destination FIFO Count Register 2 | Section 4.4.6.15 |
| 038Ch | DFDST2 | Destination FIFO Destination Address Register 2 | Section 4.4.6.16 |
| 0390h | DFBIDX2 | Destination FIFO B-Index Register 2 | Section 4.4.6.17 |
| 0394h | DFMPPRXY2 | Destination FIFO Memory Protection Proxy Register 2 | Section 4.4.6.18 |
| 03C0h | DFOPT3 | Destination FIFO Options Register 3 | Section 4.4.6.13 |
| 03C4h | DFSRC3 | Destination FIFO Source Address Register 3 | Section 4.4.6.14 |
| 03C8h | DFCNT3 | Destination FIFO Count Register 3 | Section 4.4.6.15 |

**Table 4-68. EDMA3 Transfer Controller (EDMA3TC) Registers (continued)**

| Offset | Acronym | Register Description | Section |
|---|---|---|---|
| 03CCh | DFDST3 | Destination FIFO Destination Address Register 3 | Section 4.4.6.16 |
| 03D0h | DFBIDX3 | Destination FIFO B-Index Register 3 | Section 4.4.6.17 |
| 03D4h | DFMPPRXY3 | Destination FIFO Memory Protection Proxy Register 3 | Section 4.4.6.18 |

### 4.4.1 Peripheral Identification Register (PID)

The peripheral identification register (PID) is a constant register that uniquely identifies the EDMA3TC and specific revision of the EDMA3TC. The PID is shown in Figure 4-66 and described in Table 4-69.

**Figure 4-66. Peripheral ID Register (PID)**

| 31 | 16 |
|---|---|
| PID | |
| R-4000h | |

| 15 | 0 |
|---|---|
| PID | |
| R-3B00h | |

LEGEND: R = Read only; -*n* = value after reset

**Table 4-69. Peripheral ID Register (PID) Field Descriptions**

| Bit | Field | Value | Description |
|---|---|---|---|
| 31-0 | PID | | Peripheral identifier. |
| | | 4000 3B00h | Uniquely identifies the EDMA3TC and the specific revision of the EDMA3TC. |

### 4.4.2 EDMA3TC Configuration Register (TCCFG)

The EDMA3TC configuration register (TCCFG) is shown in Figure 4-67 and described in Table 4-70.

**Figure 4-67. EDMA3TC Configuration Register (TCCFG)**

| 31 | 16 |
|---|---|
| Reserved | |
| R-0 | |

| 15 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | DREGDEPTH | | Reserved | | BUSWIDTH | | Rsvd | FIFOSIZE | |
| R-0 | | R-x | | R-0 | | R-x | | R-0 | R-x | |

LEGEND: R = Read only; -*n* = value after reset; -x = value is indeterminate after reset

**Table 4-70. EDMA3TC Configuration Register (TCCFG) Field Descriptions**

| Bit | Field | Value | Description |
|---|---|---|---|
| 31-10 | Reserved | 0 | Reserved |
| 9-8 | DREGDEPTH | 0-3h | Destination register FIFO depth parameterization. |
| | | 0 | 1 entry |
| | | 1h | 2 entry |
| | | 2h | 4 entry |
| | | 3h | Reserved |
| 7-6 | Reserved | 0 | Reserved |
| 5-4 | BUSWIDTH | 0-3h | Bus width parameterization. |
| | | 0 | 32 bit |
| | | 1h | 64 bit |
| | | 2h | 128 bit |
| | | 3h | Reserved |
| 3 | Reserved | 0 | Reserved |
| 2-0 | FIFOSIZE | 0-7h | FIFO size. |
| | | 0 | 32-byte FIFO |
| | | 1h | 64-byte FIFO |
| | | 2h | 128-byte FIFO |
| | | 3h | 256-byte FIFO |
| | | 4h | 512-byte FIFO |
| | | 5h-7h | Reserved |

### 4.4.3 EDMA3TC Channel Status Register (TCSTAT)

The EDMA3TC channel status register (TCSTAT) is shown in Figure 4-68 and described in Table 4-71.

**Figure 4-68. EDMA3TC Channel Status Register (TCSTAT)**

| 31 | 16 |
|---|---|
| Reserved | |
| R-0 | |

| 15 | 13 | 12 | 11 | 10 | 8 |
|---|---|---|---|---|---|
| Reserved | | DFSTRTPTR | | Reserved | |
| R-0 | | R-0 | | R-0 | |

| 7 | 6 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|
| Reserved | DSTACTV | | Reserved | WSACTV | SRCACTV | PROGBUSY |
| R-0 | R-0 | | R-0 | R-0 | R-0 | R-0 |

LEGEND: R = Read only; -*n* = value after reset

**Table 4-71. EDMA3TC Channel Status Register (TCSTAT) Field Descriptions**

| Bit | Field | Value | Description |
|---|---|---|---|
| 31-13 | Reserved | 0 | Reserved |
| 12-11 | DFSTRTPTR | 0-3h | Destination FIFO start pointer. The offset to the head entry of the destination register FIFO, in units of *entries*. |
| 10-7 | Reserved | 0 | Reserved |
| 6-4 | DSTACTV | 0-7h | Destination active state. Specifies the number of transfer requests (TRs) that are resident in the destination register FIFO at a given instant. This bit field can be primarily used for advanced debugging. |
| | | 0 | Destination FIFO is empty. |
| | | 1h | Destination FIFO contains 1 TR. |
| | | 2h | Destination FIFO contains 2 TR. |
| | | 3h | Destination FIFO contains 3 TR. |
| | | 4h | Destination FIFO contains 4 TR. (Full if DSTREGDEPTH == 4) |
| | | | If the destination register FIFO is empty, then any TR written to Prog Set immediately transitions to the destination register FIFO. If the destination register FIFO is not empty and not full, then any TR written to Prog Set immediately transitions to the destination register FIFO set if the source active state (SRCACTV) bit is set to idle. |
| | | | If the destination register FIFO is full, then TRs cannot transition to the destination register FIFO. The destination register FIFO becomes not full when the TR at the head of the destination register FIFO is completed. |
| | | 5h-7h | Reserved |
| 3 | Reserved | 0 | Reserved |
| 2 | WSACTV | | Write status active. |
| | | 0 | Write status is not pending. Write status has been received for all previously issued write commands. |
| | | 1 | Write status is pending. Write status has not been received for all previously issued write commands. |
| 1 | SRCACTV | | Source active state. |
| | | 0 | Source active set is idle and is available for programming by the EDMA3CC. Source active register set contains a previously processed transfer request. |
| | | 1 | Source active set is busy servicing a transfer request. |
| 0 | PROGBUSY | | Program register set busy. |
| | | 0 | Program set idle and is available for programming by the EDMA3CC. |
| | | 1 | Program set busy. |

## 4.4.4 Error Registers

### 4.4.4.1 Error Status Register (ERRSTAT)

The error status register (ERRSTAT) is shown in Figure 4-69 and described in Table 4-72.

**Figure 4-69. Error Status Register (ERRSTAT)**

| 31 | 16 |
|---|---|
| Reserved | |
| R-0 | |

| 15 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|
| Reserved | | MMRAERR | TRERR | Reserved | BUSERR |
| R-0 | | R-0 | R-0 | R-0 | R-0 |

LEGEND: R = Read only; -*n* = value after reset

**Table 4-72. Error Status Register (ERRSTAT) Field Descriptions**

| Bit | Field | Value | Description |
|---|---|---|---|
| 31-4 | Reserved | 0 | Reserved |
| 3 | MMRAERR | | MMR address error. |
| | | 0 | MMR address error is not detected. |
| | | 1 | User attempted to read or write to an invalid address in configuration memory map. |
| 2 | TRERR | | Transfer request (TR) error event. |
| | | 0 | Transfer request (TR) error is not detected. |
| | | 1 | Transfer request (TR) detected that violates constant addressing mode transfer (SAM or DAM is set to 1) alignment rules or has ACNT or BCNT == 0. |
| 1 | Reserved | 0 | Reserved |
| 0 | BUSERR | | Bus error event. |
| | | 0 | Bus error is not detected. |
| | | 1 | EDMA3TC has detected an error at source or destination address. Error information can be read from the error details register (ERRDET). |

### 4.4.4.2 Error Enable Register (ERREN)

The error enable register (ERREN) is shown in Figure 4-70 and described in Table 4-73. When any of the enable bits in ERREN is set, a bit set in the corresponding error status register (ERRSTAT) causes an assertion of the EDMA3TC interrupt.

**Figure 4-70. Error Enable Register (ERREN)**

| 31 | 16 |
|---|---|
| Reserved | |
| R-0 | |

| 15 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|
| Reserved | | MMRAERR | TRERR | Reserved | BUSERR |
| R-0 | | R/W-0 | R/W-0 | R/W-0 | R/W-0 |

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

**Table 4-73. Error Enable Register (ERREN) Field Descriptions**

| Bit | Field | Value | Description |
|---|---|---|---|
| 31-4 | Reserved | 0 | Reserved |
| 3 | MMRAERR | | Interrupt enable for MMR address error (MMRAERR). |
| | | 0 | MMRAERR is disabled. |
| | | 1 | MMRAERR is enabled and contributes to the state of EDMA3TC error interrupt generation |
| 2 | TRERR | | Interrupt enable for transfer request error (TRERR). |
| | | 0 | TRERR is disabled. |
| | | 1 | TRERR is enabled and contributes to the state of EDMA3TC error interrupt generation. |
| 1 | Reserved | 0 | Reserved. Always write 0 to this bit; writes of 1 to this bit are not supported and attempts to do so may result in undefined behavior. |
| 0 | BUSERR | | Interrupt enable for bus error (BUSERR). |
| | | 0 | BUSERR is disabled. |
| | | 1 | BUSERR is enabled and contributes to the state of EDMA3TC error interrupt generation. |

### 4.4.4.3 Error Clear Register (ERRCLR)

The error clear register (ERRCLR) is shown in Figure 4-71 and described in Table 4-74.

**Figure 4-71. Error Clear Register (ERRCLR)**

| 31 | 16 |
|---|---|
| Reserved | |
| R-0 | |

| 15 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|
| Reserved | | MMRAERR | TRERR | Reserved | BUSERR |
| R-0 | | W-0 | W-0 | R-0 | W-0 |

LEGEND: R = Read only; W = Write only; -*n* = value after reset

**Table 4-74. Error Clear Register (ERRCLR) Field Descriptions**

| Bit | Field | Value | Description |
|---|---|---|---|
| 31-4 | Reserved | 0 | Reserved |
| 3 | MMRAERR | | Interrupt enable clear for the MMR address error (MMRAERR) bit in the error status register (ERRSTAT). |
| | | 0 | No effect. |
| | | 1 | Clears the MMRAERR bit in the error status register (ERRSTAT) but does not clear the error details register (ERRDET). |
| 2 | TRERR | | Interrupt enable clear for the transfer request error (TRERR) bit in the error status register (ERRSTAT). |
| | | 0 | No effect. |
| | | 1 | Clears the TRERR bit in the error status register (ERRSTAT) but does not clear the error details register (ERRDET). |
| 1 | Reserved | 0 | Reserved |
| 0 | BUSERR | | Interrupt clear for the bus error (BUSERR) bit in the error status register (ERRSTAT). |
| | | 0 | No effect. |
| | | 1 | Clears the BUSERR bit in the error status register (ERRSTAT) and clears the error details register (ERRDET). |

### 4.4.4.4 Error Details Register (ERRDET)

The error details register (ERRDET) is shown in Figure 4-72 and described in Table 4-75.

**Figure 4-72. Error Details Register (ERRDET)**

| 31 | 18 | 17 | 16 |
|---|---|---|---|
| Reserved | | TCCHEN | TCINTEN |
| R-0 | | R-0 | R-0 |

| 15 | 14 | 13 | 8 | 7 | 4 | 3 | 0 |
|---|---|---|---|---|---|---|---|
| Reserved | | TCC | | Reserved | | STAT | |
| R-0 | | R-0 | | R-0 | | R-0 | |

LEGEND: R = Read only; -*n* = value after reset

**Table 4-75. Error Details Register (ERRDET) Field Descriptions**

| Bit | Field | Value | Description |
|---|---|---|---|
| 31-8 | Reserved | 0 | Reserved |
| 17 | TCCHEN | 0-1 | Transfer completion chaining enable. Contains the TCCHEN value in the channel options parameter (OPT) programmed by the channel controller for the read or write transaction that resulted in an error. |
| 16 | TCINTEN | 0-1 | Transfer completion interrupt enable. Contains the TCINTEN value in the channel options parameter (OPT) programmed by the channel controller for the read or write transaction that resulted in an error. |
| 15-14 | Reserved | 0 | Reserved |
| 13 - 8 | TCC | 0-3Fh | Transfer complete code. Contains the TCC value in the channel options parameter (OPT) programmed by the channel controller for the read or write transaction that resulted in an error. |
| 7-4 | Reserved | 0 | Reserved |
| 3-0 | STAT | 0-Fh | Transaction status. Stores the nonzero status/error code that was detected on the read status or write status bus. If read status and write status are returned on the same cycle, then the EDMA3TC chooses nonzero version. If both are nonzero, then the write status is treated as higher priority. |
| | | 0 | No error |
| | | 1h | Read addressing error |
| | | 2h | Read privilege error |
| | | 3h | Read timeout error |
| | | 4h | Read data error |
| | | 5h-6h | Reserved |
| | | 7h | Read exclusive operation error |
| | | 8h | Reserved |
| | | 9h | Write addressing error |
| | | Ah | Write privilege error |
| | | Bh | Write timeout error |
| | | Ch | Write data error |
| | | Dh-Eh | Reserved |
| | | Fh | Write exclusive operation error |

### 4.4.4.5 Error Interrupt Command Register (ERRCMD)

The error interrupt command register (ERRCMD) is shown in Figure 4-73 and described in Table 4-76.

**Figure 4-73. Error Interrupt Command Register (ERRCMD)**

| 31 | 16 |
|---|---|
| Reserved | |
| R-0 | |

| 15 | 1 | 0 |
|---|---|---|
| Reserved | | EVAL |
| R-0 | | W-0 |

LEGEND: R = Read only; W = Write only; -*n* = value after reset

**Table 4-76. Error Interrupt Command Register (ERRCMD) Field Descriptions**

| Bit | Field | Value | Description |
|---|---|---|---|
| 31-1 | Reserved | 0 | Reserved |
| 0 | EVAL | | Error evaluate. |
| | | 0 | No effect. |
| | | 1 | EDMA3TC error line is pulsed if any of the error status register (ERRSTAT) bits are set to 1. |

### 4.4.5 Read Command Rate Register (RDRATE)

The EDMA3 transfer controller issues Read commands at a rate controlled by the Read command rate register (RDRATE). The RDRATE defines the number of idle cycles that the Read controller must wait before issuing subsequent commands. This applies both to commands within a transfer request packet (TRP) and for commands that are issued for different transfer requests (TRs). For instance, if RDRATE is set to 4 cycles between reads, there are 3 inactive cycles between reads.

RDRATE allows flexibility in transfer controller access requests to an endpoint. For an application, RDRATE can be manipulated to slow down the access rate, so that the endpoint may service requests from other masters during the inactive EDMA3TC cycles.

The RDRATE is shown in Figure 4-74 and described in Table 4-77.

**NOTE:** It is expected that the RDRATE value for a transfer controller is static, as it is decided based on the application requirement. It is not recommended to change this setting on the fly.

**Figure 4-74. Read Command Rate Register (RDRATE)**

| 31 | 16 |
|---|---|
| Reserved | |
| R-0 | |

| 15 | 3 | 2 | 0 |
|---|---|---|---|
| Reserved | | RDRATE | |
| R-0 | | R/W-0 | |

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

**Table 4-77. Read Command Rate Register (RDRATE) Field Descriptions**

| Bit | Field | Value | Description |
|---|---|---|---|
| 31-3 | Reserved | 0 | Reserved |
| 2-0 | RDRATE | 0-7h | Read rate. Controls the number of cycles between Read commands. This is a global setting that applies to all TRs for this EDMA3TC. |
| | | 0 | Reads issued as fast as possible. |
| | | 1h | 4 cycles between reads. |
| | | 2h | 8 cycles between reads. |
| | | 3h | 16 cycles between reads. |
| | | 4h | 32 cycles between reads. |
| | | 5h-7h | Reserved |

### 4.4.6 EDMA3TC Channel Registers

The EDMA3TC channel registers are split into three parts: the programming registers, the source active registers, and the destination FIFO registers. This section describes the registers and their functions. The program register set is programmed by the channel controller and is for internal use. The source active registers and the destination FIFO registers are read-only and are provided to facilitate advanced debug capabilities. The number of destination FIFO register sets depends on the destination FIFO depth.

Both TC0 and TC1 have a destination FIFO depth of 4, and there are four sets of destination FIFO registers.

### 4.4.6.1 Source Active Options Register (SAOPT)

The source active options register (SAOPT) is shown in Figure 4-75 and described in Table 4-78.

**Figure 4-75. Source Active Options Register (SAOPT)**

| 31 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|
| Reserved | | TCCHEN | Rsvd | TCINTEN | Reserved | | TCC | |
| R-0 | | R/W-0 | R-0 | R/W-0 | R-0 | | R/W-0 | |

| 15 | 12 | 11 | 10 | 8 | 7 | 6 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| TCC | | Rsvd | FWID | | Rsvd | PRI | | Reserved | | DAM | SAM |
| R/W-0 | | R-0 | R/W-0 | | R-0 | R/W-0 | | R-0 | | R/W-0 | R/W-0 |

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

**Table 4-78. Source Active Options Register (SAOPT) Field Descriptions**

| Bit | Field | Value | Description |
|---|---|---|---|
| 31-23 | Reserved | 0 | Reserved |
| 22 | TCCHEN | | Transfer complete chaining enable. |
| | | 0 | Transfer complete chaining is disabled. |
| | | 1 | Transfer complete chaining is enabled. |
| 21 | Reserved | 0 | Reserved |
| 20 | TCINTEN | | Transfer complete interrupt enable. |
| | | 0 | Transfer complete interrupt is disabled. |
| | | 1 | Transfer complete interrupt is enabled. |
| 19-18 | Reserved | 0 | Reserved |
| 17-12 | TCC | 0-3Fh | Transfer complete code. This 6-bit code is used to set the relevant bit in CER or IPR of the EDMA3CC. |
| 11 | Reserved | 0 | Reserved |
| 10-8 | FWID | 0-7h | FIFO width. Applies if either SAM or DAM is set to constant addressing mode. |
| | | 0 | FIFO width is 8 bits. |
| | | 1h | FIFO width is 16 bits. |
| | | 2h | FIFO width is 32 bits. |
| | | 3h | FIFO width is 64 bits. |
| | | 4h | FIFO width is 128 bits. |
| | | 5h | FIFO width is 256 bits. |
| | | 6h-7h | Reserved |
| 7 | Reserved | 0 | Reserved |
| 6-4 | PRI | 0-7h | Transfer priority. Reflects the values programmed in the queue priority register (QUEPRI) in the EDMA3CC. |
| | | 0 | Priority 0 - Highest priority |
| | | 1h-6h | Priority 1 to priority 6 |
| | | 7h | Priority 7 - Lowest priority |
| 3-2 | Reserved | 0 | Reserved |
| 1 | DAM | | Destination address mode within an array. |
| | | 0 | Increment (INCR) mode. Destination addressing within an array increments. |
| | | 1 | Constant addressing (CONST) mode. Destination addressing within an array wraps around upon reaching FIFO width. |
| 0 | SAM | | Source address mode within an array. |
| | | 0 | Increment (INCR) mode. Source addressing within an array increments. |
| | | 1 | Constant addressing (CONST) mode. Source addressing within an array wraps around upon reaching FIFO width. |

### 4.4.6.2 Source Active Source Address Register (SASRC)

The source active source address register (SASRC) is shown in Figure 4-76 and described in Table 4-79.

**Figure 4-76. Source Active Source Address Register (SASRC)**

| 31 | 16 |
|---|---|
| SADDR | |
| R-0 | |

| 15 | 0 |
|---|---|
| SADDR | |
| R-0 | |

LEGEND: R = Read only; -*n* = value after reset

**Table 4-79. Source Active Source Address Register (SASRC) Field Descriptions**

| Bit | Field | Value | Description |
|---|---|---|---|
| 31-0 | SADDR | 0-FFFF FFFFh | Source address for program register set. EDMA3TC updates value according to source addressing mode (SAM bit in the source active options register, SAOPT) . |

### 4.4.6.3 Source Active Count Register (SACNT)

The source active count register (SACNT) is shown in Figure 4-77 and described in Table 4-80.

**Figure 4-77. Source Active Count Register (SACNT)**

| 31 | 16 |
|---|---|
| BCNT | |
| R-0 | |

| 15 | 0 |
|---|---|
| ACNT | |
| R-0 | |

LEGEND: R = Read only; -*n* = value after reset

**Table 4-80. Source Active Count Register (SACNT) Field Descriptions**

| Bit | Field | Value | Description |
|---|---|---|---|
| 31-16 | BCNT | 0-FFFFh | B dimension count. Number of arrays to be transferred, where each array is ACNT in length. It is decremented after each Read command appropriately. Represents the amount of data remaining to be Read. It should be 0 when transfer request (TR) is complete. |
| 15-0 | ACNT | 0-FFFFh | A dimension count. Number of bytes to be transferred in first dimension. It is decremented after each Read command appropriately. Represents the amount of data remaining to be Read. It should be 0 when transfer request (TR) is complete. |

### 4.4.6.4 Source Active Destination Address Register (SADST)

The source active destination address register (SADST) is shown in Figure 4-78 and described in Table 4-81.

**Figure 4-78. Source Active Destination Address Register (SADST)**

| 31 | 16 |
|---|---|
| DADDR | |
| R-0 | |

| 15 | 0 |
|---|---|
| DADDR | |
| R-0 | |

LEGEND: R = Read only; -*n* = value after reset

**Table 4-81. Source Active Destination Address Register (SADST) Field Descriptions**

| Bit | Field | Value | Description |
|---|---|---|---|
| 31-0 | DADDR | 0 | Always reads as 0 |

### 4.4.6.5 Source Active B-Index Register (SABIDX)

The source active B-index register (SABIDX) is shown in Figure 4-79 and described in Table 4-82.

**Figure 4-79. Source Active B-Index Register (SABIDX)**

| 31 | 16 |
|---|---|
| DSTBIDX | |
| R-0 | |

| 15 | 0 |
|---|---|
| SRCBIDX | |
| R-0 | |

LEGEND: R = Read only; -*n* = value after reset

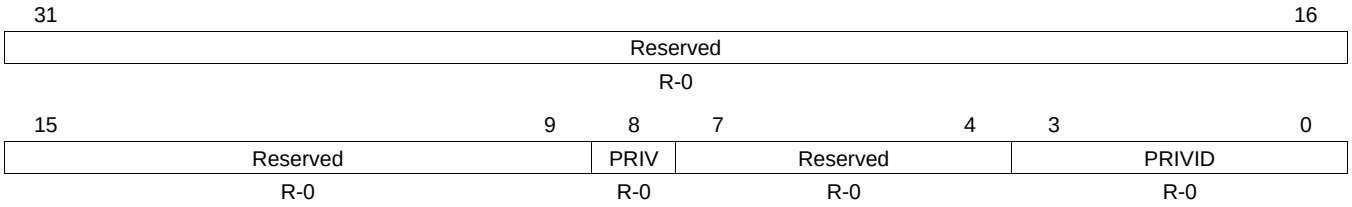
**Table 4-82. Source Active B-Index Register (SABIDX) Field Descriptions**

| Bit | Field | Value | Description |
|---|---|---|---|
| 31-16 | DSTBIDX | 0 | B-Index offset between destination arrays. Represents the offset in bytes between the starting address of each destination. Always reads as 0. |
| 15-0 | SRCBIDX | 0-FFFFh | B-Index offset between source arrays. Represents the offset in bytes between the starting address of each source array. |

### 4.4.6.6 Source Active Memory Protection Proxy Register (SAMPPRXY)

The source active memory protection proxy register (SAMPPRXY) is shown in Figure 4-80 and described in Table 4-83.

**Figure 4-80. Source Active Memory Protection Proxy Register (SAMPPRXY)**




LEGEND: R = Read only; -*n* = value after reset

**Table 4-83. Source Active Memory Protection Proxy Register (SAMPPRXY) Field Descriptions**

| Bit | Field | Value | Description |
|---|---|---|---|
| 31-9 | Reserved | 0 | Reserved |
| 8 | PRIV | | Privilege level. The privilege level used by the host to set up the parameter entry in the channel controller. This field is set up when the associated TR is submitted to the EDMA3TC.<br>The privilege ID is used while issuing Read and write command to the target endpoints so that the target endpoints can perform memory protection checks based on the PRIV of the host that set up the DMA transaction. |
| | | 0 | User-level privilege |
| | | 1 | Supervisor-level privilege |
| 7-4 | Reserved | 0 | Reserved |
| 3-0 | PRIVID | 0-Fh | Privilege ID. This contains the privilege ID of the host that set up the parameter entry in the channel controller. This field is set up when the associated TR is submitted to the EDMA3TC.<br>This PRIVID value is used while issuing Read and write commands to the target endpoints so that the target endpoints can perform memory protection checks based on the PRIVID of the host that set up the DMA transaction. |
| | | 0 | For any other master that sets up the PaRAM entry. |
| | | 1 | If DSP sets up the PaRAM entry. |

### 4.4.6.7 Source Active Count Reload Register (SACNTRLD)

The source active count reload register (SACNTRLD) is shown in Figure 4-81 and described in Table 4-84.

**Figure 4-81. Source Active Count Reload Register (SACNTRLD)**

| 31 | 16 |
|---|---|
| Reserved | |
| R-0 | |

| 15 | 0 |
|---|---|
| ACNTRLD | |
| R-0 | |

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

**Table 4-84. Source Active Count Reload Register (SACNTRLD) Field Descriptions**

| Bit | Field | Value | Description |
|---|---|---|---|
| 31-16 | Reserved | 0 | Reserved |
| 15-0 | ACNTRLD | 0-FFFFh | A-count reload value. Represents the originally programmed value of ACNT. The reload value is used to reinitialize ACNT after each array is serviced. |

### 4.4.6.8 Source Active Source Address B-Reference Register (SASRCBREF)

The source active source address B-reference register (SASRCBREF) is shown in Figure 4-82 and described in Table 4-85.

**Figure 4-82. Source Active Source Address B-Reference Register (SASRCBREF)**

| 31 | 16 |
|---|---|
| SADDRBREF | |
| R-0 | |

| 15 | 0 |
|---|---|
| SADDRBREF | |
| R-0 | |

LEGEND: R = Read only; -*n* = value after reset

**Table 4-85. Source Active Source Address B-Reference Register (SASRCBREF) Field Descriptions**

| Bit | Field | Value | Description |
|---|---|---|---|
| 31-0 | SADDRBREF | 0-FFFF FFFFh | Source address B-reference. Represents the starting address for the array currently being Read. |

### 4.4.6.9 Source Active Destination Address B-Reference Register (SADSTBREF)

The source active destination address B-reference register (SADSTBREF) is shown in Figure 4-83 and described in Table 4-86.

**Figure 4-83. Source Active Destination Address B-Reference Register (SADSTBREF)**

| 31 | 16 |
|---|---|
| DADDRBREF | |
| R-0 | |

| 15 | 0 |
|---|---|
| DADDRBREF | |
| R-0 | |

LEGEND: R = Read only; -*n* = value after reset

**Table 4-86. Source Active Destination Address B-Reference Register (SADSTBREF) Field Descriptions**

| Bit | Field | Value | Description |
|---|---|---|---|
| 31-0 | DADDRBREF | 0 | Always reads as 0 |

### 4.4.6.10 Destination FIFO Set Count Reload Register (DFCNTRLD)

The destination FIFO set count reload register (DFCNTRLD) is shown in Figure 4-84 and described in Table 4-87.

**Figure 4-84. Destination FIFO Set Count Reload Register (DFCNTRLD)**

| 31 | 16 |
|---|---|
| Reserved | |
| R-0 | |

| 15 | 0 |
|---|---|
| ACNTRLD | |
| R-0 | |

LEGEND: R = Read only; -*n* = value after reset

**Table 4-87. Destination FIFO Set Count Reload Register (DFCNTRLD) Field Descriptions**

| Bit | Field | Value | Description |
|---|---|---|---|
| 31-16 | Reserved | 0 | Reserved |
| 15-0 | ACNTRLD | 0-FFFFh | A-count reload value. Represents the originally programmed value of ACNT. The reload value is used to reinitialize ACNT after each array is serviced. |

### 4.4.6.11 Destination FIFO Set Source Address B-Reference Register (DFSRCBREF)

The destination FIFO set source address B-reference register (DFSRCBREF) is shown in Figure 4-85 and described in Table 4-88.

**Figure 4-85. Destination FIFO Set Source Address B-Reference Register (DFSRCBREF)**

| 31 | 16 |
|---|---|
| SADDRBREF | |
| R-0 | |

| 15 | 0 |
|---|---|
| SADDRBREF | |
| R-0 | |

LEGEND: R = Read only; -*n* = value after reset

**Table 4-88. Destination FIFO Set Source Address B-Reference Register (DFSRCBREF) Field Descriptions**

| Bit | Field | Value | Description |
|---|---|---|---|
| 31-0 | SADDRBREF | 0 | Not applicable. Always Read as 0. |

### 4.4.6.12 Destination FIFO Set Destination Address B-Reference (DFDSTBREF)

The destination FIFO set destination address B-reference register (DFDSTBREF) is shown in Figure 4-86 and described in Table 4-89.

**Figure 4-86. Destination FIFO Set Destination Address B-Reference Register (DFDSTBREF)**

| 31 | 16 |
|---|---|
| DADDRBREF | |
| R-0 | |

| 15 | 0 |
|---|---|
| DADDRBREF | |
| R-0 | |

LEGEND: R = Read only; -*n* = value after reset

**Table 4-89. Destination FIFO Set Destination Address B-Reference Register (DFDSTBREF) Field Descriptions**

| Bit | Field | Value | Description |
|---|---|---|---|
| 31-0 | DADDRBREF | 0-FFFF FFFFh | Destination address reference for the destination FIFO register set. Represents the starting address for the array currently being written. |

### 4.4.6.13 Destination FIFO Options Register *n* (DFOPT*n*)

The destination FIFO options register *n* (DFOPT*n*) is shown in Figure 4-87 and described in Table 4-90.

**NOTE:** The value for *n* varies from 0 to DSTREGDEPTH for the given EDMA3TC.

**Figure 4-87. Destination FIFO Options Register *n* (DFOPT*n*)**

| 31 | 23 | 22 | 21 | 20 | 19 18 | 17 16 |
|---|---|---|---|---|---|---|
| Reserved | | TCCHEN | Rsvd | TCINTEN | Reserved | TCC |
| R-0 | | R/W-0 | R-0 | R/W-0 | R-0 | R/W-0 |

| 15 12 | 11 | 10 8 | 7 | 6 4 | 3 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| TCC | Rsvd | FWID | Rsvd | PRI | Reserved | DAM | SAM |
| R/W-0 | R-0 | R/W-0 | R-0 | R/W-0 | R-0 | R/W-0 | R/W-0 |

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

**Table 4-90. Destination FIFO Options Register *n* (DFOPT*n*) Field Descriptions**

| Bit | Field | Value | Description |
|---|---|---|---|
| 31-23 | Reserved | 0 | Reserved |
| 22 | TCCHEN | | Transfer complete chaining enable. |
| | | 0 | Transfer complete chaining is disabled. |
| | | 1 | Transfer complete chaining is enabled. |
| 21 | Reserved | 0 | Reserved |
| 20 | TCINTEN | | Transfer complete interrupt enable. |
| | | 0 | Transfer complete interrupt is disabled. |
| | | 1 | Transfer complete interrupt is enabled. |
| 19-18 | Reserved | 0 | Reserved |
| 17-12 | TCC | 0-3Fh | Transfer complete code. This 6-bit code is used to set the relevant bit in CER or IPR of the EDMA3CC. |
| 11 | Reserved | 0 | Reserved |
| 10-8 | FWID | 0-7h | FIFO width. Applies if either SAM or DAM is set to constant addressing mode. |
| | | 0 | FIFO width is 8 bits. |
| | | 1h | FIFO width is 16 bits. |
| | | 2h | FIFO width is 32 bits. |
| | | 3h | FIFO width is 64 bits. |
| | | 4h | FIFO width is 128 bits. |
| | | 5h | FIFO width is 256 bits. |
| | | 6h-7h | Reserved |
| 7 | Reserved | 0 | Reserved |
| 6-4 | PRI | 0-7h | Transfer priority. |
| | | 0 | Priority 0 - Highest priority |
| | | 1h-6h | Priority 1 to priority 6 |
| | | 7h | Priority 7 - Lowest priority |
| 3-2 | Reserved | 0 | Reserved |
| 1 | DAM | | Destination address mode within an array. |
| | | 0 | Increment (INCR) mode. Destination addressing within an array increments. |
| | | 1 | Constant addressing (CONST) mode. Destination addressing within an array wraps around upon reaching FIFO width. |
| 0 | SAM | | Source address mode within an array. |
| | | 0 | Increment (INCR) mode. Source addressing within an array increments. |
| | | 1 | Constant addressing (CONST) mode. Source addressing within an array wraps around upon reaching FIFO width. |

#### 4.4.6.14 Destination FIFO Source Address Register *n* (DFSRC*n*)

The destination FIFO source address register *n* (DFSRC*n*) is shown in Figure 4-88 and described in Table 4-91.

**NOTE:** The value for *n* varies from 0 to DSTREGDEPTH for the given EDMA3TC.

**Figure 4-88. Destination FIFO Source Address Register *n* (DFSRC*n*)**

| 31 | 16 |
|---|---|
| SADDR | |
| R-0 | |

| 15 | 0 |
|---|---|
| SADDR | |
| R-0 | |

LEGEND: R = Read only; -*n* = value after reset

**Table 4-91. Destination FIFO Source Address Register *n* (DFSRC*n*) Field Descriptions**

| Bit | Field | Value | Description |
|---|---|---|---|
| 31-0 | SADDR | 0 | Always Read as 0. |

### 4.4.6.15 Destination FIFO Count Register *n* (DFCNT*n*)

The destination FIFO count register *n* (DFCNT*n*) is shown in Figure 4-89 and described in Table 4-92.

**NOTE:** The value for *n* varies from 0 to DSTREGDEPTH for the given EDMA3TC.

**Figure 4-89. Destination FIFO Count Register *n* (DFCNT*n*)**

| 31 | 16 |
|---|---|
| BCNT | |
| R-0 | |

| 15 | 0 |
|---|---|
| ACNT | |
| R-0 | |

LEGEND: R = Read only; -*n* = value after reset

**Table 4-92. Destination FIFO Count Register *n* (DFCNT*n*) Field Descriptions**

| Bit | Field | Value | Description |
|---|---|---|---|
| 31-16 | BCNT | 0-FFFFh | B-dimension count. Number of arrays to be transferred, where each array is ACNT in length. Count/count remaining for destination register set. Represents the amount of data remaining to be written. |
| 15-0 | ACNT | 0-FFFFh | A-dimension count. Number of bytes to be transferred in first dimension count/count remaining for destination register set. Represents the amount of data remaining to be written. |

### 4.4.6.16 Destination FIFO Destination Address Register *n* (DFDST*n*)

The destination FIFO destination address register *n* (DFDST*n*) is shown in Figure 4-90 and described in Table 4-93.

**NOTE:** The value for *n* varies from 0 to DSTREGDEPTH for the given EDMA3TC.

**Figure 4-90. Destination FIFO Destination Address Register *n* (DFDST*n*)**

| 31 | 16 |
|---|---|
| DADDR | |
| R-0 | |

| 15 | 0 |
|---|---|
| DADDR | |
| R-0 | |

LEGEND: R = Read only; -*n* = value after reset

**Table 4-93. Destination FIFO Destination Address Register *n* (DFDST*n*) Field Descriptions**

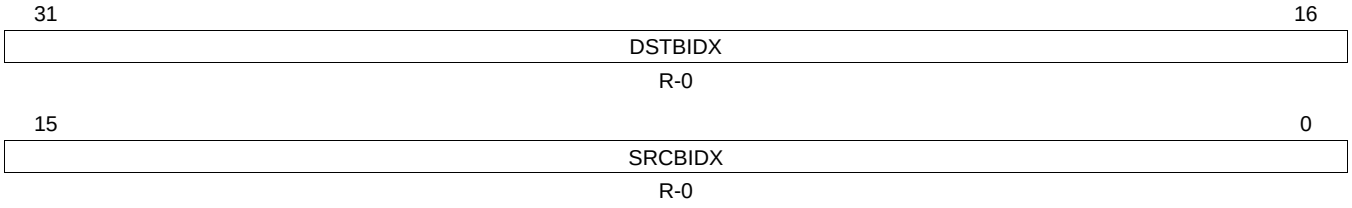
| Bit | Field | Value | Description |
|---|---|---|---|
| 31-0 | DADDR | 0 | Destination address for the destination FIFO register set. When a transfer request (TR) is complete, the final value should be the address of the last write command issued. |

### 4.4.6.17 Destination FIFO B-Index Register *n* (DFBIDX*n*)

The destination FIFO B-index register *n* (DFBIDX*n*) is shown in Figure 4-91 and described in Table 4-94.

**NOTE:** The value for *n* varies from 0 to DSTREGDEPTH for the given EDMA3TC.

**Figure 4-91. Destination FIFO B-Index Register *n* (DFBIDX*n*)**




LEGEND: R = Read only; -*n* = value after reset

**Table 4-94. Destination FIFO B-Index Register *n* (DFBIDX*n*) Field Descriptions**

| Bit | Field | Value | Description |
|---|---|---|---|
| 31-16 | DSTBIDX | 0-FFFFh | B-Index offset between destination arrays. Represents the offset in bytes between the starting address of each destination. |
| 15-0 | SRCBIDX | 0 | B-Index offset between source arrays. Represents the offset in bytes between the starting address of each source array. Always Read as 0. |

### 4.4.6.18 Destination FIFO Memory Protection Proxy Register *n* (DFMPPRXY*n*)

The destination FIFO memory protection proxy register *n* (DFMPPRXY*n*) is shown in Figure 4-92 and described in Table 4-95.

**Figure 4-92. Destination FIFO Memory Protection Proxy Register *n* (DFMPPRXY*n*)**

| 31 | 16 |
|---|---|
| Reserved | |
| R-0 | |

| 15 | 9 | 8 | 7 | 4 | 3 | 0 |
|---|---|---|---|---|---|---|
| Reserved | | PRIV | Reserved | | PRIVID | |
| R-0 | | R-0 | R-0 | | R-0 | |

LEGEND: R = Read only; -*n* = value after reset

**Table 4-95. Destination FIFO Memory Protection Proxy Register *n* (DFMPPRXY*n*) Field Descriptions**

| Bit | Field | Value | Description |
|---|---|---|---|
| 31-9 | Reserved | 0 | Reserved |
| 8 | PRIV | | Privilege level. This contains the privilege level used by the EDMA programmer to set up the parameter entry in the channel controller. This field is set up when the associated TR is submitted to the EDMA3TC.<br>The privilege ID is used while issuing Read and write command to the target endpoints so that the target endpoints can perform memory protection checks based on the PRIV of the host that set up the DMA transaction. |
| | | 0 | User-level privilege |
| | | 1 | Supervisor-level privilege |
| 7-4 | Reserved | 0 | Reserved |
| 3-0 | PRIVID | 0-Fh | Privilege ID. This contains the Privilege ID of the EDMA programmer that set up the parameter entry in the channel controller. This field is set up when the associated TR is submitted to the EDMA3TC.<br>This PRIVID value is used while issuing Read and write commands to the target endpoints so that the target endpoints can perform memory protection checks based on the PRIVID of the host that set up the DMA transaction. |
| | | 0 | For any other master that sets up the PaRAM entry |
| | | 1 | If DSP sets up the PaRAM entry |

# Tips

## A.1 Debug Checklist

This section lists some tips to keep in mind while debugging applications using the EDMA3. Table A-1 provides some common issues and their probable causes and resolutions.

**Table A-1. Debug List**

| Issue | Description/Solution |
|---|---|
| The transfer associated with the channel does not happen. The channel does not get serviced. | The EDMA3 channel controller (EDMA3CC) may not service a transfer request, even though the associated PaRAM set is programmed appropriately. Check for the following:<br>1) Verify that events are enabled, that is, if an external/peripheral event is latched in the event registers (ER/ERH), make sure that the event is enabled in the event enable registers (EER/EERH). Similarly for QDMA channels, make sure that QDMA events are appropriately enabled in the QDMA event enable register (QEER).<br>2) Verify that the DMA or QDMA secondary event register (SER/SERH/QSERH) bits corresponding to the particular event or channel are not set. |
| The secondary event register bits are set, not allowing additional transfers to occur on a channel. | It is possible that a trigger event was received when the parameter set associated with the channel/event was a NULL set for a previous transfer on the channel. This is typical in two cases:<br>1) QDMA channels: Typically if the parameter set is nonstatic and expected to be terminated by a NULL set (OPT.STATIC = 0, LINK = FFFFh), the parameter set is updated with a NULL set after submission of the last TR. Because QDMA channels are autotriggered, this update caused the generation of an event. An event generated for a NULL set causes an error condition and results in setting the bits corresponding to the QDMA channel in QEMR and QSER. This will disable further prioritization of the channel.<br>2) DMA channels used in a continuous mode: The peripheral may be set up to continuously generate infinite events (for instance, in case of the McBSP, every time the data shifts out from DXR, it generates an XEVT). The parameter set may be programmed to expect only a finite number of events and to be terminated by a NULL link. After the expected number of events, the parameter set is reloaded with a NULL parameter set. Because the peripheral will generate additional events, an error condition is set in SER.E*n* and EMR.E*n*, preventing further event prioritization.<br>You must ensure that the number of events received is limited to the expected number of events for which the parameter set is programmed, or you must ensure that bits corresponding to a particular channel or event are not set in the secondary event registers (SER/SERH/QSER) and the event missed registers (EMR/EMRH/QEMR) before trying to perform subsequent transfers for the event/channel. |
| Completion interrupts are not asserted, or no further interrupts are received after the first completion interrupt. | You must ensure the following:<br>1) The interrupt generation is enabled in the OPT of the associated PaRAM set (TCINTEN = 1 and/or ITCINTEN = 1).<br>2) The interrupts are enabled in the EDMA3 channel controller (EDMA3CC), via the interrupt enable registers (IER/IERH).<br>3) The corresponding interrupts are enabled in the device interrupt controller.<br>4) The set interrupts are cleared in the interrupt pending registers (IPR/IPRH) before exiting the transfer completion interrupt service routine (ISR). See Section 2.9.1.2 for details on writing EDMA3 ISRs.<br>5) If working with shadow region interrupts, make sure that the DMA region access enable registers (DRAE/DRAEH) are set up properly, because DRAE/DRAEH act as secondary enables for shadow region completion interrupts, along with IER/IERH.<br>If working with shadow region interrupts, make sure that the bits corresponding to the transfer completion code (TCC) value are also enabled in DRAE/DRAEH. For instance, if the PaRAM set associated with channel 0 returns a completion code of 63 (OPT.TCC = 63), make sure that DRAEH.E63 is also set for a shadow region completion interrupt because the interrupt pending register bit set will be IPRH.I63 (not IPR.I0). |

## A.2 Miscellaneous Programming/Debug Tips

1. For several registers, the setting and clearing of bits needs to be done via separate dedicated registers. For example, the event register (ER/ERH) bits can only be cleared by writing a 1 to the corresponding bits in the event clear registers (ECR/ECRH). Similarly, the event enable register (EER/EERH) bits can only be set with writes of 1 to the corresponding bits in the event enable set registers (EESR/EESRH) and can only be cleared with writes of 1 to the corresponding bits in the event enable clear registers (EECR/EECRH).
2. Writes to the shadow region memory maps are governed by region access enable registers (DRAE/DRAEH/QRAE). If the appropriate channels are not enabled in these registers, read/write access to the shadow region memory map is not enabled.
3. When working with shadow region completion interrupts, ensure that the DMA region access enable registers (DRAE/DRAEH) for every region are set in a mutually exclusive way (unless it is a requirement for an application). If there is an overlap in the allocated channels and transfer completion codes (setting of interrupt pending register bits) in the region resource allocation, it results in multiple shadow region completion interrupts. For example, if DRAE0.E0 and DRAE1.E0 are both set, then on completion of a transfer that returns a TCC = 0, they will generate both shadow region 0 and 1 completion interrupts.
4. While programming a non-dummy parameter set, ensure the CCNT is not left to zero.
5. Enable the EDMA3CC error interrupt in the device controller and attach an interrupt service routine (ISR) to ensure that error conditions are not missed in an application and are appropriately addressed with the ISR.
6. Depending on the application, you may want to break large transfers into smaller transfers and use self-chaining to prevent starvation of other events in an event queue.
7. In applications where a large transfer is broken into sets of small transfers using chaining or other methods, you might choose to use the early chaining option to reduce the time between the sets of transfers and increase the throughput. However, keep in mind that with early completion, all data might have not been received at the end point when completion is reported because the EDMA3CC internally signals completion when the TR is submitted to the EDMA3TC, potentially before any data has been transferred.
8. The event queue entries can be observed to determine the last few events if there is a system failure (provided the entries were not bypassed).
9. In order to put the EDMA3CC and EDMA3TC in power-down modes, you should ensure that there is no activity with the EDMA3CC and EDMA3TC. The EDMA3CC status register (CCSTAT) and the EDMA3TC channel status register (TCSTAT) should be used.

# Setting Up a Transfer

The following list provides a quick guide for the typical steps involved in setting up a transfer.

1. Initiating a DMA/QDMA channel:
   (a) Determine the type of channel (QDMA or DMA) to be used.
   (b) If using a QDMA channel, program the QDMA channel *n* mapping register (QCHMAP*n*) with the parameter set number to which the channel maps and the trigger word.
   (c) If the channel is being used in the context of a shadow region, ensure the DMA region access enable registers (DRAE/DRAEH) for the region is properly set up to allow read/write accesses to bits in the event registers and interrupt registers in the shadow region memory map. The subsequent steps in this process should be done using the respective shadow region registers. (Shadow region descriptions and usage are provided in Section 2.7.1.)
   (d) Determine the type of triggering used.
      (i) If external events are used for triggering (DMA channels), enable the respective event in EER/EERH by writing into EESR/EESRH.
      (ii) If a QDMA channel is used, enable the channel in QEER by writing into QEESR.
   (e) Queue setup.
      (i) If a QDMA channel is used, set up QDMAQNUM to map the channel to the respective event queue.
      (ii) If a DMA channel is used, set up DMAQNUM to map the event to the respective event queue.
2. Parameter set setup: Program the PaRAM set number associated with the channel. Note that if it is a QDMA channel, the PaRAM entry that is configured as trigger word is written last. Alternatively, enable the QDMA channel just before the write to the trigger word.

   See Chapter 3 for parameter set field setups for different types of transfers. See the sections on chaining (Section 2.8) and interrupt completion (Section 2.9) on how to set up final/intermediate completion chaining and/or interrupts.
3. Interrupt setup:
   (a) If working in the context of a shadow region, ensure the relevant bits in DRAE/DRAEH are set.
   (b) Enable the interrupt in IER/IERH by writing into IESR/IESRH.
   (c) Ensure that the EDMA3CC completion interrupt is enabled properly in the device interrupt controller.
   (d) Set up the interrupt controller properly to receive the expected EDMA3 interrupt.
4. Initiate transfer (this step is highly dependent on the event trigger source):
   (a) If the source is an external event coming from a peripheral, the peripheral will be enabled to start generating relevant EDMA3 events that can be latched to the ER transfer.
   (b) For QDMA events, writes to the trigger word will initiate the transfer.
   (c) Manually-triggered transfers will be initiated by writes to the event set registers (ESR/ESRH).
   (d) Chained-trigger events initiate when a previous transfer returns a transfer completion code equal to the chained channel number.

5. Wait for completion:
   (a) If the interrupts are enabled as mentioned in step 3, then the EDMA3CC generates a completion interrupt to the CPU whenever transfer completion results in setting the corresponding bits in the interrupt pending register (IPR/IPRH). The set bits must be cleared in IPR\IPRH by writing to the corresponding bit in ICR\ICRH.
   (b) If polling for completion (interrupts not enabled in the device controller), then the application code can wait on the expected bits to be set in IPR\IPRH. Again, the set bits in IPR\IPRH must be manually cleared by writing to ICR\ICRH before the next set of transfers is performed for the same transfer completion code values.

# Revision History

Table C-1 lists the changes made since the previous version of this document.

**Table C-1. Document Revision History**

| Reference | Additions/Modifications/Deletions |
|---|---|
| Section 2.11.4 | Changed paragraph. |
| Section 3.4.3 | Changed second paragraph. |

## IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, modifications, enhancements, improvements, and other changes to its products and services at any time and to discontinue any product or service without notice. Customers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All products are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

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

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

TI does not warrant or represent that any license, either express or implied, is granted under any TI patent right, copyright, mask work right, or other TI intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information published by TI regarding third-party products or services does not constitute a license from TI 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 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. Reproduction of this information with alteration is an unfair and deceptive business practice. TI is not responsible or liable for such altered documentation. Information of third parties may be subject to additional restrictions.

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

TI products are not authorized for use in safety-critical applications (such as life support) where a failure of the TI product would reasonably be expected to cause severe personal injury or death, unless officers of the parties have executed an agreement specifically governing such use. Buyers represent that they have all necessary expertise in the safety and regulatory ramifications of their applications, and acknowledge and agree that they are solely responsible for all legal, regulatory and safety-related requirements concerning their products and any use of TI products in such safety-critical applications, notwithstanding any applications-related information or support that may be provided by TI. Further, Buyers must fully indemnify TI and its representatives against any damages arising out of the use of TI products in such safety-critical applications.

TI products are neither designed nor intended for use in military/aerospace applications or environments unless the TI products are specifically designated by TI as military-grade or "enhanced plastic." Only products designated by TI as military-grade meet military specifications. Buyers acknowledge and agree that any such use of TI products which TI has not designated as military-grade is solely at the Buyer's risk, and that they are solely responsible for compliance with all legal and regulatory requirements in connection with such use.

TI products are neither designed nor intended for use in automotive applications or environments unless the specific TI products are designated by TI as compliant with ISO/TS 16949 requirements. Buyers acknowledge and agree that, if they use any non-designated products in automotive applications, TI will not be responsible for any failure to meet such requirements.

Following are URLs where you can obtain information on other Texas Instruments products and application solutions:

| **Products** | | **Applications** | |
|---|---|---|---|
| Amplifiers | amplifier.ti.com | Audio | www.ti.com/audio |
| Data Converters | dataconverter.ti.com | Automotive | www.ti.com/automotive |
| DLP® Products | www.dlp.com | Communications and Telecom | www.ti.com/communications |
| DSP | dsp.ti.com | Computers and Peripherals | www.ti.com/computers |
| Clocks and Timers | www.ti.com/clocks | Consumer Electronics | www.ti.com/consumer-apps |
| Interface | interface.ti.com | Energy | www.ti.com/energy |
| Logic | logic.ti.com | Industrial | www.ti.com/industrial |
| Power Mgmt | power.ti.com | Medical | www.ti.com/medical |
| Microcontrollers | microcontroller.ti.com | Security | www.ti.com/security |
| RFID | www.ti-rfid.com | Space, Avionics & Defense | www.ti.com/space-avionics-defense |
| RF/IF and ZigBee® Solutions | www.ti.com/lprf | Video and Imaging | www.ti.com/video |
| | | Wireless | www.ti.com/wireless-apps |

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265