Chapter 6
Enhanced Direct Memory Access (eDMA)

6.1 Chip-specific eDMA information

Table 6-1. Reference links to related information

Topic	Related module(s)	Reference
System memory map	-	System Memory Map
Clocking	CCM	Clock Management Clock Control Module (CCM)
Power management	PMU	Power Management Power Management Unit
Signal multiplexing	IOMUX	External Signals and Pin Multiplexing IOMUX
Interrupts, DMA Events and XBAR Assignments	-	Interrupts, DMA Events and XBAR Assignments

6.2 Overview

The enhanced direct memory access (eDMA) controller is a second-generation module capable of performing complex data transfers with minimal intervention from a host processor. The hardware microarchitecture includes:

- A DMA engine that performs:
  - Source address and destination address calculations
  - Data-movement operations
- Local memory containing transfer control descriptors for each of the 32 channels

6.2.1 Block diagram

The following figure illustrates the components of the eDMA system, including the eDMA module ("engine").

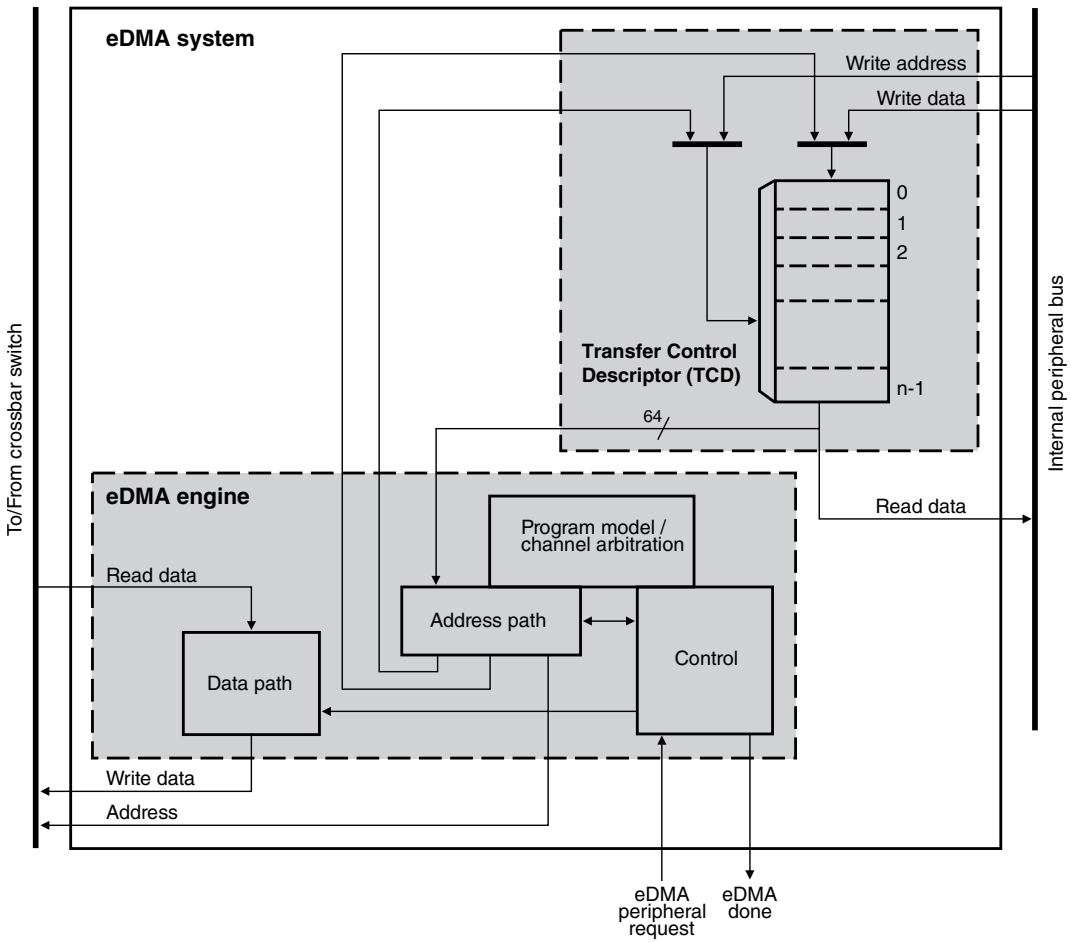


Figure 6-1. Block diagram

6.2.2 Block parts

The eDMA module comprises two major modules: the eDMA engine and the transfer-control descriptor local memory.

Table 6-2 describes the eDMA engine submodules.

Table 6-2. eDMA engine submodules

Submodule	Function
Address path	The address path block: • Provides registered versions of two channel Transfer Control Descriptors (TCDs)—channel x (normal start) and channel y (preemption start) • Manages all master bus-address calculations

Table continues on the next page...

Table 6-2. eDMA engine submodules (continued)

Submodule	Function
	All channels provide the same functionality. This structure enables preemption of data transfers associated with an active channel (after completion of a read/write sequence) if the eDMA engine asserts a higher priority channel activation. After eDMA activates a channel, it runs until the minor loop completes, unless preempted by a higher priority channel. This provides a mechanism (enabled by DCHPRI*n*[ECP]) in which the eDMA engine can preempt a large data move operation to minimize the time another channel stalls. When the eDMA engine selects a channel to execute, it reads the contents of the channel TCD from local memory and loads it into one of the following: • The address path channel x registers (normal start) • The address path channel y registers (preemption start) After the minor loop execution completes, the address path hardware writes the new values for the TCD*n*_{SADDR, DADDR, CITER} back to local memory. If the major iteration count completes, the eDMA engine performs additional processing, including: • Final address pointer updates • Reloading the TCD*n*_CITER field • A possible fetch of the next TCD*n* from memory as part of a scatter/gather operation.
Data path	The data path block implements the bus master read/write data path. It includes a data buffer and the necessary multiplex logic to support any required data alignment. The internal read data bus is the primary input, and the internal write data bus is the primary output. The address and data path modules directly support the two-stage pipelined internal bus. The address path module represents the first stage of the bus pipeline (address phase). The data path module implements the second stage of the pipeline (data phase).
Programming model/ channel arbitration	This block implements: • The first section of the eDMA programming model • Channel arbitration logic The programming model registers connect to the chip's internal peripheral bus. The eDMA peripheral request inputs and interrupt request outputs also connect to this block (via control logic).
Control	The control block provides all control functions for the eDMA engine. For data transfers in which the source size (SSIZE) and destination size (DSIZE) are equal, the eDMA engine performs a series of source read/destination write operations until it has transferred the number of bytes specified in the minor loop byte count (NBYTES). For TCDs in which the source and destination sizes are not equal, the eDMA engine executes multiple accesses of the smaller size data for each reference of the larger size. For example, if the source size (SSIZE) references 16-bit data and the destination size (DSIZE) is 32-bit data, eDMA performs two reads, then one 32-bit write.

Table 6-3 explains the partitioning of the TCD local memory.

Table 6-3. Transfer control descriptor memory

Submodule	Description
Memory controller	The Memory controller logic implements the required dual-ported controller, managing accesses from the eDMA engine as well as references from the internal peripheral bus. If simultaneous accesses occur, the eDMA engine receives priority and the peripheral transaction stalls.
Memory array	The Memory array provides TCD storage for the transfer profile for each channel.

6.2.3 Features

The eDMA module is a highly programmable data-transfer engine optimized to minimize any required intervention from the host processor. Use it for applications where you statically know the size of the data to be transferred and do not define the size within the transferred data itself. The eDMA module features:

- All data movement via dual-address transfers: read from source, write to destination
  - Programmable source and destination addresses and transfer size
  - Support for enhanced addressing modes
- 32-channel implementation performs complex data transfers with minimal intervention from a host processor
  - Internal data buffer, used as temporary storage to support 16- and 32-byte transfers
  - Connections to the crossbar switch (AXBS) for bus mastering the data movement
- TCD supports two-deep, nested transfer operations
  - 32-byte TCD stored in local memory for each channel
  - An inner data transfer loop defined by a minor byte transfer count
  - An outer data transfer loop defined by a major iteration count
- Channel activation via one of three methods:
  - Explicit software initiation
  - Initiation via a channel-to-channel linking mechanism for continuous transfers
  - Peripheral-paced hardware requests, one per channel
- Fixed-priority and round-robin channel arbitration
- Channel completion notification via programmable interrupt requests
  - One interrupt per channel. eDMA engine can generate an interrupt when major iteration count completes
  - Programmable error terminations per channel and logically summed together to form one error interrupt to the interrupt controller

- Programmable support for scatter/gather DMA processing
- Support for complex data structures

NOTE

In the discussion of this module, *n* is the channel number.

6.3 Functional description

The operation of the eDMA is described in the following subsections.

6.3.1 Modes of operation

eDMA operates in the following modes:

Table 6-4. Modes of operation

Mode	Description
Normal	In Normal mode, eDMA transfers data from a source to a destination. The source and destination can be a memory block or an I/O block capable of operation with eDMA. A *service request* initiates a transfer of a specific number of bytes (NBYTES) as specified in the TCD. • The *minor loop* is the sequence of read and write operations that transfers the NBYTES of data for a service request. • Each service request executes one iteration of the major loop, transferring NBYTES of data.
Debug	DMA operation is configurable in Debug mode via Control (CR) • If CR[EDBG] = 0, eDMA continues to operate normally when the chip is in debug mode. • If CR[EDBG] = 1, eDMA stops transferring data when the chip enters debug mode. If a channel is active when eDMA enters Debug mode, eDMA continues operation until the channel retires.
Wait	Before entering Wait mode, eDMA attempts to complete any transfer that is in progress. After the transfer completes, the chip enters Wait mode.

6.3.2 eDMA basic data flow

The basic flow of a data transfer can be partitioned into three segments.

As shown in the following diagram, the first segment involves the channel activation:

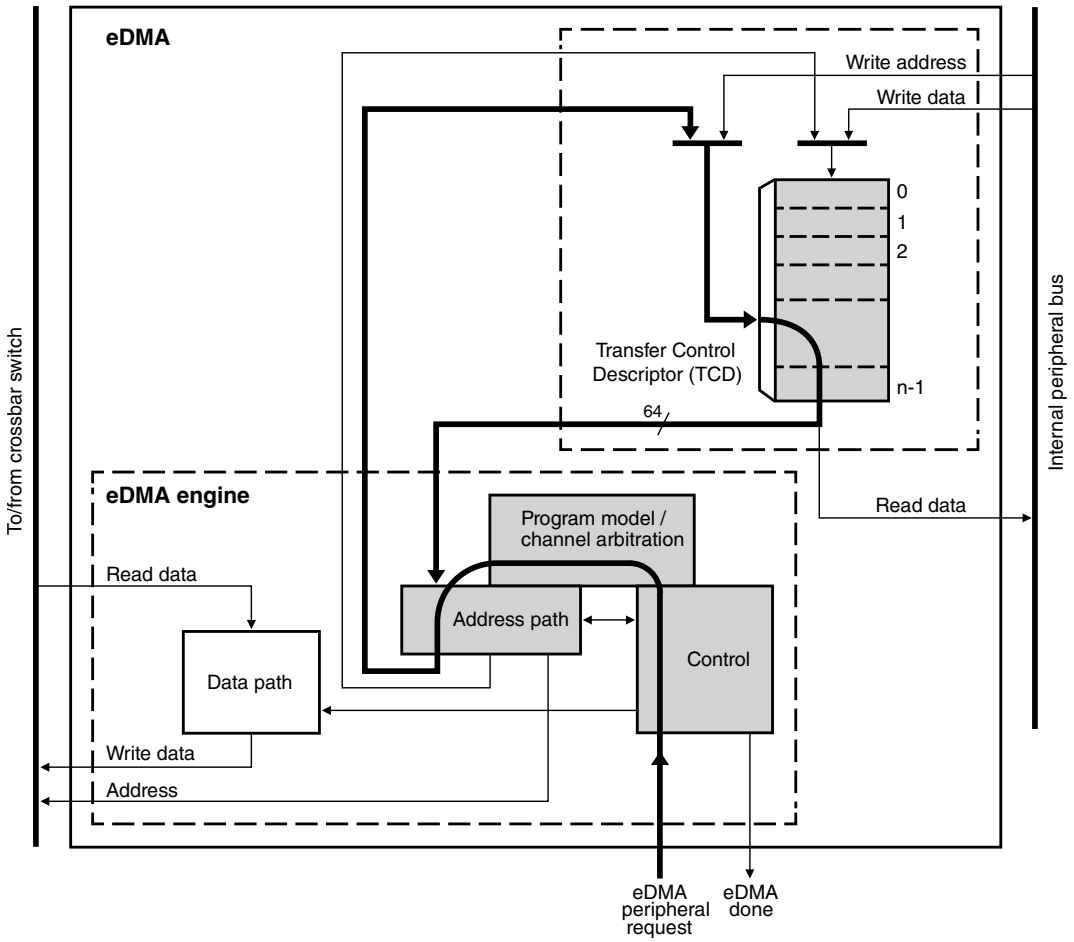


Figure 6-2. eDMA operation, part 1

This example uses the assertion of the eDMA peripheral request signal to request service for channel *n*. Channel activation via software and the TCD*n*_CSR[START] bit follows the same basic flow as peripheral requests. The eDMA request input signal is registered internally and then routed through the eDMA engine: first through the control module, then into the program model and channel arbitration. In the next cycle, the channel arbitration executes, using either the fixed-priority or round-robin algorithm. After arbitration is complete, the activated channel number is sent through the address path and converted into the required address to access the local memory for TCD*n*. Next, the TCD memory is accessed and the required descriptor read from the local memory and loaded into the eDMA engine's internal register file. The TCD memory is 64 bits wide to minimize the time needed to fetch the activated channel descriptor and load it into the internal register file.

The following diagram illustrates the second part of the basic data flow:

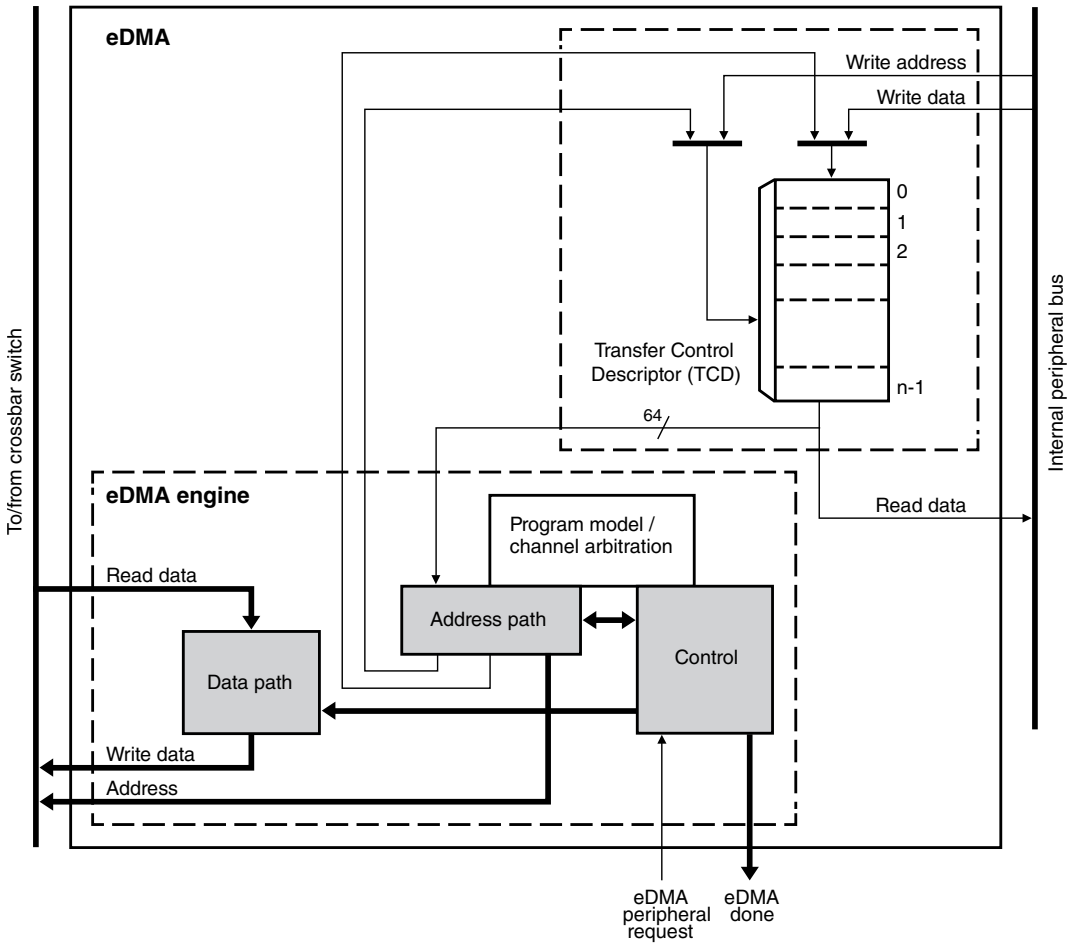


Figure 6-3. eDMA operation, part 2

The modules associated with the data transfer (address path, data path, and control) execute sequentially through the required source reads and destination writes to perform the data movement. The source reads are initiated and the fetched data is temporarily stored in the data path block until it is gated onto the internal bus during the destination write. This source read/destination write processing continues until the minor byte count has transferred.

After the minor byte count has moved, the final phase of the basic data flow is performed. In this segment, the address path logic performs the required updates to certain fields in the appropriate TCD, for example, SADDR, DADDR, CITER. If the major iteration count is exhausted, additional operations are performed. These include the final address adjustments and reloading of the BITER field into the CITER. Assertion of an optional interrupt request also occurs at this time, as does a possible fetch of a new TCD from memory using the scatter/gather address pointer included in the descriptor (if scatter/gather is enabled). The updates to the TCD memory and the assertion of an interrupt request are shown in the following diagram.

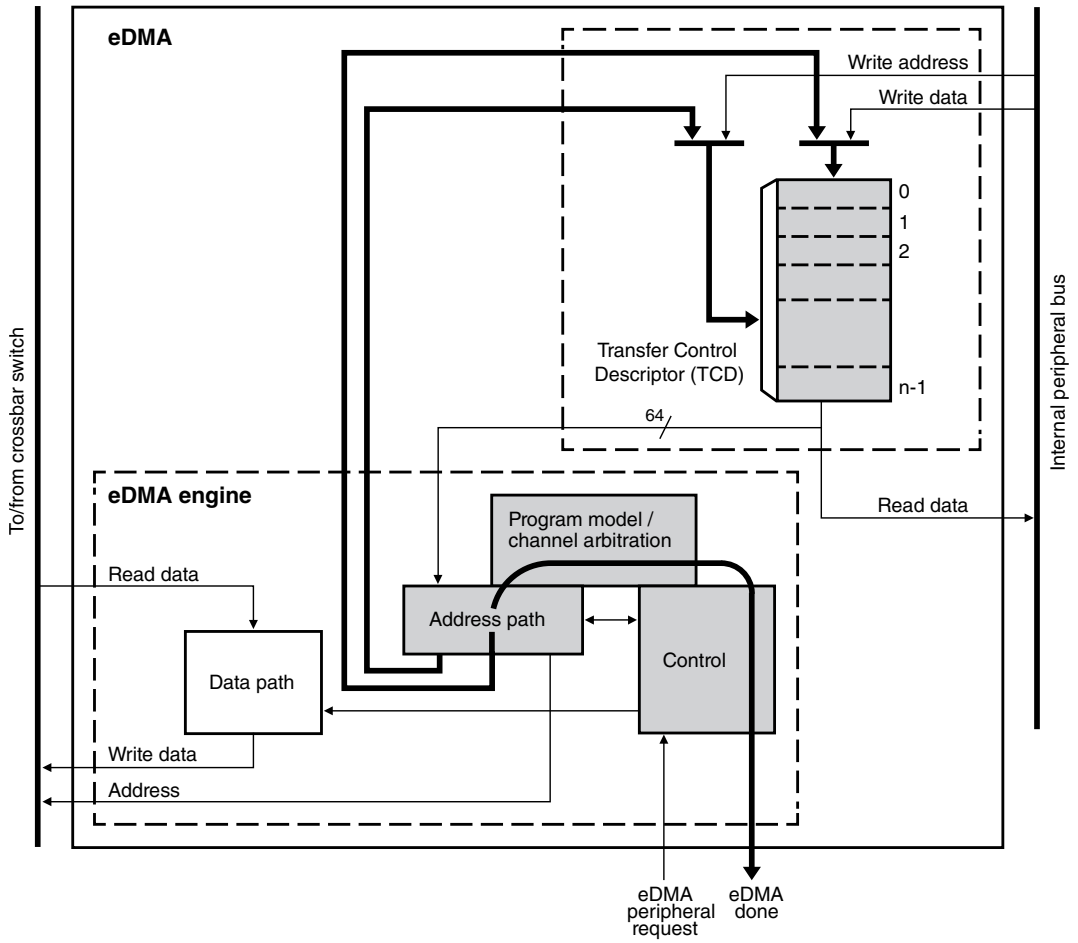


Figure 6-4. eDMA operation, part 3

6.3.3 Fault reporting and handling

Channel errors are reported in the Error Status register (DMAx_ES) and can be caused by:

- A configuration error, which is an illegal setting in the transfer-control descriptor or an illegal priority register setting in Fixed-Arbitration mode, or
- An error termination to a bus master read or write cycle

A configuration error is reported when the starting source or destination address, source or destination offsets, minor loop byte count, or the transfer size represent an inconsistent state. Each of these possible causes is detailed below:

- The addresses and offsets must be aligned on 0-modulo transfer size boundaries.
- The minor loop byte count must be a multiple of the source and destination transfer sizes.

- All source reads and destination writes must be configured to the natural boundary of the programmed transfer size respectively.
- In fixed arbitration mode, a configuration error is caused by any two channel priorities being equal. All channel priority levels must be unique when fixed arbitration mode is enabled.

NOTE

When two channels have the same priority, a channel priority error exists and is reported in the Error Status register. However, the channel number is not reported in the Error Status register. When all of the channel priorities within a group are not unique, the channel number selected by arbitration is undetermined.

To aid in Channel Priority Error (CPE) debug, set the Halt On Error bit in the DMA's Control register. If all channel priorities within a group are not unique, the DMA is halted after the CPE error is recorded. The DMA remains halted and does not process any channel service requests. After all of the channel priorities are set to unique numbers, the DMA may be enabled again by clearing the HALT bit.

- If a scatter/gather operation is enabled upon channel completion, a configuration error is reported if the scatter/gather address (DLAST_SGA) is not aligned on a 32-byte boundary.
- If minor loop channel linking is enabled upon channel completion, a configuration error is reported when the link is attempted if the TCDn_CITER[ELINK] bit does not equal the TCDn_BITER[ELINK] bit.

If enabled, all configuration error conditions, except the scatter/gather and minor-loop link errors, report as the channel activates and asserts an error interrupt request. A scatter/gather configuration error is reported when the scatter/gather operation begins at major loop completion, when properly enabled. A minor loop channel link configuration error is reported when the link operation is serviced at minor loop completion.

If a system bus read or write is terminated with an error, the data transfer is stopped and the appropriate bus error flag set. In this case, the state of the channel's transfer control descriptor is updated by the eDMA engine with the current source address, destination address, and current iteration count at the point of the fault. When a system bus error occurs, the channel terminates after the next transfer. Due to pipeline effect, the next transfer is already in progress when the bus error is received by the eDMA. If a bus error occurs on the last read prior to beginning the write sequence, the write executes using the

data captured during the bus error. If a bus error occurs on the last write prior to switching to the next read sequence, the read sequence executes before the channel terminates due to the destination bus error.

A transfer may be canceled by software with the CR[CX] bit. When a cancel transfer request is recognized, the DMA engine stops processing the channel. The current read-write sequence is allowed to finish. If the cancel occurs on the last read-write sequence of a major or minor loop, the cancel request is discarded and the channel retires normally.

The error cancel transfer is the same as a cancel transfer except the Error Status register (DMAx_ES) is updated with the canceled channel number and ECX is set. The TCD of a canceled channel contains the source and destination addresses of the last transfer saved in the TCD. If the channel needs to be restarted, you must re-initialize the TCD because the aforementioned fields no longer represent the original parameters. When a transfer is canceled by the error cancel transfer mechanism, the channel number is loaded into DMA_ES[ERRCHN] and ECX and VLD are set. In addition, an error interrupt may be generated if enabled.

NOTE

The cancel transfer request enables you to stop a large data transfer when the full data transfer is no longer needed. The cancel transfer bit does not abort the channel. It simply stops the transferring of data and then retires the channel through its normal shutdown sequence. The application software must manage the context of the cancel. If an interrupt is desired (or not), then the interrupt should be enabled (or disabled) before the cancel request. The application software must clean up the transfer control descriptor because the full transfer did not occur.

The occurrence of any error causes the eDMA engine to stop normal processing of the active channel immediately (it goes to its error processing states and the transaction to the system bus still has pipeline effect), and the appropriate channel bit in the eDMA error register is asserted. At the same time, the details of the error condition are loaded into the Error Status register (DMAx_ES). The major loop complete indicators, setting the transfer control descriptor DONE flag and the possible assertion of an interrupt request, are not affected when an error is detected. After the error status has been updated, the eDMA engine continues operating by servicing the next appropriate channel. A channel that experiences an error condition is not automatically disabled. If a channel is terminated by an error and then issues another service request before the error is fixed, that channel executes and terminates with the same error condition.

6.3.4 Channel preemption

Channel preemption is enabled on a per-channel basis by setting DCHPRIn[ECP]. Channel preemption enables the executing channel's data transfers to temporarily be suspended in favor of starting a higher priority channel. After the preempting channel has completed its minor loop data transfers, the preempted channel is restored and resumes execution. After the restored channel completes one read/write sequence, it is again eligible for preemption. If any higher priority channel is requesting service, the restored channel is suspended and the higher priority channel is serviced. Nested preemption, that is, attempting to preempt a preempting channel, is not supported. After a preempting channel begins execution, it cannot be preempted. Preemption is available only when fixed arbitration is selected.

A channel's ability to preempt another channel can be disabled by setting DCHPRIn[DPA]. When a channel's preemption ability is disabled, that channel cannot suspend a lower priority channel's data transfer, regardless of the lower priority channel's ECP setting. This enables a pool of low priority, large data-moving channels to be defined. These low priority channels can be configured to not preempt each other, thus preventing a low priority channel from consuming the preempt slot normally available to a true, high priority channel.

6.3.5 Clocks

The following table describes the clock sources for eDMA. Please see Clock Controller Module (CCM) for clock setting, configuration and gating information.

Table 6-5. eDMA clocks

Clock name	Description
edma_hclk	Module clock
ipg_clk	Peripheral clock

6.4 Initialization/application information

The following sections discuss initialization of the eDMA and programming considerations.

6.4.1 eDMA initialization

To initialize the eDMA:

1. Write to the CR if a configuration other than the default is desired.
2. Write the channel priority levels to the DCHPRI*n* registers if a configuration other than the default is desired.
3. Enable error interrupts in the EEI register if desired.
4. Write the 32-byte TCD for each channel that may request service.
5. Enable any hardware service requests via the ERQ register.
6. Request channel service via either:
   - Software: setting TCD*n*_CSR[START]
   - Hardware: slave device asserting its eDMA peripheral request signal

After any channel requests service, a channel is selected for execution based on the arbitration and priority levels in the programming model. The eDMA engine reads the entire TCD, including the TCD control and status fields, as shown in Table 6-6, for the selected channel into its internal address path module.

As the TCD is read, the first transfer is initiated on the system bus, unless a configuration error is detected. Transfers from the source, as defined by TCD*n*_SADDR, to the destination, as defined by TCD*n*_DADDR, continue until the number of bytes specified by TCD*n*_NBYTES have been transferred.

When the transfer is complete, the eDMA engine's local TCD*n*_SADDR, TCD*n*_DADDR, and TCD*n*_CITER are written back to the main TCD memory and any minor loop channel linking is performed, if enabled. If the major loop is exhausted, further post-processing executes, such as interrupts, major loop channel linking, and scatter/gather operations, if enabled.

Table 6-6. TCD control and status fields

TCD*n*_CSR field name	Description
START	Control bit to start channel explicitly when using a software-initiated DMA service (automatically cleared by hardware)
ACTIVE	Status bit indicating the channel is currently in execution
DONE	Status bit indicating major loop completion (cleared by software when using a software-initiated DMA service)
DREQ	Control bit to disable DMA request at end of major loop completion when using a hardware-initiated DMA service
BWC	Control bits for throttling bandwidth control of a channel
ESG	Control bit to enable scatter/gather feature
INTHALF	Control bit to enable interrupt when major loop is half complete
INTMAJOR	Control bit to enable interrupt when major loop completes

The following figure shows how each DMA request initiates one minor-loop transfer, or iteration, without CPU intervention. DMA arbitration can occur after each minor loop, and one level of minor loop DMA preemption is allowed. The number of minor loops in a major loop is specified by the beginning iteration count (BITER).

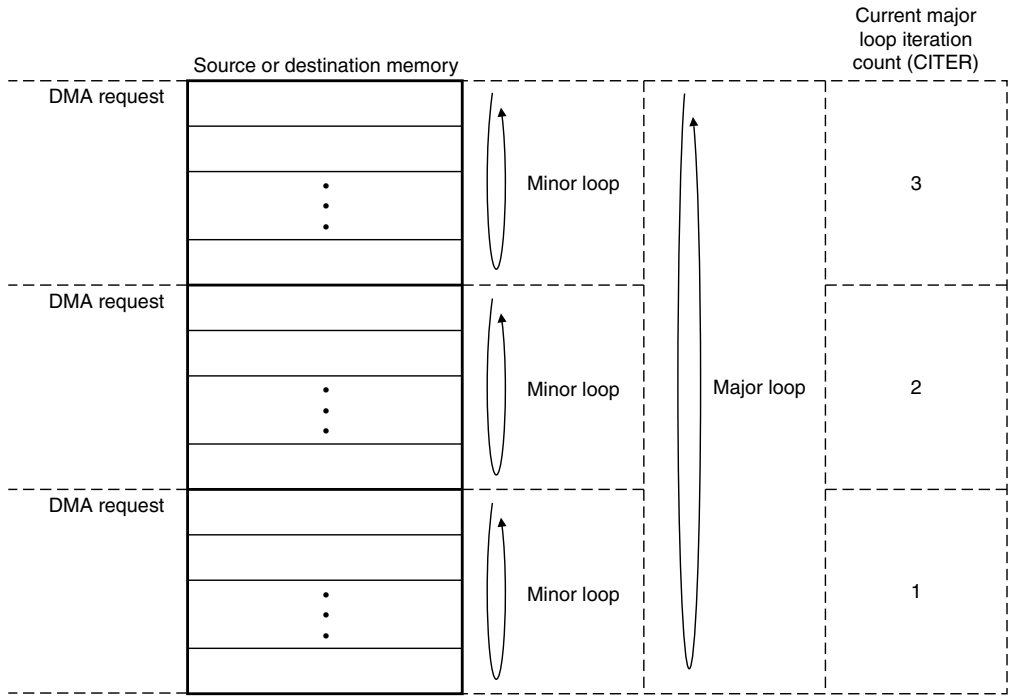


Figure 6-5. Example of multiple loop iterations

The following figure lists the memory array terms and how the TCD settings interrelate.

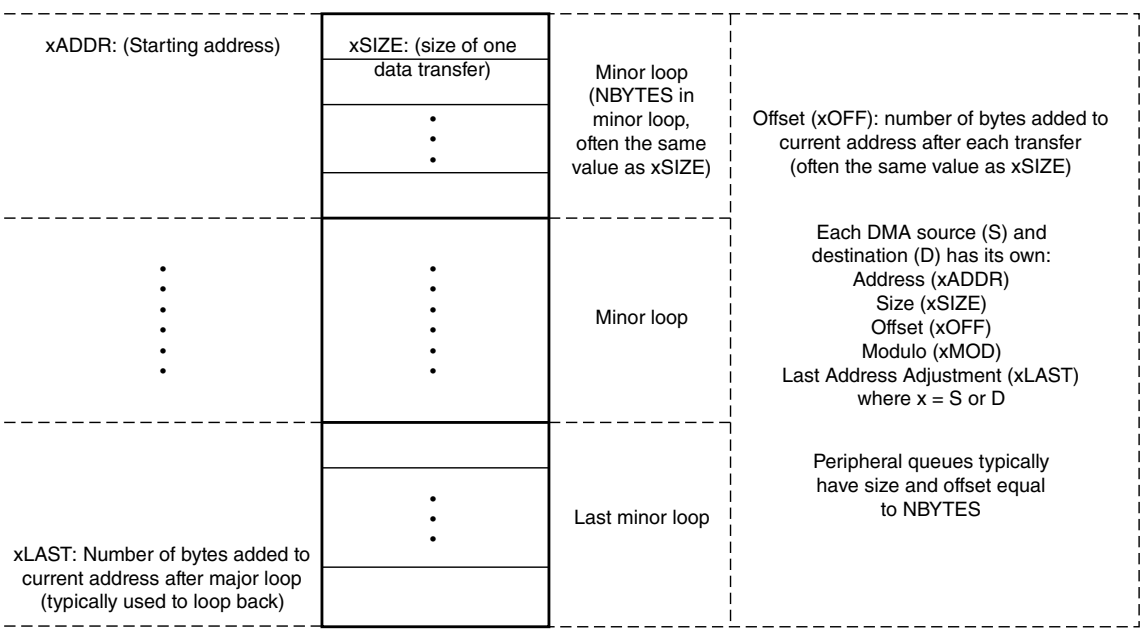


Figure 6-6. Memory array terms

6.4.2 Programming errors

The eDMA performs various tests on the transfer control descriptor to verify consistency in the descriptor data. Most programming errors are reported on a per-channel basis with the exception of channel priority error (ES[CPE]).

For all error types other than group or channel priority errors, the channel number causing the error is recorded in the Error Status register (DMAx_ES). If the error source is not removed before the next activation of the problem channel, the error is detected and recorded again.

Channel priority errors are identified within a group after that group has been selected as the active group. For example:

1. The eDMA is configured for fixed group and fixed channel arbitration modes.
2. Group 1 is the highest priority and all channels are unique in that group.
3. Group 0 is the next highest priority and has two channels with the same priority level.
4. If Group 1 has any service requests, those requests will be executed.
5. After all of Group 1 requests have completed, Group 0 will be the next active group.
6. If Group 0 has a service request, then an undefined channel in Group 0 will be selected and a channel priority error will occur.
7. This repeats until the all of Group 0 requests have been removed or a higher priority Group 1 request comes in.

In this sequence, for item 2, the eDMA acknowledge lines will assert only if the selected channel is requesting service via the eDMA peripheral request signal. If interrupts are enabled for all channels, the user will get an error interrupt, but the channel number for the ERR register and the error interrupt request line may be wrong because they reflect the selected channel. A group priority error is global and any request in any group will cause a group priority error.

If priority levels are not unique, when any channel requests service, a channel priority error is reported. The highest channel/group priority with an active request is selected, but the lowest numbered channel with that priority is selected by arbitration and executed by the eDMA engine. The hardware service request handshake signals, error interrupts, and error reporting are associated with the selected channel.

6.4.3 Arbitration mode considerations

This section discusses arbitration considerations for the eDMA.

6.4.3.1 Fixed group arbitration, Fixed channel arbitration

In this mode, the channel service request from the highest priority channel in the highest priority group is selected to execute. If the eDMA is programmed so that the channels within one group use "fixed" priorities, and that group is assigned the highest "fixed" priority of all groups, that group can take all the bandwidth of the eDMA controller. That is, no other groups will be serviced if there is always at least one DMA request pending on a channel in the highest priority group when the controller arbitrates the next DMA request. The advantage of this scenario is that latency can be small for channels that need to be serviced quickly. Preemption is available in this scenario only.

6.4.3.2 Fixed group arbitration, Round-robin channel arbitration

The highest priority group with a request will be serviced. Lower priority groups will be serviced if no pending requests exist in the higher priority groups.

Within each group, c hannels are serviced starting with the highest channel number and rotating through to the lowest channel number without regard to the channel priority levels assigned within the group.

This scenario could cause the same bandwidth consumption problem as indicated in Fixed group arbitration, Fixed channel arbitration, but all the channels in the highest priority group will be serviced. Service latency will be short on the highest priority group, but could potentially be very much longer as the group priority decreases.

6.4.4 DMA transfer examples

This section presents examples of how to perform DMA transfers with the eDMA.

6.4.4.1 Single request

To perform a simple transfer of n bytes of data with one activation, set the major loop to one (TCD*n*_CITER = TCD*n*_BITER = 1). The data transfer begins after the channel service request is acknowledged and the channel is selected to execute. After the transfer is complete, TCD*n*_CSR[DONE] is set and an interrupt generates if properly enabled.

For example, the following TCD entry is configured to transfer 16 bytes of data. The eDMA is programmed for one iteration of the major loop transferring 16 bytes per iteration. The source memory has an 8-bit memory port located at 0x1000. The

destination memory has a 32-bit port located at 0x2000. The address offsets are programmed in increments to match the transfer size: one byte for the source and four bytes for the destination. The final source and destination addresses are adjusted to return to their beginning values.

```
TCDn_CITER = TCDn_BITER = 1
TCDn_NBYTES = 16
TCDn_SADDR = 0x1000
TCDn_SOFF = 1
TCDn_ATTR[SSIZE] = 0
TCDn_SLAST = -16
TCDn_DADDR = 0x2000
TCDn_DOFF = 4
TCDn_ATTR[DSIZE] = 2
TCDn_DLAST_SGA= -16
TCDn_CSR[INTMAJOR] = 1
TCDn_CSR[START] = 1 (Should be written last after all other fields have been initialized)
All other TCDn fields = 0
```

This generates the following event sequence:

1. User write to the TCD*n*_CSR[START] bit requests channel service.
2. The channel is selected by arbitration for servicing.
3. eDMA engine writes: TCD*n*_CSR[DONE] = 0, TCD*n*_CSR[START] = 0, TCD*n*_CSR[ACTIVE] = 1.
4. eDMA engine reads: channel TCD data from local memory to internal register file.
5. The source-to-destination transfers are executed as follows:
   a. Read byte from location 0x1000, read byte from location 0x1001, read byte from 0x1002, read byte from 0x1003.
   b. Write 32 bits to location 0x2000 → first iteration of the minor loop.
   c. Read byte from location 0x1004, read byte from location 0x1005, read byte from 0x1006, read byte from 0x1007.
   d. Write 32 bits to location 0x2004 → second iteration of the minor loop.
   e. Read byte from location 0x1008, read byte from location 0x1009, read byte from 0x100A, read byte from 0x100B.
   f. Write 32 bits to location 0x2008 → third iteration of the minor loop.
   g. Read byte from location 0x100C, read byte from location 0x100D, read byte from 0x100E, read byte from 0x100F.
   h. Write 32 bits to location 0x200C → last iteration of the minor loop → major loop complete.
6. The eDMA engine writes: TCD*n*_SADDR = 0x1000, TCD*n*_DADDR = 0x2000, TCD*n*_CITER = 1 (TCD*n*_BITER).
7. The eDMA engine writes: TCD*n*_CSR[ACTIVE] = 0, TCD*n*_CSR[DONE] = 1, INT[*n*] = 1.
8. The channel retires and the eDMA goes idle or services the next channel.

6.4.4.2 Multiple requests

The following example transfers 32 bytes via two hardware requests, but is otherwise the same as the previous example. The only fields that change are the major loop iteration count and the final address offsets. The eDMA is programmed for two iterations of the major loop, transferring 16 bytes per iteration. After the channel's hardware requests are enabled in the ERQ register, the slave device initiates channel service requests.

```
TCDn_CITER = TCDn_BITER = 2
TCDn_SLAST = -32
TCDn_DLAST_SGA = -32
```

This would generate the following sequence of events:

1. First hardware, that is, the eDMA peripheral, requests channel service.
2. The channel is selected by arbitration for servicing.
3. eDMA engine writes: TCD*n*_CSR[DONE] = 0, TCD*n*_CSR[START] = 0, TCD*n*_CSR[ACTIVE] = 1.
4. eDMA engine reads: channel TCD*n* data from local memory to internal register file.
5. The source to destination transfers are executed as follows:
   a. Read byte from location 0x1000, read byte from location 0x1001, read byte from 0x1002, read byte from 0x1003.
   b. Write 32 bits to location 0x2000 → first iteration of the minor loop.
   c. Read byte from location 0x1004, read byte from location 0x1005, read byte from 0x1006, read byte from 0x1007.
   d. Write 32 bits to location 0x2004 → second iteration of the minor loop.
   e. Read byte from location 0x1008, read byte from location 0x1009, read byte from 0x100A, read byte from 0x100B.
   f. Write 32 bits to location 0x2008 → third iteration of the minor loop.
   g. Read byte from location 0x100C, read byte from location 0x100D, read byte from 0x100E, read byte from 0x100F.
   h. Write 32 bits to location 0x200C → last iteration of the minor loop.
6. eDMA engine writes: TCD*n*_SADDR = 0x1010, TCD*n*_DADDR = 0x2010, TCD*n*_CITER = 1.
7. eDMA engine writes: TCD*n*_CSR[ACTIVE] = 0.
8. The channel retires → one iteration of the major loop. The eDMA goes idle or services the next channel.
9. Second hardware, that is, eDMA peripheral, requests channel service.
10. The channel is selected by arbitration for servicing.
11. eDMA engine writes: TCD*n*_CSR[DONE] = 0, TCD*n*_CSR[START] = 0, TCD*n*_CSR[ACTIVE] = 1.
12. eDMA engine reads channel TCD data from local memory to internal register file.
13. The source to destination transfers are executed as follows:

a. Read byte from location 0x1010, read byte from location 0x1011, read byte from 0x1012, read byte from 0x1013.
b. Write 32 bits to location 0x2010 → first iteration of the minor loop.
c. Read byte from location 0x1014, read byte from location 0x1015, read byte from 0x1016, read byte from 0x1017.
d. Write 32 bits to location 0x2014 → second iteration of the minor loop.
e. Read byte from location 0x1018, read byte from location 0x1019, read byte from 0x101A, read byte from 0x101B.
f. Write 32 bits to location 0x2018 → third iteration of the minor loop.
g. Read byte from location 0x101C, read byte from location 0x101D, read byte from 0x101E, read byte from 0x101F.
h. Write 32 bits to location 0x201C → last iteration of the minor loop → major loop complete.

14. eDMA engine writes: TCD*n*_SADDR = 0x1000, TCD*n*_DADDR = 0x2000, TCD*n*_CITER = 2 (TCD*n*_BITER).
15. eDMA engine writes: TCD*n*_CSR[ACTIVE] = 0, TCD*n*_CSR[DONE] = 1, INT[n] = 1.
16. The channel retires → major loop complete. The eDMA goes idle or services the next channel.

6.4.4.3 Using the modulo feature

The modulo feature of the eDMA provides the ability to implement a circular data queue in which the size of the queue is a power of 2. MOD is a 5-bit field for the source and destination in the TCD, and it specifies which lower address bits increment from their original value after the address+offset calculation. All upper address bits remain the same as in the original value. A setting of zero for this field disables the modulo feature.

The following table shows how the transfer addresses are specified based on the setting of the MOD field. Here a circular buffer is created where the address wraps to the original value while the 28 upper address bits (0x1234567*x*) retain their original value. In this example the source address is set to 0x12345670, the offset is set to 4 bytes, and the MOD field is set to 4, allowing for a 2^4 byte (16-byte) size queue.

Table 6-7. Modulo example

Transfer number	Address
1	0x12345670
2	0x12345674
3	0x12345678
4	0x1234567C

Table continues on the next page...

Table 6-7. Modulo example (continued)

Transfer number	Address
5	0x12345670
6	0x12345674

6.4.5 Monitoring transfer descriptor status

This section discusses how to monitor eDMA status.

6.4.5.1 Testing for minor loop completion

There are two methods to test for minor loop completion when using software-initiated service requests. The first is to read the TCD*n*_CITER field and test for a change. Another method may be extracted from the sequence shown below. The second method is to test TCD*n*_CSR[START] and TCD*n*_CSR[ACTIVE]. The minor-loop-complete condition is indicated by both bits reading zero after TCD*n*_CSR[START] was set. Polling the TCD*n*_CSR[ACTIVE] bit may be inconclusive, because the active status may be missed if the channel execution is short in duration.

The TCD status bits execute the following sequence for a software activated channel:

Stage	TCD*n*_CSR bits			State
	START	ACTIVE	DONE	
1	1	0	0	Channel service request via software
2	0	1	0	Channel is executing
3a	0	0	0	Channel has completed the minor loop and is idle
3b	0	0	1	Channel has completed the major loop and is idle

The best method to test for minor-loop completion when using service requests initiated by hardware, that is, peripherals, is to read the TCD*n*_CITER field and test for a change. The hardware request and acknowledge handshake signals are not visible in the programmer's model.

The TCD status bits execute the following sequence for a hardware-activated channel:

Stage	TCD*n*_CSR bits			State
	START	ACTIVE	DONE	
1	0	0	0	Channel service request via hardware (peripheral request asserted)
2	0	1	0	Channel is executing
3a	0	0	0	Channel has completed the minor loop and is idle
3b	0	0	1	Channel has completed the major loop and is idle

For both activation types, the major-loop-complete status is explicitly indicated via the TCD*n*_CSR[DONE] bit.

The TCD*n*_CSR[START] bit is cleared automatically when the channel begins execution regardless of how the channel activates.

6.4.5.2 Reading the transfer descriptors of active channels

The eDMA reads back the true TCD*n*_SADDR, TCD*n*_DADDR, and TCD*n*_NBYTES values if read when a channel executes. The true values of SADDR, DADDR, and NBYTES are the values the eDMA engine currently uses in its internal register file and not the values in the TCD local memory for that channel. The addresses, SADDR and DADDR, and NBYTES, which decrement to zero as the transfer progresses, can give an indication of the progress of the transfer. All other values are read back from the TCD local memory.

6.4.5.3 Checking channel preemption status

Preemption is available only when fixed arbitration is selected for both group and channel arbitration modes. A preemptive situation is one in which a preempt-enabled channel runs and a higher priority request becomes active. When the eDMA engine is not operating in fixed group, fixed channel arbitration mode, determination of the actively running relative priority outstanding requests become undefined. Channel and/or group priorities are treated as equal, that is, constantly rotating, when Round-Robin Arbitration mode is selected.

The TCD*n*_CSR[ACTIVE] bit for the preempted channel remains asserted throughout the preemption. The preempted channel is temporarily suspended while the preempting channel executes one major loop iteration. If two TCD*n*_CSR[ACTIVE] bits are set simultaneously in the global TCD map, a higher priority channel is actively preempting a lower priority channel.

6.4.6 Channel linking

Channel linking (or chaining) is a mechanism where one channel sets the TCD*n*_CSR[START] bit of another channel (or itself), thus initiating a service request for that channel. When properly enabled, the EDMA engine automatically performs this operation at the major or minor loop completion.

The minor loop channel linking occurs at the completion of the minor loop (or one iteration of the major loop). The TCD*n*_CITER[ELINK] field determines whether a minor loop link is requested. When enabled, the channel link is made after each iteration of the major loop except for the last. When the major loop is exhausted, only the major loop channel link fields are used to determine if a channel link should be made. For example, the initial fields of:

```
TCDn_CITER[ELINK] = 1
TCDn_CITER[LINKCH] = 0xC
TCDn_CITER[CITER] value = 0x4
TCDn_CSR[MAJOR_ELINK] = 1
TCDn_CSR[MAJOR_LINKCH] = 0x3
```

executes as:

1. Minor loop done → set TCD2_CSR[START] bit.
2. Minor loop done → set TCD2_CSR[START] bit.
3. Minor loop done → set TCD2_CSR[START] bit.
4. Minor loop done, major loop done → set TCD3_CSR[START] bit.

When minor loop linking is enabled (TCD*n*_CITER[ELINK] = 1), the TCD*n*_CITER[CITER] field uses a nine-bit vector to form the current iteration count. When minor loop linking is disabled (TCD*n*_CITER[ELINK] = 0), the TCD*n*_CITER[CITER] field uses a 15-bit vector to form the current iteration count. The bits associated with the TCD*n*_CITER[LINKCH] field are concatenated onto the CITER value to increase the range of the CITER.

Note

The TCD*n*_CITER[ELINK] bit and the TCD*n*_BITER[ELINK] bit must be equal or a configuration error is reported. The CITER and BITER vector widths must be equal to calculate the major loop, half-way done interrupt point.

The following table summarizes how a DMA channel can link to another DMA channel, that is, use another channel's TCD, at the end of a loop.

Table 6-8. Channel linking parameters

Desired link behavior	TCD control field name	Description
Link at end of minor loop	CITER[ELINK]	Enable channel-to-channel linking on minor loop completion (current iteration)
	CITER[LINKCH]	Link channel number when linking at end of minor loop (current iteration)
Link at end of major loop	CSR[MAJOR_ELINK]	Enable channel-to-channel linking on major loop completion
	CSR[MAJOR_LINKCH]	Link channel number when linking at end of major loop

6.4.7 Dynamic programming

This section provides recommended methods to change the programming model during channel execution.

6.4.7.1 Dynamically changing the channel priority

The following two options are recommended for dynamically changing channel priority levels:

1. Switch to Round-Robin Channel Arbitration mode, change the channel priorities, then switch back to Fixed Arbitration mode
2. Disable all the channels, change the channel priorities, then enable the appropriate channels.

6.4.7.2 Dynamic channel linking

Dynamic channel linking is the process of setting TCDn_CSR[MAJORELINK] during channel execution (see the diagram in TCD structure). This field is read from the TCD local memory at the end of channel execution, thus enabling you to enable the feature during channel execution.

Because you can change the configuration during execution, a coherency model is needed. Consider the scenario where you attempt to execute a dynamic channel link by enabling TCDn_CSR[MAJORELINK] at the same time the eDMA engine is retiring the channel. TCDn_CSR[MAJORELINK] would be set in the programmer's model, but it would be unclear whether the actual link was made before the channel retired.

The following coherency model is recommended when executing a dynamic channel link request.

1. Write one to TCDn_CSR[MAJORELINK].
2. Read back TCDn_CSR[MAJORELINK].
3. Test the TCDn_CSR[MAJORELINK] request status:
   - If TCDn_CSR[MAJORELINK] = 1, the dynamic link attempt was successful.
   - If TCDn_CSR[MAJORELINK] = 0, the attempted dynamic link did not succeed (the channel was already retiring).

For this request, the TCD local memory controller forces TCDn_CSR[MAJORELINK] to zero on any writes to a channel's TCD.word7 after that channel's TCD.done bit is set, indicating the major loop is complete.

NOTE

You must clear TCDn_CSR[DONE] before writing TCDn_CSR[MAJORELINK]. The eDMA engine automatically clears TCDn_CSR[DONE] after a channel begins execution.

6.4.7.3 Dynamic scatter/gather

Scatter/gather is the process of automatically loading a new TCD into a channel. It enables a DMA channel to use multiple TCDs; this enables a DMA channel to scatter the DMA data to multiple destinations or gather it from multiple sources. When scatter/gather is enabled and the channel has finished its major loop, a new TCD is fetched from system memory and loaded into that channel's descriptor location in eDMA programmer's model, thus replacing the current descriptor.

Because you are able to change the configuration during execution, a coherency model is needed. Consider the scenario where you attempt to execute a dynamic scatter/gather operation by enabling the TCDn_CSR[ESG] bit at the same time the eDMA engine is retiring the channel. The ESG bit would be set in the programmer's model, but it would be unclear whether the actual scatter/gather request was honored before the channel retired.

Two methods for this coherency model are shown in the following subsections. Method 1 has the advantage of reading the MAJORLINKCH field and the ESG bit with a single read. For both dynamic channel linking and scatter/gather requests, the TCD local memory controller forces the TCD MAJOR[ELINK] and ESG bits to zero on any writes to a channel's TCD word 7 if that channel's TCD[DONE] bit is set, indicating the major loop is complete.

NOTE

The user must clear the TCDn_CSR[DONE] bit before writing the MAJORELINK or ESG bits. The TCDn_CSR[DONE] bit is cleared automatically by the eDMA engine after a channel begins execution.

6.4.7.3.1 Method 1 (channel not using major loop channel linking)

For a channel not using major loop channel linking, the coherency model described here may be used for a dynamic scatter/gather request.

When TCDn_CSR[MAJORELINK] is zero, TCDn_CSR[MAJORLINKCH] is not used by the eDMA. In this case, MAJORLINKCH may be used for other purposes. This method uses the MAJORLINKCH field as a TCD identification (ID).

1. When the descriptors are built, write a unique TCD ID in TCDn_CSR[MAJORLINKCH] for each TCD associated with a channel using dynamic scatter/gather.
2. Write one to TCDn_CSR[DREQ].

   Should a dynamic scatter/gather attempt fail, setting the DREQ bit prevents a future hardware activation of the channel. This stops the channel from executing with a destination address (DADDR) that was calculated using a scatter/gather address (written in the next step) instead of a DLAST_SGA final offset value.

3. Write the TCDn_DLASTSGA register with the scatter/gather address.
4. Write one to TCDn_CSR[ESG].
5. Read back the 16-bit TCD control/status field.
6. Test the ESG request status and MAJORLINKCH value in the TCDn_CSR register:

   If ESG = 1, the dynamic link attempt was successful.

   If ESG = 0 and MAJORLINKCH (ID) did not change, the attempted dynamic link did not succeed (the channel was already retiring).

   If ESG = 0 and MAJORLINKCH (ID) changed, the dynamic link attempt was successful (the new TCD's ESG value cleared the ESG bit).

6.4.7.3.2 Method 2 (channel using major loop channel linking)

For a channel using major loop channel linking, the coherency model described here may be used for a dynamic scatter/gather request. This method uses the TCD[DLAST_SGA] field as a TCD identification (ID).

1. Write one to TCDn_CSR[DREQ].

Should a dynamic scatter/gather attempt fail, setting TCDn_CSR[DREQ] prevents a future hardware activation of the channel. This stops the channel from executing with a destination address (DADDR) that was calculated using a scatter/gather address (written in the next step) instead of a DLAST_SGA final offset value.

2. Write the TCDn_DLAST_SGA register with the scatter/gather address.
3. Write one to TCDn_CSR[ESG].
4. Read back TCDn_CSR[ESG].
5. Test the ESG request status:

   If ESG = 1, the dynamic link attempt was successful.

   If ESG = 0, read the 32-bit TCDn_DLAST_SGA field.

   If ESG = 0 and TCDn_DLAST_SGA did not change, the attempted dynamic link did not succeed (the channel was already retiring).

   If ESG = 0 and TCDn_DLAST_SGA changed, the dynamic link attempt was successful (the new TCD's ESG value cleared the ESG bit).

6.4.8 Suspend/resume a DMA channel with active hardware service requests

The DMA enables you to move data from memory or peripheral registers to another location in memory or peripheral registers without CPU interaction. After the DMA and peripherals have been configured and are active, it is rare to suspend a peripheral's service request dynamically. In this scenario, there are certain restrictions to disabling a DMA hardware service request. For coherency, a specific procedure must be followed. This section provides guidance on how to coherently suspend and resume a Direct Memory Access (DMA) channel when the DMA is triggered by a slave module such as the Serial Peripheral Interface (SPI), ADC, or other module.

6.4.8.1 Suspend an active DMA channel

To suspend an active DMA channel:

1. Stop the DMA service request at the peripheral first. Confirm it has been disabled by reading back the appropriate register in the peripheral.
2. Check the DMA's Hardware Request Status register (DMA_HRSn) to ensure there is no service request to the DMA channel being suspended. Then disable the hardware service request by clearing the ERQ bit on appropriate DMA channel.

6.4.8.2 Resume a DMA channel

To resume a DMA channel:

1. Enable the DMA service request on the appropriate channel by setting the relevant ERQ bit.
2. Enable the DMA service request at the peripheral.

For example, assume the SPI is set as a master for transmitting data via a DMA service request when the SPI_TXFIFO has an empty slot. The DMA transfers the next command and data to the TXFIFO upon the request. You must suspend the DMA/SPI transfer loop and perform the following steps:

1. Disable the DMA service request at the source by writing zero to SPI_RSER[TFFF_RE]. Confirm that SPI_RSER[TFFF_RE] is zero.
2. Ensure there is no DMA service request from the SPI by verifying that DMA_HRS[HRS*n*] is zero for the appropriate channel. If no service request is present, disable the DMA channel by clearing the channel's ERQ bit. If a service request is present, wait until the request has been processed and the HRS bit reads zero.

6.5 Memory map/register definition

The eDMA programming model consists of registers that provide:

- Control and status functions
- Channel configuration functions
- TCD definition functions

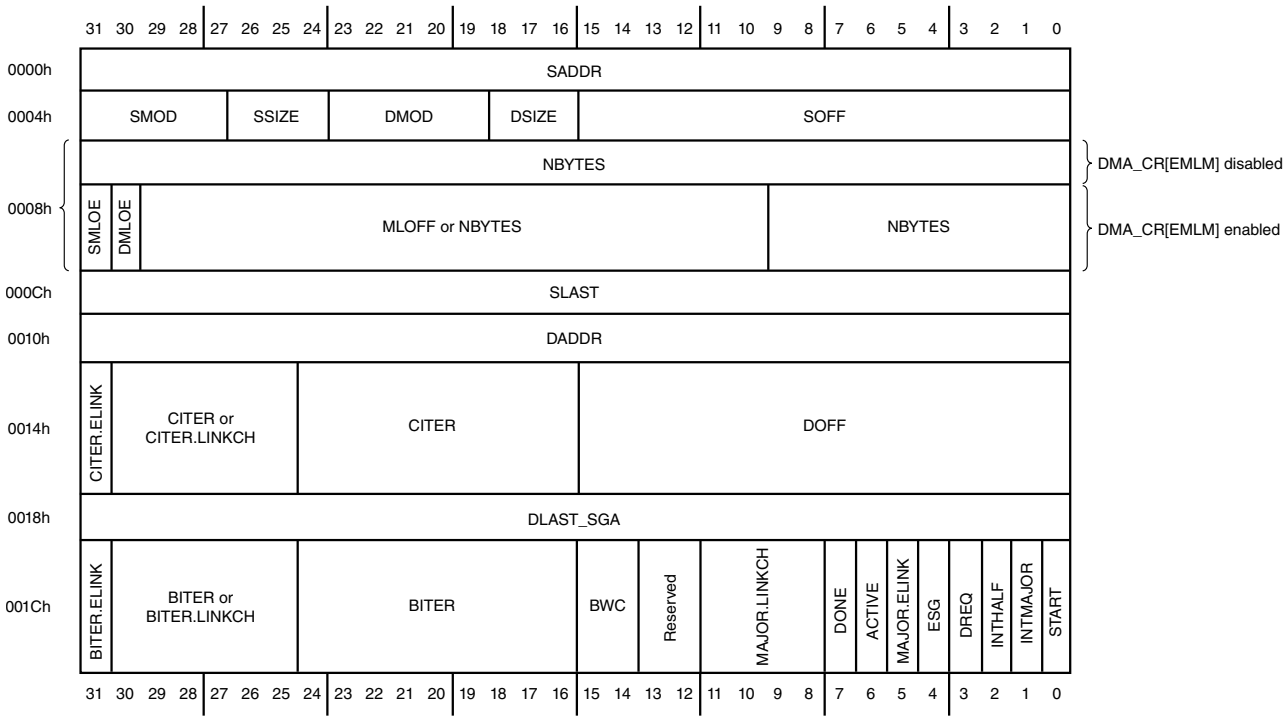
6.5.1 TCD memory

Each channel requires a 32-byte TCD to define the desired data movement operation. The channel descriptors are in local memory in sequential order: channel 0, channel 1, ... channel 31. Each TCD*n* definition comprises 11 registers of 16 or 32 bits.

6.5.2 TCD initialization

Before activating a channel, you must initialize its TCD with the appropriate transfer profile.

6.5.3 TCD structure



6.5.4 Reserved memory and fields

- Reading reserved fields in a register returns the value of zero.
- The eDMA ignores writes to reserved bits in a register.
- Reading or writing a reserved memory location generates a bus error.

6.5.5 DMA register descriptions

6.5.5.1 DMA memory map

DMA base address: 400E_8000h

Offset	Register	Width (In bits)	Access	Reset value
0h	Control (CR)	32	RW	Table 6-8

Table continues on the next page...

Offset	Register	Width (In bits)	Access	Reset value
4h	Error Status (ES)	32	RO	0000_0000h
Ch	Enable Request (ERQ)	32	RW	0000_0000h
14h	Enable Error Interrupt (EEI)	32	RW	0000_0000h
18h	Clear Enable Error Interrupt (CEEI)	8	WORZ	00h
19h	Set Enable Error Interrupt (SEEI)	8	WORZ	00h
1Ah	Clear Enable Request (CERQ)	8	WORZ	00h
1Bh	Set Enable Request (SERQ)	8	WORZ	00h
1Ch	Clear DONE Status Bit (CDNE)	8	WORZ	00h
1Dh	Set START Bit (SSRT)	8	WORZ	00h
1Eh	Clear Error (CERR)	8	WORZ	00h
1Fh	Clear Interrupt Request (CINT)	8	WORZ	00h
24h	Interrupt Request (INT)	32	W1C	0000_0000h
2Ch	Error (ERR)	32	W1C	0000_0000h
34h	Hardware Request Status (HRS)	32	RO	0000_0000h
44h	Enable Asynchronous Request in Stop (EARS)	32	RW	0000_0000h
100h	Channel Priority (DCHPRI3)	8	RW	03h
101h	Channel Priority (DCHPRI2)	8	RW	02h
102h	Channel Priority (DCHPRI1)	8	RW	01h
103h	Channel Priority (DCHPRI0)	8	RW	00h
104h	Channel Priority (DCHPRI7)	8	RW	07h
105h	Channel Priority (DCHPRI6)	8	RW	06h
106h	Channel Priority (DCHPRI5)	8	RW	05h
107h	Channel Priority (DCHPRI4)	8	RW	04h
108h	Channel Priority (DCHPRI11)	8	RW	0Bh
109h	Channel Priority (DCHPRI10)	8	RW	0Ah
10Ah	Channel Priority (DCHPRI9)	8	RW	09h
10Bh	Channel Priority (DCHPRI8)	8	RW	08h
10Ch	Channel Priority (DCHPRI15)	8	RW	0Fh
10Dh	Channel Priority (DCHPRI14)	8	RW	0Eh
10Eh	Channel Priority (DCHPRI13)	8	RW	0Dh
10Fh	Channel Priority (DCHPRI12)	8	RW	0Ch
110h	Channel Priority (DCHPRI19)	8	RW	13h
111h	Channel Priority (DCHPRI18)	8	RW	12h
112h	Channel Priority (DCHPRI17)	8	RW	11h
113h	Channel Priority (DCHPRI16)	8	RW	10h
114h	Channel Priority (DCHPRI23)	8	RW	17h
115h	Channel Priority (DCHPRI22)	8	RW	16h
116h	Channel Priority (DCHPRI21)	8	RW	15h
117h	Channel Priority (DCHPRI20)	8	RW	14h
118h	Channel Priority (DCHPRI27)	8	RW	1Bh

Table continues on the next page...

Offset	Register	Width (In bits)	Access	Reset value
119h	Channel Priority (DCHPRI26)	8	RW	1Ah
11Ah	Channel Priority (DCHPRI25)	8	RW	19h
11Bh	Channel Priority (DCHPRI24)	8	RW	18h
11Ch	Channel Priority (DCHPRI31)	8	RW	1Fh
11Dh	Channel Priority (DCHPRI30)	8	RW	1Eh
11Eh	Channel Priority (DCHPRI29)	8	RW	1Dh
11Fh	Channel Priority (DCHPRI28)	8	RW	1Ch
1000h - 13E0h	TCD Source Address (TCD0_SADDR - TCD31_SADDR)	32	RW	Table 6-8
1004h - 13E4h	TCD Signed Source Address Offset (TCD0_SOFF - TCD31_SOFF)	16	RW	Table 6-8
1006h - 13E6h	TCD Transfer Attributes (TCD0_ATTR - TCD31_ATTR)	16	RW	Table 6-8
1008h - 13E8h	TCD Minor Byte Count (Minor Loop Mapping Disabled) (TCD0_NBYTES_MLNO - TCD31_NBYTES_MLNO)	32	RW	Table 6-8
1008h - 13E8h	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (TCD0_NBYTES_MLOFFNO - TCD31_NBYTES_MLOFFNO)	32	RW	Table 6-8
1008h - 13E8h	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (TCD0_NBYTES_MLOFFYES - TCD31_NBYTES_MLOFFYES)	32	RW	Table 6-8
100Ch - 13ECh	TCD Last Source Address Adjustment (TCD0_SLAST - TCD31_SLAST)	32	RW	Table 6-8
1010h - 13F0h	TCD Destination Address (TCD0_DADDR - TCD31_DADDR)	32	RW	Table 6-8
1014h - 13F4h	TCD Signed Destination Address Offset (TCD0_DOFF - TCD31_DOFF)	16	RW	Table 6-8
1016h - 13F6h	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) (TCD0_CITER_ELINKNO - TCD31_CITER_ELINKNO)	16	RW	Table 6-8
1016h - 13F6h	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (TCD0_CITER_ELINKYES - TCD31_CITER_ELINKYES)	16	RW	Table 6-8
1018h - 13F8h	TCD Last Destination Address Adjustment/Scatter Gather Address (TCD0_DLASTSGA - TCD31_DLASTSGA)	32	RW	Table 6-8
101Ch - 13FCh	TCD Control and Status (TCD0_CSR - TCD31_CSR)	16	RW	Table 6-8
101Eh - 13FEh	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (TCD0_BITER_ELINKNO - TCD31_BITER_ELINKNO)	16	RW	Table 6-8
101Eh - 13FEh	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (TCD0_BITER_ELINKYES - TCD31_BITER_ELINKYES)	16	RW	Table 6-8

6.5.5.2 Control (CR)

6.5.5.2.1 Offset

Register	Offset
CR	0h

6.5.5.2.2 Function

This register defines the basic operating configuration of the eDMA module. eDMA arbitrates channel service requests in two groups of 16 channels each:

- Group 1 contains channels 31-16
- Group 0 contains channels 15-0

You can configure arbitration within a group to use either a fixed-priority or a round-robin scheme. For fixed-priority arbitration, eDMA selects and executes the highest-priority channel that requests service. The channel priority registers assign the priorities (see the Channel Priority (DCHPRI0 - DCHPRI31) registers). For round-robin arbitration, the eDMA engine ignores channel priorities and cycles through channels within each group from high to low channel number without regard to priority.

NOTE

For correct operation, you must write to this register only when the eDMA channels are inactive—that is, when TCD*n*_CSR[ACTIVE] = 0.

The group priorities operate in a similar fashion. In group fixed priority arbitration mode, channel service requests in the highest priority group are executed first, where priority level 1 is the highest and priority level 0 is the lowest. The group priorities are assigned in the GRP*n*PRI fields of the Control register (CR). All group priorities must have unique values prior to any channel service requests occurring; otherwise, a configuration error is reported. For group round robin arbitration, eDMA ignores the group priorities and the groups are cycled through (from high to low group number) without regard to priority.

Minor loop offsets are address-offset values to be added to the final source address (TCDn_SADDR) or destination address (TCDn_DADDR) when the minor loop completes. When you enable minor loop offsets, eDMA adds the minor loop offset (MLOFF) value to the final source address (TCDn_SADDR), to the final destination address (TCDn_DADDR), or to both, before it writes the addresses back into the TCD. If the major loop is complete, eDMA ignores the minor loop offset, and uses the major loop address offsets (TCDn_SLAST and TCDn_DLAST_SGA) to compute the next TCDn_SADDR and TCDn_DADDR values.

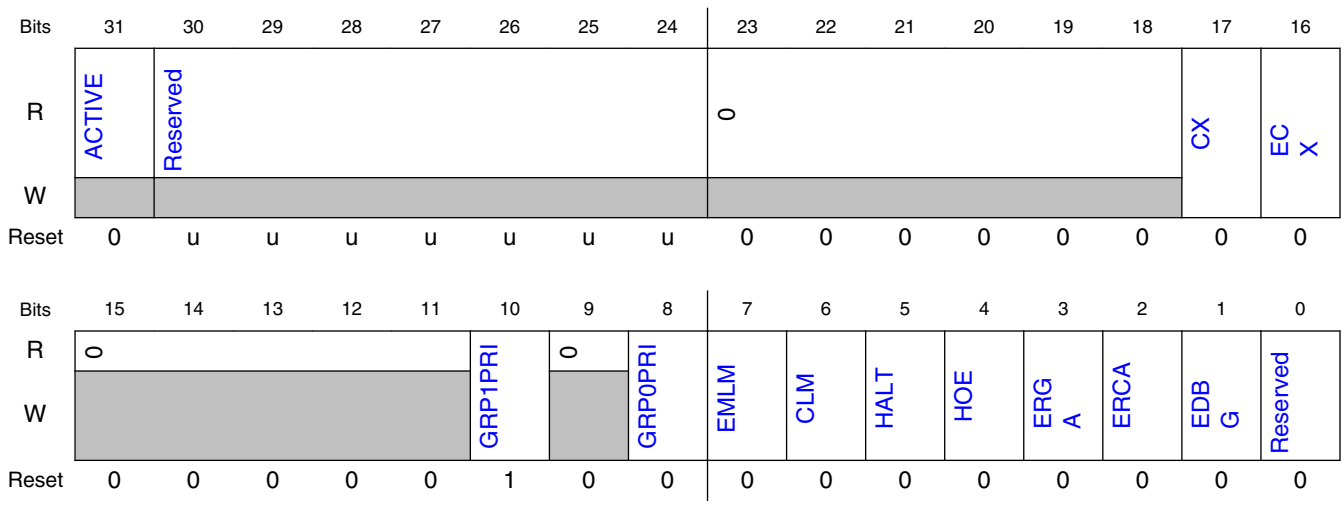
Enabling minor loop mapping (EMLM = 1) redefines TCDn word2. eDMA uses a portion of TCDn word2 for multiple fields:

- A source enable field (SMLOE) to specify the minor loop offset is to be applied to the source address (TCDn_SADDR) when the minor loop completes
- A destination enable field (DMLOE) to specify the minor loop offset to be applied to the destination address (TCDn_DADDR) when the minor loop completes
- The sign extended minor loop offset value (MLOFF).

eDMA uses the same offset value (MLOFF) for both source and destination minor loop offsets. When you enable either minor loop offset (SMLOE = 1 or DMLOE = 1), the NBYTES field reduces in size to 10 bits. When you disable both minor loop offsets (SMLOE = 0 and and DMLOE = 0), the NBYTES field is a 30-bit vector.

When you disable minor loop mapping (EMLM = 0), the NBYTES field contains all 32 bits of TCDn word2.

6.5.5.2.3 Diagram



6.5.5.2.4 Fields

Field	Function
31 ACTIVE	eDMA Active Status 0b - eDMA is idle 1b - eDMA is executing a channel
30-24 —	Reserved
23-18 —	Reserved
17 CX	Cancel Transfer When you write 1 to this field, the following actions take place:

Table continues on the next page...

Field	Function
	• Stop the executing channel • Force the minor loop to finish. The cancellation takes effect after the last write of the current read/write sequence. This field is automatically written with 0 after the cancellation completes. The cancellation retires the channel normally as if the minor loop completed. 0b - Normal operation 1b - Cancel the remaining data transfer
16 ECX	Error Cancel Transfer When you write a 1 to this field, the following actions take place: • Stop the executing channel • Force the minor loop to finish. The cancellation takes effect after the last write of the current read/write sequence. This field is automatically reset to 0 after the cancellation completes. In addition to cancelling the transfer, eDMA: • Treats the cancel as an error condition • Updates the Error Status register (DMAx_ES) • Optionally generates an error interrupt. 0b - Normal operation 1b - Cancel the remaining data transfer
15-11 —	Reserved
10 GRP1PRI	Channel Group 1 Priority Group 1 priority level when fixed priority group arbitration is enabled.
9 —	Reserved
8 GRP0PRI	Channel Group 0 Priority Group 0 priority level when fixed priority group arbitration is enabled.
7 EMLM	Enable Minor Loop Mapping When the value of this field is 0, TCDn.word2 is a 32-bit NBYTES field. When the value of this field is 1, TCDn.word2 includes: • Individual enable fields • An offset field • The NBYTES field. The individual enable fields allow the minor loop offset to be applied to the source address, the destination address, or both. The NBYTES field reduces in size when either offset is enabled. 0b - Disabled 1b - Enabled
6 CLM	Continuous Link Mode When the value of this field is 0, a minor loop channel link made to itself goes through channel arbitration before being activated again. When the value of this field is 1, a minor loop channel link made to itself does not go through channel arbitration before being activated again. When the minor loop completes, the channel activates again if that channel has a minor loop channel link enabled and the link channel is itself. This effectively applies the minor loop offsets and restarts the next minor loop. **NOTE:** Do not use continuous link mode with a channel linking to itself if there is only one minor loop iteration per service request, for example, if the channel's NBYTES value is the same as either the source or destination size. The same data transfer profile can be achieved by simply increasing the NBYTES value, which provides more efficient, faster processing. 0b - Continuous link mode is off

Table continues on the next page...

Field	Function
	1b - Continuous link mode is on
5 HALT	Halt eDMA Operations When this field is 1 the following actions take place: • eDMA stalls the start of any new channels • Executing channels are allowed to complete. When you write 0 to this field, channel execution resumes. 0b - Normal operation 1b - eDMA operations halted
4 HOE	Halt On Error When this field is 1, any error causes the eDMA engine to write 1 to the HALT field. Subsequently, the eDMA engine ignores all service requests until you write 0 to the HALT field. 0b - Normal operation 1b - Error causes HALT field to be automatically set to 1
3 ERGA	Enable Round Robin Group Arbitration When you write 1 to this field, eDMA uses round robin arbitration for selection among the groups. Otherwise, eDMA uses fixed priority arbitration. 0b - Fixed priority arbitration 1b - Round robin arbitration
2 ERCA	Enable Round Robin Channel Arbitration When you write 1 to this field, eDMA uses round robin arbitration for channel selection. Otherwise, eDMA uses fixed priority arbitration for channel selection. 0b - Fixed priority arbitration within each group 1b - Round robin arbitration within each group
1 EDBG	Enable Debug When this field is 0 and the chip enters Debug mode, eDMA continues operation. When this field is 1, entry of the chip into Debug mode causes the eDMA to stall the start of a new channel. Executing channels are allowed to complete. Channel execution resumes when the chip exits Debug mode or you write 0 to this field. 0b - When the chip is in Debug mode, the eDMA continues to operate. 1b - When the chip is in debug mode, the DMA stalls the start of a new channel. Executing channels are allowed to complete.
0 —	Reserved

6.5.5.3 Error Status (ES)

6.5.5.3.1 Offset

Register	Offset
ES	4h

6.5.5.3.2 Function

The ES register provides information concerning the most-recently recorded channel error. Channel errors can be caused by:

- A configuration error, that is:
  - An illegal setting in the transfer-control descriptor
  - An illegal priority register setting in fixed arbitration
- An error termination to a bus master read or write cycle
- A cancel transfer with error field that is 1 when a transfer is canceled via the corresponding cancel transfer control field

See Fault reporting and handling for more details.

6.5.5.3.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	VLD	0														ECX
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	GPE	CPE	0	ERRCHN					SAE	SOE	DAE	DOE	NCE	SGE	SBE	DBE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

6.5.5.3.4 Fields

Field	Function
31 VLD	Logical OR of all ERR status fields 0b - No ERR fields are 1 1b - At least one ERR field has a value of 1, indicating a valid error exists that has not been cleared
30-17 —	Reserved
16 ECX	Transfer Canceled 0b - No canceled transfers 1b - The most-recently recorded entry was a canceled transfer initiated by the error cancel transfer field
15 GPE	Group Priority Error 0b - No group priority error. 1b - The most-recently recorded error was a configuration error among the group priorities. All group priorities are not unique.
14 CPE	Channel Priority Error 0b - No channel priority error. 1b - The most-recently recorded error was a configuration error in the channel priorities within a group. Channel priorities within a group are not unique.

Table continues on the next page...

Field	Function
13 —	Reserved
12-8 ERRCHN	Error Channel Number or Canceled Channel Number The channel number of the most-recently recorded error, excluding GPE and CPE errors or most-recently recorded error canceled transfer.
7 SAE	Source Address Error 0b - No source address configuration error. 1b - The most-recently recorded error was a configuration error detected in the TCDn_SADDR field. TCDn_SADDR is inconsistent with TCDn_ATTR[SSIZE].
6 SOE	Source Offset Error 0b - No source offset configuration error. 1b - The most-recently recorded error was a configuration error detected in the TCDn_SOFF field. TCDn_SOFF is inconsistent with TCDn_ATTR[SSIZE].
5 DAE	Destination Address Error 0b - No destination address configuration error. 1b - The most-recently recorded error was a configuration error detected in the TCDn_DADDR field. TCDn_DADDR is inconsistent with TCDn_ATTR[DSIZE].
4 DOE	Destination Offset Error 0b - No destination offset configuration error. 1b - The most-recently recorded error was a configuration error detected in the TCDn_DOFF field. TCDn_DOFF is inconsistent with TCDn_ATTR[DSIZE].
3 NCE	NBYTES/CITER Configuration Error 0b - No NBYTES/CITER configuration error. 1b - The most-recently recorded error was a configuration error detected in the TCDn_NBYTES or TCDn_CITER fields. TCDn_NBYTES is not a multiple of TCDn_ATTR[SSIZE] and TCDn_ATTR[DSIZE], or TCDn_CITER[CITER] = 0, or TCDn_CITER[ELINK] is not equal to TCDn_BITER[ELINK].
2 SGE	Scatter/Gather Configuration Error When 1, this field indicates the most-recently recorded error was a configuration error detected in the TCDn_DLASTSGA field. eDMA checks This field at the beginning of a scatter/gather operation after major loop completion if TCDn_CSR[ESG] is enabled. TCDn_DLASTSGA is not on a 32-byte boundary. 0b - No scatter/gather configuration error. 1b - The most-recently recorded error was a configuration error detected in the TCDn_DLASTSGA field.
1 SBE	Source Bus Error 0b - No source bus error. 1b - The most-recently recorded error was a bus error on a source read.
0 DBE	Destination Bus Error 0b - No destination bus error. 1b - The most-recently recorded error was a bus error on a destination write.

6.5.5.4 Enable Request (ERQ)

6.5.5.4.1 Offset

Register	Offset
ERQ	Ch

6.5.5.4.2 Function

The ERQ register provides a bit map for the 32 channels to enable the request signal for each channel. The state of any given channel enable is directly affected by writes to this register; it is also affected by writes to the SERQ and CERQ registers. These registers are provided so the request enable for a single channel can easily be modified without needing to perform a read-modify-write sequence to this register.

DMA request input signals and this enable request field must be set to 1 before a channel's hardware service request is accepted. The state of the DMA enable request field does not affect a channel service request made explicitly through software or a linked channel request.

NOTE

Disable a channel's hardware service request at the source before writing 0 to the channel's ERQ field.

6.5.5.4.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	ERQ31	ERQ30	ERQ29	ERQ28	ERQ27	ERQ26	ERQ25	ERQ24	ERQ23	ERQ22	ERQ21	ERQ20	ERQ19	ERQ18	ERQ17	ERQ16
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	ERQ15	ERQ14	ERQ13	ERQ12	ERQ11	ERQ10	ERQ9	ERQ8	ERQ7	ERQ6	ERQ5	ERQ4	ERQ3	ERQ2	ERQ1	ERQ0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

6.5.5.4.4 Fields

Field	Function
31 ERQ31	Enable DMA Request 31 0b - The DMA request signal for channel 31 is disabled 1b - The DMA request signal for channel 31 is enabled

Table continues on the next page...

Field	Function
30 ERQ30	Enable DMA Request 30 0b - The DMA request signal for channel 30 is disabled 1b - The DMA request signal for channel 30 is enabled
29 ERQ29	Enable DMA Request 29 0b - The DMA request signal for channel 29 is disabled 1b - The DMA request signal for channel 29 is enabled
28 ERQ28	Enable DMA Request 28 0b - The DMA request signal for channel 28 is disabled 1b - The DMA request signal for channel 28 is enabled
27 ERQ27	Enable DMA Request 27 0b - The DMA request signal for channel 27 is disabled 1b - The DMA request signal for channel 27 is enabled
26 ERQ26	Enable DMA Request 26 0b - The DMA request signal for channel 26 is disabled 1b - The DMA request signal for channel 26 is enabled
25 ERQ25	Enable DMA Request 25 0b - The DMA request signal for channel 25 is disabled 1b - The DMA request signal for channel 25 is enabled
24 ERQ24	Enable DMA Request 24 0b - The DMA request signal for channel 24 is disabled 1b - The DMA request signal for channel 24 is enabled
23 ERQ23	Enable DMA Request 23 0b - The DMA request signal for channel 23 is disabled 1b - The DMA request signal for channel 23 is enabled
22 ERQ22	Enable DMA Request 22 0b - The DMA request signal for channel 22 is disabled 1b - The DMA request signal for channel 22 is enabled
21 ERQ21	Enable DMA Request 21 0b - The DMA request signal for channel 21 is disabled 1b - The DMA request signal for channel 21 is enabled
20 ERQ20	Enable DMA Request 20 0b - The DMA request signal for channel 20 is disabled 1b - The DMA request signal for channel 20 is enabled
19 ERQ19	Enable DMA Request 19 0b - The DMA request signal for channel 19 is disabled 1b - The DMA request signal for channel 19 is enabled
18 ERQ18	Enable DMA Request 18 0b - The DMA request signal for channel 18 is disabled 1b - The DMA request signal for channel 18 is enabled
17 ERQ17	Enable DMA Request 17 0b - The DMA request signal for channel 17 is disabled 1b - The DMA request signal for channel 17 is enabled
16 ERQ16	Enable DMA Request 16 0b - The DMA request signal for channel 16 is disabled 1b - The DMA request signal for channel 16 is enabled
15 ERQ15	Enable DMA Request 15 0b - The DMA request signal for channel 15 is disabled 1b - The DMA request signal for channel 15 is enabled
14 ERQ14	Enable DMA Request 14 0b - The DMA request signal for channel 14 is disabled

Table continues on the next page...

Field	Function
	1b - The DMA request signal for channel 14 is enabled
13 ERQ13	Enable DMA Request 13 0b - The DMA request signal for channel 13 is disabled 1b - The DMA request signal for channel 13 is enabled
12 ERQ12	Enable DMA Request 12 0b - The DMA request signal for channel 12 is disabled 1b - The DMA request signal for channel 12 is enabled
11 ERQ11	Enable DMA Request 11 0b - The DMA request signal for channel 11 is disabled 1b - The DMA request signal for channel 11 is enabled
10 ERQ10	Enable DMA Request 10 0b - The DMA request signal for channel 10 is disabled 1b - The DMA request signal for channel 10 is enabled
9 ERQ9	Enable DMA Request 9 0b - The DMA request signal for channel 9 is disabled 1b - The DMA request signal for channel 9 is enabled
8 ERQ8	Enable DMA Request 8 0b - The DMA request signal for channel 8 is disabled 1b - The DMA request signal for channel 8 is enabled
7 ERQ7	Enable DMA Request 7 0b - The DMA request signal for channel 7 is disabled 1b - The DMA request signal for channel 7 is enabled
6 ERQ6	Enable DMA Request 6 0b - The DMA request signal for channel 6 is disabled 1b - The DMA request signal for channel 6 is enabled
5 ERQ5	Enable DMA Request 5 0b - The DMA request signal for channel 5 is disabled 1b - The DMA request signal for channel 5 is enabled
4 ERQ4	Enable DMA Request 4 0b - The DMA request signal for channel 4 is disabled 1b - The DMA request signal for channel 4 is enabled
3 ERQ3	Enable DMA Request 3 0b - The DMA request signal for channel 3 is disabled 1b - The DMA request signal for channel 3 is enabled
2 ERQ2	Enable DMA Request 2 0b - The DMA request signal for channel 2 is disabled 1b - The DMA request signal for channel 2 is enabled
1 ERQ1	Enable DMA Request 1 0b - The DMA request signal for channel 1 is disabled 1b - The DMA request signal for channel 1 is enabled
0 ERQ0	Enable DMA Request 0 0b - The DMA request signal for channel 0 is disabled 1b - The DMA request signal for channel 0 is enabled

6.5.5.5 Enable Error Interrupt (EEI)

6.5.5.5.1 Offset

Register	Offset
EEI	14h

6.5.5.5.2 Function

The EEI register provides a bit map for the 32 channels to enable the error interrupt signal for each channel. The state of any given channel's error interrupt enable is directly affected by writes to this register; it is also affected by writes to the SEEI and CEEI registers. These registers are provided so that the error interrupt enable for a single channel can easily be modified without the need to perform a read-modify-write sequence to the EEI register.

The DMA error indicator and the error interrupt enable field must be set to 1 before an error interrupt request for a given channel is sent to the interrupt controller.

6.5.5.5.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	EEI31	EEI30	EEI29	EEI28	EEI27	EEI26	EEI25	EEI24	EEI23	EEI22	EEI21	EEI20	EEI19	EEI18	EEI17	EEI16
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	EEI15	EEI14	EEI13	EEI12	EEI11	EEI10	EEI 9	EEI 8	EEI 7	EEI 6	EEI 5	EEI 4	EEI 3	EEI 2	EEI 1	EEI 0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

6.5.5.5.4 Fields

Field	Function
31 EEI31	Enable Error Interrupt 31 0b - An error on channel 31 does not generate an error interrupt 1b - An error on channel 31 generates an error interrupt request
30 EEI30	Enable Error Interrupt 30 0b - An error on channel 30 does not generate an error interrupt 1b - An error on channel 30 generates an error interrupt request
29 EEI29	Enable Error Interrupt 29 0b - An error on channel 29 does not generate an error interrupt 1b - An error on channel 29 generates an error interrupt request

Table continues on the next page...

Field	Function
28 EEI28	Enable Error Interrupt 28 0b - An error on channel 28 does not generate an error interrupt 1b - An error on channel 28 generates an error interrupt request
27 EEI27	Enable Error Interrupt 27 0b - An error on channel 27 does not generate an error interrupt 1b - An error on channel 27 generates an error interrupt request
26 EEI26	Enable Error Interrupt 26 0b - An error on channel 26 does not generate an error interrupt 1b - An error on channel 26 generates an error interrupt request
25 EEI25	Enable Error Interrupt 25 0b - An error on channel 25 does not generate an error interrupt 1b - An error on channel 25 generates an error interrupt request
24 EEI24	Enable Error Interrupt 24 0b - An error on channel 24 does not generate an error interrupt 1b - An error on channel 24 generates an error interrupt request
23 EEI23	Enable Error Interrupt 23 0b - An error on channel 23 does not generate an error interrupt 1b - An error on channel 23 generates an error interrupt request
22 EEI22	Enable Error Interrupt 22 0b - An error on channel 22 does not generate an error interrupt 1b - An error on channel 22 generates an error interrupt request
21 EEI21	Enable Error Interrupt 21 0b - An error on channel 21 does not generate an error interrupt 1b - An error on channel 21 generates an error interrupt request
20 EEI20	Enable Error Interrupt 20 0b - An error on channel 20 does not generate an error interrupt 1b - An error on channel 20 generates an error interrupt request
19 EEI19	Enable Error Interrupt 19 0b - An error on channel 19 does not generate an error interrupt 1b - An error on channel 19 generates an error interrupt request
18 EEI18	Enable Error Interrupt 18 0b - An error on channel 18 does not generate an error interrupt 1b - An error on channel 18 generates an error interrupt request
17 EEI17	Enable Error Interrupt 17 0b - An error on channel 17 does not generate an error interrupt 1b - An error on channel 17 generates an error interrupt request
16 EEI16	Enable Error Interrupt 16 0b - An error on channel 16 does not generate an error interrupt 1b - An error on channel 16 generates an error interrupt request
15 EEI15	Enable Error Interrupt 15 0b - An error on channel 15 does not generate an error interrupt 1b - An error on channel 15 generates an error interrupt request
14 EEI14	Enable Error Interrupt 14 0b - An error on channel 14 does not generate an error interrupt 1b - An error on channel 14 generates an error interrupt request
13 EEI13	Enable Error Interrupt 13 0b - An error on channel 13 does not generate an error interrupt 1b - An error on channel 13 generates an error interrupt request
12 EEI12	Enable Error Interrupt 12 0b - An error on channel 12 does not generate an error interrupt

Table continues on the next page...

Field	Function
	1b - An error on channel 12 generates an error interrupt request
11 EEI11	Enable Error Interrupt 11 0b - An error on channel 11 does not generate an error interrupt 1b - An error on channel 11 generates an error interrupt request
10 EEI10	Enable Error Interrupt 10 0b - An error on channel 10 does not generate an error interrupt 1b - An error on channel 10 generates an error interrupt request
9 EEI9	Enable Error Interrupt 9 0b - An error on channel 9 does not generate an error interrupt 1b - An error on channel 9 generates an error interrupt request
8 EEI8	Enable Error Interrupt 8 0b - An error on channel 8 does not generate an error interrupt 1b - An error on channel 8 generates an error interrupt request
7 EEI7	Enable Error Interrupt 7 0b - An error on channel 7 does not generate an error interrupt 1b - An error on channel 7 generates an error interrupt request
6 EEI6	Enable Error Interrupt 6 0b - An error on channel 6 does not generate an error interrupt 1b - An error on channel 6 generates an error interrupt request
5 EEI5	Enable Error Interrupt 5 0b - An error on channel 5 does not generate an error interrupt 1b - An error on channel 5 generates an error interrupt request
4 EEI4	Enable Error Interrupt 4 0b - An error on channel 4 does not generate an error interrupt 1b - An error on channel 4 generates an error interrupt request
3 EEI3	Enable Error Interrupt 3 0b - An error on channel 3 does not generate an error interrupt 1b - An error on channel 3 generates an error interrupt request
2 EEI2	Enable Error Interrupt 2 0b - An error on channel 2 does not generate an error interrupt 1b - An error on channel 2 generates an error interrupt request
1 EEI1	Enable Error Interrupt 1 0b - An error on channel 1 does not generate an error interrupt 1b - An error on channel 1 generates an error interrupt request
0 EEI0	Enable Error Interrupt 0 0b - An error on channel 0 does not generate an error interrupt 1b - An error on channel 0 generates an error interrupt request

6.5.5.6 Clear Enable Error Interrupt (CEEI)

6.5.5.6.1 Offset

Register	Offset
CEEI	18h

6.5.5.6.2 Function

The CEEI provides a simple memory-mapped mechanism to write 0 to a given field in the EEI register to disable the error interrupt for a given channel. The data value on a register write causes the corresponding field in the EEI register to be written to 0. Writing 1 to the CAEE field provides a global clear to 0 function, forcing the EEI contents to be written to 0, disabling all DMA request inputs.

If the NOP field is written with 1, the command is ignored. This enables you to write 1 to a single, byte-wide register with a 32-bit write that does not affect the other registers addressed in the write. In such a case the other three bytes of the word must all have their NOP field set to 1 so that these registers are not affected by the write.

Reads of this register return all zeroes.

6.5.5.6.3 Diagram

Bits	7	6	5	4	3	2	1	0
R	0	0		0				
W	NOP	CAEE	0	CEEI				
Reset	0	0	0	0	0	0	0	0

6.5.5.6.4 Fields

Field	Function
7 NOP	No Op Enable 0b - Normal operation 1b - No operation, ignore the other fields in this register
6 CAEE	Clear All Enable Error Interrupts 0b - Write 0 only to the EEI field specified in the CEEI field 1b - Write 0 to all fields in EEI
5 —	Reserved
4-0 CEEI	Clear Enable Error Interrupt Writes 0 to the corresponding field in EEI

6.5.5.7 Set Enable Error Interrupt (SEEI)

6.5.5.7.1 Offset

Register	Offset
SEEI	19h

6.5.5.7.2 Function

The SEEI register provides a simple memory-mapped mechanism to write 1 to a given field in the EEI register to enable the error interrupt for a given channel. The data value on a register write causes the corresponding field in the EEI to be written to 1. Writing 1 to the SAEE field provides a global set to 1 function, forcing the entire EEI register contents to be written with 1.

If the NOP field is 1, the command is ignored. This enables you to write 1 to a single, byte-wide register with a 32-bit write that does not affect the other registers addressed in the write. In such a case the other three bytes of the word must all have their NOP field set to 1 so that these registers are not affected by the write.

Reads of this register return all zeroes.

6.5.5.7.3 Diagram

Bits	7	6	5	4	3	2	1	0
R	0	0		0				
W	NOP	SAEE	0	SEEI				
Reset	0	0	0	0	0	0	0	0

6.5.5.7.4 Fields

Field	Function
7 NOP	No Op Enable 0b - Normal operation 1b - No operation, ignore the other fields in this register
6 SAEE	Set All Enable Error Interrupts 0b - Write 1 only to the EEI field specified in the SEEI field 1b - Writes 1 to all fields in EEI
5 —	Reserved
4-0 SEEI	Set Enable Error Interrupt Writes 1 to the corresponding field in EEI

6.5.5.8 Clear Enable Request (CERQ)

6.5.5.8.1 Offset

Register	Offset
CERQ	1Ah

6.5.5.8.2 Function

The CERQ provides a simple memory-mapped mechanism to write 0 to a given field in the ERQ register to disable the DMA request for a given channel. The data value on a register write causes the corresponding field in the ERQ register to be written with 0. Setting the CAER field provides a global clear to 0 function, forcing the entire contents of the ERQ register to be written with 0, disabling all DMA request inputs.

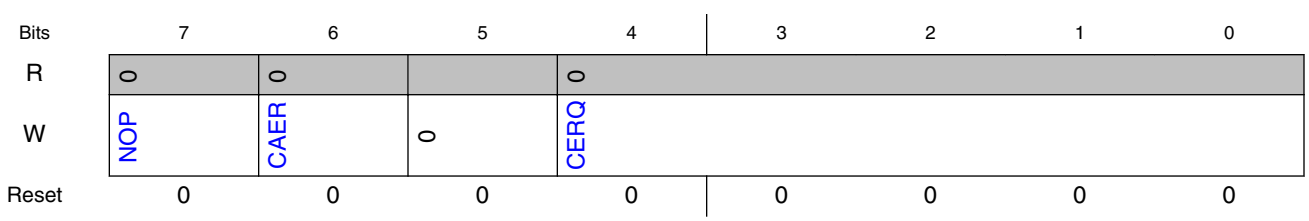
If the NOP field is 1, the command is ignored. This enables you to write 1 to a single, byte-wide register with a 32-bit write that does not affect the other registers addressed in the write. In such a case the other three bytes of the word must all have their NOP field written with 1 so that these registers are not affected by the write.

Reads of this register return all zeroes.

NOTE

Disable a channel's hardware service request at the source before writing 0 to the channel's ERQ field.

6.5.5.8.3 Diagram



6.5.5.8.4 Fields

Field	Function
7	No Op Enable 0b - Normal operation

Table continues on the next page...

Field	Function
NOP	1b - No operation, ignore the other fields in this register
6 CAER	Clear All Enable Requests 0b - Write 0 to only the ERQ field specified in the CERQ field 1b - Write 0 to all fields in ERQ
5 —	Reserved
4-0 CERQ	Clear Enable Request Writes 0 to the corresponding field in ERQ.

6.5.5.9 Set Enable Request (SERQ)

6.5.5.9.1 Offset

Register	Offset
SERQ	1Bh

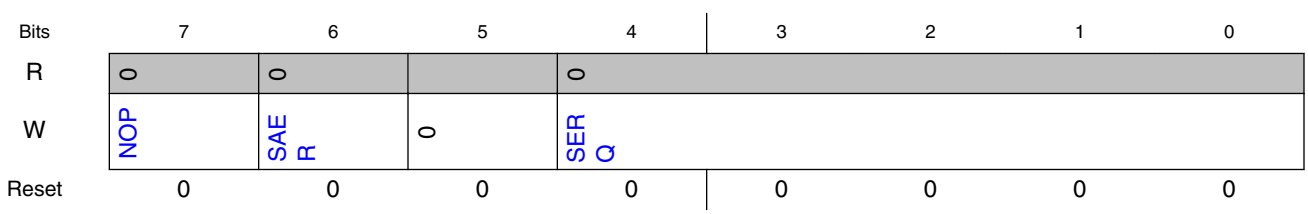
6.5.5.9.2 Function

The SERQ provides a simple memory-mapped mechanism to write 1 to a given field in the ERQ register to enable the DMA request for a given channel. The data value on a register write causes the corresponding field in the ERQ register to be set. Writing 1 to the SAER field provides a global set to 1 function, forcing the entire contents of ERQ register to be 1.

If the NOP field is 1, the command is ignored. This enables you to write 1 to a single, byte-wide register with a 32-bit write that does not affect the other registers addressed in the write. In such a case the other three bytes of the word must all have their NOP field written with 1 so that these registers are not affected by the write.

Reads of this register returns all zeroes.

6.5.5.9.3 Diagram



6.5.5.9.4 Fields

Field	Function
7 NOP	No Op Enable 0b - Normal operation 1b - No operation, ignore the other fields in this register
6 SAER	Set All Enable Requests 0b - Write 1 to only the ERQ field specified in the SERQ field 1b - Write 1 to all fields in ERQ
5 —	Reserved
4-0 SERQ	Set Enable Request Writes 1 to the corresponding field in ERQ.

6.5.5.10 Clear DONE Status Bit (CDNE)

6.5.5.10.1 Offset

Register	Offset
CDNE	1Ch

6.5.5.10.2 Function

The CDNE provides a simple memory-mapped mechanism to write 0 to the DONE field in the TCD of the given channel. The data value on a register write causes the DONE field in the corresponding TCD to be written with 0. Writing 1 to the CADN field provides a global clear function, forcing all DONE fields to be written with 0.

If the NOP field is 1, the command is ignored. This enables you to write 1 to a single, byte-wide register with a 32-bit write that does not affect the other registers addressed in the write. In such a case the other three bytes of the word must all have their NOP field written with 1 so that these registers are not affected by the write.

Reads of this register return all zeroes.

6.5.5.10.3 Diagram

Bits	7	6	5	4	3	2	1	0
R	0	0		0				
W	NOP	CADN	0	CDNE				
Reset	0	0	0	0	0	0	0	0

6.5.5.10.4 Fields

Field	Function
7 NOP	No Op Enable 0b - Normal operation 1b - No operation; all other fields in this register are ignored.
6 CADN	Clears All DONE fields 0b - Writes 0 to only the TCDn_CSR[DONE] field specified in the CDNE field 1b - Writes 0 to all bits in TCDn_CSR[DONE]
5 —	Reserved
4-0 CDNE	Clear DONE field Writes 0 to the corresponding field in TCDn_CSR[DONE]

6.5.5.11 Set START Bit (SSRT)

6.5.5.11.1 Offset

Register	Offset
SSRT	1Dh

6.5.5.11.2 Function

The SSRT register provides a simple memory-mapped mechanism to write 1 to the START field in the TCD of the given channel. The data value on a register write causes the START field in the corresponding TCD to be written with 1. Writing 1 to the SAST field provides a global set to 1 function, forcing all START fields to be written with 1.

If the NOP field is 1, the command is ignored. This enables you to write 1 to a single, byte-wide register with a 32-bit write that does not affect the other registers addressed in the write. In such a case the other three bytes of the word must all have their NOP field written with 1 so that these registers are not affected by the write.

Reads of this register return all zeroes.

6.5.5.11.3 Diagram

Bits	7	6	5	4	3	2	1	0
R	0	0		0				
W	NOP	SAST	0	SSRT				
Reset	0	0	0	0	0	0	0	0

6.5.5.11.4 Fields

Field	Function
7 NOP	No Op Enable 0b - Normal operation 1b - No operation; all other fields in this register are ignored.
6 SAST	Set All START fields (activates all channels) 0b - Write 1 to only the TCDn_CSR[START] field specified in the SSRT field 1b - Write 1 to all bits in TCDn_CSR[START]
5 —	Reserved
4-0 SSRT	Set START field Sets the corresponding field in TCDn_CSR[START]

6.5.5.12 Clear Error (CERR)

6.5.5.12.1 Offset

Register	Offset
CERR	1Eh

6.5.5.12.2 Function

The CERR provides a simple memory-mapped mechanism to write 0 to a given field in the ERR register to disable the error condition field for a given channel. The given value on a register write causes the corresponding field in the ERR register to be written with 0. Writing 1 to the CAEI field provides a global clear to 0 function, forcing the ERR register contents to be written with 0, clearing all channel error indicators. If the NOP field is 1, the command is ignored. This enables you to write multiple-byte registers as a 32-bit word. Reads of this register return all zeroes.

6.5.5.12.3 Diagram

Bits	7	6	5	4	3	2	1	0
R	0	0		0				
W	NOP	CAEI	0	CERR				
Reset	0	0	0	0	0	0	0	0

6.5.5.12.4 Fields

Field	Function
7 NOP	No Op Enable 0b - Normal operation 1b - No operation; all other fields in this register are ignored.
6 CAEI	Clear All Error Indicators 0b - Write 0 to only the ERR field specified in the CERR field 1b - Write 0 to all fields in ERR
5 —	Reserved
4-0 CERR	Clear Error Indicator Writes 0 to the corresponding field in ERR

6.5.5.13 Clear Interrupt Request (CINT)

6.5.5.13.1 Offset

Register	Offset
CINT	1Fh

6.5.5.13.2 Function

The CINT register provides a simple, memory-mapped mechanism to clear a given field in the INT register to disable the interrupt request for a given channel. The given value on a register write causes the corresponding field in the INT register to be cleared. Setting the CAIR field provides a global clear function, forcing the entire contents of the INT to be cleared, disabling all DMA interrupt requests.

If the NOP field is 1, the command is ignored. This enables you to set a single, byte-wide register with a 32-bit write that does not affect the other registers addressed in the write. In such a case the other three bytes of the word would all have their NOP field set to 1 so that these registers are not affected by the write.

Reads of this register return all zeroes.

6.5.5.13.3 Diagram

Bits	7	6	5	4	3	2	1	0
R	0	0		0				
W	NOP	CAIR	0	CINT				
Reset	0	0	0	0	0	0	0	0

6.5.5.13.4 Fields

Field	Function
7 NOP	No Op Enable 0b - Normal operation 1b - No operation; all other fields in this register are ignored.
6 CAIR	Clear All Interrupt Requests 0b - Clear only the INT field specified in the CINT field 1b - Clear all bits in INT
5 —	Reserved
4-0 CINT	Clear Interrupt Request Clears the corresponding field in INT

6.5.5.14 Interrupt Request (INT)

6.5.5.14.1 Offset

Register	Offset
INT	24h

6.5.5.14.2 Function

The INT register provides a bit map for the 32 channels signaling the presence of an interrupt request for each channel. Depending on the appropriate bit setting in the transfer-control descriptors, the eDMA engine generates an interrupt on data transfer completion. The outputs of this register are directly routed to the interrupt controller. During the interrupt-service routine associated with any given channel, it is the software's responsibility to write 0 to the appropriate bit, negating the interrupt request. Typically, a write to the CINT register in the interrupt service routine is used for this purpose.

The state of any given channel's interrupt request is directly affected by writes to this register; it is also affected by writes to the CINT register. On writes to INT, a 1 in any bit position clears the corresponding channel's interrupt request. A 0 in any bit position has no effect on the corresponding channel's current interrupt status. The CINT register is provided so the interrupt request for a single channel can easily be cleared without the need to perform a read-modify-write sequence to the INT register.

6.5.5.14.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	INT31	INT30	INT29	INT28	INT27	INT26	INT25	INT24	INT23	INT22	INT21	INT20	INT19	INT18	INT17	INT16
W	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	INT15	INT14	INT13	INT12	INT11	INT10	INT9	INT8	INT7	INT6	INT5	INT4	INT3	INT2	INT1	INT0
W	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

6.5.5.14.4 Fields

Field	Function
31 INT31	Interrupt Request 31 0b - The interrupt request for channel 31 is cleared 1b - The interrupt request for channel 31 is active
30 INT30	Interrupt Request 30 0b - The interrupt request for channel 30 is cleared 1b - The interrupt request for channel 30 is active
29 INT29	Interrupt Request 29 0b - The interrupt request for channel 29 is cleared 1b - The interrupt request for channel 29 is active
28 INT28	Interrupt Request 28 0b - The interrupt request for channel 28 is cleared 1b - The interrupt request for channel 28 is active
27 INT27	Interrupt Request 27 0b - The interrupt request for channel 27 is cleared 1b - The interrupt request for channel 27 is active
26 INT26	Interrupt Request 26 0b - The interrupt request for channel 26 is cleared 1b - The interrupt request for channel 26 is active
25 INT25	Interrupt Request 25 0b - The interrupt request for channel 25 is cleared 1b - The interrupt request for channel 25 is active
24 INT24	Interrupt Request 24 0b - The interrupt request for channel 24 is cleared 1b - The interrupt request for channel 24 is active
23 INT23	Interrupt Request 23 0b - The interrupt request for channel 23 is cleared 1b - The interrupt request for channel 23 is active
22 INT22	Interrupt Request 22 0b - The interrupt request for channel 22 is cleared 1b - The interrupt request for channel 22 is active
21 INT21	Interrupt Request 21 0b - The interrupt request for channel 21 is cleared 1b - The interrupt request for channel 21 is active
20 INT20	Interrupt Request 20 0b - The interrupt request for channel 20 is cleared 1b - The interrupt request for channel 20 is active
19 INT19	Interrupt Request 19 0b - The interrupt request for channel 19 is cleared 1b - The interrupt request for channel 19 is active
18 INT18	Interrupt Request 18 0b - The interrupt request for channel 18 is cleared 1b - The interrupt request for channel 18 is active
17 INT17	Interrupt Request 17 0b - The interrupt request for channel 17 is cleared 1b - The interrupt request for channel 17 is active
16 INT16	Interrupt Request 16 0b - The interrupt request for channel 16 is cleared 1b - The interrupt request for channel 16 is active

Table continues on the next page...

Field	Function
15 INT15	Interrupt Request 15 0b - The interrupt request for channel 15 is cleared 1b - The interrupt request for channel 15 is active
14 INT14	Interrupt Request 14 0b - The interrupt request for channel 14 is cleared 1b - The interrupt request for channel 14 is active
13 INT13	Interrupt Request 13 0b - The interrupt request for channel 13 is cleared 1b - The interrupt request for channel 13 is active
12 INT12	Interrupt Request 12 0b - The interrupt request for channel 12 is cleared 1b - The interrupt request for channel 12 is active
11 INT11	Interrupt Request 11 0b - The interrupt request for channel 11 is cleared 1b - The interrupt request for channel 11 is active
10 INT10	Interrupt Request 10 0b - The interrupt request for channel 10 is cleared 1b - The interrupt request for channel 10 is active
9 INT9	Interrupt Request 9 0b - The interrupt request for channel 9 is cleared 1b - The interrupt request for channel 9 is active
8 INT8	Interrupt Request 8 0b - The interrupt request for channel 8 is cleared 1b - The interrupt request for channel 8 is active
7 INT7	Interrupt Request 7 0b - The interrupt request for channel 7 is cleared 1b - The interrupt request for channel 7 is active
6 INT6	Interrupt Request 6 0b - The interrupt request for channel 6 is cleared 1b - The interrupt request for channel 6 is active
5 INT5	Interrupt Request 5 0b - The interrupt request for channel 5 is cleared 1b - The interrupt request for channel 5 is active
4 INT4	Interrupt Request 4 0b - The interrupt request for channel 4 is cleared 1b - The interrupt request for channel 4 is active
3 INT3	Interrupt Request 3 0b - The interrupt request for channel 3 is cleared 1b - The interrupt request for channel 3 is active
2 INT2	Interrupt Request 2 0b - The interrupt request for channel 2 is cleared 1b - The interrupt request for channel 2 is active
1 INT1	Interrupt Request 1 0b - The interrupt request for channel 1 is cleared 1b - The interrupt request for channel 1 is active
0 INT0	Interrupt Request 0 0b - The interrupt request for channel 0 is cleared 1b - The interrupt request for channel 0 is active

6.5.5.15 Error (ERR)

6.5.5.15.1 Offset

Register	Offset
ERR	2Ch

6.5.5.15.2 Function

The ERR register provides a bit map for the 32 channels, signaling the presence of an error for each channel. The eDMA engine signals the occurrence of an error condition by setting the appropriate field in this register. The outputs of this register are enabled by the contents of the EEI register, then logically summed across groups of 16 and 32 channels to form several group error interrupt requests, which are then routed to the interrupt controller. During the execution of the interrupt service routine associated with any DMA errors, it is software's responsibility to reset the appropriate bit to 0, negating the error-interrupt request. Typically, a write to the CERR in the interrupt service routine is used for this purpose. The normal DMA channel completion indicators (setting the TCD DONE field to 1 and the possible generation of an interrupt request) are not affected when an error is detected.

The contents of this register can also be polled because a non-zero value indicates the presence of a channel error regardless of the state of the EEI fields. The state of any given channel's error indicators is affected by writes to this register; it is also affected by writes to the CERR. On writes to the ERR, a 1 in any bit position clears the corresponding channel's error status. A 0 in any bit position has no effect on the corresponding channel's current error status. The CERR is provided so the error indicator for a single channel can easily be reset to 0.

6.5.5.15.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	ERR31	ERR30	ERR29	ERR28	ERR27	ERR26	ERR25	ERR24	ERR23	ERR22	ERR21	ERR20	ERR19	ERR18	ERR17	ERR16
W	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	ERR15	ERR14	ERR13	ERR12	ERR11	ERR10	ERR9	ERR8	ERR7	ERR6	ERR5	ERR4	ERR3	ERR2	ERR1	ERR0
W	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

6.5.5.15.4 Fields

Field	Function
31 ERR31	Error In Channel 31 0b - No error in this channel has occurred 1b - An error in this channel has occurred
30 ERR30	Error In Channel 30 0b - No error in this channel has occurred 1b - An error in this channel has occurred
29 ERR29	Error In Channel 29 0b - No error in this channel has occurred 1b - An error in this channel has occurred
28 ERR28	Error In Channel 28 0b - No error in this channel has occurred 1b - An error in this channel has occurred
27 ERR27	Error In Channel 27 0b - No error in this channel has occurred 1b - An error in this channel has occurred
26 ERR26	Error In Channel 26 0b - No error in this channel has occurred 1b - An error in this channel has occurred
25 ERR25	Error In Channel 25 0b - No error in this channel has occurred 1b - An error in this channel has occurred
24 ERR24	Error In Channel 24 0b - No error in this channel has occurred 1b - An error in this channel has occurred
23 ERR23	Error In Channel 23 0b - No error in this channel has occurred 1b - An error in this channel has occurred

Table continues on the next page...

Field	Function
22 ERR22	Error In Channel 22 0b - No error in this channel has occurred 1b - An error in this channel has occurred
21 ERR21	Error In Channel 21 0b - No error in this channel has occurred 1b - An error in this channel has occurred
20 ERR20	Error In Channel 20 0b - No error in this channel has occurred 1b - An error in this channel has occurred
19 ERR19	Error In Channel 19 0b - No error in this channel has occurred 1b - An error in this channel has occurred
18 ERR18	Error In Channel 18 0b - No error in this channel has occurred 1b - An error in this channel has occurred
17 ERR17	Error In Channel 17 0b - No error in this channel has occurred 1b - An error in this channel has occurred
16 ERR16	Error In Channel 16 0b - No error in this channel has occurred 1b - An error in this channel has occurred
15 ERR15	Error In Channel 15 0b - No error in this channel has occurred 1b - An error in this channel has occurred
14 ERR14	Error In Channel 14 0b - No error in this channel has occurred 1b - An error in this channel has occurred
13 ERR13	Error In Channel 13 0b - No error in this channel has occurred 1b - An error in this channel has occurred
12 ERR12	Error In Channel 12 0b - No error in this channel has occurred 1b - An error in this channel has occurred
11 ERR11	Error In Channel 11 0b - No error in this channel has occurred 1b - An error in this channel has occurred
10 ERR10	Error In Channel 10 0b - No error in this channel has occurred 1b - An error in this channel has occurred
9 ERR9	Error In Channel 9 0b - No error in this channel has occurred 1b - An error in this channel has occurred
8 ERR8	Error In Channel 8 0b - No error in this channel has occurred 1b - An error in this channel has occurred
7 ERR7	Error In Channel 7 0b - No error in this channel has occurred 1b - An error in this channel has occurred
6 ERR6	Error In Channel 6 0b - No error in this channel has occurred

Table continues on the next page...

Field	Function
	1b - An error in this channel has occurred
5 ERR5	Error In Channel 5 0b - No error in this channel has occurred 1b - An error in this channel has occurred
4 ERR4	Error In Channel 4 0b - No error in this channel has occurred 1b - An error in this channel has occurred
3 ERR3	Error In Channel 3 0b - No error in this channel has occurred 1b - An error in this channel has occurred
2 ERR2	Error In Channel 2 0b - No error in this channel has occurred 1b - An error in this channel has occurred
1 ERR1	Error In Channel 1 0b - No error in this channel has occurred 1b - An error in this channel has occurred
0 ERR0	Error In Channel 0 0b - No error in this channel has occurred 1b - An error in this channel has occurred

6.5.5.16 Hardware Request Status (HRS)

6.5.5.16.1 Offset

Register	Offset
HRS	34h

6.5.5.16.2 Function

The HRS register provides a bit map for the DMA channels, signaling the presence of a hardware request for each channel. The hardware request status bits reflect the current state of the register and qualified (via the ERQ fields) DMA request signals, as seen by the DMA's arbitration logic. This view into the hardware request signals may be used for debug purposes.

NOTE

These bits reflect the state of the request as seen by the arbitration logic. Therefore, this status is affected by the ERQ bits.

Each HRS field for its respective channel is 1 when a hardware request is present on the channel. After the request is completed and channel is free, the HRS field is automatically changed to 0 by hardware.

6.5.5.16.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	HRS31	HRS30	HRS29	HRS28	HRS27	HRS26	HRS25	HRS24	HRS23	HRS22	HRS21	HRS20	HRS19	HRS18	HRS17	HRS16
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	HRS15	HRS14	HRS13	HRS12	HRS11	HRS10	HRS 9	HRS 8	HRS 7	HRS 6	HRS 5	HRS 4	HRS 3	HRS 2	HRS 1	HRS 0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

6.5.5.16.4 Fields

Field	Function
31 HRS31	Hardware Request Status Channel 31 0b - A hardware service request for channel 31 is not present 1b - A hardware service request for channel 31 is present
30 HRS30	Hardware Request Status Channel 30 0b - A hardware service request for channel 30 is not present 1b - A hardware service request for channel 30 is present
29 HRS29	Hardware Request Status Channel 29 0b - A hardware service request for channel 29 is not preset 1b - A hardware service request for channel 29 is present
28 HRS28	Hardware Request Status Channel 28 0b - A hardware service request for channel 28 is not present 1b - A hardware service request for channel 28 is present
27 HRS27	Hardware Request Status Channel 27 0b - A hardware service request for channel 27 is not present 1b - A hardware service request for channel 27 is present
26 HRS26	Hardware Request Status Channel 26 0b - A hardware service request for channel 26 is not present 1b - A hardware service request for channel 26 is present
25 HRS25	Hardware Request Status Channel 25 0b - A hardware service request for channel 25 is not present 1b - A hardware service request for channel 25 is present
24 HRS24	Hardware Request Status Channel 24 0b - A hardware service request for channel 24 is not present 1b - A hardware service request for channel 24 is present

Table continues on the next page...

Field	Function
23 HRS23	Hardware Request Status Channel 23 0b - A hardware service request for channel 23 is not present 1b - A hardware service request for channel 23 is present
22 HRS22	Hardware Request Status Channel 22 0b - A hardware service request for channel 22 is not present 1b - A hardware service request for channel 22 is present
21 HRS21	Hardware Request Status Channel 21 0b - A hardware service request for channel 21 is not present 1b - A hardware service request for channel 21 is present
20 HRS20	Hardware Request Status Channel 20 0b - A hardware service request for channel 20 is not present 1b - A hardware service request for channel 20 is present
19 HRS19	Hardware Request Status Channel 19 0b - A hardware service request for channel 19 is not present 1b - A hardware service request for channel 19 is present
18 HRS18	Hardware Request Status Channel 18 0b - A hardware service request for channel 18 is not present 1b - A hardware service request for channel 18 is present
17 HRS17	Hardware Request Status Channel 17 0b - A hardware service request for channel 17 is not present 1b - A hardware service request for channel 17 is present
16 HRS16	Hardware Request Status Channel 16 0b - A hardware service request for channel 16 is not present 1b - A hardware service request for channel 16 is present
15 HRS15	Hardware Request Status Channel 15 0b - A hardware service request for channel 15 is not present 1b - A hardware service request for channel 15 is present
14 HRS14	Hardware Request Status Channel 14 0b - A hardware service request for channel 14 is not present 1b - A hardware service request for channel 14 is present
13 HRS13	Hardware Request Status Channel 13 0b - A hardware service request for channel 13 is not present 1b - A hardware service request for channel 13 is present
12 HRS12	Hardware Request Status Channel 12 0b - A hardware service request for channel 12 is not present 1b - A hardware service request for channel 12 is present
11 HRS11	Hardware Request Status Channel 11 0b - A hardware service request for channel 11 is not present 1b - A hardware service request for channel 11 is present
10 HRS10	Hardware Request Status Channel 10 0b - A hardware service request for channel 10 is not present 1b - A hardware service request for channel 10 is present
9 HRS9	Hardware Request Status Channel 9 0b - A hardware service request for channel 9 is not present 1b - A hardware service request for channel 9 is present
8 HRS8	Hardware Request Status Channel 8 0b - A hardware service request for channel 8 is not present 1b - A hardware service request for channel 8 is present
7 HRS7	Hardware Request Status Channel 7 0b - A hardware service request for channel 7 is not present

Table continues on the next page...

Field	Function
	1b - A hardware service request for channel 7 is present
6 HRS6	Hardware Request Status Channel 6 0b - A hardware service request for channel 6 is not present 1b - A hardware service request for channel 6 is present
5 HRS5	Hardware Request Status Channel 5 0b - A hardware service request for channel 5 is not present 1b - A hardware service request for channel 5 is present
4 HRS4	Hardware Request Status Channel 4 0b - A hardware service request for channel 4 is not present 1b - A hardware service request for channel 4 is present
3 HRS3	Hardware Request Status Channel 3 0b - A hardware service request for channel 3 is not present 1b - A hardware service request for channel 3 is present
2 HRS2	Hardware Request Status Channel 2 0b - A hardware service request for channel 2 is not present 1b - A hardware service request for channel 2 is present
1 HRS1	Hardware Request Status Channel 1 0b - A hardware service request for channel 1 is not present 1b - A hardware service request for channel 1 is present
0 HRS0	Hardware Request Status Channel 0 0b - A hardware service request for channel 0 is not present 1b - A hardware service request for channel 0 is present

6.5.5.17 Enable Asynchronous Request in Stop (EARS)

6.5.5.17.1 Offset

Register	Offset
EARS	44h

6.5.5.17.2 Function

The EARS register is used to enable or disable the DMA requests in Enable Request (ERQ) by AND'ing the bits of these two registers.

6.5.5.17.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	EDREQ_31	EDREQ_30	EDREQ_29	EDREQ_28	EDREQ_27	EDREQ_26	EDREQ_25	EDREQ_24	EDREQ_23	EDREQ_22	EDREQ_21	EDREQ_20	EDREQ_19	EDREQ_18	EDREQ_17	EDREQ_16
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	EDREQ_15	EDREQ_14	EDREQ_13	EDREQ_12	EDREQ_11	EDREQ_10	EDREQ_9	EDREQ_8	EDREQ_7	EDREQ_6	EDREQ_5	EDREQ_4	EDREQ_3	EDREQ_2	EDREQ_1	EDREQ_0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

6.5.5.17.4 Fields

Field	Function
31 EDREQ_31	Enable asynchronous DMA request in stop mode for channel 31. 0b - Disable asynchronous DMA request for channel 31 1b - Enable asynchronous DMA request for channel 31
30 EDREQ_30	Enable asynchronous DMA request in stop mode for channel 30. 0b - Disable asynchronous DMA request for channel 30 1b - Enable asynchronous DMA request for channel 30
29 EDREQ_29	Enable asynchronous DMA request in stop mode for channel 29. 0b - Disable asynchronous DMA request for channel 29 1b - Enable asynchronous DMA request for channel 29
28 EDREQ_28	Enable asynchronous DMA request in stop mode for channel 28. 0b - Disable asynchronous DMA request for channel 28 1b - Enable asynchronous DMA request for channel 28
27 EDREQ_27	Enable asynchronous DMA request in stop mode for channel 27. 0b - Disable asynchronous DMA request for channel 27 1b - Enable asynchronous DMA request for channel 27
26 EDREQ_26	Enable asynchronous DMA request in stop mode for channel 26. 0b - Disable asynchronous DMA request for channel 26 1b - Enable asynchronous DMA request for channel 26
25 EDREQ_25	Enable asynchronous DMA request in stop mode for channel 25. 0b - Disable asynchronous DMA request for channel 25 1b - Enable asynchronous DMA request for channel 25
24 EDREQ_24	Enable asynchronous DMA request in stop mode for channel 24. 0b - Disable asynchronous DMA request for channel 24 1b - Enable asynchronous DMA request for channel 24
23 EDREQ_23	Enable asynchronous DMA request in stop mode for channel 23. 0b - Disable asynchronous DMA request for channel 23 1b - Enable asynchronous DMA request for channel 23
22	Enable asynchronous DMA request in stop mode for channel 22. 0b - Disable asynchronous DMA request for channel 22

Table continues on the next page...

Field	Function
EDREQ_22	1b - Enable asynchronous DMA request for channel 22
21 EDREQ_21	Enable asynchronous DMA request in stop mode for channel 21. 0b - Disable asynchronous DMA request for channel 21 1b - Enable asynchronous DMA request for channel 21
20 EDREQ_20	Enable asynchronous DMA request in stop mode for channel 20. 0b - Disable asynchronous DMA request for channel 20 1b - Enable asynchronous DMA request for channel 20
19 EDREQ_19	Enable asynchronous DMA request in stop mode for channel 19. 0b - Disable asynchronous DMA request for channel 19 1b - Enable asynchronous DMA request for channel 19
18 EDREQ_18	Enable asynchronous DMA request in stop mode for channel 18. 0b - Disable asynchronous DMA request for channel 18 1b - Enable asynchronous DMA request for channel 18
17 EDREQ_17	Enable asynchronous DMA request in stop mode for channel 17. 0b - Disable asynchronous DMA request for channel 17 1b - Enable asynchronous DMA request for channel 17
16 EDREQ_16	Enable asynchronous DMA request in stop mode for channel 16. 0b - Disable asynchronous DMA request for channel 16 1b - Enable asynchronous DMA request for channel 16
15 EDREQ_15	Enable asynchronous DMA request in stop mode for channel 15. 0b - Disable asynchronous DMA request for channel 15 1b - Enable asynchronous DMA request for channel 15
14 EDREQ_14	Enable asynchronous DMA request in stop mode for channel 14. 0b - Disable asynchronous DMA request for channel 14 1b - Enable asynchronous DMA request for channel 14
13 EDREQ_13	Enable asynchronous DMA request in stop mode for channel 13. 0b - Disable asynchronous DMA request for channel 13 1b - Enable asynchronous DMA request for channel 13
12 EDREQ_12	Enable asynchronous DMA request in stop mode for channel 12. 0b - Disable asynchronous DMA request for channel 12 1b - Enable asynchronous DMA request for channel 12
11 EDREQ_11	Enable asynchronous DMA request in stop mode for channel 11. 0b - Disable asynchronous DMA request for channel 11 1b - Enable asynchronous DMA request for channel 11
10 EDREQ_10	Enable asynchronous DMA request in stop mode for channel 10. 0b - Disable asynchronous DMA request for channel 10 1b - Enable asynchronous DMA request for channel 10
9 EDREQ_9	Enable asynchronous DMA request in stop mode for channel 9. 0b - Disable asynchronous DMA request for channel 9 1b - Enable asynchronous DMA request for channel 9
8 EDREQ_8	Enable asynchronous DMA request in stop mode for channel 8. 0b - Disable asynchronous DMA request for channel 8 1b - Enable asynchronous DMA request for channel 8
7 EDREQ_7	Enable asynchronous DMA request in stop mode for channel 7. 0b - Disable asynchronous DMA request for channel 7 1b - Enable asynchronous DMA request for channel 7
6 EDREQ_6	Enable asynchronous DMA request in stop mode for channel 6. 0b - Disable asynchronous DMA request for channel 6 1b - Enable asynchronous DMA request for channel 6
5	Enable asynchronous DMA request in stop mode for channel 5.

Table continues on the next page...

Field	Function
EDREQ_5	0b - Disable asynchronous DMA request for channel 5 1b - Enable asynchronous DMA request for channel 5
4 EDREQ_4	Enable asynchronous DMA request in stop mode for channel 4. 0b - Disable asynchronous DMA request for channel 4 1b - Enable asynchronous DMA request for channel 4
3 EDREQ_3	Enable asynchronous DMA request in stop mode for channel 3. 0b - Disable asynchronous DMA request for channel 3 1b - Enable asynchronous DMA request for channel 3
2 EDREQ_2	Enable asynchronous DMA request in stop mode for channel 2. 0b - Disable asynchronous DMA request for channel 2 1b - Enable asynchronous DMA request for channel 2
1 EDREQ_1	Enable asynchronous DMA request in stop mode for channel 1. 0b - Disable asynchronous DMA request for channel 1 1b - Enable asynchronous DMA request for channel 1
0 EDREQ_0	Enable asynchronous DMA request in stop mode for channel 0. 0b - Disable asynchronous DMA request for channel 0 1b - Enable asynchronous DMA request for channel 0

6.5.5.18 Channel Priority (DCHPRI0 - DCHPRI31)

6.5.5.18.1 Offset

For n = 0 to 31:

Register	Offset
DCHPRIn	100h + (n + 3 - 2 × (n mod 4))

6.5.5.18.2 Function

When fixed-priority channel arbitration is enabled (CR[ERCA] = 0), the contents of these registers define the unique priorities associated with each channel within a group. The channel priorities are evaluated by numeric value; for example, 0 is the lowest priority, 1 is the next higher priority, then 2, 3, and so on. Software must program the channel priorities with unique values; otherwise, a configuration error is reported. The range of the priority value is limited to the values of 0 through 15. When read, the GRPPRI bits of the DCHPRIn register reflect the current priority level of the group of channels in which the corresponding channel resides. GRPPRI bits are not affected by writes to the DCHPRIn registers. The group priority is assigned in the DMA control register.

6.5.5.18.3 Diagram

Bits	7	6	5	4	3	2	1	0
R	ECP	DPA	GRPPRI		CHPRI			
W								
Reset	See Register reset values.							

6.5.5.18.4 Register reset values

Register	Reset value
DCHPRI0	00h
DCHPRI1	01h
DCHPRI2	02h
DCHPRI3	03h
DCHPRI4	04h
DCHPRI5	05h
DCHPRI6	06h
DCHPRI7	07h
DCHPRI8	08h
DCHPRI9	09h
DCHPRI10	0Ah
DCHPRI11	0Bh
DCHPRI12	0Ch
DCHPRI13	0Dh
DCHPRI14	0Eh
DCHPRI15	0Fh
DCHPRI16	10h
DCHPRI17	11h
DCHPRI18	12h
DCHPRI19	13h
DCHPRI20	14h
DCHPRI21	15h
DCHPRI22	16h
DCHPRI23	17h
DCHPRI24	18h
DCHPRI25	19h
DCHPRI26	1Ah
DCHPRI27	1Bh
DCHPRI28	1Ch
DCHPRI29	1Dh

Table continues on the next page...

Register	Reset value
DCHPRI30	1Eh
DCHPRI31	1Fh

6.5.5.18.5 Fields

Field	Function
7 ECP	Enable Channel Preemption. This field resets to 0. 0b - Channel n cannot be suspended by a higher priority channel's service request 1b - Channel n can be temporarily suspended by the service request of a higher priority channel
6 DPA	Disable Preempt Ability. This field resets to 0. 0b - Channel n can suspend a lower priority channel 1b - Channel n cannot suspend any channel, regardless of channel priority
5-4 GRPPRI	Channel n Current Group Priority Group priority assigned to this channel group when fixed-priority arbitration is enabled. This field is read-only; writes are ignored.
3-0 CHPRI	Channel n Arbitration Priority Channel priority when fixed-priority arbitration is enabled.

6.5.5.19 TCD Source Address (TCD0_SADDR - TCD31_SADDR)

6.5.5.19.1 Offset

For n = 0 to 31:

Register	Offset
TCDn_SADDR	1000h + (n × 20h)

6.5.5.19.2 Function

This register contains the source address of the transfer.

6.5.5.19.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	SADDR															
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	SADDR															
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

6.5.5.19.4 Fields

Field	Function
31-0 SADDR	Source Address Memory address pointing to the source data.

6.5.5.20 TCD Signed Source Address Offset (TCD0_SOFF - TCD31_SOFF)

6.5.5.20.1 Offset

For n = 0 to 31:

Register	Offset
TCDn_SOFF	1004h + (n × 20h)

6.5.5.20.2 Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	SOFF															
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

6.5.5.20.3 Fields

Field	Function
15-0 SOFF	Source address signed offset Sign-extended offset applied to the current source address to form the next-state value as each source read is completed.

6.5.5.21 TCD Transfer Attributes (TCD0_ATTR - TCD31_ATTR)

6.5.5.21.1 Offset

For n = 0 to 31:

Register	Offset
TCDn_ATTR	1006h + (n × 20h)

6.5.5.21.2 Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	SMOD					SSIZE			DMOD					DSIZE		
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

6.5.5.21.3 Fields

Field	Function
15-11 SMOD	Source Address Modulo Any non-zero value in this field defines a specific address range specified to be the value after SADDR + SOFF calculation is performed on the original register value. Setting this field provides the ability to implement a circular data queue easily. For data queues requiring power-of-2 size bytes, the queue should start at a 0-modulo-size address and the SMOD field should be set to the appropriate value for the queue, freezing the desired number of upper address bits. The value programmed into this field specifies the number of lower address bits allowed to change. For a circular queue application, the SOFF is typically set to the transfer size to implement post-increment addressing with the SMOD function constraining the addresses to a 0-modulo-size range. 00000b - Source address modulo feature is disabled 00001-11111b - Value defines address range used to set up circular data queue
10-8 SSIZE	Source data transfer size **NOTE:** 1. Using a reserved value causes a configuration error. 2. The eDMA defaults to privileged data access for all transactions.

Table continues on the next page...

Field	Function
	000b - 8-bit 001b - 16-bit 010b - 32-bit 011b - 64-bit 100b - Reserved 101b - 32-byte burst (4 beats of 64 bits) 110b - Reserved 111b - Reserved
7-3 DMOD	Destination Address Modulo See the SMOD definition.
2-0 DSIZE	Destination data transfer size See the SSIZE definition.

6.5.5.22 TCD Minor Byte Count (Minor Loop Mapping Disabled) (TCD0_NBYTES_MLNO - TCD31_NBYTES_MLNO)

6.5.5.22.1 Offset

For n = 0 to 31:

Register	Offset
TCDn_NBYTES_MLNO	1008h + (n × 20h)

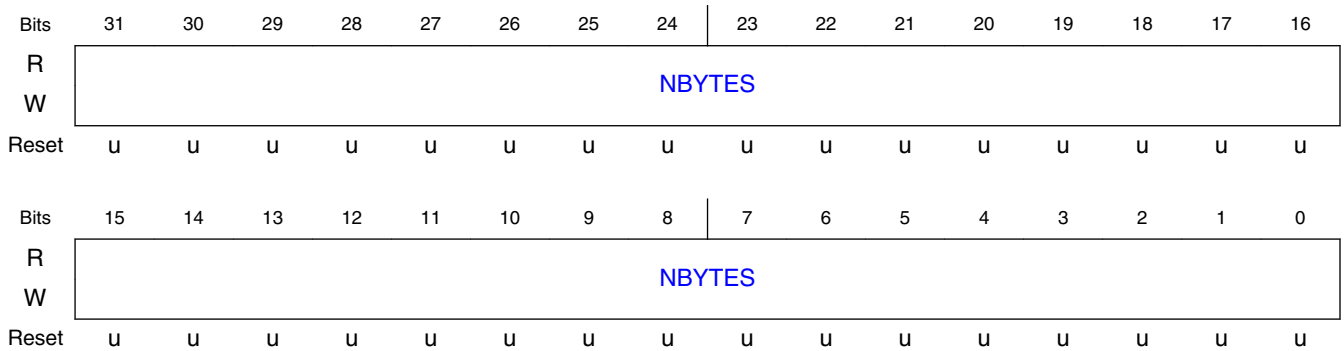
6.5.5.22.2 Function

This register, or one of the next two registers (TCD_NBYTES_MLOFFNO, TCD_NBYTES_MLOFFYES), that defines the number of bytes to transfer per request. Which register to use depends on whether minor loop mapping is disabled, is enabled but not used for this channel, or is enabled and used.

TCD word 2 is defined as follows if minor loop mapping is disabled (CR[EMLM] = 0).

If minor loop mapping is enabled, see the TCD_NBYTES_MLOFFNO and TCD_NBYTES_MLOFFYES register descriptions for the definition of TCD word 2.

6.5.5.22.3 Diagram



6.5.5.22.4 Fields

Field	Function
31-0 NBYTES	Minor Byte Transfer Count Number of bytes to be transferred in each service request of the channel. As a channel activates, the appropriate TCD contents load into the eDMA engine, and the appropriate reads and writes are performed until the minor byte transfer count has transferred. This is an indivisible operation and cannot be halted. It can, however, be stalled by using the bandwidth control field, or via preemption. After the minor count is exhausted, the SADDR and DADDR values are written back into the TCD memory, and the major iteration count is decremented and restored to the TCD memory. If the major iteration count is completed, additional processing is performed. **NOTE:** An NBYTES value of 0x0000_0000 is interpreted as a 4 GB transfer.

6.5.5.23 TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (TCD0_NBYTES_MLOFFNO - TCD31_NBYTES_MLOFFNO)

6.5.5.23.1 Offset

For n = 0 to 31:

Register	Offset
TCDn_NBYTES_MLOFFNO	1008h + (n × 20h)

6.5.5.23.2 Function

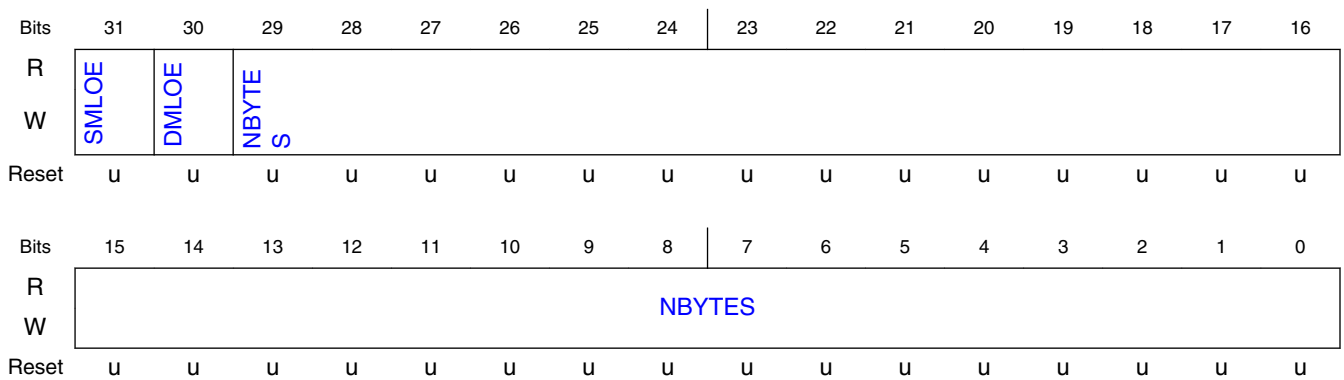
One of three registers (this register, TCD_NBYTES_MLNO, or TCD_NBYTES_MLOFFYES), that defines the number of bytes to transfer per request. Which register to use depends on whether minor loop mapping is disabled, is enabled but not used for this channel, or is enabled and used.

TCD word 2 is defined as follows if:

- Minor loop mapping is enabled (CR[EMLM] = 1) and
- SMLOE = 0 and DMLOE = 0

If minor loop mapping is enabled and SMLOE = 1 or DMLOE = 1, refer to the TCD_NBYTES_MLOFFYES register description. If minor loop mapping is disabled, refer to the TCD_NBYTES_MLNO register description.

6.5.5.23.3 Diagram



6.5.5.23.4 Fields

Field	Function
31 SMLOE	Source Minor Loop Offset Enable Specifies whether the minor loop offset is applied to the source address when the minor loop completes. 0b - The minor loop offset is not applied to the SADDR 1b - The minor loop offset is applied to the SADDR
30 DMLOE	Destination Minor Loop Offset Enable Specifies whether the minor loop offset is applied to the destination address when the minor loop completes. 0b - The minor loop offset is not applied to the DADDR 1b - The minor loop offset is applied to the DADDR
29-0 NBYTES	Minor Byte Transfer Count Number of bytes to be transferred in each service request of the channel.

Field	Function
	As a channel activates, the appropriate TCD contents load into the eDMA engine, and the appropriate reads and writes are performed until the minor byte transfer count has transferred. This is an indivisible operation and cannot be halted. It can, however, be stalled by using the bandwidth control field, or via preemption. After the minor count is exhausted, the SADDR and DADDR values are written back into the TCD memory, and the major iteration count is decremented and restored to the TCD memory. If the major iteration count is completed, additional processing is performed.

6.5.5.24 TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (TCD0_NBYTES_MLOFFYES - TCD31_NBYTES_MLOFFYES)

6.5.5.24.1 Offset

For n = 0 to 31:

Register	Offset
TCDn_NBYTES_MLOFFYES	1008h + (n × 20h)

6.5.5.24.2 Function

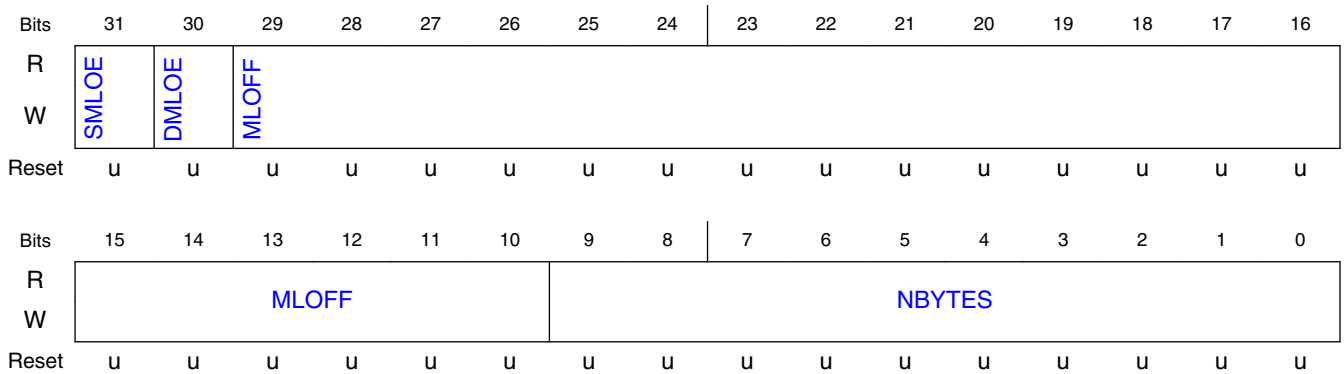
One of three registers (this register, TCD_NBYTES_MLNO, or TCD_NBYTES_MLOFFNO), that defines the number of bytes to transfer per request. Which register to use depends on whether minor loop mapping is disabled, is enabled but not used for this channel, or is enabled and used.

TCD word 2 is defined as follows if:

- Minor loop mapping is enabled (CR[EMLM] = 1) and
- Minor loop offset is enabled (SMLOE or DMLOE = 1)

If minor loop mapping is enabled and SMLOE = 0 and DMLOE = 0, refer to the TCD_NBYTES_MLOFFNO register description. If minor loop mapping is disabled, refer to the TCD_NBYTES_MLNO register description.

6.5.5.24.3 Diagram



6.5.5.24.4 Fields

Field	Function
31 SMLOE	Source Minor Loop Offset Enable Specifies whether the minor loop offset is applied to the source address when the minor loop completes. 0b - The minor loop offset is not applied to the SADDR 1b - The minor loop offset is applied to the SADDR
30 DMLOE	Destination Minor Loop Offset Enable Specifies whether the minor loop offset is applied to the destination address when the minor loop completes. 0b - The minor loop offset is not applied to the DADDR 1b - The minor loop offset is applied to the DADDR
29-10 MLOFF	If SMLOE = 1 or DMLOE = 1, this field represents a sign-extended offset applied to the source or destination address to form the next-state value after the minor loop completes.
9-0 NBYTES	Minor Byte Transfer Count Number of bytes to be transferred in each service request of the channel. As a channel activates, the appropriate TCD contents load into the eDMA engine, and the appropriate reads and writes are performed until the minor byte transfer count has transferred. This is an indivisible operation and cannot be halted. It can, however, be stalled by using the bandwidth control field, or via preemption. After the minor count is exhausted, the SADDR and DADDR values are written back into the TCD memory, and the major iteration count is decremented and restored to the TCD memory. If the major iteration count is completed, additional processing is performed.

6.5.5.25 TCD Last Source Address Adjustment (TCD0_SLAST - TCD31_SLAST)

6.5.5.25.1 Offset

For n = 0 to 31:

Register	Offset
TCDn_SLAST	100Ch + (n × 20h)

6.5.5.25.2 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	SLAST															
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	SLAST															
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

6.5.5.25.3 Fields

Field	Function
31-0 SLAST	Last Source Address Adjustment Adjustment value added to the source address at the completion of the major iteration count. This value can be applied to restore the source address to the initial value, or adjust the address to reference the next data structure. This register uses two's complement notation; the overflow bit is discarded.

6.5.5.26 TCD Destination Address (TCD0_DADDR - TCD31_DADDR)

6.5.5.26.1 Offset

For n = 0 to 31:

Register	Offset
TCDn_DADDR	1010h + (n × 20h)

6.5.5.26.2 Function

This register contains the destination address of the transfer.

6.5.5.26.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	DADDR															
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	DADDR															
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

6.5.5.26.4 Fields

Field	Function
31-0 DADDR	Destination Address Memory address pointing to the destination data.

6.5.5.27 TCD Signed Destination Address Offset (TCD0_DOFF - TCD31_DOFF)

6.5.5.27.1 Offset

For n = 0 to 31:

Register	Offset
TCDn_DOFF	1014h + (n × 20h)

6.5.5.27.2 Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	DOFF															
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

6.5.5.27.3 Fields

Field	Function
15-0 DOFF	Destination Address Signed Offset Sign-extended offset applied to the current destination address to form the next-state value as each destination write is completed.

6.5.5.28 TCD Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) (TCD0_CITER_ELINKNO - TCD31_CITER_ELINKNO)

6.5.5.28.1 Offset

For n = 0 to 31:

Register	Offset
TCDn_CITER_ELINKNO	1016h + (n × 20h)

6.5.5.28.2 Function

This register contains the minor-loop channel-linking configuration and the channel's current iteration count. It is the same register as TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (TCD0_CITER_ELINKYES - TCD31_CITER_ELINKYES), but its fields are defined differently based on the state of the ELINK field. If the ELINK field is 0, this register is defined as follows.

6.5.5.28.3 Diagram

6.5.5.28.4 Fields

Field	Function
15 ELINK	Enable channel-to-channel linking on minor-loop complete As the channel completes the minor loop, this field enables linking to another channel, defined by the LINKCH field. The link target channel initiates a channel service request via an internal mechanism that sets TCDn_CSR[START] of the specified channel. If channel linking is disabled, the CITER value is extended to 15 bits in place of a link channel number. If the major loop is exhausted, this link mechanism is suppressed in favor of the MAJORELINK channel linking. **NOTE:** This field must be equal to BITER[ELINK]; otherwise, a configuration error is reported. 0b - Channel-to-channel linking is disabled 1b - Channel-to-channel linking is enabled
14-0 CITER	Current Major Iteration Count This field is the current major loop count for the channel. It is decremented each time the minor loop is completed and updated in the transfer control descriptor memory. After the major iteration count is exhausted, the channel performs a number of operations, for example, final source and destination address calculations. It optionally generates an interrupt to signal channel completion before reloading the CITER field from the Beginning Iteration Count (BITER) field. **NOTE:** 1. When the CITER field is initially loaded by software, it must be set to the same value as that contained in the BITER field. 2. If the channel is configured to execute a single service request, the initial values of BITER and CITER should be 0x0001.

6.5.5.29 TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (TCD0_CITER_ELINKYES - TCD31_CITER_ELINKYES)

6.5.5.29.1 Offset

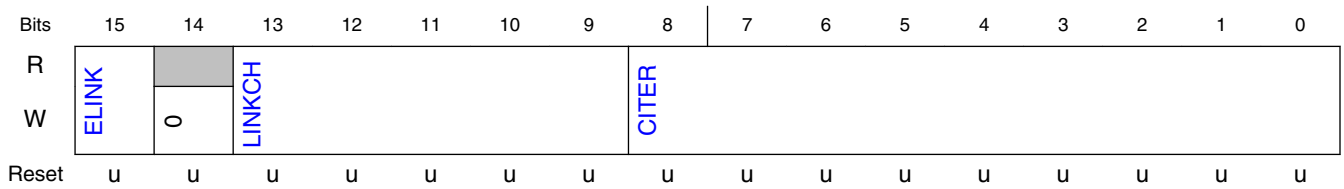
For n = 0 to 31:

Register	Offset
TCDn_CITER_ELINKYES	1016h + (n × 20h)

6.5.5.29.2 Function

This register contains the minor-loop channel-linking configuration and the channel's current iteration count. It is the same register as TCD Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) (TCD0_CITER_ELINKNO - TCD31_CITER_ELINKNO), but its fields are defined differently based on the state of the ELINK field. If the ELINK field is 1, this register is defined as follows.

6.5.5.29.3 Diagram



6.5.5.29.4 Fields

Field	Function
15 ELINK	Enable channel-to-channel linking on minor-loop complete As the channel completes the minor loop, this field enables linking to another channel, defined by the LINKCH field. The link target channel initiates a channel service request via an internal mechanism that sets TCDn_CSR[START] of the specified channel. If channel linking is disabled, the CITER value is extended to 15 bits in place of a link channel number. If the major loop is exhausted, this link mechanism is suppressed in favor of the MAJORELINK channel linking. **NOTE:** This field must be equal to BITER[ELINK]; otherwise, a configuration error is reported. 0b - Channel-to-channel linking is disabled 1b - Channel-to-channel linking is enabled
14 —	Reserved
13-9 LINKCH	Minor Loop Link Channel Number If channel-to-channel linking is enabled (ELINK = 1), then after the minor loop is exhausted, the eDMA engine initiates a channel service request to the channel defined by this field, by setting that channel's TCDn_CSR[START].
8-0 CITER	Current Major Iteration Count This field is the current major loop count for the channel. It is decremented each time the minor loop is completed and updated in the transfer control descriptor memory. After the major iteration count is exhausted, the channel performs a number of operations, for example, final source and destination address calculations. It optionally generates an interrupt to signal channel completion before reloading the CITER field from the Beginning Iteration Count (BITER) field. **NOTE:** 1. When the CITER field is initially loaded by software, it must be set to the same value as that contained in the BITER field. 2. If the channel is configured to execute a single service request, the initial values of BITER and CITER should be 0x0001.

6.5.5.30 TCD Last Destination Address Adjustment/Scatter Gather Address (TCD0_DLASTSGA - TCD31_DLASTSGA)

6.5.5.30.1 Offset

For n = 0 to 31:

Register	Offset
TCDn_DLASTSGA	1018h + (n × 20h)

6.5.5.30.2 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	DLASTSGA															
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	DLASTSGA															
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

6.5.5.30.3 Fields

Field	Function
31-0 DLASTSGA	Destination last address adjustment, or next memory address TCD for channel (scatter/gather) If (TCDn_CSR[ESG] = 0) then: • This is the adjustment value added to the destination address at the completion of the major iteration count. This value can apply to restore the destination address to the initial value or adjust the address to reference the next data structure. • This field uses two's complement notation for the final destination address adjustment. Otherwise: • This address points to the beginning of a 0-modulo 32-byte region containing the next TCD to be loaded into this channel. This channel reload is performed as the major iteration count completes. The scatter/gather address must be 0-modulo 32-byte; otherwise a configuration error is reported.

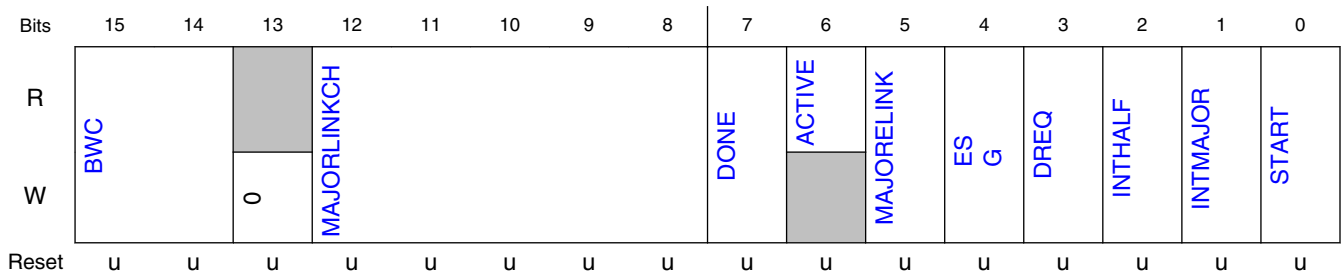
6.5.5.31 TCD Control and Status (TCD0_CSR - TCD31_CSR)

6.5.5.31.1 Offset

For n = 0 to 31:

Register	Offset
TCDn_CSR	101Ch + (n × 20h)

6.5.5.31.2 Diagram



6.5.5.31.3 Fields

Field	Function
15-14 BWC	Bandwidth Control Throttles the amount of bus bandwidth consumed by the eDMA. Generally, as the eDMA processes the minor loop, it continuously generates read/write sequences until the minor count is exhausted. This field forces the eDMA to stall after the completion of each read/write access to control the bus request bandwidth seen by the crossbar switch. **NOTE:** If the source and destination sizes are equal, this field is ignored between the first and second transfers and after the last write of each minor loop. This behavior is a side effect of reducing start-up latency. 00b - No eDMA engine stalls 01b - Reserved 10b - eDMA engine stalls for 4 cycles after each R/W 11b - eDMA engine stalls for 8 cycles after each R/W
13 —	Reserved
12-8 MAJORLINKCH	Major Loop Link Channel Number If (MAJORELINK = 0) then: • No channel-to-channel linking, or chaining, is performed after the major loop counter is exhausted. Otherwise: • After the major loop counter is exhausted, the eDMA engine initiates a channel service request at the channel defined by this field by setting that channel's START bit.
7 DONE	Channel Done This field indicates whether the eDMA has completed the major loop. The eDMA engine sets the value of this field to 1 when the CITER count reaches zero. The value of this field is reset to 0 by the hardware (when the channel is activated) or by software. **NOTE:** This field must be 0 to write the MAJORELINK or ESG fields.
6	Channel Active

Table continues on the next page...

Field	Function
ACTIVE	This field indicates whether the channel is currently in execution. The eDMA sets the value of this field to 1 when channel service begins, and resets it to 0 as the minor loop completes or when any error condition is detected.
5 MAJORELINK	Enable channel-to-channel linking on major loop complete As the channel completes the major loop, this field controls linking to another channel, defined by MAJORLINKCH. The link target channel initiates a channel service request via an internal mechanism that sets TCDn_CSR[START] of the specified channel. **NOTE:** To support the dynamic linking coherency model, this field is forced to zero when written to when TCDn_CSR[DONE] is set. 0b - Channel-to-channel linking is disabled 1b - Channel-to-channel linking is enabled
4 ESG	Enable Scatter/Gather Processing As the channel completes the major loop, this field controls scatter/gather processing in the current channel. If enabled, the eDMA engine uses DLASTSGA as a memory pointer to a 0-modulo 32-bit address containing a 32-byte data structure loaded as the TCD into local memory. **NOTE:** To support the dynamic scatter/gather coherency model, this field is forced to zero when written to when TCDn_CSR[DONE] is set. 0b - The current channel's TCD is normal format 1b - The current channel's TCD specifies a scatter gather format
3 DREQ	Disable Request If the value of this field is 1, eDMA hardware automatically writes 0 to the corresponding ERQ field when the current major iteration count reaches zero. 0b - The channel's ERQ field is not affected 1b - The channel's ERQ field value changes to 0 when the major loop is complete
2 INTHALF	Enable an interrupt when major counter is half complete. If the value of this field is 1, the channel generates an interrupt request by setting the appropriate field in the INT register when the current major iteration count reaches the halfway point. Specifically, the comparison performed by the eDMA engine is (CITER == (BITER >> 1)). This halfway point interrupt request is provided to support double-buffered, also known as ping-pong, schemes or other types of data movement where the processor needs an early indication of the transfer's progress. **NOTE:** If BITER = 1, do not use INTHALF. Use INTMAJOR instead. 0b - Half-point interrupt is disabled 1b - Half-point interrupt is enabled
1 INTMAJOR	Enable an interrupt when major iteration count completes. If the value of this field is 1, the channel generates an interrupt request by setting the appropriate field in the INT when the current major iteration count reaches zero. 0b - End of major loop interrupt is disabled 1b - End of major loop interrupt is enabled
0 START	Channel Start If the value of this field is 1, the channel is requesting service. eDMA hardware automatically writes 0 to this field after the channel begins execution. 0b - Channel is not explicitly started 1b - Channel is explicitly started via a software initiated service request

6.5.5.32 TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (TCD0_BITER_ELINKNO - TCD31_BITER_ELINKNO)

6.5.5.32.1 Offset

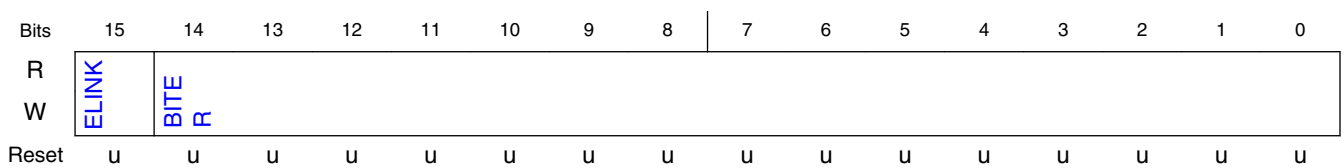
For n = 0 to 31:

Register	Offset
TCDn_BITER_ELINKNO	101Eh + (n × 20h)

6.5.5.32.2 Function

If TCDn_BITER[ELINK] is 0, the TCDn_BITER register is defined as follows.

6.5.5.32.3 Diagram



6.5.5.32.4 Fields

Field	Function
15 ELINK	Enables channel-to-channel linking on minor loop complete As the channel completes the minor loop, this field enables linking to another channel, defined by BITER[LINKCH]. The link target channel initiates a channel service request via an internal mechanism that sets TCDn_CSR[START] of the specified channel. If channel linking is disabled, the BITER value extends to 15 bits in place of a link channel number. If the major loop is exhausted, this link mechanism is suppressed in favor of the MAJORELINK channel linking. **NOTE:** When the software loads the TCD, this field must be set equal to the corresponding CITER field; otherwise, a configuration error is reported. As the major iteration count is exhausted, the contents of this field are reloaded into the CITER field. 0b - Channel-to-channel linking is disabled 1b - Channel-to-channel linking is enabled
14-0 BITER	Starting Major Iteration Count As the TCD is first loaded by software, this 9-bit (ELINK = 1) or 15-bit (ELINK = 0) field must be equal to the value in the CITER field. As the major iteration count is exhausted, the contents of this field are reloaded into the CITER field. **NOTE:** When the software loads the TCD, this field must be set equal to the corresponding CITER field; otherwise, a configuration error is reported. As the major iteration count is exhausted, the contents of this field are reloaded into the CITER field. If the channel is configured to execute a single service request, the initial values of BITER and CITER should be 0x0001.

6.5.5.33 TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (TCD0_BITER_ELINKYES - TCD31_BITER_ELINKYES)

6.5.5.33.1 Offset

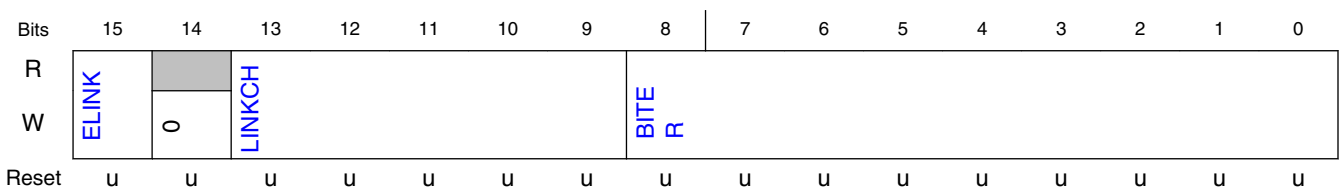
For n = 0 to 31:

Register	Offset
TCDn_BITER_ELINKYES	101Eh + (n × 20h)

6.5.5.33.2 Function

If TCDn_BITER[ELINK] is 1, the TCDn_BITER register is defined as follows.

6.5.5.33.3 Diagram



6.5.5.33.4 Fields

Field	Function
15 ELINK	Enables channel-to-channel linking on minor loop complete As the channel completes the minor loop, this field enables linking to another channel, defined by BITER[LINKCH]. The link target channel initiates a channel service request via an internal mechanism that sets TCDn_CSR[START] of the specified channel. If channel linking disables, the BITER value extends to 15 bits in place of a link channel number. If the major loop is exhausted, this link mechanism is suppressed in favor of the MAJORELINK channel linking. **NOTE:** When the software loads the TCD, this field must be set equal to the corresponding CITER field; otherwise, a configuration error is reported. As the major iteration count is exhausted, the contents of this field are reloaded into the CITER field. 0b - Channel-to-channel linking is disabled 1b - Channel-to-channel linking is enabled
14 —	Reserved

Table continues on the next page...

Field	Function
13-9 LINKCH	Link Channel Number If channel-to-channel linking is enabled (ELINK = 1), then after the minor loop is exhausted, the eDMA engine initiates a channel service request at the channel defined by this field, by setting that channel's TCDn_CSR[START]. **NOTE:** When the software loads the TCD, this field must be set equal to the corresponding CITER field; otherwise, a configuration error is reported. As the major iteration count is exhausted, the contents of this field are reloaded into the CITER field.
8-0 BITER	Starting major iteration count As the TCD is first loaded by software, this 9-bit (ELINK = 1) or 15-bit (ELINK = 0) field must be equal to the value in the CITER field. As the major iteration count is exhausted, the contents of this field are reloaded into the CITER field. **NOTE:** When the software loads the TCD, this field must be set equal to the corresponding CITER field; otherwise, a configuration error is reported. As the major iteration count is exhausted, the contents of this field are reloaded into the CITER field. If the channel is configured to execute a single service request, the initial values of BITER and CITER should be 0x0001.