

- 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:

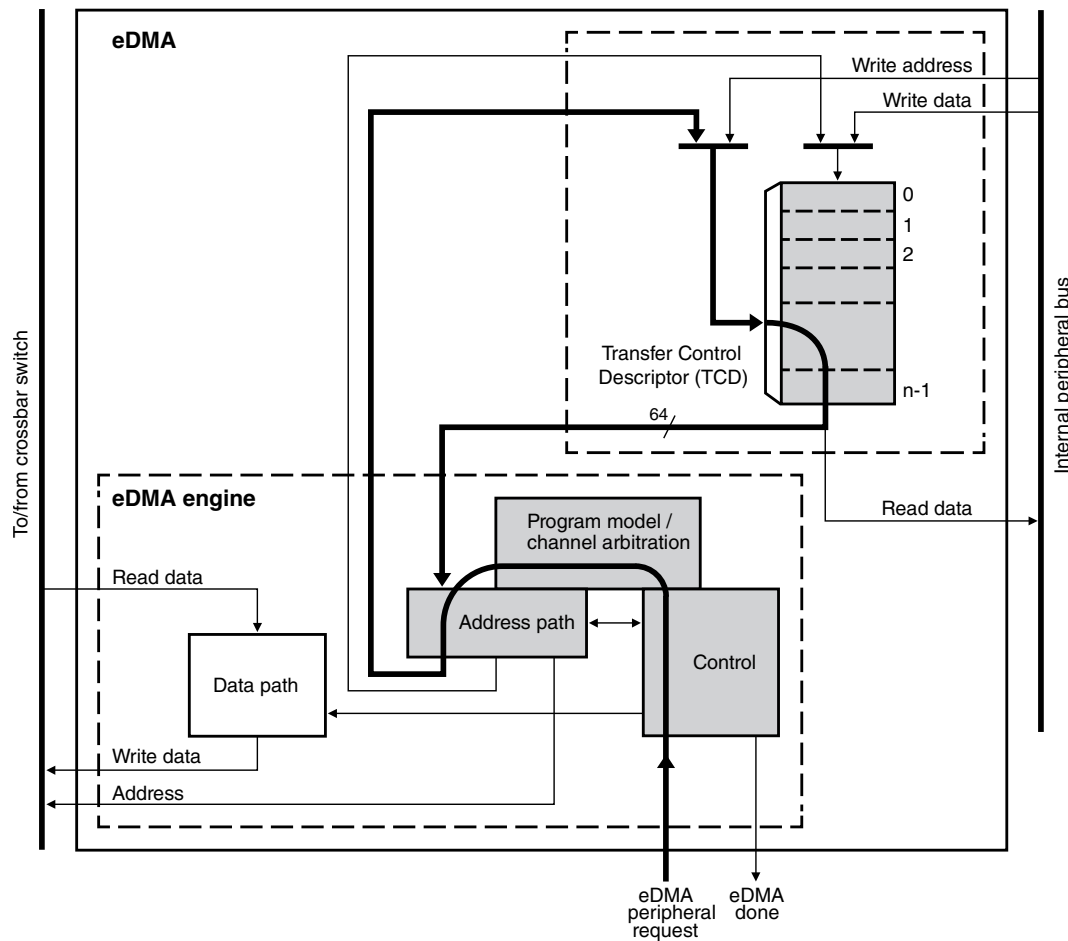
**Table 6-4. Modes of operation**

Mode	Description
Normal	<p>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.</p> <p>A <i>service request</i> initiates a transfer of a specific number of bytes (NBYTES) as specified in the TCD.</p> <ul style="list-style-type: none"> <li>• The <i>minor loop</i> is the sequence of read and write operations that transfers the NBYTES of data for a service request.</li> <li>• Each service request executes one iteration of the major loop, transferring NBYTES of data.</li> </ul>
Debug	<p>DMA operation is configurable in Debug mode via <a href="#">Control (CR)</a></p> <ul style="list-style-type: none"> <li>• If CR[EDBG] = 0, eDMA continues to operate normally when the chip is in debug mode.</li> <li>• 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.</li> </ul>
Wait	<p>Before entering Wait mode, eDMA attempts to complete any transfer that is in progress. After the transfer completes, the chip enters Wait mode.</p>

### 6.3.2 eDMA basic data flow

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

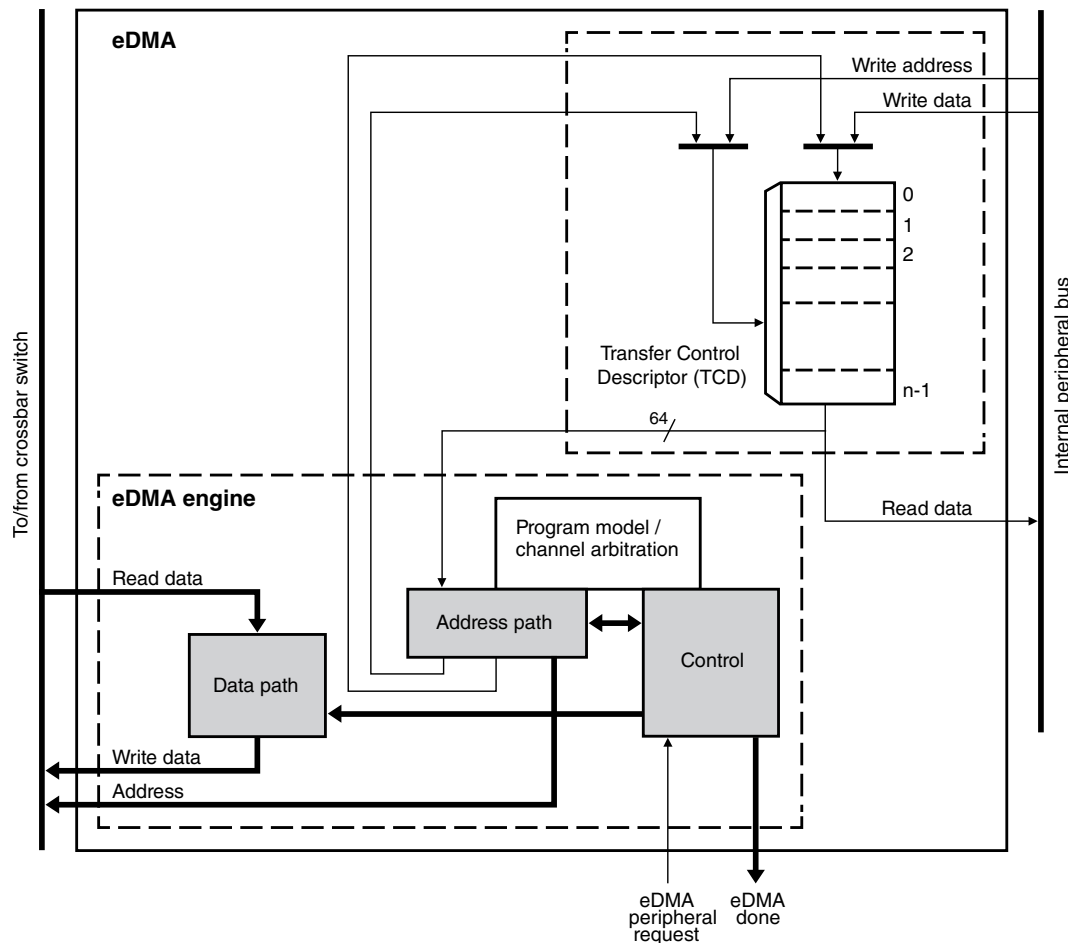
As shown in the following diagram, the first segment involves the channel activation:



**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  $TCDn\_CSR[START]$  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  $TCDn$ . 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.

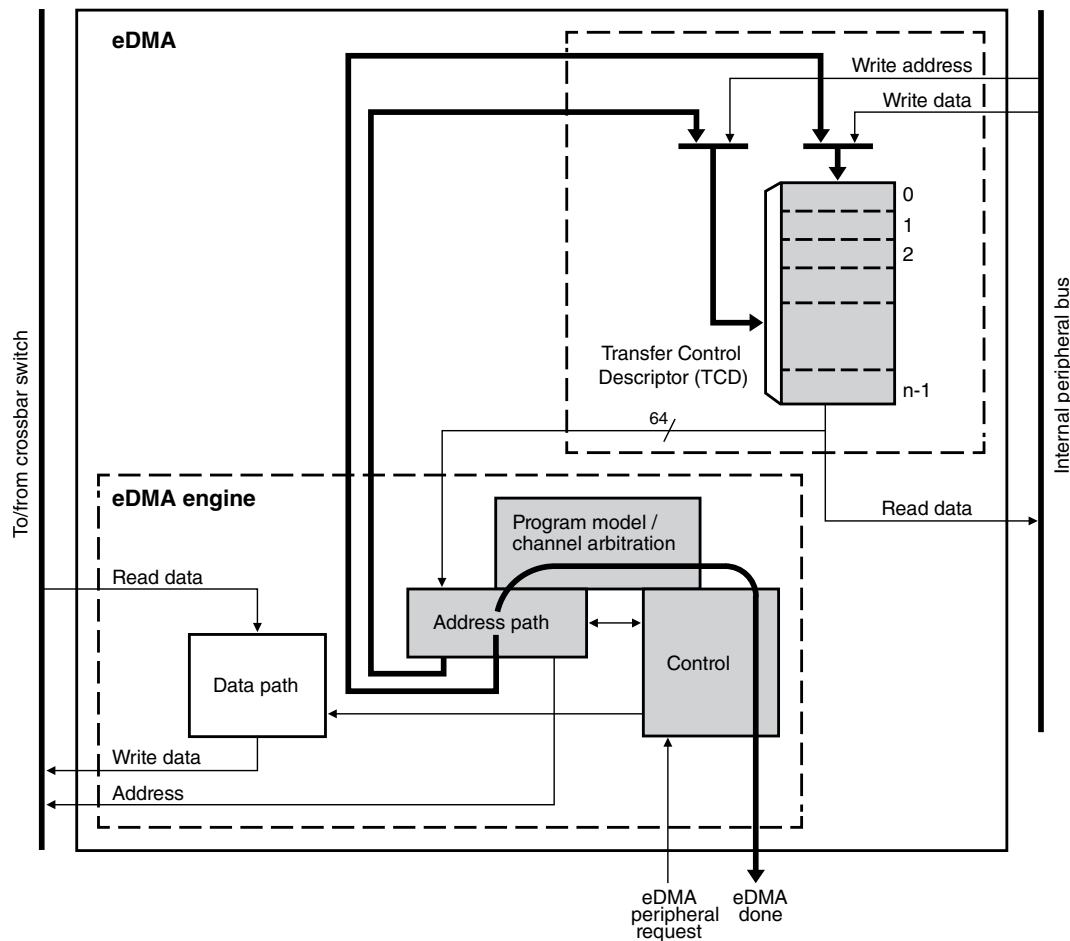


Figure 6-4. eDMA operation, part 3

### 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.