# Direct Memory Access (DMA) Controller Module

**NOTE:** This chapter is an excerpt from the *MSP430x5xx and MSP430x6xx Family User's Guide*. The most recent version of the full user's guide is available from http://www.ti.com/lit/pdf/slau208.

The direct memory access (DMA) controller module transfers data from one address to another without CPU intervention. This chapter describes the operation of the DMA controller.

**Topic** **Page**

## 1.1 Direct Memory Access (DMA) Introduction

The DMA controller transfers data from one address to another, without CPU intervention, across the entire address range. For example, the DMA controller can move data from the ADC conversion memory to RAM.

Devices that contain a DMA controller may have up to eight DMA channels available. Therefore, depending on the number of DMA channels available, some features described in this chapter are not applicable to all devices. See the device-specific data sheet for number of channels supported.

Using the DMA controller can increase the throughput of peripheral modules. It can also reduce system power consumption by allowing the CPU to remain in a low-power mode, without having to awaken to move data to or from a peripheral.

DMA controller features include:

- Up to eight independent transfer channels
- Configurable DMA channel priorities
- Requires only two MCLK clock cycles per transfer
- Byte or word and mixed byte and word transfer capability
- Block sizes up to 65535 bytes or words
- Configurable transfer trigger selections
- Selectable edge- or level-triggered transfer
- Four addressing modes
- Single, block, or burst-block transfer modes

Figure 1-1 shows the DMA controller block diagram.

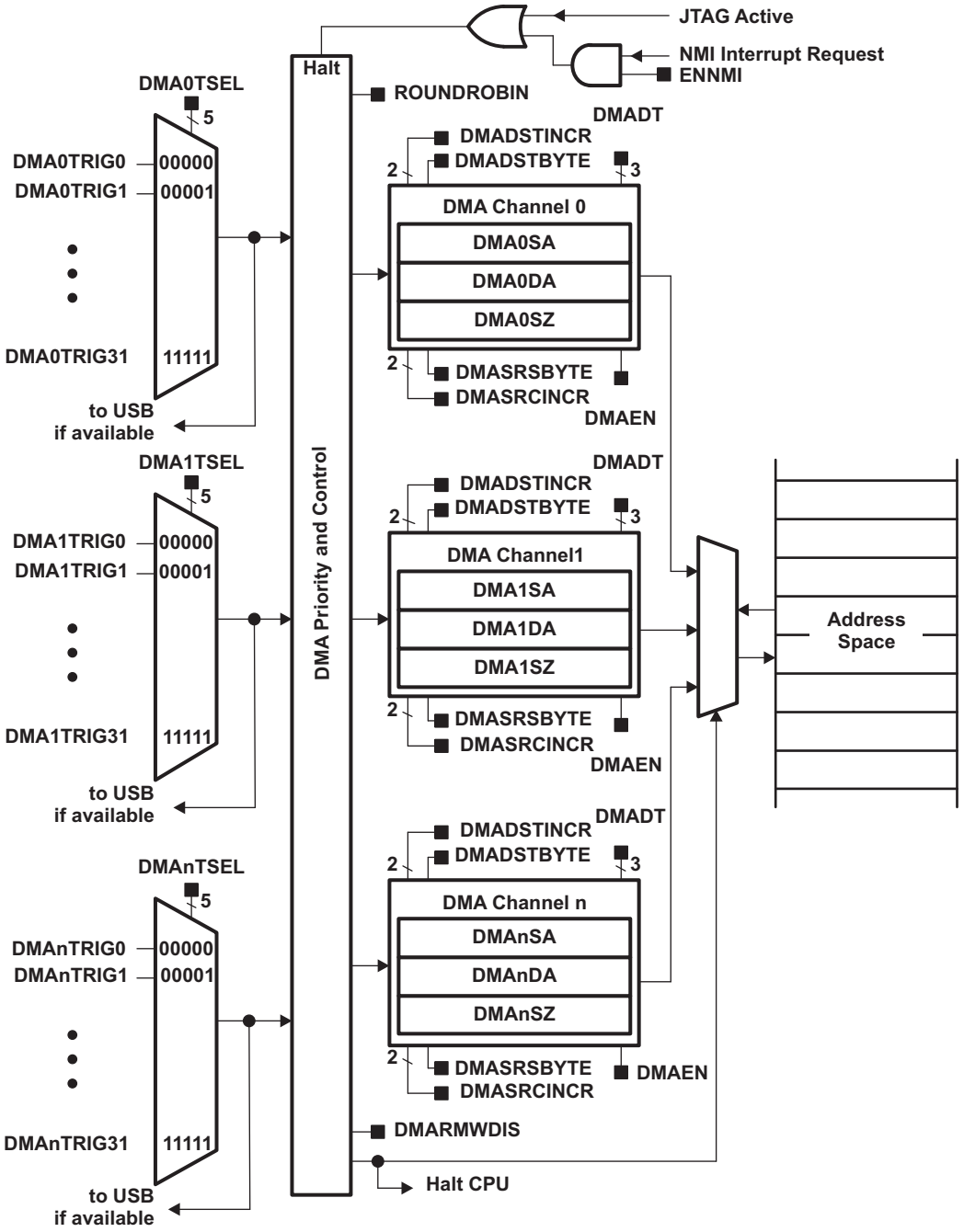


**Figure 1-1. DMA Controller Block Diagram**

## 1.2 DMA Operation

The DMA controller is configured with user software. The setup and operation of the DMA is discussed in the following sections.

### *1.2.1 DMA Addressing Modes*

The DMA controller has four addressing modes. The addressing mode for each DMA channel is independently configurable. For example, channel 0 may transfer between two fixed addresses, while channel 1 transfers between two blocks of addresses. Figure 1-2 shows the addressing modes. The addressing modes are:

- Fixed address to fixed address
- Fixed address to block of addresses
- Block of addresses to fixed address
- Block of addresses to block of addresses

The addressing modes are configured with the DMASRCINCR and DMADSTINCR control bits. The DMASRCINCR bits select if the source address is incremented, decremented, or unchanged after each transfer. The DMADSTINCR bits select if the destination address is incremented, decremented, or unchanged after each transfer.

Transfers may be byte to byte, word to word, byte to word, or word to byte. When transferring word to byte, only the lower byte of the source word is transfered. When transferring byte to word, the upper byte of the destination word is cleared when the transfer occurs.

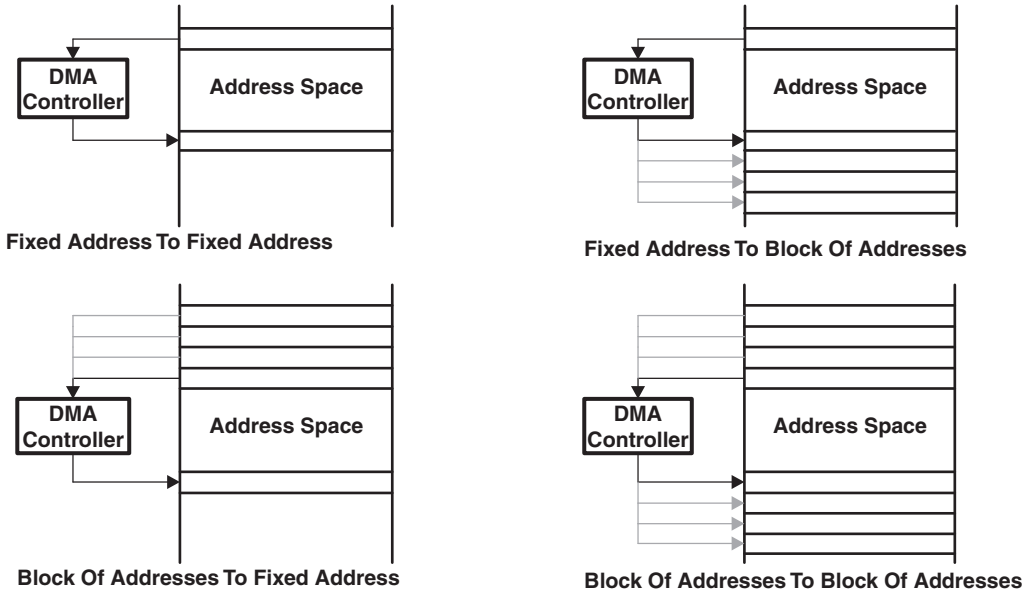


**Figure 1-2. DMA Addressing Modes**

### 1.2.2 DMA Transfer Modes

The DMA controller has six transfer modes selected by the DMADT bits (see Table 1-1). Each channel is individually configurable for its transfer mode. For example, channel 0 may be configured in single transfer mode, while channel 1 is configured for burst-block transfer mode, and channel 2 operates in repeated block mode. The transfer mode is configured independently from the addressing mode. Any addressing mode can be used with any transfer mode.

Two types of data can be transferred selectable by the DMAxCTL DSTBYTE and SRCBYTE fields. The source and destination location can be either byte or word data. It is also possible to transfer byte to byte, word to word, or any combination.

**Table 1-1. DMA Transfer Modes**

| DMADT | Transfer Mode | Description |
|---|---|---|
| 000 | Single transfer | Each transfer requires a trigger. DMAEN is automatically cleared when DMAxSZ transfers have been made. |
| 001 | Block transfer | A complete block is transferred with one trigger. DMAEN is automatically cleared at the end of the block transfer. |
| 010, 011 | Burst-block transfer | CPU activity is interleaved with a block transfer. DMAEN is automatically cleared at the end of the burst-block transfer. |
| 100 | Repeated single transfer | Each transfer requires a trigger. DMAEN remains enabled. |
| 101 | Repeated block transfer | A complete block is transferred with one trigger. DMAEN remains enabled. |
| 110, 111 | Repeated burst-block transfer | CPU activity is interleaved with a block transfer. DMAEN remains enabled. |

### 1.2.2.1 Single Transfer

In single transfer mode, each byte or word transfer requires a separate trigger. Figure 1-3 shows the single transfer state diagram.

The DMAxSZ register defines the number of transfers to be made. The DMADSTINCR and DMASRCINCR bits select if the destination address and the source address are incremented or decremented after each transfer. If DMAxSZ = 0, no transfers occur.

The DMAxSA, DMAxDA, and DMAxSZ registers are copied into temporary registers. The temporary values of DMAxSA and DMAxDA are incremented or decremented after each transfer. The DMAxSZ register is decremented after each transfer. When the DMAxSZ register decrements to zero, it is reloaded from its temporary register and the corresponding DMAIFG flag is set. When DMADT = {0}, the DMAEN bit is cleared automatically when DMAxSZ decrements to zero and must be set again for another transfer to occur.

In repeated single transfer mode, the DMA controller remains enabled with DMAEN = 1, and a transfer occurs every time a trigger occurs.

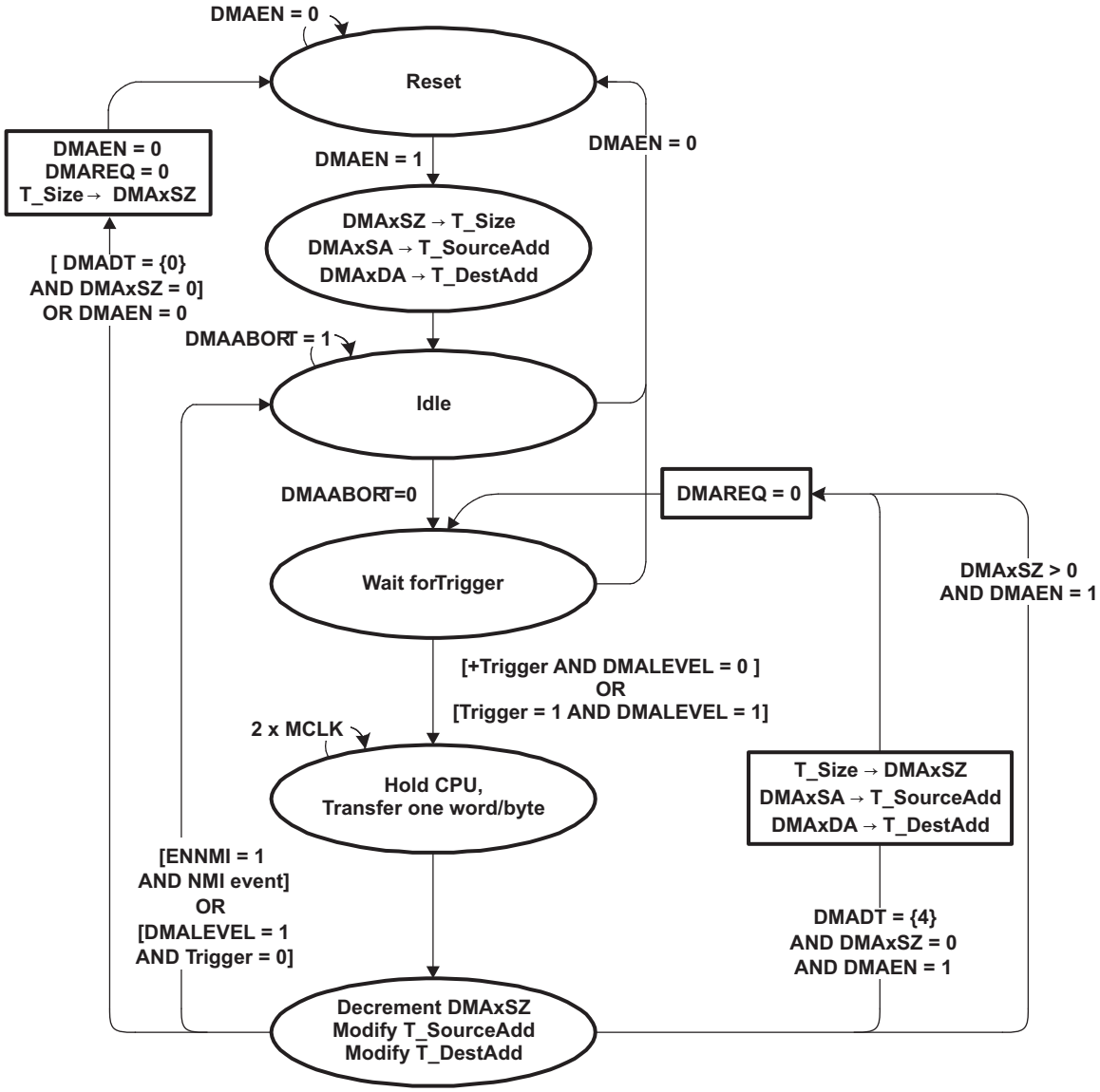


**Figure 1-3. DMA Single Transfer State Diagram**

#### 1.2.2.2 Block Transfer

In block transfer mode, a transfer of a complete block of data occurs after one trigger. When DMADT = {1} ,the DMAEN bit is cleared after the completion of the block transfer and must be set again before another block transfer can be triggered. After a block transfer has been triggered, further trigger signals occurring during the block transfer are ignored. Figure 1-4 shows the block transfer state diagram.

The DMAxSZ register is used to define the size of the block, and the DMADSTINCR and DMASRCINCR bits select if the destination address and the source address are incremented or decremented after each transfer of the block. If DMAxSZ = 0, no transfers occur.

The DMAxSA, DMAxDA, and DMAxSZ registers are copied into temporary registers. The temporary values of DMAxSA and DMAxDA are incremented or decremented after each transfer in the block. The DMAxSZ register is decremented after each transfer of the block and shows the number of transfers remaining in the block. When the DMAxSZ register decrements to zero, it is reloaded from its temporary register and the corresponding DMAIFG flag is set.

During a block transfer, the CPU is halted until the complete block has been transferred. The block transfer takes 2 × MCLK × DMAxSZ clock cycles to complete. CPU execution resumes with its previous state after the block transfer is complete.

In repeated block transfer mode, the DMAEN bit remains set after completion of the block transfer. The next trigger after the completion of a repeated block transfer triggers another block transfer.

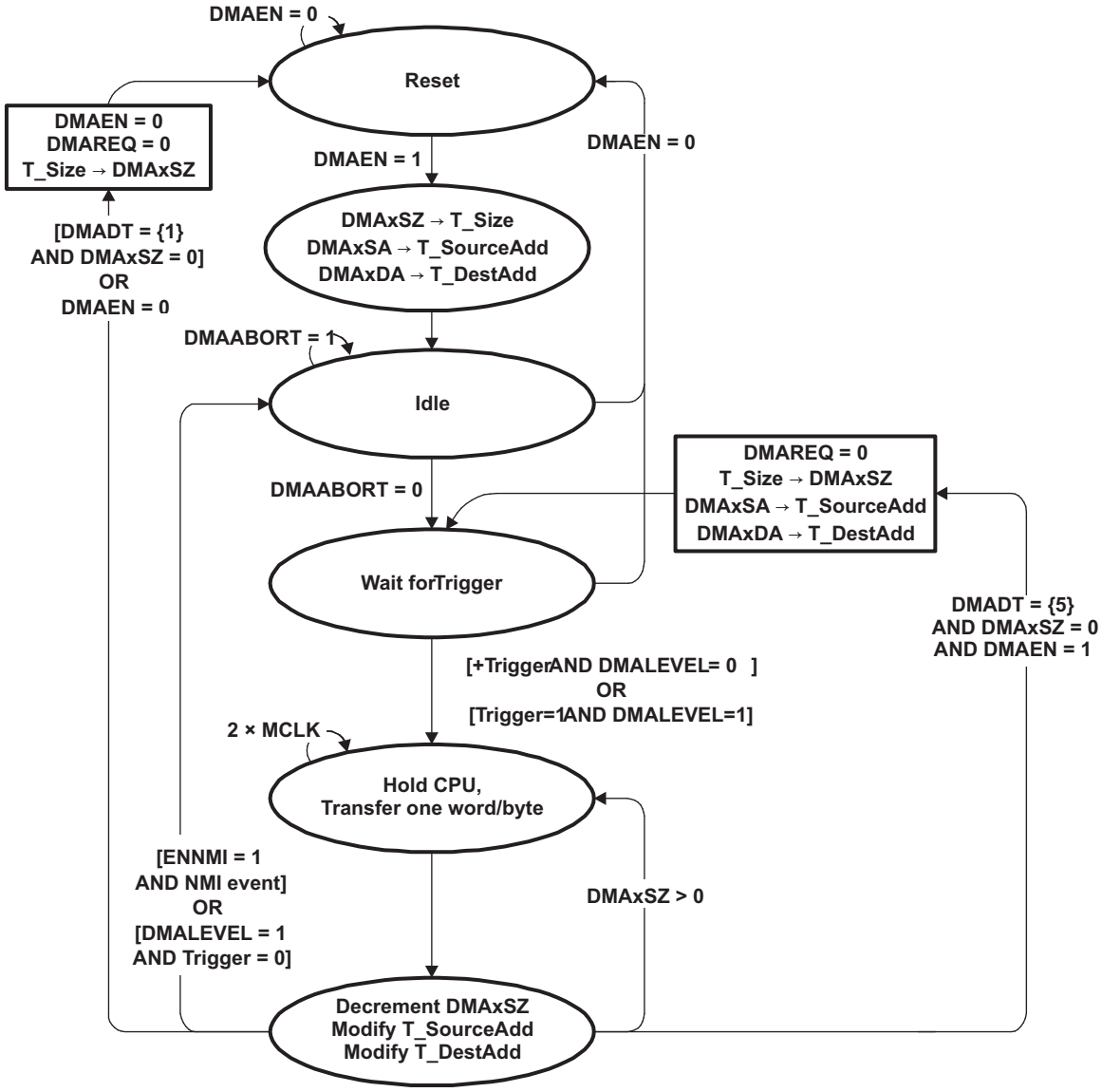


**Figure 1-4. DMA Block Transfer State Diagram**

#### 1.2.2.3 Burst-Block Transfer

In burst-block mode, transfers are block transfers with CPU activity interleaved. The CPU executes two MCLK cycles after every four byte/word transfers of the block, resulting in 20% CPU execution capacity. After the burst-block, CPU execution resumes at 100% capacity and the DMAEN bit is cleared. DMAEN must be set again before another burst-block transfer can be triggered. After a burst-block transfer has been triggered, further trigger signals occurring during the burst-block transfer are ignored. Figure 1-5 shows the burst-block transfer state diagram.

The DMAxSZ register is used to define the size of the block, and the DMADSTINCR and DMASRCINCR bits select if the destination address and the source address are incremented or decremented after each transfer of the block. If DMAxSZ = 0, no transfers occur.

The DMAxSA, DMAxDA, and DMAxSZ registers are copied into temporary registers. The temporary values of DMAxSA and DMAxDA are incremented or decremented after each transfer in the block. The DMAxSZ register is decremented after each transfer of the block and shows the number of transfers remaining in the block. When the DMAxSZ register decrements to zero, it is reloaded from its temporary register and the corresponding DMAIFG flag is set.

In repeated burst-block mode, the DMAEN bit remains set after completion of the burst-block transfer and no further trigger signals are required to initiate another burst-block transfer. Another burst-block transfer begins immediately after completion of a burst-block transfer. In this case, the transfers must be stopped by clearing the DMAEN bit, or by an (non)maskable interrupt (NMI) when ENNMI is set. In repeated burst-block mode, the CPU executes at 20% capacity continuously until the repeated burst-block transfer is stopped.

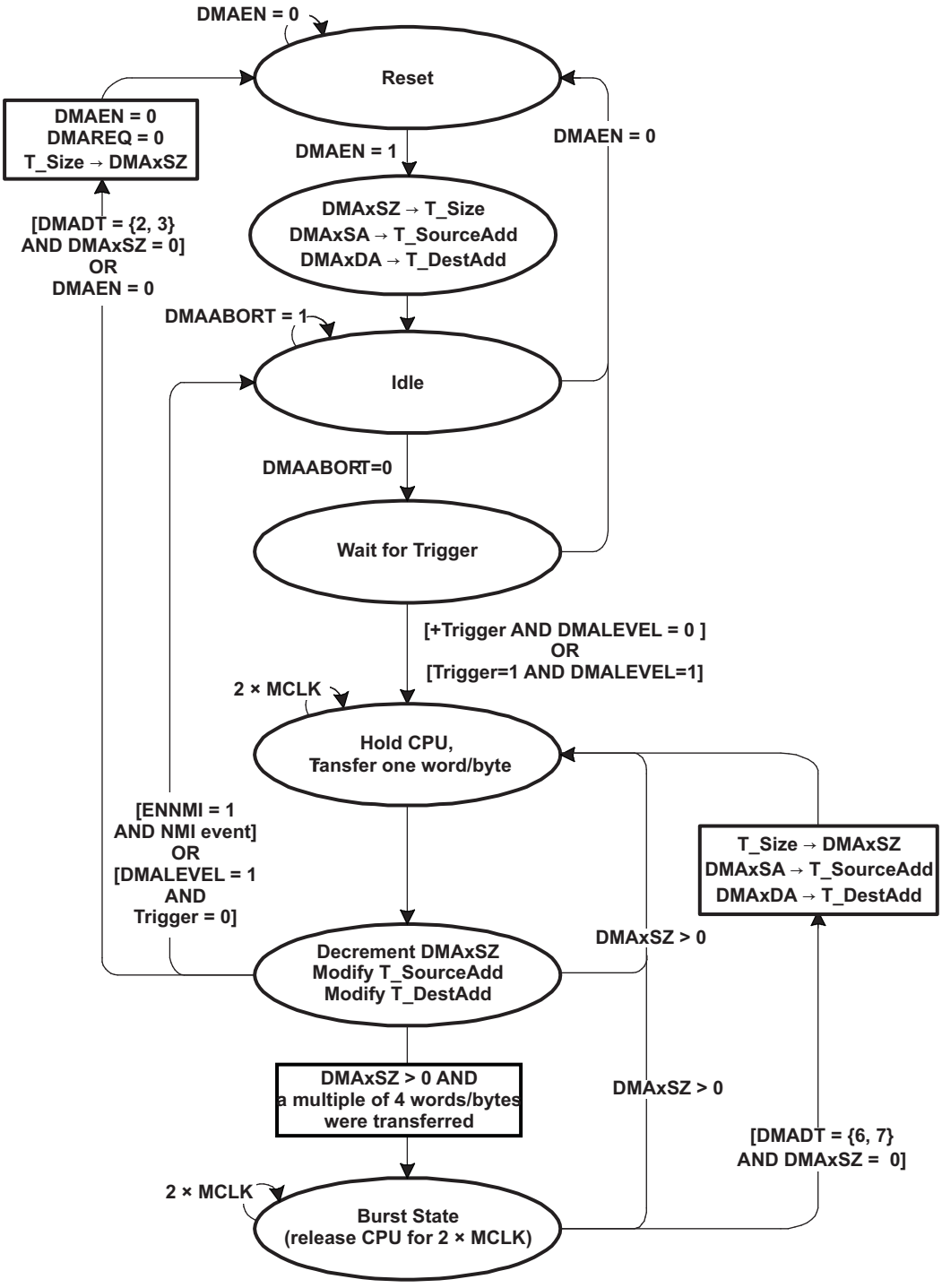


**Figure 1-5. DMA Burst-Block Transfer State Diagram**

### 1.2.3 Initiating DMA Transfers

The trigger source for each DMA channel is independently configured by DMAxTSEL. The DMAxTSEL bits should be modified only when the DMAxCTL DMAEN bit is 0. Otherwise, unpredictable DMA triggers may occur. Table 1-2 describes the trigger operation for each type of module. See the device-specific data sheet for the list of triggers available, along with their respective DMAxTSEL values.

When selecting the trigger, the trigger must not have already occurred, or the transfer does not take place.

**NOTE: DMA trigger selection and USB**

On devices that contain a USB module, the triggers selection from DMA channels 0, 1, or 2 can be used for the USB time stamp event selection (see the USB module description for further details).

#### 1.2.3.1 Edge-Sensitive Triggers

When DMALEVEL = 0, edge-sensitive triggers are used, and the rising edge of the trigger signal initiates the transfer. In single-transfer mode, each transfer requires its own trigger. When using block or burst-block modes, only one trigger is required to initiate the block or burst-block transfer.

#### 1.2.3.2 Level-Sensitive Triggers

When DMALEVEL = 1, level-sensitive triggers are used. For proper operation, level-sensitive triggers can only be used when external trigger DMAE0 is selected as the trigger. DMA transfers are triggered as long as the trigger signal is high and the DMAEN bit remains set.

The trigger signal must remain high for a block or burst-block transfer to complete. If the trigger signal goes low during a block or burst-block transfer, the DMA controller is held in its current state until the trigger goes back high or until the DMA registers are modified by software. If the DMA registers are not modified by software, when the trigger signal goes high again, the transfer resumes from where it was when the trigger signal went low.

When DMALEVEL = 1, transfer modes selected by DMADT = {0, 1, 2, or 3} are recommended because the DMAEN bit is automatically reset after the configured transfer.

### 1.2.4 Halting Executing Instructions for DMA Transfers

The DMARMWDIS bit controls when the CPU is halted for DMA transfers. When DMARMWDIS = 0, the CPU is halted immediately and the transfer begins when a trigger is received. In this case, it is possible that CPU read-modify-write operations can be interrupted by a DMA transfer. When DMARMWDIS = 1, the CPU finishes the currently executing read-modify-write operation before the DMA controller halts the CPU and the transfer begins (see Table 1-2).

**Table 1-2. DMA Trigger Operation**

| Module | Operation |
|---|---|
| DMA | A transfer is triggered when the DMAREQ bit is set. The DMAREQ bit is automatically reset when the transfer starts.<br>A transfer is triggered when the DMAxIFG flag is set. DMA0IFG triggers channel 1, DMA1IFG triggers channel 2, and DMA2IFG triggers channel 0. None of the DMAxIFG flags are automatically reset when the transfer starts.<br>A transfer is triggered by the external trigger DMAE0. |
| Timer_A | A transfer is triggered when the TAxCCR0 CCIFG flag is set. The TAxCCR0 CCIFG flag is automatically reset when the transfer starts. If the TAxCCR0 CCIE bit is set, the TAxCCR0 CCIFG flag dies not trigger a transfer.<br>A transfer is triggered when the TAxCCR2 CCIFG flag is set. The TAxCCR2 CCIFG flag is automatically reset when the transfer starts. If the TAxCCR2 CCIE bit is set, the TAxCCR2 CCIFG flag does not trigger a transfer. |
| Timer_B | A transfer is triggered when the TBxCCR0 CCIFG flag is set. The TBxCCR0 CCIFG flag is automatically reset when the transfer starts. If the TBxCCR0 CCIE bit is set, the TBxCCR0 CCIFG flag does not trigger a transfer.<br>A transfer is triggered when the TBxCCR2 CCIFG flag is set. The TBxCCR2 CCIFG flag is automatically reset when the transfer starts. If the TBxCCR2 CCIE bit is set, the TBxCCR2 CCIFG flag does not trigger a transfer. |
| USCI_Ax | A transfer is triggered when USCI_Ax receives new data. UCAxRXIFG is automatically reset when the transfer starts. If UCAxRXIE is set, the UCAxRXIFG does not trigger a transfer.<br>A transfer is triggered when USCI_Ax is ready to transmit new data. UCAxTXIFG is automatically reset when the transfer starts. If UCAxTXIE is set, the UCAxTXIFG does not trigger a transfer. |
| USCI_Bx | A transfer is triggered when USCI_Bx receives new data. UCBxRXIFG is automatically reset when the transfer starts. If UCBxRXIE is set, the UCBxRXIFG does not trigger a transfer.<br>A transfer is triggered when USCI_Bx is ready to transmit new data. UCBxTXIFG is automatically reset when the transfer starts. If UCBxTXIE is set, the UCBxTXIFG does not trigger a transfer. |
| DAC12_A | A transfer is triggered when the DAC12_xCTL0 DAC12IFG flag is set. The DAC12_xCTL0 DAC12IFG flag is automatically cleared when the transfer starts. If the DAC12_xCTL0 DAC12IE bit is set, the DAC12_xCTL0 DAC12IFG flag does not trigger a transfer. |
| ADC10_A | A transfer is triggered by an ADC10IFG0 flag with the ADC10IE0 bit reset. A transfer is triggered when the conversion is completed and the ADC10IFG0 is set. Setting the ADC10IFG0 with software does not trigger a transfer. The ADC10IFG0 flag is automatically reset when the ADC10MEM0 register is accessed by the DMA controller. |
| ADC12_A | A transfer is triggered by an ADC12IFG flag with the corresponding ADC12IE bit reset. When single-channel conversions are performed, the corresponding ADC12IFG is the trigger. When sequences are used, the ADC12IFG for the last conversion in the sequence is the trigger. A transfer is triggered when the conversion is completed and the ADC12IFG is set. Setting the ADC12IFG with software does not trigger a transfer. All ADC12IFG flags are automatically reset when the associated ADC12MEMx register is accessed by the DMA controller. |
| MPY | A transfer is triggered when the hardware multiplier is ready for a new operand. |
| Reserved | No transfer is triggered. |

### 1.2.5 Stopping DMA Transfers

There are two ways to stop DMA transfers in progress:

- A single, block, or burst-block transfer may be stopped with an NMI, if the ENNMI bit is set in register DMACTL1.
- A burst-block transfer may be stopped by clearing the DMAEN bit.

### 1.2.6 DMA Channel Priorities

The default DMA channel priorities are DMA0 through DMA7. If two or three triggers happen simultaneously or are pending, the channel with the highest priority completes its transfer (single, block, or burst-block transfer) first, then the second priority channel, then the third priority channel. Transfers in progress are not halted if a higher-priority channel is triggered. The higher-priority channel waits until the transfer in progress completes before starting.

The DMA channel priorities are configurable with the ROUNDROBIN bit. When the ROUNDROBIN bit is set, the channel that completes a transfer becomes the lowest priority. The *order* of the priority of the channels always stays the same, DMA0-DMA1-DMA2, for example, for three channels. When the ROUNDROBIN bit is cleared, the channel priority returns to the default priority.

| DMA Priority | Transfer Occurs | New DMA Priority |
|---|---|---|
| DMA0-DMA1-DMA2 | DMA1 | DMA2-DMA0-DMA1 |
| DMA2-DMA0-DMA1 | DMA2 | DMA0-DMA1-DMA2 |
| DMA0-DMA1-DMA2 | DMA0 | DMA1-DMA2-DMA0 |

### 1.2.7 DMA Transfer Cycle Time

The DMA controller requires one or two MCLK clock cycles to synchronize before each single transfer or complete block or burst-block transfer. Each byte/word transfer requires two MCLK cycles after synchronization, and one cycle of wait time after the transfer. Because the DMA controller uses MCLK, the DMA cycle time depends on the MSP430 operating mode and clock system setup.

If the MCLK source is active but the CPU is off, the DMA controller uses the MCLK source for each transfer, without enabling the CPU. If the MCLK source is off, the DMA controller temporarily restarts MCLK, sourced with DCOCLK, for the single transfer or complete block or burst-block transfer. The CPU remains off, and MCLK is turned off after the transfer completes. Table 1-3 lists the maximum DMA cycle times for all operating modes.

**Table 1-3. Maximum Single-Transfer DMA Cycle Time**

| CPU Operating Mode, Clock Source | Maximum DMA Cycle Time |
|---|---|
| Active mode, MCLK = DCOCLK | 4 MCLK cycles |
| Active mode, MCLK = LFXT1CLK | 4 MCLK cycles |
| Low-power mode LPM0 or LPM1 MCLK = DCOCLK | 5 MCLK cycles |
| Low-power mode LPM3 or LPM4, MCLK = DCOCLK | 5 MCLK cycles + 5 µs(1) |
| Low-power mode LPM0 or LPM1, MCLK = LFXT1CLK | 5 MCLK cycles |
| Low-power mode LPM3, MCLK = LFXT1CLK | 5 MCLK cycles |
| Low-power mode LPM4, MCLK = LFXT1CLK | 5 MCLK cycles + 5 µs(1) |

(1) The additional 5 µs are needed to start the DCOCLK. It is the $t_{(LPMx)}$ parameter in the data sheet.

### 1.2.8 Using DMA With System Interrupts

DMA transfers are not interruptible by system interrupts. System interrupts remain pending until the completion of the transfer. NMIs can interrupt the DMA controller if the ENNMI bit is set.

System interrupt service routines are interrupted by DMA transfers. If an interrupt service routine or other routine must execute with no interruptions, the DMA controller should be disabled prior to executing the routine.

### 1.2.9 DMA Controller Interrupts

Each DMA channel has its own DMAIFG flag. Each DMAIFG flag is set in any mode when the corresponding DMAxSZ register counts to zero. If the corresponding DMAIE and GIE bits are set, an interrupt request is generated.

All DMAIFG flags are prioritized, with DMA0IFG being the highest, and combined to source a single interrupt vector. The highest-priority enabled interrupt generates a number in the DMAIV register. This number can be evaluated or added to the program counter (PC) to automatically enter the appropriate software routine. Disabled DMA interrupts do not affect the DMAIV value.

Any access (read or write) of the DMAIV register automatically resets the highest pending interrupt flag. If another interrupt flag is set, another interrupt is immediately generated after servicing the initial interrupt. For example, assume that DMA0 has the highest priority. If the DMA0IFG and DMA2IFG flags are set when the interrupt service routine accesses the DMAIV register, DMA0IFG is reset automatically. After the RETI instruction of the interrupt service routine is executed, the DMA2IFG generates another interrupt.

### 1.2.9.1 DMAIV Software Example

The following software example shows the recommended use of DMAIV and the handling overhead for an eight channel DMA controller. The DMAIV value is added to the PC to automatically jump to the appropriate routine.

The numbers at the right margin show the necessary CPU cycles for each instruction. The software overhead for different interrupt sources includes interrupt latency and return-from-interrupt cycles, but not the task handling itself.

```
;Interrupt handler for DMAxIFG                    Cycles

DMA_HND      ...              ; Interrupt latency          6
         ADD      &DMAIV,PC   ; Add offset to Jump table   3
         RETI                 ; Vector  0: No interrupt    5
         JMP      DMA0_HND    ; Vector  2: DMA channel 0   2
         JMP      DMA1_HND    ; Vector  4: DMA channel 1   2
         JMP      DMA2_HND    ; Vector  6: DMA channel 2   2
         JMP      DMA3_HND    ; Vector  8: DMA channel 3   2
         JMP      DMA4_HND    ; Vector 10: DMA channel 4   2
         JMP      DMA5_HND    ; Vector 12: DMA channel 5   2
         JMP      DMA6_HND    ; Vector 14: DMA channel 6   2
         JMP      DMA7_HND    ; Vector 16: DMA channel 7   2

DMA7_HND                     ; Vector 16: DMA channel 7
         ...                 ; Task starts here
         RETI                ; Back to main program       5

DMA6_HND                     ; Vector 14: DMA channel 6
         ...                 ; Task starts here
         RETI                ; Back to main program       5

DMA5_HND                     ; Vector 12: DMA channel 5
         ...                 ; Task starts here
         RETI                ; Back to main program       5

DMA4_HND                     ; Vector 10: DMA channel 4
         ...                 ; Task starts here
         RETI                ; Back to main program       5

DMA3_HND                     ; Vector 8: DMA channel 3
         ...                 ; Task starts here
         RETI                ; Back to main program       5

DMA2_HND                     ; Vector 6: DMA channel 2
         ...                 ; Task starts here
         RETI                ; Back to main program       5

DMA1_HND                     ; Vector 4: DMA channel 1
         ...                 ; Task starts here
         RETI                ; Back to main program       5

DMA0_HND                     ; Vector 2: DMA channel 0
         ...                 ; Task starts here
         RETI                ; Back to main program       5
```

### 1.2.10 Using the USCI_B I²C Module With the DMA Controller

The USCI_B I²C module provides two trigger sources for the DMA controller. The USCI_B I²C module can trigger a transfer when new I²C data is received and when the transmit data is needed.

### 1.2.11 Using ADC10 With the DMA Controller

MSP430 devices with an integrated DMA controller can automatically move data from the ADC10MEM0 register to another location. DMA transfers are done without CPU intervention and are independent of any low-power modes. The DMA controller increases throughput of the ADC10 module and enhances low-power applications by allowing the CPU to remain off while data transfers occur.

A transfer is triggered when the conversion is completed and the ADC10IFG0 is set as long as the ADC10IE0 bit is reset. Setting the ADC10IFG0 with software does not trigger a transfer. The ADC10IFG0 flag is automatically reset when the ADC10MEM0 register is accessed by the DMA controller.

### 1.2.12 Using ADC12 With the DMA Controller

MSP430 devices with an integrated DMA controller can automatically move data from any ADC12MEMx register to another location. DMA transfers are done without CPU intervention and are independent of any low-power modes. The DMA controller increases throughput of the ADC12 module and enhances low-power applications by allowing the CPU to remain off while data transfers occur.

DMA transfers can be triggered from any ADC12IFG flag as long as the corresponding ADC12IE bit is reset. When CONSEQx = {0,2}, the ADC12IFG flag for the ADC12MEMx used for the conversion can trigger a DMA transfer. When CONSEQx = {1,3}, the ADC12IFG flag for the last ADC12MEMx in the sequence can trigger a DMA transfer. Any ADC12IFG flag is automatically cleared when the DMA controller accesses the corresponding ADC12MEMx.

### 1.2.13 Using DAC12 With the DMA Controller

MSP430 devices with an integrated DMA controller can automatically move data to the DAC12_xDAT register. DMA transfers are done without CPU intervention and are independent of any low-power modes. The DMA controller increases throughput to the DAC12 module and enhances low-power applications by allowing the CPU to remain off while data transfers occur.

Applications that require periodic waveform generation can benefit from using the DMA controller with the DAC12. For example, an application that produces a sinusoidal waveform may store the sinusoid values in a table. The DMA controller can continuously and automatically transfer the values to the DAC12 at specific intervals to create the sinusoid with no CPU execution. The DAC12_xCTL DAC12IFG flag is automatically cleared when the DMA controller accesses the DAC12_xDAT register.

## 1.3 DMA Registers

The DMA module registers are listed in Table 1-4. The base addresses can be found in the device-specific data sheet. Each channel starts at its respective base address. The address offsets are listed in Table 1-4.

**Table 1-4. DMA Registers**

| Offset | Acronym | Register Name | Type | Access | Reset | Section |
|---|---|---|---|---|---|---|
| 00h | DMACTL0 | DMA Control 0 | Read/write | Word | 0000h | Section 1.3.1 |
| 02h | DMACTL1 | DMA Control 1 | Read/write | Word | 0000h | Section 1.3.2 |
| 04h | DMACTL2 | DMA Control 2 | Read/write | Word | 0000h | Section 1.3.3 |
| 06h | DMACTL3 | DMA Control 3 | Read/write | Word | 0000h | Section 1.3.4 |
| 08h | DMACTL4 | DMA Control 4 | Read/write | Word | 0000h | Section 1.3.5 |
| 0Eh | DMAIV | DMA Interrupt Vector | Read only | Word | 0000h | Section 1.3.10 |
| 00h | DMA0CTL | DMA Channel 0 Control | Read/write | Word | 0000h | Section 1.3.6 |
| 02h | DMA0SA | DMA Channel 0 Source Address | Read/write | Word, double word | undefined | Section 1.3.7 |
| 06h | DMA0DA | DMA Channel 0 Destination Address | Read/write | Word, double word | undefined | Section 1.3.8 |
| 0Ah | DMA0SZ | DMA Channel 0 Transfer Size | Read/write | Word | undefined | Section 1.3.9 |
| 00h | DMA1CTL | DMA Channel 1 Control | Read/write | Word | 0000h | Section 1.3.6 |
| 02h | DMA1SA | DMA Channel 1 Source Address | Read/write | Word, double word | undefined | Section 1.3.7 |
| 06h | DMA1DA | DMA Channel 1 Destination Address | Read/write | Word, double word | undefined | Section 1.3.8 |
| 0Ah | DMA1SZ | DMA Channel 1 Transfer Size | Read/write | Word | undefined | Section 1.3.9 |
| 00h | DMA2CTL | DMA Channel 2 Control | Read/write | Word | 0000h | Section 1.3.6 |
| 02h | DMA2SA | DMA Channel 2 Source Address | Read/write | Word, double word | undefined | Section 1.3.7 |
| 06h | DMA2DA | DMA Channel 2 Destination Address | Read/write | Word, double word | undefined | Section 1.3.8 |
| 0Ah | DMA2SZ | DMA Channel 2 Transfer Size | Read/write | Word | undefined | Section 1.3.9 |
| 00h | DMA3CTL | DMA Channel 3 Control | Read/write | Word | 0000h | Section 1.3.6 |
| 02h | DMA3SA | DMA Channel 3 Source Address | Read/write | Word, double word | undefined | Section 1.3.7 |
| 06h | DMA3DA | DMA Channel 3 Destination Address | Read/write | Word, double word | undefined | Section 1.3.8 |
| 0Ah | DMA3SZ | DMA Channel 3 Transfer Size | Read/write | Word | undefined | Section 1.3.9 |
| 00h | DMA4CTL | DMA Channel 4 Control | Read/write | Word | 0000h | Section 1.3.6 |
| 02h | DMA4SA | DMA Channel 4 Source Address | Read/write | Word, double word | undefined | Section 1.3.7 |
| 06h | DMA4DA | DMA Channel 4 Destination Address | Read/write | Word, double word | undefined | Section 1.3.8 |
| 0Ah | DMA4SZ | DMA Channel 4 Transfer Size | Read/write | Word | undefined | Section 1.3.9 |
| 00h | DMA5CTL | DMA Channel 5 Control | Read/write | Word | 0000h | Section 1.3.6 |
| 02h | DMA5SA | DMA Channel 5 Source Address | Read/write | Word, double word | undefined | Section 1.3.7 |
| 06h | DMA5DA | DMA Channel 5 Destination Address | Read/write | Word, double word | undefined | Section 1.3.8 |
| 0Ah | DMA5SZ | DMA Channel 5 Transfer Size | Read/write | Word | undefined | Section 1.3.9 |

**Table 1-4. DMA Registers (continued)**

| Offset | Acronym | Register Name | Type | Access | Reset | Section |
|---|---|---|---|---|---|---|
| 00h | DMA6CTL | DMA Channel 6 Control | Read/write | Word | 0000h | Section 1.3.6 |
| 02h | DMA6SA | DMA Channel 6 Source Address | Read/write | Word, double word | undefined | Section 1.3.7 |
| 06h | DMA6DA | DMA Channel 6 Destination Address | Read/write | Word, double word | undefined | Section 1.3.8 |
| 0Ah | DMA6SZ | DMA Channel 6 Transfer Size | Read/write | Word | undefined | Section 1.3.9 |
| 00h | DMA7CTL | DMA Channel 7 Control | Read/write | Word | 0000h | Section 1.3.6 |
| 02h | DMA7SA | DMA Channel 7 Source Address | Read/write | Word, double word | undefined | Section 1.3.7 |
| 06h | DMA7DA | DMA Channel 7 Destination Address | Read/write | Word, double word | undefined | Section 1.3.8 |
| 0Ah | DMA7SZ | DMA Channel 7 Transfer Size | Read/write | Word | undefined | Section 1.3.9 |

### 1.3.1 DMACTL0 Register

DMA Control 0 Register

**Figure 1-6. DMACTL0 Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| Reserved | | | DMA1TSEL | | | | |
| r0 | r0 | r0 | rw-(0) | rw-(0) | rw-(0) | rw-(0) | rw-(0) |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | | | DMA0TSEL | | | | |
| r0 | r0 | r0 | rw-(0) | rw-(0) | rw-(0) | rw-(0) | rw-(0) |

**Table 1-5. DMACTL0 Register Description**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-13 | Reserved | R | 0h | Reserved. Always reads as 0. |
| 12-8 | DMA1TSEL | RW | 0h | DMA 1 trigger select. These bits select the DMA transfer trigger. See the device-specific data sheet for number of channels and trigger assignment.<br>00000b = DMA1TRIG0<br>00001b = DMA1TRIG1<br>00010b = DMA1TRIG2<br>⋮<br>11110b = DMA1TRIG30<br>11111b = DMA1TRIG31 |
| 7-5 | Reserved | R | 0h | Reserved. Always reads as 0. |
| 4-0 | DMA0TSEL | RW | 0h | DMA 0 trigger select. These bits select the DMA transfer trigger. See the device-specific data sheet for number of channels and trigger assignment.<br>00000b = DMA0TRIG0<br>00001b = DMA0TRIG1<br>00010b = DMA0TRIG2<br>⋮<br>11110b = DMA0TRIG30<br>11111b = DMA0TRIG31 |

### 1.3.2 DMACTL1 Register

DMA Control 1 Register

**Figure 1-7. DMACTL1 Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| Reserved | | | DMA3TSEL | | | | |
| r0 | r0 | r0 | rw-(0) | rw-(0) | rw-(0) | rw-(0) | rw-(0) |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | | | DMA2TSEL | | | | |
| r0 | r0 | r0 | rw-(0) | rw-(0) | rw-(0) | rw-(0) | rw-(0) |

**Table 1-6. DMACTL1 Register Description**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-13 | Reserved | R | 0h | Reserved. Always reads as 0. |
| 12-8 | DMA3TSEL | RW | 0h | DMA 3 trigger select. These bits select the DMA transfer trigger. See the device-specific data sheet for number of channels and trigger assignment.<br>00000b = DMA3TRIG0<br>00001b = DMA3TRIG1<br>00010b = DMA3TRIG2<br>⋮<br>11110b = DMA3TRIG30<br>11111b = DMA3TRIG31 |
| 7-5 | Reserved | R | 0h | Reserved. Always reads as 0. |
| 4-0 | DMA2TSEL | RW | 0h | DMA 2 trigger select. These bits select the DMA transfer trigger. See the device-specific data sheet for number of channels and trigger assignment.<br>00000b = DMA2TRIG0<br>00001b = DMA2TRIG1<br>00010b = DMA2TRIG2<br>⋮<br>11110b = DMA2TRIG30<br>11111b = DMA2TRIG31 |

### 1.3.3 DMACTL2 Register

DMA Control 2 Register

**Figure 1-8. DMACTL2 Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| Reserved | | | DMA5TSEL | | | | |
| r0 | r0 | r0 | rw-(0) | rw-(0) | rw-(0) | rw-(0) | rw-(0) |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | | | DMA4TSEL | | | | |
| r0 | r0 | r0 | rw-(0) | rw-(0) | rw-(0) | rw-(0) | rw-(0) |

**Table 1-7. DMACTL2 Register Description**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-13 | Reserved | R | 0h | Reserved. Always reads as 0. |
| 12-8 | DMA5TSEL | RW | 0h | DMA 5 trigger select. These bits select the DMA transfer trigger. See the device-specific data sheet for number of channels and trigger assignment.<br>00000b = DMA5TRIG0<br>00001b = DMA5TRIG1<br>00010b = DMA5TRIG2<br>⋮<br>11110b = DMA5TRIG30<br>11111b = DMA5TRIG31 |
| 7-5 | Reserved | R | 0h | Reserved. Always reads as 0. |
| 4-0 | DMA4TSEL | RW | 0h | DMA 4 trigger select. These bits select the DMA transfer trigger. See the device-specific data sheet for number of channels and trigger assignment.<br>00000b = DMA4TRIG0<br>00001b = DMA4TRIG1<br>00010b = DMA4TRIG2<br>⋮<br>11110b = DMA4TRIG30<br>11111b = DMA4TRIG31 |

### 1.3.4 DMACTL3 Register

DMA Control 3 Register

**Figure 1-9. DMACTL3 Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| Reserved | | | DMA7TSEL | | | | |
| r0 | r0 | r0 | rw-(0) | rw-(0) | rw-(0) | rw-(0) | rw-(0) |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | | | DMA6TSEL | | | | |
| r0 | r0 | r0 | rw-(0) | rw-(0) | rw-(0) | rw-(0) | rw-(0) |

**Table 1-8. DMACTL3 Register Description**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-13 | Reserved | R | 0h | Reserved. Always reads as 0. |
| 12-8 | DMA7TSEL | RW | 0h | DMA 7 trigger select. These bits select the DMA transfer trigger. See the device-specific data sheet for number of channels and trigger assignment.<br>00000b = DMA7TRIG0<br>00001b = DMA7TRIG1<br>00010b = DMA7TRIG2<br>⋮<br>11110b = DMA7TRIG30<br>11111b = DMA7TRIG31 |
| 7-5 | Reserved | R | 0h | Reserved. Always reads as 0. |
| 4-0 | DMA6TSEL | RW | 0h | DMA 6 trigger select. These bits select the DMA transfer trigger. See the device-specific data sheet for number of channels and trigger assignment.<br>00000b = DMA6TRIG0<br>00001b = DMA6TRIG1<br>00010b = DMA6TRIG2<br>⋮<br>11110b = DMA6TRIG30<br>11111b = DMA6TRIG31 |

### 1.3.5 DMACTL4 Register

DMA Control 4 Register

**Figure 1-10. DMACTL4 Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | |
| r0 | r0 | r0 | r0 | r0 | r0 | r0 | r0 |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | | | | | DMARMWDIS | ROUNDROBIN | ENNMI |
| r0 | r0 | r0 | r0 | r0 | rw-(0) | rw-(0) | rw-(0) |

**Table 1-9. DMACTL4 Register Description**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-3 | Reserved | R | 0h | Reserved. Always reads as 0. |
| 2 | DMARMWDIS | RW | 0h | Read-modify-write disable. When set, this bit inhibits any DMA transfers from occurring during CPU read-modify-write operations.<br>0b = DMA transfers can occur during read-modify-write CPU operations.<br>1b = DMA transfers inhibited during read-modify-write CPU operations |
| 1 | ROUNDROBIN | RW | 0h | Round robin. This bit enables the round-robin DMA channel priorities.<br>0b = DMA channel priority is DMA0-DMA1-DMA2 - ..... -DMA7.<br>1b = DMA channel priority changes with each transfer. |
| 0 | ENNMI | RW | 0h | Enable NMI. This bit enables the interruption of a DMA transfer by an NMI. When an NMI interrupts a DMA transfer, the current transfer is completed normally, further transfers are stopped and DMAABORT is set.<br>0b = NMI does not interrupt DMA transfer.<br>1b = NMI interrupts a DMA transfer. |

### 1.3.6 DMAxCTL Register

DMA Channel x Control Register

**Figure 1-11. DMAxCTL Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| Reserved | DMADT | | | DMADSTINCR | | DMASRCINCR | |
| r0 | rw-(0) | rw-(0) | rw-(0) | rw-(0) | rw-(0) | rw-(0) | rw-(0) |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| DMADSTBYTE | DMASRCBYTE | DMALEVEL | DMAEN | DMAIFG | DMAIE | DMAABORT | DMAREQ |
| rw-(0) | rw-(0) | rw-(0) | rw-(0) | rw-(0) | rw-(0) | rw-(0) | rw-(0) |

**Table 1-10. DMAxCTL Register Description**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15 | Reserved | R | 0h | Reserved. Always reads as 0. |
| 14-12 | DMADT | RW | 0h | DMA transfer mode<br>000b = Single transfer<br>001b = Block transfer<br>010b = Burst-block transfer<br>011b = Burst-block transfer<br>100b = Repeated single transfer<br>101b = Repeated block transfer<br>110b = Repeated burst-block transfer<br>111b = Repeated burst-block transfer |
| 11-10 | DMADSTINCR | RW | 0h | DMA destination increment. This bit selects automatic incrementing or decrementing of the destination address after each byte or word transfer. When DMADSTBYTE = 1, the destination address increments/decrements by one. When DMADSTBYTE = 0, the destination address increments/decrements by two. The DMAxDA is copied into a temporary register and the temporary register is incremented or decremented. DMAxDA is not incremented or decremented.<br>00b = Destination address is unchanged.<br>01b = Destination address is unchanged.<br>10b = Destination address is decremented.<br>11b = Destination address is incremented. |
| 9-8 | DMASRCINCR | RW | 0h | DMA source increment. This bit selects automatic incrementing or decrementing of the source address for each byte or word transfer. When DMASRCBYTE = 1, the source address increments/decrements by one. When DMASRCBYTE = 0, the source address increments/decrements by two. The DMAxSA is copied into a temporary register and the temporary register is incremented or decremented. DMAxSA is not incremented or decremented.<br>00b = Source address is unchanged.<br>01b = Source address is unchanged.<br>10b = Source address is decremented.<br>11b = Source address is incremented. |
| 7 | DMADSTBYTE | RW | 0h | DMA destination byte. This bit selects the destination as a byte or word.<br>0b = Word<br>1b = Byte |
| 6 | DMASRCBYTE | RW | 0h | DMA source byte. This bit selects the source as a byte or word.<br>0b = Word<br>1b = Byte |
| 5 | DMALEVEL | RW | 0h | DMA level. This bit selects between edge-sensitive and level-sensitive triggers.<br>0b = Edge sensitive (rising edge)<br>1b = Level sensitive (high level) |
| 4 | DMAEN | RW | 0h | DMA enable<br>0b = Disabled<br>1b = Enabled |

**Table 1-10. DMAxCTL Register Description (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 3 | DMAIFG | RW | 0h | DMA interrupt flag<br>0b = No interrupt pending<br>1b = Interrupt pending |
| 2 | DMAIE | RW | 0h | DMA interrupt enable<br>0b = Disabled<br>1b = Enabled |
| 1 | DMAABORT | RW | 0h | DMA abort. This bit indicates if a DMA transfer was interrupt by an NMI.<br>0b = DMA transfer not interrupted<br>1b = DMA transfer interrupted by NMI |
| 0 | DMAREQ | RW | 0h | DMA request. Software-controlled DMA start. DMAREQ is reset automatically.<br>0b = No DMA start<br>1b = Start DMA |

### 1.3.7 DMAxSA Register

DMA Channel x Source Address Register

**Figure 1-12. DMAxSA Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | |
| r0 | r0 | r0 | r0 | r0 | r0 | r0 | r0 |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Reserved | | | | DMAxSA | | | |
| r0 | r0 | r0 | r0 | rw | rw | rw | rw |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| DMAxSA | | | | | | | |
| rw | rw | rw | rw | rw | rw | rw | rw |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| DMAxSA | | | | | | | |
| rw | rw | rw | rw | rw | rw | rw | rw |

**Table 1-11. DMAxSA Register Description**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-20 | Reserved | R | 0h | Reserved. Always reads as 0. |
| 19-0 | DMAxSA | RW | undefined | DMA source address. The source address register points to the DMA source address for single transfers or the first source address for block transfers. The source address register remains unchanged during block and burst-block transfers. There are two words for the DMAxSA register. Bits 31-20 are reserved and always read as zero. Reading or writing bits 19-16 requires the use of extended instructions. When writing to DMAxSA with word instructions, bits 19-16 are cleared. |

### 1.3.8 DMAxDA Register

DMA Channel x Destination Address Register

**Figure 1-13. DMAxDA Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | |
| r0 | r0 | r0 | r0 | r0 | r0 | r0 | r0 |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Reserved | | | | DMAxDA | | | |
| r0 | r0 | r0 | r0 | rw | rw | rw | rw |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| DMAxDA | | | | | | | |
| rw | rw | rw | rw | rw | rw | rw | rw |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| DMAxDA | | | | | | | |
| rw | rw | rw | rw | rw | rw | rw | rw |

**Table 1-12. DMAxDA Register Description**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-20 | Reserved | R | 0h | Reserved. Always reads as 0. |
| 19-0 | DMAxDA | RW | undefined | DMA destination address. The destination address register points to the DMA destination address for single transfers or the first destination address for block transfers. The destination address register remains unchanged during block and burst-block transfers. There are two words for the DMAxDA register. Bits 31-20 are reserved and always read as zero. Reading or writing bits 19-16 requires the use of extended instructions. When writing to DMAxDA with word instructions, bits 19-16 are cleared. |

### 1.3.9 DMAxSZ Register

DMA Channel x Size Address Register

**Figure 1-14. DMAxSZ Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| DMAxSZ | | | | | | | |
| rw | rw | rw | rw | rw | rw | rw | rw |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| DMAxSZ | | | | | | | |
| rw | rw | rw | rw | rw | rw | rw | rw |

**Table 1-13. DMAxSZ Register Description**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-0 | DMAxSZ | RW | undefined | DMA size. The DMA size register defines the number of byte/word data per block transfer. DMAxSZ register decrements with each word or byte transfer. When DMAxSZ decrements to 0, it is immediately and automatically reloaded with its previously initialized value.<br>00000h = Transfer is disabled.<br>00001h = One byte or word is transferred.<br>00002h = Two bytes or words are transferred.<br>⋮<br>0FFFFh = 65535 bytes or words are transferred. |

### 1.3.10 DMAIV Register

DMA Interrupt Vector Register

**Figure 1-15. DMAIV Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| DMAIV | | | | | | | |
| r0 | r0 | r0 | r0 | r0 | r0 | r0 | r0 |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| DMAIV | | | | | | | |
| r0 | r0 | r-(0) | r-(0) | r-(0) | r-(0) | r-(0) | r0 |

**Table 1-14. DMAIV Register Description**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-0 | DMAIV | R | 0h | DMA interrupt vector value<br>00h = No interrupt pending<br>02h = Interrupt Source: DMA channel 0; Interrupt Flag: DMA0IFG; Interrupt Priority: Highest<br>04h = Interrupt Source: DMA channel 1; Interrupt Flag: DMA1IFG<br>06h = Interrupt Source: DMA channel 2; Interrupt Flag: DMA2IFG<br>08h = Interrupt Source: DMA channel 3; Interrupt Flag: DMA3IFG<br>0Ah = Interrupt Source: DMA channel 4; Interrupt Flag: DMA4IFG<br>0Ch = Interrupt Source: DMA channel 5; Interrupt Flag: DMA5IFG<br>0Eh = Interrupt Source: DMA channel 6; Interrupt Flag: DMA6IFG<br>10h = Interrupt Source: DMA channel 7; Interrupt Flag: DMA7IFG; Interrupt Priority: Lowest |

## IMPORTANT NOTICE FOR TI DESIGN INFORMATION AND RESOURCES

Texas Instruments Incorporated ('TI") technical, application or other design advice, services or information, including, but not limited to, reference designs and materials relating to evaluation modules, (collectively, "TI Resources") are intended to assist designers who are developing applications that incorporate TI products; by downloading, accessing or using any particular TI Resource in any way, you (individually or, if you are acting on behalf of a company, your company) agree to use it solely for this purpose and subject to the terms of this Notice.

TI's provision of TI Resources does not expand or otherwise alter TI's applicable published warranties or warranty disclaimers for TI products, and no additional obligations or liabilities arise from TI providing such TI Resources. TI reserves the right to make corrections, enhancements, improvements and other changes to its TI Resources.

You understand and agree that you remain responsible for using your independent analysis, evaluation and judgment in designing your applications and that you have full and exclusive responsibility to assure the safety of your applications and compliance of your applications (and of all TI products used in or for your applications) with all applicable regulations, laws and other applicable requirements. You represent that, with respect to your applications, you have all the necessary expertise to create and implement safeguards that (1) anticipate dangerous consequences of failures, (2) monitor failures and their consequences, and (3) lessen the likelihood of failures that might cause harm and take appropriate actions. You agree that prior to using or distributing any applications that include TI products, you will thoroughly test such applications and the functionality of such TI products as used in such applications. TI has not conducted any testing other than that specifically described in the published documentation for a particular TI Resource.

You are authorized to use, copy and modify any individual TI Resource only in connection with the development of applications that include the TI product(s) identified in such TI Resource. NO OTHER LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE TO ANY OTHER TI INTELLECTUAL PROPERTY RIGHT, AND NO LICENSE TO ANY TECHNOLOGY OR INTELLECTUAL PROPERTY RIGHT OF TI OR ANY THIRD PARTY IS GRANTED HEREIN, including but not limited to any patent right, copyright, mask work right, or other intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information regarding or referencing third-party products or services does not constitute a license to use such products or services, or a warranty or endorsement thereof. Use of TI Resources may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

TI RESOURCES ARE PROVIDED "AS IS" AND WITH ALL FAULTS. TI DISCLAIMS ALL OTHER WARRANTIES OR REPRESENTATIONS, EXPRESS OR IMPLIED, REGARDING TI RESOURCES OR USE THEREOF, INCLUDING BUT NOT LIMITED TO ACCURACY OR COMPLETENESS, TITLE, ANY EPIDEMIC FAILURE WARRANTY AND ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NON-INFRINGEMENT OF ANY THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

TI SHALL NOT BE LIABLE FOR AND SHALL NOT DEFEND OR INDEMNIFY YOU AGAINST ANY CLAIM, INCLUDING BUT NOT LIMITED TO ANY INFRINGEMENT CLAIM THAT RELATES TO OR IS BASED ON ANY COMBINATION OF PRODUCTS EVEN IF DESCRIBED IN TI RESOURCES OR OTHERWISE. IN NO EVENT SHALL TI BE LIABLE FOR ANY ACTUAL, DIRECT, SPECIAL, COLLATERAL, INDIRECT, PUNITIVE, INCIDENTAL, CONSEQUENTIAL OR EXEMPLARY DAMAGES IN CONNECTION WITH OR ARISING OUT OF TI RESOURCES OR USE THEREOF, AND REGARDLESS OF WHETHER TI HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

You agree to fully indemnify TI and its representatives against any damages, costs, losses, and/or liabilities arising out of your non-compliance with the terms and provisions of this Notice.

This Notice applies to TI Resources. Additional terms apply to the use and purchase of certain types of materials, TI products and services. These include; without limitation, TI's standard terms for semiconductor products http://www.ti.com/sc/docs/stdterms.htm), evaluation modules, and samples (http://www.ti.com/sc/docs/sampterms.htm).

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265