TMS320VC5501/5502 DSP Direct Memory Access (DMA) Controller Reference Guide

Literature Number: SPRU613G
March 2005

TEXAS INSTRUMENTS

Read This First

About This Manual

This manual describes the features and operation of the direct memory access (DMA) controller that is available on the TMS320VC5501 and TMS320VC5502 digital signal processors (DSPs) in the TMS320C55x™ (C55x™) DSP generation. This DMA controller allows movement of data among internal memory, external memory, and peripherals to occur without intervention from the CPU and in the background of CPU operation.

Notational Conventions

This document uses the following conventions:

❑ In most cases, hexadecimal numbers are shown with the suffix h. For example, the following number is a hexadecimal 40 (decimal 64):

40h

❑ Similarly, binary numbers often are shown with the suffix b. For example, the following number is the decimal number 4 shown in binary form:

0100b

Related Documentation From Texas Instruments

The following documents describe the C55x devices and related support tools. Copies of these documents are available on the Internet at www.ti.com. *Tip:* Enter the literature number in the search box provided at www.ti.com.

TMS320C55x Technical Overview (literature number SPRU393). This overview is an introduction to the TMS320C55x DSPs, the latest generation of fixed-point DSPs in the TMS320C5000™ DSP platform. Like the previous generations, this processor is optimized for high performance and low-power operation. This book describes the CPU architecture, low-power enhancements, and embedded emulation features.

TMS320C55x DSP CPU Reference Guide (literature number SPRU371) describes the architecture, registers, and operation of the CPU for the TMS320C55x DSPs.

TMS320C55x DSP Peripherals Reference Guide (literature number SPRU317) describes the peripherals, interfaces, and related hardware that are available on TMS320C55x DSPs.

TMS320C55x DSP Algebraic Instruction Set Reference Guide (literature number SPRU375) describes the TMS320C55x DSP algebraic instructions individually. Also includes a summary of the instruction set, a list of the instruction opcodes, and a cross-reference to the mnemonic instruction set.

TMS320C55x DSP Mnemonic Instruction Set Reference Guide (literature number SPRU374) describes the TMS320C55x DSP mnemonic instructions individually. Also includes a summary of the instruction set, a list of the instruction opcodes, and a cross-reference to the algebraic instruction set.

TMS320C55x Optimizing C/C++ Compiler User's Guide (literature number SPRU281) describes the TMS320C55x™ C/C++ Compiler. This C/C++ compiler accepts ISO standard C and C++ source code and produces assembly language source code for TMS320C55x devices.

TMS320C55x Assembly Language Tools User's Guide (literature number SPRU280) describes the assembly language tools (assembler, linker, and other tools used to develop assembly language code), assembler directives, macros, common object file format, and symbolic debugging directives for TMS320C55x devices.

TMS320C55x DSP Programmer's Guide (literature number SPRU376) describes ways to optimize C and assembly code for the TMS320C55x DSPs and explains how to write code that uses special features and instructions of the DSPs.

Trademarks

TMS320, TMS320C5000, TMS320C55x, and C55x are trademarks of Texas Instruments.

Trademarks are the property of their respective owners.

Contents

Figures

Tables

DMA Controller

This document describes the direct memory access (DMA) controller of the TMS320VC5501/TMS320VC5502 digital signal processors (DSPs). This DMA controller allows movement of data among internal memory, external memory, and peripherals to occur without intervention from the CPU and in the background of CPU operation.

1 Introduction to the DMA Controller

Acting in the background of CPU operation, the DMA controller can transfer data among internal memory, external memory, and on-chip peripherals.

1.1 Key Features of the DMA Controller

The DMA controller has the following important features:

- ❑ Operation that is independent of the CPU.
- ❑ Four standard ports: Two for internal dual-access RAM (DARAM), one for external memory, and one for peripherals.
- ❑ Six channels, which allow the DMA controller to keep track of the context of six independent block transfers among the standard ports.
- ❑ Bits for assigning each channel a low priority or a high priority. For details, see *Service Chain* on page 19.
- ❑ Event synchronization. DMA transfers in each channel can be made dependent on the occurrence of selected events. For details, see *Synchronizing Channel Activity* on page 30.
- ❑ An interrupt for each channel. Each channel can send an interrupt to the CPU on completion of certain operational events. See *Monitoring Channel Activity* on page 32.
- ❑ Software-selectable options for updating addresses for the sources and destinations of data transfers.
- ❑ A dedicated idle domain. You can put the DMA controller into a low-power state by turning off this domain. Each multichannel buffered serial port (McBSP) on the C55x DSP has the ability to temporarily take the DMA domain out of this idle state when the McBSP needs the DMA controller.

To read about the registers used to program the DMA controller, see section 15 on page 38.

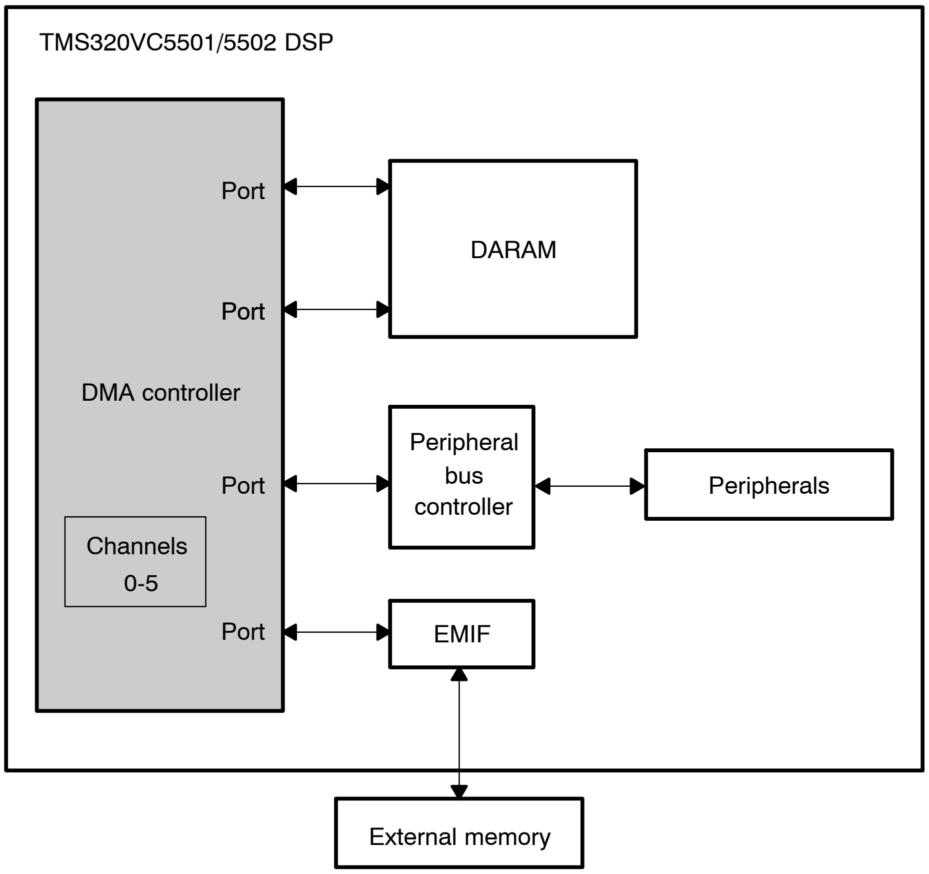
1.2 Block Diagram of the DMA Controller

Figure 1 is a conceptual diagram of connections between the DMA controller and other parts of the DSP. The DMA controller has four ports:

- ❑ Two ports for internal dual-access RAM (DARAM). For ease of reference, these ports are called internal memory port 0 and internal memory port 1 throughout this document.
- ❑ One port for external memory. The external memory interface (EMIF) connects the port to the external memory.
- ❑ One port for peripherals. A peripheral bus controller connects the port to the peripherals.

Data transfers among the ports occur in the six DMA channels. (The DMA channels are described on page 12.) It is possible for multiple channels to request access to the same port at the same time. To arbitrate simultaneous requests, the DMA controller has one programmable service chain that is used by each of the ports. For details on the service chain, see page 19.

Figure 1. Conceptual Block Diagram of the DMA Controller Connections



1.3 DMA Requests Versus CPU Requests for Internal Memory

If the CPU and the DMA controller simultaneously request access to the same DARAM block in internal memory, CPU requests always have priority over DMA requests. The DMA requests to a DARAM block will be serviced when there are no more CPU requests. Refer to the device-specific data manual for specific information on the start and end addresses for each DARAM block.

2 Channels and Port Accesses

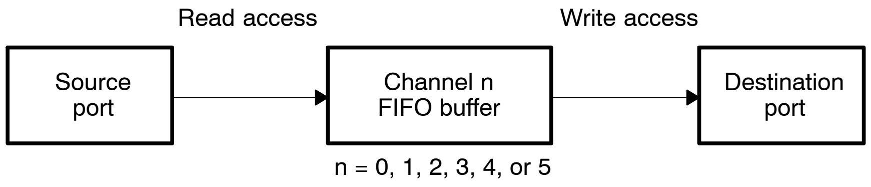
The DMA controller has six paths, called **channels**, to transfer data among the four ports (two for DARAM, one for external memory, and one for peripherals). Each channel reads data from one port (from the source) and writes data to that same port or another port (to the destination).

Each channel has a first in, first out (FIFO) buffer that allows the data transfer to occur in two stages (see Figure 2):

Port read access	Transfer of data from the source port to the channel FIFO buffer.
Port write access	Transfer of data from the channel FIFO buffer to the destination port.

The FIFO buffer in each channel is eight 32-bit words deep.

Figure 2. The Two Parts of a DMA Transfer



The different ports in the DMA can be categorized as event driven and non−event driven (see Table 1). This difference between ports is significant when synchronization is used to transfer data, as described in section 11, page 30.

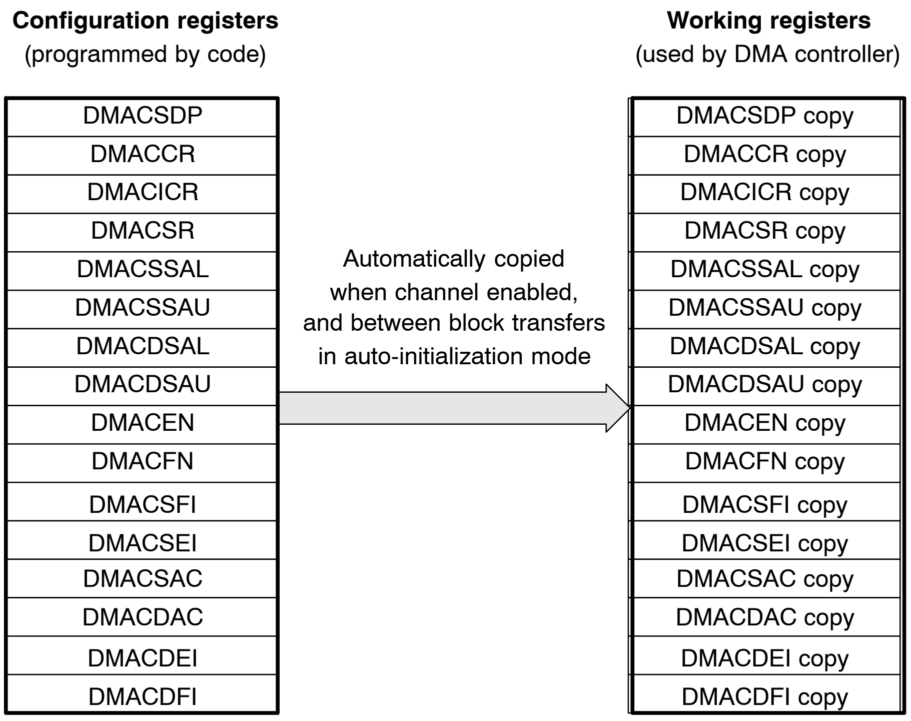
Table 1. Event and Non-Event Driven Ports

Port	Category
DARAM	Non-event driven
External memory	Non-event driven
Peripheral	Event driven

The set of conditions under which transfers occur in a channel is called the **channel context**. Each of the six channels contains a register structure for programming and updating the channel context (see Figure 3). Your code modifies the configuration registers. When it is time for data transferring, the contents of the configuration registers are copied to the working registers, and the DMA controller uses the working register values to control channel activity. The copy from the configuration registers to the working registers occurs whenever your code enables the channel (EN = 1 in DMACCR). In addition, if the auto-initialization mode is on (AUTOINIT = 1 in DMACCR), the copy occurs between block transfers. For more information about the DMA controller registers, see page 38.

Some configuration registers can be programmed for the next block transfer while the DMA is still running the current context from the working registers. The next transfer will use the new configuration without stopping the DMA. The configuration registers that should not be configured in this manner are DMAGCR, DMAGTCR, DMACSDP, DMACCR, DMACICR and DMACSR. Modification of these registers while the DMA channel is running may cause unpredictable operation of the channel.

Figure 3. Registers for Controlling a Channel's Context



3 Channel Auto-initialization Capability

After a block transfer is completed (all of the elements and frames in a block have been moved), the DMA controller automatically disables the channel. If it is necessary for the channel to be used again, the CPU can reprogram the new channel context and re-enable the DMA channel, or the DMA controller can automatically initialize the new context and re-enable the channel.

When auto-initialization is used, after each block transfer is completed, the DMA controller automatically recopies the channel context from the configuration registers to the working registers and re-enables the channel allowing the channel to run again. Auto-initialization is enabled by setting the AUTOINIT bit in the channel controller register (DMACCR).

Two additional bits in DMACCR, REPEAT and ENDPROG, are used during the auto-initialization operation. REPEAT controls whether the DMA controller waits for an indication from the CPU that the configuration registers are ready to be copied. ENDPROG is a handshaking bit used to communicate between the CPU and the DMA controller regarding the state of the register copy process. Figure 4 shows DMACCR and Table 2 describes AUTOINIT, REPEAT, and ENDPROG. For a complete description of DMACCR, see section 15.3.

There are two methods for using auto-initialization. The same channel context can be repeated on each block transfer, or a new context can be provided for each transfer. These two cases are explained in the following sections.

Figure 4. Channel Control Register (DMACCR)

15	14	13	12	11	10	9	8
				ENDPROG		REPEAT	AUTOINIT
				R/W-0		R/W-0	R/W-0

7	6	5	4	3	2	1	0

Legend: R = Read; W = Write; -*n* = Value after DSP reset

Table 2. Channel Control Register (DMACCR) Field Descriptions

Bit	Field	Value	Description
11	ENDPROG		End-of-programming bit. Each DMA channel has two sets of registers: configuration registers and working registers. When block transfers occur repeatedly because of auto-initialization (AUTOINIT = 1), you can change the context for the next DMA transfer by writing to the configuration registers during the current block transfer. At the end of the current transfer, the contents of the configuration registers are copied into the working registers, and the DMA controller begins the next transfer using the new context. For proper auto-initialization, the CPU must finish programming the configuration registers before the DMA controller copies their contents. The DMA controller automatically clears the ENDPROG bit after copying the configuration registers to the working registers. The CPU can then program the DMA channel context for the next iteration of the transfer by programming the configuration registers. To make sure auto-initialization waits for the CPU, follow this procedure: 1) Make auto-initialization wait for ENDPROG = 1 by clearing the REPEAT bit (REPEAT = 0) 2) Poll for ENDPROG = 0, which indicates that the DMA controller has finished copying the previous context. The configuration registers can now be programmed for the next iteration. 3) Program the configuration registers. 4) Set ENDPROG (ENDPROG = 1) to indicate the end of register programming.
		0	Configuration registers ready for programming / Programming in progress.
		1	End of programming.
9	REPEAT		Repeat condition bit. If auto-initialization is selected for a channel (AUTOINIT = 1), REPEAT specifies one of two special repeat conditions:
		0	Repeat only if ENDPROG = 1. Once the current DMA transfer is complete, auto-initialization will wait for the end-of-programming bit (ENDPROG) bit to be set.
		1	Repeat regardless of ENDPROG. Once the current DMA transfer is complete, auto-initialization occurs regardless of whether ENDPROG is 0 or 1.

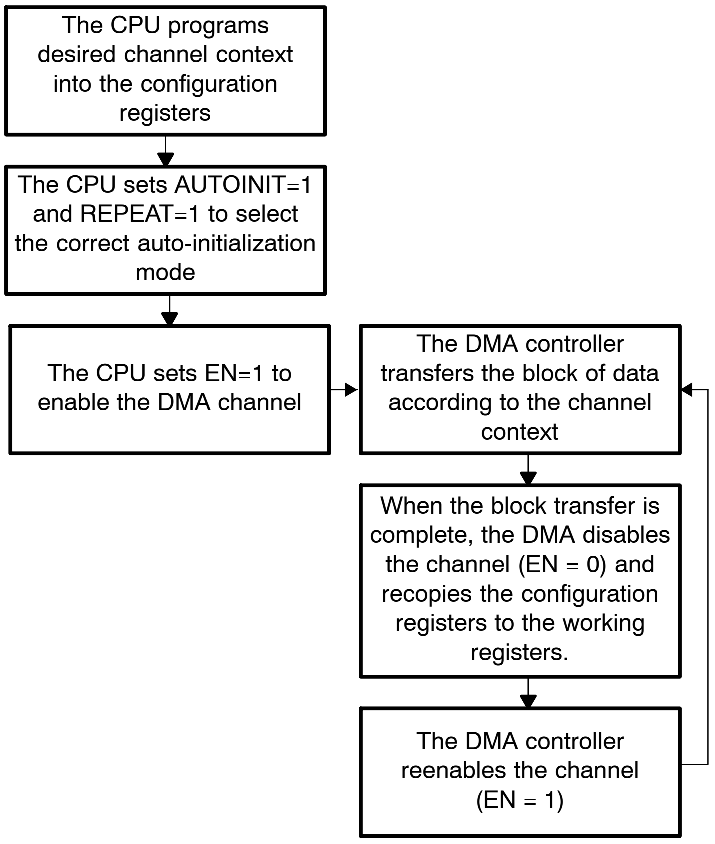
Table 2. Channel Control Register (DMACCR) Field Descriptions (Continued)

Bit	Field	Value	Description
8	AUTOINIT		Auto-initialization bit. The DMA controller supports auto-initialization, which is the automatic reinitialization of the channel between DMA block transfers. Use AUTOINIT to enable or disable this feature.
		0	Auto-initialization is disabled. Activity in the channel stops at the end of the current block transfer. To stop a transfer immediately, clear the channel enable bit (EN).
		1	Auto-initialization is enabled. Once the current block transfer is complete, the DMA controller reinitializes the channel and starts a new block transfer. To stop activity in the channel you have two options: ❑ To stop activity immediately, clear the channel enable bit (EN = 0). ❑ To stop activity after the current block transfer, clear AUTOINIT (AUTOINIT= 0).

3.1 Auto-initialization with Unchanging Context

If the desired context for the channel needs to be repeated but does not need to be changed, then the DMA controller is configured with AUTOINIT = 1 and REPEAT = 1. When REPEAT = 1, the DMA controller ignores the state of the ENDPROG handshaking bit. After the CPU has initially configured the DMA channel, no other CPU intervention is required to keep the channel running. A detailed sequence of events in this mode is shown in Figure 5.

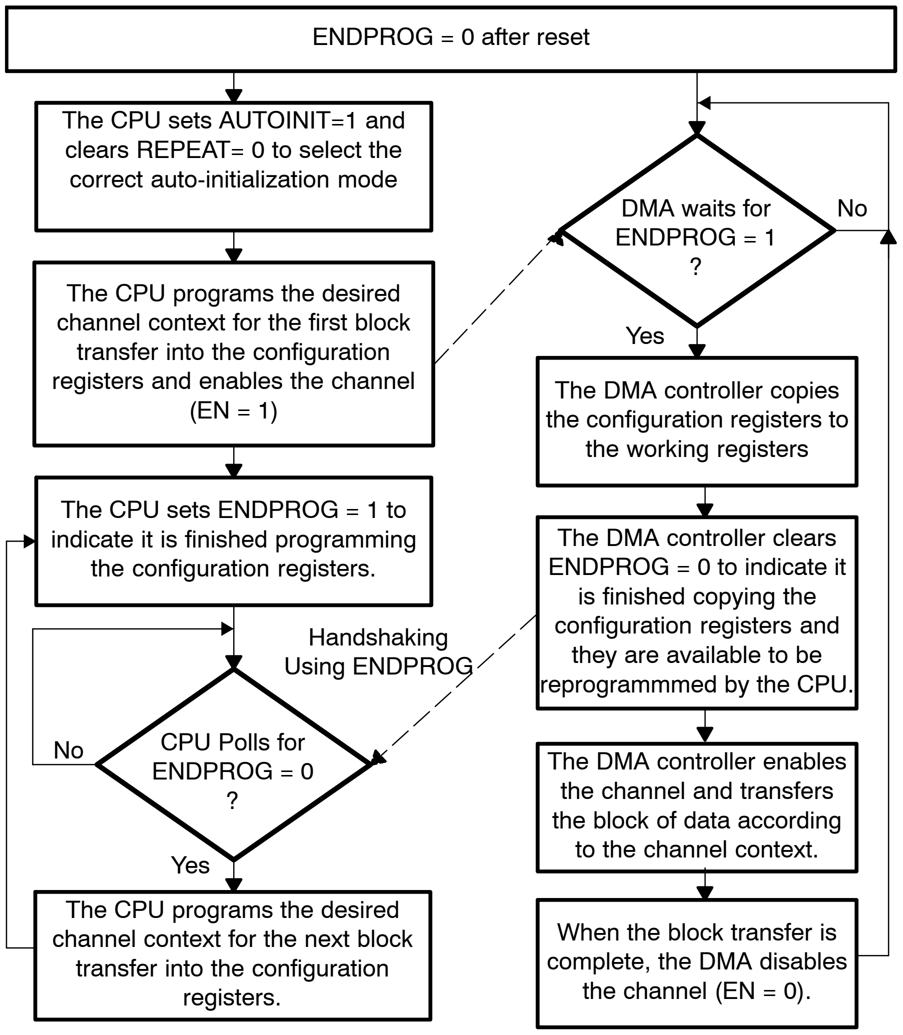
Figure 5. Auto-initialization Sequence with Unchanging Context (REPEAT = 1)


The CPU programs desired channel context into the configuration registers
The CPU sets AUTOINIT=1 and REPEAT=1 to select the correct auto-initialization mode
The CPU sets EN=1 to enable the DMA channel
The DMA controller transfers the block of data according to the channel context
When the block transfer is complete, the DMA disables the channel (EN = 0) and recopies the configuration registers to the working registers.
The DMA controller reenables the channel (EN = 1)

3.2 Auto-initialization with Changing Context

If the desired context for the channel needs to be repeated and is not the same on each block transfer, then the DMA controller must be configured with AUTOINIT = 1 and REPEAT = 0. When REPEAT = 0, the DMA controller waits for the CPU to write ENDPROG = 1 before it copies the configuration registers. This provides handshaking for the DMA to prevent it from copying the registers while they are still being configured by the CPU. A detailed sequence of events in this mode is shown in Figure 6.

Figure 6. Auto-initialization Sequence with Changing Context (REPEAT = 0)

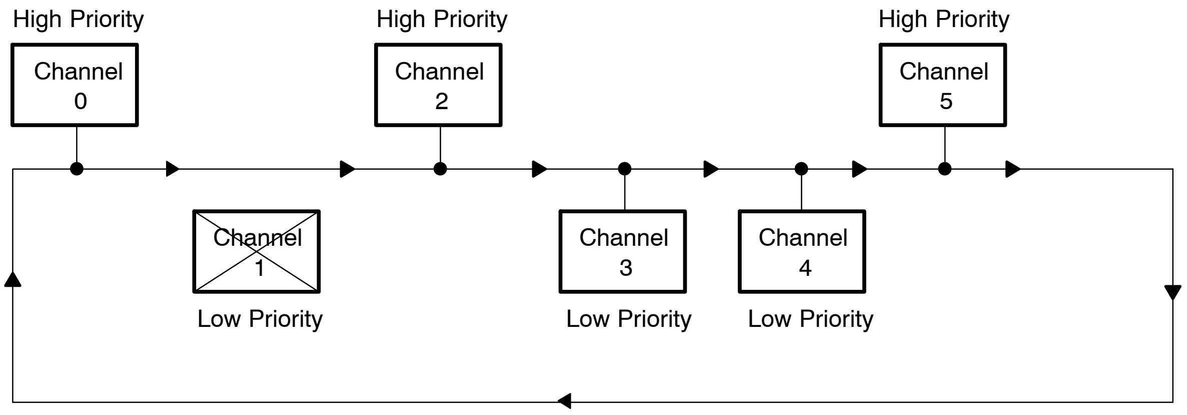

ENDPROG = 0 after reset
The CPU sets AUTOINIT=1 and clears REPEAT= 0 to select the correct auto-initialization mode
The CPU programs the desired channel context for the first block transfer into the configuration registers and enables the channel (EN = 1)
The CPU sets ENDPROG = 1 to indicate it is finished programming the configuration registers.
Handshaking Using ENDPROG
No
CPU Polls for ENDPROG = 0 ?
Yes
The CPU programs the desired channel context for the next block transfer into the configuration registers.
DMA waits for ENDPROG = 1 ?
No
Yes
The DMA controller copies the configuration registers to the working registers
The DMA controller clears ENDPROG = 0 to indicate it is finished copying the configuration registers and they are available to be reprogrammmed by the CPU.
The DMA controller enables the channel and transfers the block of data according to the channel context.
When the block transfer is complete, the DMA disables the channel (EN = 0).

4 Service Chain

Each of the ports of the DMA controller can arbitrate simultaneous access requests sent by the six DMA channels. Each of the ports has an independently functioning service chain—a software- and hardware-controlled scheme for servicing access requests. Although the four service chains function independently, they share a common configuration. For example, if you disable channel 2, it is disabled in all four ports, and if you make channel 4 high-priority, it is high-priority in all four of the ports. One possible configuration for the service chains is shown in Figure 7. Important characteristics of the service chain are listed after the figure.

Section 4.1 contains an example that shows a service chain configuration applied to three ports.

Figure 7. One Possible Configuration for the Service Chains



- ❑ The channels have a programmable priority level. Each channel has a PRIO bit in DMACCR for selecting a high priority or a low priority. The DMA controller only services the low-priority items when all the high-priority items are done or stalled. After a DSP reset, all channels are low priority.

  In the figure, channels 0, 2, and 5 are high-priority (in each of these channels, PRIO = 1). DMA channels 1, 3, and 4 are low priority (in each of these channels, PRIO = 0).

- ❑ The channels have fixed positions in the service chain. Regardless of the programmed priorities, the port checks the channels in a repeating circular sequence: 0, 1, 2, 3, 4, 5, 0, 1, 2, 3, 4, 5, and so on. At each position in the service chain, the port checks whether the channel is ready and able to be serviced. If so, it is serviced; otherwise, the port skips to the next position. After a DSP reset, the port restarts its circular sequence, beginning with channel 0.

- The channels can be individually connected or disconnected from the service chain through software. If a channel is enabled (EN = 1 in DMACCR), it is connected to the service chain; if it is disabled (EN = 0), it is disconnected. After a DSP reset, all channels are disconnected.

  In the figure, only channel 1 is disconnected. As a port checks the channels in its repeating circular sequence, it will keep skipping channel 1 until the channel is reconnected.

- If a channel is tied to a synchronization event, the channel does not generate a DMA request (and, therefore, cannot be serviced) until the synchronization event occurs.

4.1 Service Chain Example

Figure 8 shows a DMA service chain applied to a DARAM port, the external memory port, and the peripheral port. The service chain has the following programmed characteristics.

- Channels 0, 2, and 5 are high-priority (PRIO = 1 in DMACCR). Channels 1, 3, and 4 are low-priority (PRIO = 0).

- Channels 1, 2, and 4 are enabled (EN = 1 in DMACCR). Channels 0, 3, and 5 are disabled (EN = 0).

Table 3 summarizes the activity at the ports in the figure.

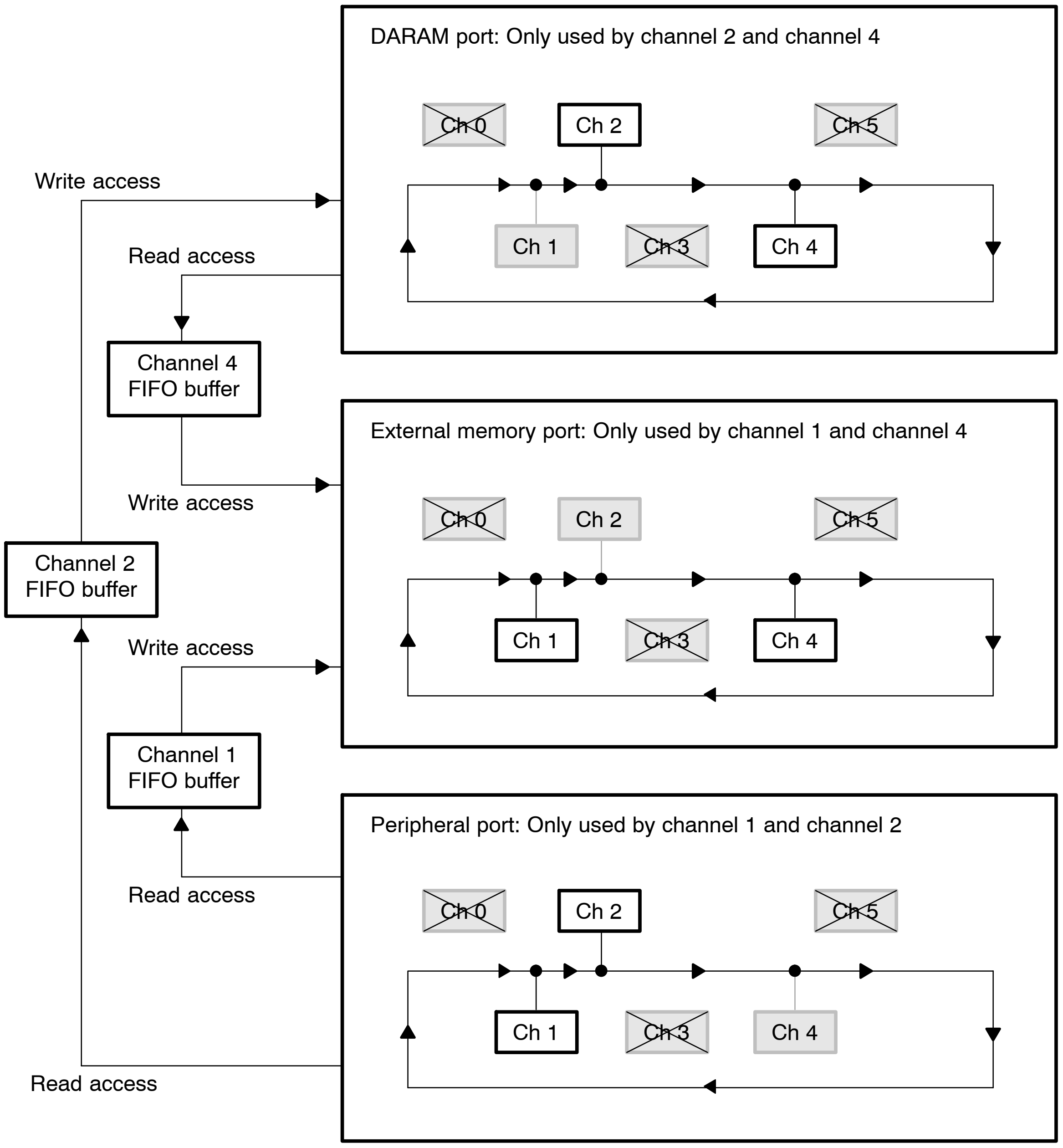
Table 3. Activity Shown in Figure 8

Port	This Port Arbitrates …
DARAM	Write access requests from channel 2 Read access requests from channel 4
External memory	Write access requests from channel 1 Write access requests from channel 4
Peripheral	Read access requests from channel 1 Read access requests from channel 2

Finally, notice that for each port in the figure, there is a channel that is connected to the service chain but does not use the port. For example, the peripheral port is not used by channel 4. If channel 4 were redefined to include the peripheral port as source or destination, the port would handle channel 4 according to its position and priority in the service chain.

Figure 8. Service Chain Applied to Three DMA Ports

Configuration for the service chains	High-priority: 0, 2, 5 Low-priority: 1, 3, 4	Disabled: 0, 3, 5 Enabled: 1, 2, 4


DARAM port: Only used by channel 2 and channel 4
Ch 0
Ch 2
Ch 5
Ch 1
Ch 3
Ch 4
Write access
Read access
Channel 4 FIFO buffer
External memory port: Only used by channel 1 and channel 4
Write access
Channel 2 FIFO buffer
Write access
Channel 1 FIFO buffer
Read access
Peripheral port: Only used by channel 1 and channel 2
Read access

5 Units of Data: Byte, Element, Frame, and Block

This documentation on the DMA controller refers to data in four levels of granularity:

Byte	An 8-bit value. A byte is the smallest unit of data transferred in a DMA channel.
Element	One or more bytes to be transferred as a unit. Depending on the programmed data type, an element is an 8-bit, 16-bit, or a 32-bit value. An element transfer cannot be interrupted; all of its bytes are transferred to a port before another channel can take control of the port.
Frame	One or more elements to be transferred as a unit. A frame transfer can be interrupted between element transfers.
Block	One or more frames to be transferred as a unit. Each channel can transfer one block of data (once or multiple times). A block transfer can be interrupted between frame transfers and element transfers.

For each of the six DMA channels, you can define the number of frames in a block (with DMACFN), the number of elements in a frame (with DMACEN), and the number of bytes in an element (with the DATATYPE bits in DMACSDP). For descriptions of DMACFN, DMACEN, DMACSDP, and other registers of the DMA controller, see section 15 on page 38.

6 Start Addresses in a Channel

During a data transfer in a DMA channel, the first address at which data is read is called the source start address. The first address to which the data is written is called the destination start address. These are byte addresses. From the standpoint of the DMA controller, every 8 bits in memory or I/O space has its own address. Each channel contains the following registers for specifying the start addresses:

Table 4. Registers Used to Define the Start Addresses for a DMA Transfer

Register	**Load With** …
DMACSSAL	Source start address (lower part)
DMACSSAU	Source start address (upper part)
DMACDSAL	Destination start address (lower part)
DMACDSAU	Destination start address (upper part)

The following sections explain how to load the start address registers for memory accesses and I/O accesses. The DMA controller can access all of the internal and external memory and all of I/O space (which contains registers for the DSP peripherals).

6.1 Start Address in Memory

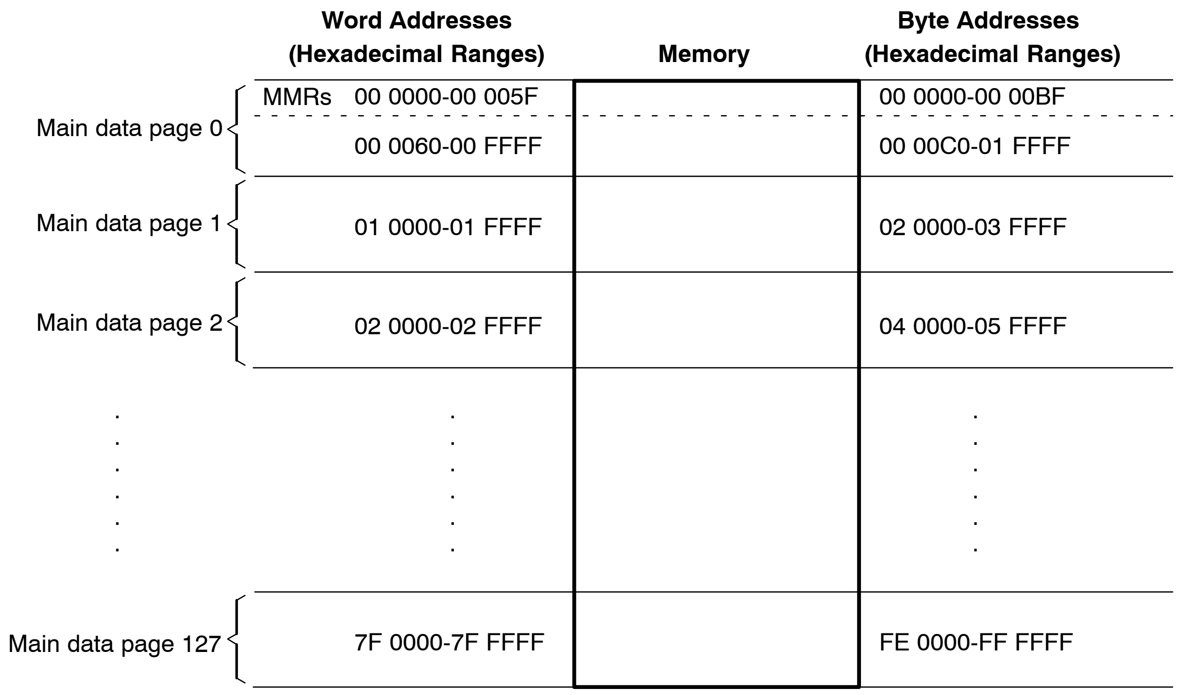
Figure 9 is a high-level memory map for TMS320C55x DSPs. The diagram shows both the word addresses (23-bit addresses) used by the CPU and byte addresses (24-bit addresses) used by the DMA controller. To load the source/destination start address registers:

1) Identify the correct start address. Check for any alignment constraint for the data type; see the description for the DATATYPE bits in section 15.5 (page 55). If you have a word address, shift it left by 1 bit to form a byte address with 24 bits. For example, word address 02 4000h should be converted to byte address 04 8000h.
2) Load the 16 least significant bits (LSBs) of the byte address into DMACSSAL (for source) or DMACDSAL (for destination).
3) Load the 8 most significant bits (MSBs) of the byte address into the 8 LSBs of DMACSSAU (for source) or DMACDSAU (for destination).

Note:

Word addresses 00 0000h–00 005Fh (which correspond to byte addresses 00 0000h–00 00BFh) are reserved for the memory-mapped registers (MMRs) of the DSP CPU.

Figure 9. High-Level Memory Map for TMS320C55x DSPs



6.2 Start Address in I/O Space

Figure 10 is an I/O space map for TMS320C55x DSPs. The diagram shows both the word addresses (16-bit addresses) used by the CPU and byte addresses (17-bit addresses) used by the DMA controller. To load the source/destination start address registers:

1) Identify the correct start address. Check for any alignment constraint for the data type; see the description for the DATATYPE bits in section 15.5 (page 55). If you have a word address, shift it left by 1 bit to form a byte address with 17 bits. For example, word address 8000h should be converted to byte address 1 0000h.

2) Load the 16 least significant bits (LSBs) of the byte address into DMACSSAL (for source) or DMACDSAL (for destination).

3) Load the most significant bit (MSB) of the byte address into the LSB of DMACSSAU (for source) or DMACDSAU (for destination).

Figure 10. High-Level I/O Map for TMS320C55x DSPs

Word Addresses (Hexadecimal Range)	**I/O Space**	**Byte Addresses (Hexadecimal Range)**
0000-FFFF		0 0000-1 FFFF

7 Updating Addresses in a Channel

During data transfers in a DMA channel, the DMA controller begins its read and write accesses at the start addresses you specify (as described in section 6). In many cases, after a data transfer has begun, these addresses must be updated so that data is read and written at consecutive or indexed locations. You can configure address updates at two levels:

❑ Block-level address updates. In the auto-initialization mode (AUTOINIT = 1 in DMACCR), block transfers can occur one after another until you turn off auto-initialization or disable the channel. If you want different start addresses for the block transfers, you can update the start addresses between the block transfers.

❑ Element-level address updates. You can have the DMA controller update the source address and/or the destination address after each element transfer. You can make sure the source address points to the start of the next element, and you can make sure the element will be precisely positioned at the destination. Choose an addressing mode for the source with the SRCAMODE bits in DMACCR. Choose an addressing mode for the destination with the DSTAMODE bits in DMACCR.

8 Data Burst Capability

Data bursts can be used to improve DMA throughput if one or both of the ports associated with the DMA channel support burst capability. When burst is enabled, the DMA controller executes a burst of four elements each time a channel is serviced instead of moving a single element. The DARAM ports support burst capability. The EMIF port supports bursts only if the requested address range is configured as a synchronous (burst) memory type. If the requested address is configured as asynchronous memory, the DMA will perform four single accesses to move the burst of data. The peripheral port does not support burst capability. The DMA will perform four single peripheral-port accesses to move the burst data.

If burst is used, the start addresses for the source and destination should be aligned on a burst boundary. Burst boundaries correspond to byte addresses with 0h as the least significant 4 bits.

To use burst, the following conditions should be met:

- The start address for the port on which burst is enabled should be on a burst boundary.
- The element index should be 1.
- The frame index should cause each burst access to align on a burst boundary.
- (Element number x Element size) should align on a burst boundary. This means at the end of each frame the address should be aligned on a burst boundary.

If both the source and destination have burst enabled, but the source address does not start on a burst boundary, the source burst will be automatically disabled internally. The source will load the channel FIFO and, when enough data is available, a destination burst will be executed. If the destination does not start on a burst boundary, the destination accesses will be performed as single accesses.

If the frame size is not a multiple of 4 elements, the remaining 1 to 3 elements at the end of the frame will be transferred as single (nonburst) accesses.

9 Data Packing

The DMA controller can perform data packing to double or quadruple the amount of data passed to the destination or source in a single transfer. For example, if an 8-bit data type is selected and the destination port has a 32-bit data bus, four 8-bit pieces of data can be packed into 32 bits before being sent to the destination. The resultant packed data depends on the bus size for the destination or source port as shown in Table 5.

Table 5. Data Packing Performed by the DMA Controller

Port	Data Type	Data Packing
DARAM	8-bit	Four 8-bit data values packed into 32 bits
	16-bit	Two 16-bit data values packed into 32 bits
EMIF	8-bit	Four 8-bit data values packed into 32 bits
	16-bit	Two 16-bit data values packed into 32 bits
Peripheral	8-bit	Two 8-bit data values packed into 16 bits

The addressing mode used for the destination and source ports also affects whether or not the DMA controller tries to pack data. The conditions that have to be met for both 32-bit and 16-bit data packing are listed in Table 6 and Table 7, respectively.

Table 6. 32-bit Data Packing Conditions†

Data Type	Byte Address Lower Bits	Addressing Mode	Element Index	Access
8-bit	00b	Post Increment	-	Packed
		Single/Double Index	1	Packed
			Other	Single
		Constant	-	Single
	11b	Single/Double Index	-1	Packed
		Other	-	Single
	Other	-	-	Single
16-bit	00b	Post Increment	-	Packed
		Single/Double Index	1	Packed
			Other	Single
		Constant	-	Single
	10b	Single/Double Index	-3	Packed
		Other	-	Single
	Other	-	-	Single

† Remaining bytes to be transferred is greater than or equal to 4 bytes.

Table 7. 16-bit Data Packing Conditions†

Data Type	Byte Address Lower Bits	Addressing Mode	Element Index	Access
8-bit	00b or 10b	Post Increment	-	Packed
		Single/Double Index	1	Packed
			Other	Single
		Constant	-	Single
	11b or 01b	Single/Double Index	-1	Packed
		Other	-	Single

† Remaining bytes to be transferred is greater than or equal to 2 bytes.

When using element synchronization, no data packing is performed when reading from the source port if that source port is event driven (see section 2, page 12). If the source port is non-event driven, data packing will be performed if packing is enabled at the source (SRCPACK = 1), regardless of whether or not element synchronization is used. However, only one element will be transferred to the destination on every event.

10 Write Posting: Buffering Writes to Internal Memory

The DMA controller can take advantage of the write-posting capability of the internal memory interface. When the write posting bit is set (WP = 1 in DMACCR), the DMA controller can initiate a write and then can receive acknowledgement from the internal memory interface before the data is actually written to memory. The DMA controller is free to begin the next operation while the internal memory interface assumes control of the posted data.

When write posting is disabled (WP = 0), the DMA controller waits for the internal memory interface to finish the memory access before continuing with the next operation. It might be useful during debugging to disable write posting.

11 Synchronizing Channel Activity to an Event

Activity in a channel can be synchronized to an event in a DSP peripheral or to an event signaled by the driving of an external interrupt pin. Using the SYNC bits of DMACCR, you can specify which synchronization event (if any) triggers activity.

Each channel also has an FS bit in DMACCR that allows you to choose among two synchronization modes:

❑ Element synchronization mode (FS = 0) requires one event per element transfer. When the selected synchronization event occurs, an element is transferred from the channel FIFO buffer to the destination port. When all the bytes of the current element are transferred, the channel makes no more requests to the destination port until the next occurrence of the synchronization event. The channel will request data from the source port as described below.

❑ Frame synchronization mode (FS = 1) requires one event to trigger an entire frame of elements. When the event occurs, an entire frame of elements is transferred from the channel FIFO buffer to the destination port. When all the elements in the frame are transferred, the channel makes no more requests to the destination port until the next occurrence of the event. The channel will request data from the source port as described below.

If a synchronization event is specified and the source port is event driven (see Table 8), the channel will not place an access request to the source port until the event occurs. When the event occurs, the channel will take data from the source port, place it in its FIFO buffer, and place an access request to the destination port. Requests received by the source or destination ports are handled according to the predefined position and the programmed priority of the channel in the DMA service chain (see section 4, page 19).

Table 8. Event and Non-Event Driven Ports

Port	Category
DARAM	Non-event driven
External memory	Non-event driven
Peripheral	Event driven

On the other hand, if the source port being used is non-event driven, the channel will place an access request to the source port immediately after it is enabled (EN = 1 in DMACCR) without waiting for the event. For the remainder of the block, the channel will continue to make source read requests to keep the FIFO buffer filled. The read requests from the source will stop only when the FIFO is full or when all of the source data for the current block has been moved into the FIFO. The channel will not access more than one block at a time. After the event occurs, the channel will transfer data from FIFO buffer to the destination port. The amount of data that is transferred to the destination port at each event depends on the use of frame or element synchronization. The channel will request more data from the source port as soon as there is space in the channel FIFO and as long as the entire block has not been completely copied to the channel FIFO. Read or write requests to the different ports are handled according to the predefined position and the programmed priority of the channel in the DMA service chain (see page 6).

If you choose not to synchronize the channel (SYNC = 00000b), the channel sends an access request to the source port as soon as the channel is enabled (EN = 1 in DMACCR). Setting EN = 1 initiates the transfer of the entire block defined for the channel.

11.1 Checking the Synchronization Status

Each channel has a synchronization flag (SYNC) in its status register, DMACSR. When the synchronization event occurs, the DMA controller sets the flag (SYNC = 1). The flag is cleared (SYNC = 0) as follows:

- ❑ If the source port is event driven, the SYNC bit is cleared when the source port initiates the request placed by the DMA channel.
- ❑ If the source port is non-event driven, the SYNC bit is cleared when the destination port initiates the request placed by the DMA channel.

11.2 Dropped Synchronization Event

If a synchronization event occurs before the DMA controller is done servicing the previous one (before the DMA controller clears the SYNC bit in DMACSR), a synchronization event has been dropped. The DMA controller responds to an event drop in the following manner:

- ❑ After the current element transfer, activity in the channel stops.
- ❑ If the corresponding interrupt enable bit is set (DROPIE = 1 in DMACICR), the DMA controller also sets the event drop status bit (DROP = 1 in DMACSR) and sends an interrupt request to the CPU. For more details, see *Monitoring Channel Activity* on page 32.

Before initiating the next DMA transfer, the CPU must clear the error condition by making EN = 0 in DMACCR.

12 Monitoring Channel Activity

The DMA controller can send an interrupt to the CPU in response to the operational events listed in the following table. Each channel has interrupt enable (IE) bits in the interrupt control register (DMACICR) and some corresponding status bits in the status register (DMACSR). (DMACICR and DMACSR are described in section 15.4 on page 49.) If one of the operational events in the table occurs, the DMA controller checks the corresponding IE bit and acts accordingly:

- ❑ If the IE bit is 1 (the interrupt is enabled), the DMA controller sets the corresponding status bit and sends the associated interrupt request to the CPU. DMACSR is automatically cleared if your program reads the register.
- ❑ If the IE bit is 0, the DMA controller sets the corresponding status bit but does not send an interrupt to the CPU.

DMACSR also has a SYNC bit that is used if you choose a synchronization event for the channel. SYNC indicates when the selected synchronization event has occurred (SYNC = 1) and when it has been serviced (SYNC = 0). For more details about synchronization events, see *Synchronizing Channel Activity* on page 30.

Table 9. DMA Controller Operational Events and Their Associated Bits and Interrupts

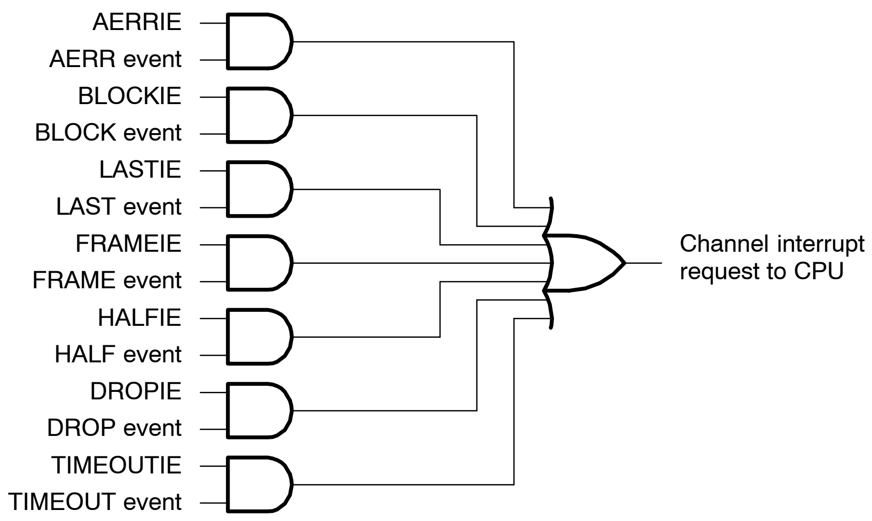
Operational Event	Interrupt Enable Bit	Status Bit	Associated Interrupt
Address error has occurred	AERRIE	AERR	Channel interrupt
Block transfer is complete	BLOCKIE	BLOCK	Channel interrupt
Last frame transfer has started	LASTIE	LAST	Channel interrupt
Frame transfer is complete	FRAMEIE	FRAME	Channel interrupt
First half of current frame has been transferred†	HALFIE	HALF	Channel interrupt
Synchronization event has been dropped	DROPIE	DROP	Channel interrupt
Time-out error has occurred	TIMEOUTIE	TIMEOUT	Channel interrupt

† For a frame with an odd number of elements, the half-frame event occurs as soon as the number of elements transferred is greater than the number that remain to be transferred. For example, for a frame of five elements, the half-frame event occurs when the DMA controller has transferred three of the elements.

12.1 Channel Interrupt

Each of the six channels has its own interrupt. As shown Figure 11, the channel interrupt is the logical OR of all the enabled operational events. You can choose any combination of these events by setting or clearing the appropriate interrupt enable (IE) bits in the interrupt control register (DMACICR) for the channel. You can determine which event(s) caused the interrupt by reading the bits in the status register (DMACSR) for the channel. A read of DMACSR clears all of the status bits. DMACSR should be read each time an interrupt occurs to clear the pending status bits.

Figure 11. Triggering a Channel Interrupt Request



As an example of using the interrupt enable bits, suppose you are monitoring activity in channel 1, and suppose that in DMACICR:

AERRIE = 0
BLOCKIE = 0
LASTIE = 0
FRAMEIE = 1
HALFIE = 0
DROPIE = 1
TIMEOUTIE = 0

When the current frame transfer is done or if a synchronization event is dropped (see section 11.2 on page 31), the channel 1 interrupt request is sent to the CPU. No other event can generate the channel 1 interrupt. To determine whether one or both of the events triggered the interrupt, you can read the DROP and FRAME bits in DMACSR.

The channel 1 interrupt sets its corresponding flag bit in an interrupt flag register of the CPU. The CPU can respond to the interrupt or ignore the interrupt.

For more details about DMACICR and DMACSR, see section 15.4 on page 49.

12.2 Time-Out Error Condition

The DMA controller has a time-out counter for each of the four DMA ports (internal memory port 0, internal memory port 1, the external memory port, and the peripheral port). The clock that controls the DMA controller runs on the fast peripherals clock (SYSCLK1) that has been programmed inside the TMS320VC5501/5502 DSPs. Once a DMA transfer is requested at one of the ports, the corresponding time-out counter is incremented every SYSCLK1 cycle. If the transfer has not been completed within 512 SYSCLK1 cycles, a time-out error signal is generated. The time-out counter for all four DMA ports is disabled by default, but can be enabled for each individual port through the DMAGTCR.

In response to a time-out error signal, activity in the affected DMA channel stops. If the corresponding interrupt enable bit is set (TIMEOUTIE = 1 in DMACICR), the DMA controller also sets the time-out status bit (TIMEOUT = 1 in DMACSR) and sends a channel interrupt request to the CPU. The CPU can respond to the interrupt request or ignore the interrupt request.

Before the next DMA transfer is initiated, the CPU must clear the time-out error condition by making EN = 0 in DMACCR.

12.3 Address Error Condition

If the DMA controller accesses a reserved address in the I/O space of the DSP, an address error signal is generated in the DMA controller. In response, activity in the effected DMA channel stops. If the corresponding interrupt enable bit is set (AERRIE = 1 in DMACICR), the DMA controller also sets the address error status bit (AERR = 1 in DMACSR) and sends a channel interrupt request to the CPU. The CPU can respond to the interrupt request or ignore the interrupt request.

Before the next DMA transfer is initiated, the CPU must clear the address error condition by making EN = 0 in DMACCR.

12.4 Bus Error Interrupt

The following actions by the CPU will cause the DMA controller to send a bus error interrupt (BERRINT) request to the CPU. The CPU can respond to the interrupt request or ignore the interrupt request.

- The CPU attempts to access a reserved address in the DMA register map.
- The CPU attempts to write an illegal/reserved value to a register or to a field inside a register. Here are two important examples:
  - The CPU attempts to load an address register with an unaligned address (an address that is not properly aligned according to the chosen data type).
  - The CPU attempts to load an index register with an index that would create an unaligned address.

13 Latency in DMA Transfers

Each element transfer in a channel is composed of a read access (a transfer from the source location to the channel buffer) and a write access (a transfer from the channel buffer to the destination location). The time to complete this activity depends on factors such as:

- ❑ The selected frequency of the fast peripherals clock (SYSCLK1) signal. This signal, as propagated to the DMA controller, determines the timing for all DMA transfers.
- ❑ Wait states or other extra cycles added by or resulting from an interface.
- ❑ Activity on other channels. Since channels are serviced in a sequential order, the number of pending DMA service requests in the other channels affects how often a given channel can be serviced. For more details on how the channels are serviced, see *Service Chain* on page 19.
- ❑ The timing of synchronization events (if the channel is synchronized). The DMA controller cannot service a synchronized channel until the synchronization event has occurred. For more details on synchronization, see *Synchronizing Channel Activity* on page 30.

The minimum (best-case) latency is determined by the ports used. On the DARAM ports, one access can be performed every cycle if the DMA is not competing with the CPU for access to the same memory block. The best-case transfer rate for channels using the DARAM ports would be one cycle to read at the source and one cycle to write at the destination. The minimum latency for the EMIF port is determined by the EMIF settings, including the memory type used, programmable timings, and any delays caused by the memory itself (such as control of the ARDY pin). The latency for the peripheral port is dependent on the peripherals being accessed and the configuration of the fast and slow peripheral clocks supplied to those peripherals. As explained in section 1.3, page 11, the CPU will always have higher priority than the DMA controller for accesses to the same DARAM block in internal memory.

14 Power, Emulation, and Reset Considerations

The following sections describe how to put the DMA controller into a low-power state, how to program the response of the DMA controller to debugger breakpoints, and what values the DMA controller registers have after a DSP reset.

14.1 Reducing Power Consumed by the DMA Controller (Idle Configurations)

The DSP is divided into idle domains that can be programmed to be idle or active. The state of all domains is called the idle configuration. Any idle configuration that disables the clock generator domain and/or the DMA domain stops the DMA clock and, therefore, stops activity in the DMA controller. The type of channel synchronization (if any) determines how quickly the DMA controller stops:

- ❑ No synchronization (SYNC = 00000b in DMACCR). The DMA controller stops after the entire block transfer is completed.
- ❑ Frame synchronization (SYNC is nonzero and FS = 1 in DMACCR). The DMA controller stops after the current frame transfer is completed.
- ❑ Element synchronization (SYNC is nonzero and FS = 0 in DMACCR). The DMA controller stops after the current element transfer is completed.

When the DMA domain is idle, there is one case when it can be temporarily reactivated without a change in the idle configuration. If one of the multichannel buffered serial ports (McBSPs) needs the DMA controller for a data transfer, the DMA controller will leave its idle state to perform the data transfer and then enter its idle state again.

14.2 Emulation Modes of the DMA Controller

The FREE bit of DMAGCR controls the behavior of the DMA controller when an emulation breakpoint is encountered. If FREE = 0 (the reset value), a breakpoint suspends DMA transfers. If FREE = 1, DMA transfers are not interrupted by a breakpoint.

14.3 DMA Controller after DSP Reset

A DSP reset resets the DMA controller and the DMA configuration registers. The register definitions that follow indicate the effects of a DSP reset on the register contents.

15 DMA Controller Registers

Table 10 lists the types of registers in the direct memory access (DMA) controller. There are two registers that affect all channels: the global control register (DMAGCR) and the global time-out control register (DMAGTCR). In addition, for each of the DMA channels, there are 16 channel configuration registers. For the I/O address of each register, see the data manual for your TMS320C55x DSP.

If the CPU attempts to access a reserved address in the DMA register map, the DMA controller will send a bus error interrupt (BERRINT) request to the CPU. The CPU can respond to the interrupt request or ignore the interrupt request.

Table 10. Registers of the DMA Controller

Register	Description	For Details, See ...
DMAGCR	Global control register (only one)	Page 39
DMAGTCR	Global time-out control register (only one)	Page 40
DMACCR	Channel control register (one for each channel)	Page 42
DMACICR	Interrupt control register (one for each channel)	Page 49
DMACSR	Status register (one for each channel)	Page 49
DMACSDP	Source and destination parameters register (one for each channel)	Page 55
DMACSSAL	Source start address (lower part) register (one for each channel)	Page 60
DMACSSAU	Source start address (upper part) register (one for each channel)	Page 60
DMACDSAL	Destination start address (lower part) register (one for each channel)	Page 61
DMACDSAU	Destination start address (upper part) register (one for each channel)	Page 61
DMACEN	Element number register (one for each channel)	Page 62
DMACFN	Frame number register (one for each channel)	Page 62

Table 10. Registers of the DMA Controller (Continued)

Register	Description	For Details, See ...
DMACSEI	Source element index register (one for each channel)	Page 64
DMACSFI	Source frame index register (one for each channel)	Page 64
DMACDEI	Destination element index register (one for each channel)	Page 64
DMACDFI	Destination frame index register (one for each channel)	Page 64
DMACSAC	Source address counter register (one for each channel)	Page 67
DMACDAC	Destination address counter register (one for each channel)	Page 67

15.1 Global Control Register (DMAGCR)

The global control register (see Figure 12) is a 16-bit I/O-mapped register used to set the emulation mode of the DMA controller.

Figure 12. Global Control Register (DMAGCR)

15–8
Reserved
R-0

7–3	2	1–0
Reserved	FREE	Reserved
R-0	R/W-0	R-0

Legend: R = Read; W = Write; -*n* = Value after reset

Table 11. DMAGCR Bit FIeld Descriptions

Bit	Field	Value	Description
15-3	Reserved		Writing to these bits has no effect. Reading these bits returns 0s.
2	FREE		Emulation mode bit. FREE controls the behavior of the DMA controller when an emulation breakpoint is encountered:
		0	A breakpoint suspends DMA transfers.
		1	DMA transfers continue uninterrupted when a breakpoint occurs.
1-0	Reserved		Writing to these bits has no effect. Reading these bits returns 0s.

15.2 Global Time-Out Control Register (DMAGTCR)

The global time-out control register is a 16-bit read/write register used to enable or disable time-out counters on the DMA ports. If the time-out counters are disabled, the DMA controller will never generate a time-out error condition for these ports. For more details about the time-out error condition, see section 12.2 on page 34.

Figure 13. Global Time-Out Control Register (DMAGTCR)

15–8
Reserved
R-0

7–4	3	2	1	0
Reserved	PTE	ETE	ITE1	ITE0
R-0	R/W-0	R/W-0	R/W-0	R/W-0

Legend: R = Read; W = Write; -*n* = Value after reset

Table 12. DMAGTCR Bit Field Descriptions

Bit	Field	Value	Description
15-4	Reserved		Writing to these bits has no effect. Reading these bits returns 0s.
3	PTE		Peripheral port time-out counter enable bit. This bit enables/disables the time-out counter used to monitor delays on DMA requests to the peripheral port.
		0b	Time-out counter disabled
		1b	Time-out counter enabled
2	ETE		External memory port time-out counter enable bit. This bit enables/disables the time-out counter used to monitor delays on DMA requests to the external memory port.
		0b	Time-out counter disabled
		1b	Time-out counter enabled
1	ITE1		Internal memory port 1 time-out counter enable bit. This bit enables/disables the time-out counter used to monitor delays on DMA requests to DARAM via internal memory port 1.
		0b	Time-out counter disabled
		1b	Time-out counter enabled
0	ITE0		Internal memory port 0 time-out counter enable bit. This bit enables/disables the time-out counter used to monitor delays on DMA requests to DARAM via internal memory port 0.
		0b	Time-out counter disabled
		1b	Time-out counter enabled

15.3 Channel Control Register (DMACCR)

Each channel has a channel control register of the form shown in the following figure. This I/O-mapped register enables you to:

- ❑ Choose how the source and destination addresses are updated (SRCAMODE and DSTAMODE)
- ❑ Enable and control repeated DMA transfers (AUTOINIT, REPEAT, and ENDPROG)
- ❑ Enable or disable write posting for accesses to the internal memory (WP)
- ❑ Enable or disable the channel (EN)
- ❑ Choose a low or high priority level for the channel (PRIO)
- ❑ Select element synchronization or frame synchronization (FS)
- ❑ Determine what synchronization event (if any) initiates a transfer in the channel (SYNC)

Figure 14. Channel Control Register (DMACCR)

15	14	13	12	11	10	9	8
DSTAMODE		SRCAMODE		ENDPROG	WP	REPEAT	AUTOINIT
R/W-00		R/W-00		R/W-0	R/W-0	R/W-0	R/W-0

7	6	5	4				0
EN	PRIO	FS	SYNC				
R/W-0	R/W-0	R/W-0	R/W-0 0000				

Legend: R = Read; W = Write; -*n* = Value after reset

Table 13. DMACCR Bit Field Descriptions

Bit	Field	Value	Description
15-14	DSTAMODE		Destination addressing mode. DSTAMODE determines the addressing mode used by the DMA controller when it writes to the destination port of the channel:
		00b	Constant address The same address is used for each element transfer.
		01b	Automatic post increment After each element transfer, the address is incremented according to the selected data type: If data type is 8-bit Address = Address + 1 If data type is 16-bit Address = Address + 2 If data type is 32-bit Address = Address + 4
		10b	Single index After each element transfer, the address is incremented by the programmed element index amount: Address = Address + element index
		11b	Double index (sort) After each element transfer, the address is incremented by the appropriate index amount: If there are more elements to transfer in the current frame Address = Address + element index If the last element in the frame has been transferred Address = Address + frame index

Table 13. DMACCR Bit Field Descriptions (Continued)

Bit	Field	Value	Description
13-12	SRCAMODE		Source addressing mode. SRCAMODE determines the addressing mode used by the DMA controller when it reads from the source port of the channel:
		00b	Constant address
			The same address is used for each element transfer.
		01b	Automatic post increment
			After each element transfer, the address is incremented according to the selected data type:
			If data type is 8-bit Address = Address + 1
			If data type is 16-bit Address = Address + 2
			If data type is 32-bit Address = Address + 4
		10b	Single index
			After each element transfer, the address is incremented by the programmed element index amount:
			Address = Address + element index
		11b	Double index (sort)
			After each element transfer, the address is incremented by the appropriate index amount:
			If there are more elements to transfer in the current frame Address = Address + element index
			If the last element in the frame has been transferred Address = Address + frame index

Table 13. DMACCR Bit Field Descriptions (Continued)

Bit	Field	Value	Description
11	ENDPROG		End-of-programming bit. Each DMA channel has two sets of registers: configuration registers and working registers. When block transfers occur repeatedly because of auto-initialization (AUTOINIT = 1), you can change the context for the next DMA transfer by writing to the configuration registers during the current block transfer. At the end of the current transfer, the contents of the configuration registers are copied into the working registers, and the DMA controller begins the next transfer using the new context. For proper auto-initialization, the CPU must finish programming the configuration registers before the DMA controller copies their contents. The DMA controller automatically clears the ENDPROG bit after copying the configuration registers to the working registers. The CPU can then program the DMA channel context for the next iteration of the transfer by programming the configuration registers. To make sure auto-initialization waits for the CPU, follow this procedure: 1) Make auto-initialization wait for ENDPROG = 1 by clearing the REPEAT bit (REPEAT = 0) 2) Poll for ENDPROG = 0, which indicates that the DMA controller has finished copying the previous context. The configuration registers can now be programmed for the next iteration. 3) Program the configuration registers. 4) Set ENDPROG (ENDPROG = 1) to indicate the end of register programming.
		0	Configuration registers ready for programming / Programming in progress
		1	End of programming
10	WP		Write posting bit. This bit enables or disables the write posting capability described on page 29.
		0	Write posting disabled.
		1	Write posting enabled. After initiating a write, the DMA controller can receive acknowledgement from the internal memory interface before the data is actually written to memory.

Table 13. DMACCR Bit Field Descriptions (Continued)

Bit	Field	Value	Description
9	REPEAT		Repeat condition bit. If auto-initialization is selected for a channel (AUTOINIT = 1), REPEAT specifies one of two special repeat conditions:
		0	Repeat only if ENDPROG = 1 Once the current DMA transfer is complete, auto-initialization only occurs if the end-of-programmation bit (ENDPROG) is set.
		1	Repeat regardless of ENDPROG Once the current DMA transfer is complete, auto-initialization occurs regardless of whether ENDPROG is 0 or 1.
8	AUTOINIT		Auto-initialization bit. The DMA controller supports auto-initialization, which is the automatic reinitialization of the channel between DMA block transfers. Use AUTOINIT to enable or disable this feature:
		0	Auto-initialization is disabled Activity in the channel stops at the end of the current block transfer. To stop a transfer immediately, clear the channel enable bit (EN).
		1	Auto-initialization is enabled Once the current block transfer is complete, the DMA controller reinitializes the channel and starts a new block transfer. To stop activity in the channel you have two options: ❑ To stop activity immediately, clear the channel enable bit (EN = 0). ❑ To stop activity after the current block transfer, clear AUTOINIT (AUTOINIT= 0)
7	EN		Channel enable bit. Use EN to enable or disable transfers in the channel. The DMA controller clears EN once a block transfer in the channel is complete. **Note:** If the CPU attempts to write to EN at the same time that the DMA controller must clear EN, the DMA controller is given higher priority. EN is cleared, and the value from the CPU is discarded.
		0	Channel is disabled The channel cannot be serviced by the DMA controller. If a DMA transfer is already active in the channel, the DMA controller stops the transfer and resets the channel.
		1	Channel is enabled The channel can be serviced by the DMA controller at the next available time slot.

Table 13. DMACCR Bit Field Descriptions (Continued)

Bit	Field	Value	Description
6	PRIO		Channel priority bit. All six of the DMA channels are given a fixed position and programmable priority level on the service chain of the DMA controller. PRIO determines whether the associated channel has a high priority or a low priority. High-priority channels are serviced before low-priority channels.
		0	Low priority
		1	High priority
5	FS		Frame/element synchronization bit. You can use the SYNC bits of DMACCR to specify a synchronization event for the channel. The FS bit determines whether the synchronization event initiates the transfer of an element or an entire frame of data:
		0	Element synchronization When the selected synchronization event occurs, one element is transferred in the channel. Each element transfer waits for the synchronization event.
		1	Frame synchronization When the selected synchronization event occurs, an entire frame is transferred in the channel. Each frame transfer waits for the synchronization event.
4-0	SYNC	See Table 14	Synchronization control bits. SYNC in DMACCR determines which event in the DSP (for example, a timer countdown) initiates a DMA transfer in the channel. Multiple channels can have the same SYNC value; in other words, one synchronization event can initiate activity in multiple channels. A DSP reset selects SYNC = 00000b (no synchronization event). When SYNC = 00000b, the DMA controller does not wait for a synchronization event before beginning a DMA transfer in the channel; channel activity begins as soon as the channel is enabled (EN = 1). If the CPU attempts to write a reserved value to the SYNC bits, the DMA controller will send a bus error interrupt (BERRINT) request to the CPU.

Table 14. Synchronization Event Mapping for the TMS320VC5501/5502 DSPs

SYNC Field of DMACCR	Synchronization Event For The Channel
00000b	No synchronization event
00001b	McBSP 0 receive event
00010b	McBSP 0 transmit event
00011b	Reserved (do not use this value)
00100b	Reserved (do not use this value)
00101b	McBSP 1 receive event
00110b	McBSP 1 transmit event
00111b	Reserved (do not use this value)
01000b	Reserved (do not use this value)
01001b	Reserved/McBSP event Serial Port Mode† = 0: Reserved Serial Port Mode = 1: McBSP 2 receive event Not available on the TMS320VC5501
01010b	Reserved/McBSP event Serial Port Mode† = 0: Reserved Serial Port Mode = 1: McBSP 2 transmit event Not available on the TMS320VC5501
01011b	Reserved/UART event Serial Port Mode† = 0: UART receive event Serial Port Mode = 1: Reserved
01100b	Reserved/UART event Serial Port Mode† = 0: UART transmit event Serial Port Mode = 1: Reserved
01101b	Timer 0 interrupt event
01110b	Timer 1 interrupt event
01111b	External interrupt 0
10000b	External interrupt 1
10001b	External interrupt 2

† For details on the Serial Port Mode bit, see the device-specific data manual.

Table 14. Synchronization Event Mapping for the TMS320VC5501/5502 DSPs (Continued)

SYNC Field of DMACCR	Synchronization Event For The Channel
10010b	External interrupt 3
10011b	I2C module receive event
10100b	I2C module transmit event
Other values	Reserved (do not use these values)

† For details on the Serial Port Mode bit, see the device-specific data manual.

15.4 Interrupt Control Register (DMACICR) and Status Register (DMACSR)

Each channel has an interrupt control register (DMACICR) and a status register (DMACSR). DMACICR and DMACSR are I/O-mapped registers. Their bits are shown in Figure 15 and described in Table 15 and Table 16.

Use DMACICR to specify that one or more operational events in the DMA controller will trigger an interrupt. If an operational event occurs and its interrupt enable (IE) bit is 1, an interrupt request is sent to the DSP CPU, where it can be serviced or ignored. Each channel has its own interrupt line to the CPU and one set of flag and enable bits in the CPU.

To see which operational event or events have occurred, your program can read DMACSR. The DMA controller sets one of the interrupt flag bits (bits 7 and 5-0) when the operational event occurs. The interrupt flag bits stay set until your program reads DMACSR, at which point all of its bits are cleared automatically.

The AERR, DROP, and TIMEOUT bits indicate error conditions. Once an error condition occurs, it must be cleared before the next DMA transfer is initiated. To clear the error condition, the CPU must write 0 to the EN bit of DMACCR.

The SYNC bit (bit 6) can be used to detect when a synchronization event has occurred (SYNC = 1) and when the resulting access request has been serviced (SYNC = 0).

Figure 15. Interrupt Control Register (DMACICR) and Status Register (DMACSR)

DMACICR

15–9	8
Reserved	Reserved[†]
R-0	R/W-1

7	6	5	4	3	2	1	0
AERRIE	Reserved	BLOCKIE	LASTIE	FRAMEIE	HALFIE	DROPIE	TIMEOUTIE
R/W-1	R-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-1	R/W-1

DMACSR

15–9	8
Reserved	Reserved[‡]
R-0	R-x

7	6	5	4	3	2	1	0
AERR	SYNC	BLOCK	LAST	FRAME	HALF	DROP	TIMEOUT
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

Legend: R = Read; W = Write; -*n* = Value after reset; -x = Value after reset is not defined

[†] Always write 0 to bit 8 of DMACICR. After reset, change this bit from 1 to 0.
[‡] The read state of bit 8 of DMACSR is not defined.

Table 15. DMACICR Bit Field Descriptions

Bit	Field	Value	Description
15-9	Reserved		Writing to these bits has no effect. Reading these bits returns 0s.
8	Reserved		Always write 0 to this bit. After a reset, change this bit from 1 to 0.
7	AERRIE		Address error interrupt enable bit. AERRIE determines how the DMA controller responds to an address error at the source port or the destination port of the channel. The address error condition is described in section 12.3 on page 34.
		0	Do not send the channel interrupt request to the CPU when this error occurs.
		1	Send the channel interrupt request to the CPU when this error occurs.

Table 15. DMACICR Bit Field Descriptions (Continued)

Bit	Field	Value	Description
6	Reserved		Writing to this bit has no effect. Reading this bit returns 0.
5	BLOCKIE		Whole block interrupt enable bit. BLOCKIE determines how the DMA controller responds when all of the current block has been transferred to the destination port.
		0	Do not send the channel interrupt request to the CPU when this error occurs.
		1	Send the channel interrupt request to the CPU when this error occurs.
4	LASTIE		Last frame interrupt enable bit. LASTIE determines how the DMA controller responds to a last-frame event: If the peripheral port is the source, the last-frame event occurs when the first element of the last frame is being read from the source. If an external or internal memory port is the source, the last-frame event occurs when the first element of the last frame is being received at the destination.
		0	Do not send the channel interrupt request to the CPU when this error occurs.
		1	Send the channel interrupt request to the CPU when this error occurs.
3	FRAMEIE		Whole frame interrupt enable bit. FRAMEIE determines how the DMA controller responds when the all of the current frame has been transferred to the destination port.
		0	Do not send the channel interrupt request to the CPU when this error occurs.
		1	Send the channel interrupt request to the CPU when this error occurs.

Table 15. DMACICR Bit Field Descriptions (Continued)

Bit	Field	Value	Description
2	HALFIE		Half frame interrupt enable bit. HALFIE determines how the DMA controller responds when the first half of the current frame has been transferred to the destination port. For a frame with an odd number of elements, the half-frame event occurs as soon as the number of elements transferred is greater than the number that remain to be transferred. For example, for a frame of five elements, the half-frame event occurs when the DMA controller has transferred three of the elements.
		0	Do not send the channel interrupt request to the CPU when this error occurs.
		1	Send the channel interrupt request to the CPU when this error occurs.
1	DROPIE		Synchronization event drop interrupt enable bit. If a DMA synchronization event occurs again before the DMA controller has finished servicing the previous DMA request, an error has occurred—a synchronization event drop. DROPIE determines how the DMA controller responds when a synchronization event drop occurs in the channel.
		0	Do not send the channel interrupt request to the CPU when this error occurs.
		1	Send the channel interrupt request to the CPU when this error occurs.
0	TIMEOUTIE		Time-out interrupt enable bit. TIMEOUTIE determines how the DMA controller responds to a time-out error at the source port or the destination port of the channel. The time-out error condition is described in section 12.2 on page 34.
		0	Do not send the channel interrupt request to the CPU when this error occurs.
		1	Send the channel interrupt request to the CPU when this error occurs.

Table 16. DMACSR Bit Field Descriptions

Bit	Field	Value	Description
15-9	Reserved		Writing to these bits has no effect. Reading these bits returns 0s.
8	Reserved		The read state of bit 8 is not defined.
7	AERR		Address error status bit. The DMA controller sets AERR when an address error has occurred at the source port or the destination port of the channel. The address error condition is described in section 12.3 on page 34.
		0	An address error has not occurred, or AERR has been cleared.
		1	An address error has occurred. A channel interrupt request has been sent to the CPU.
6	SYNC		Synchronization event status bit. The DMA controller updates SYNC to indicate when the synchronization event for the channel has occurred and when the synchronized channel has been serviced.
		0	The DMA controller has finished servicing the previous access request.
		1	The synchronization event has occurred. The SYNC bit is automatically cleared by the DMA as described in section 11.1 on page 31.
			Note 1: If a synchronization event occurs again before the DMA controller has finished servicing the previous DMA request, an error has occurred- a synchronization event drop. You can track this type of error using the DROPIE bit and the DROP bit.
			Note 2: To select a synchronization event for a channel, use the SYNC bits of DMACCR. The SYNC bit is set to 0 if the SYNC bits in the DMACCR are set to 00000b.
5	BLOCK		Whole block status bit. The DMA controller sets BLOCK when all of the current block has been transferred to the destination port.
		0	The whole-block event has not occurred yet, or BLOCK has been cleared.
		1	The whole block has been transferred. A channel interrupt request has been sent to the CPU.

Table 16. DMACSR Bit Field Descriptions (Continued)

Bit	Field	Value	Description
4	LAST		Last frame status bit. The DMA controller sets LAST when a last-frame event occurs. If the peripheral port is the source, the last-frame event occurs when the first element of the last frame is being read from the source. If an external or internal memory port is the source, the last-frame event occurs when the first element of the last frame is being received at the destination.
		0	The last-frame event has not occurred yet, or LAST has been cleared.
		1	The DMA controller has started transferring the last frame. A channel interrupt request has been sent to the CPU.
3	FRAME		Whole frame status bit. The DMA controller sets FRAME when all of the current frame has been transferred to the destination port.
		0	The whole-frame event has not occurred yet, or FRAME has been cleared.
		1	The whole frame has been transferred. A channel interrupt request has been sent to the CPU.
2	HALF		Half frame status bit. The DMA controller sets HALF when the first half of the current frame has been transferred to the destination port. For a frame with an odd number of elements, the half-frame event occurs as soon as the number of elements transferred is greater than the number that remain to be transferred. For example, for a frame of five elements, the half-frame event occurs when the DMA controller has transferred three of the elements.
		0	The half-frame event has not occurred yet, or HALF has been cleared.
		1	The first half of the frame has been transferred. A channel interrupt request has been sent to the CPU.

Table 16. DMACSR Bit Field Descriptions (Continued)

Bit	Field	Value	Description
1	DROP		Synchronization event drop status bit. If a DMA synchronization event occurs again before the DMA controller has finished servicing the previous DMA request, an error has occurred− a synchronization event drop. The DMA controller sets DROP only if DROPIE = 1 in DMACICR and a synchronization event drop has occurred in the channel.
		0	A synchronization event drop has not occurred, or DROP has been cleared.
		1	A synchronization event drop has occurred. A channel interrupt request has been sent to the CPU.
			Note 1: The DROP bit is cleared after the DMACSR is read.
			Note 2: The DROP bit is set to 0 if the SYNC bits in the DMACCR are set to 00000b.
0	TIMEOUT		Time-out status bit. The DMA controller sets TIMEOUT only if TIMEOUTIE = 1 in DMACICR and a time-out error has occurred at the source port or the destination port of the channel. The time-out error condition is described in section 12.2 on page 34.
		0	A time-out error has not occurred, or TIMEOUT has been cleared.
		1	A time-out error has occurred. A channel interrupt request has been sent to the CPU.

15.5 Source and Destination Parameters Register (DMACSDP)

Each channel has a source and destination parameters register of the form shown in Figure 16. This I/O-mapped register enables you to choose a source port (SRC) and a destination port (DST), specify a data type (DATATYPE) for port accesses, enable or disable data packing (SRCPACK and DSTPACK), and enable or disable burst transfers (SRCBEN and DSTBEN).

Figure 16. Source and Destination Parameters Register (DMACSDP)

15–14	13	12–9
DSTBEN	DSTPACK	DST
R/W-00	R/W-0	R/W-0000

8–7	6	5–2	1–0
SRCBEN	SRCPACK	SRC	DATATYPE
R/W-00	R/W-0	R/W-0000	R/W-00

Legend: R = Read; W = Write; -*n* = Value after reset

Table 17. DMACSDP Bit Field Descriptions

Bit	Field	Value	Description
15-14	DSTBEN		Destination burst enable bit. A burst in the DMA controller is four consecutive 32-bit accesses at a DMA port. DSTBEN determines whether the DMA controller performs a burst at the destination port of the channel.
		00b	Bursting disabled (single access enabled) at the destination
		01b	Bursting disabled (single access enabled) at the destination
		10b	Bursting enabled at the destination. When writing to the destination, the DMA controller performs four consecutive 32-bit accesses.
		11b	Reserved. If the CPU attempts to write 11b to the DSTBEN bits, the DMA controller will send a bus error interrupt (BERRINT) request to the CPU.
13	DSTPACK		Destination packing enable bit. The DMA controller can perform data packing to double or quadruple the amount of data passed to the destination in a single transfer. For example, if an 8-bit data type is selected and the destination port has a 32-bit data bus, four 8-bit pieces of data can be packed into 32 bits before being sent to the destination. DSTPACK determines whether data packing is used at the destination port.
		0	Packing disabled at the destination
		1	Packing enabled at the destination. Where possible, the DMA controller packs data before each write to the destination. Section 9 (page 27) shows the instances where data packing is performed.

Table 17. DMACSDP Bit Field Descriptions (Continued)

Bit	Field	Value	Description
12-9	DST		Destination selection bit. DST selects which DMA port is the destination for data transfers in the channel:
		0000b	DARAM via internal memory port 0
		0001b	DARAM via internal memory port 1
		0010b	External memory via the external memory interface (EMIF)
		0011b	Peripherals via the peripheral bus controller
		Other	Reserved. If the CPU attempts to write a reserved value to the DST bits, the DMA controller will send a bus error interrupt (BERRINT) request to the CPU.
8-7	SRCBEN		Source burst enable bit. A burst in the DMA controller is four consecutive 32-bit accesses at a DMA port. SRCBEN determines whether the DMA controller performs a burst at the source port of the channel. This field will be ignored if: ❑ the source port does not support burst capability, or ❑ constant address mode is selected for the source port, or ❑ the channel is element synchronized.
		00b	Bursting disabled (single access enabled) at the source
		01b	Bursting disabled (single access enabled) at the source
		10b	Bursting enabled at the source. When reading from the source, the DMA controller performs four consecutive 32-bit accesses.
		11b	Reserved. If the CPU attempts to write 11b to the SRCBEN bits, the DMA controller will send a bus error interrupt (BERRINT) request to the CPU.
6	SRCPACK		Source packing enable bit. The DMA controller can perform data packing to double or quadruple the amount of data gathered at the source before a transfer. For example, if an 8-bit data type is selected and the source port has a 32-bit data bus, four 8-bit pieces of data can be packed into 32 bits before being sent through the channel. SRCPACK determines whether data packing is used at the source port.
		0	Packing disabled at the source
		1	Packing enabled at the source. Where possible, the DMA controller packs data from the source before beginning a data transfer in the channel. Section 9 (page 27) shows the instances where data packing is performed.

Table 17. DMACSDP Bit Field Descriptions (Continued)

Bit	Field	Value	Description
5-2	SRC		Source selection bit. SRC selects which DMA port is the source for data transfers in the channel:
		0000b	DARAM via internal memory port 0
		0001b	DARAM via internal memory port 1
		0010b	External memory via the external memory interface (EMIF)
		0011b	Peripherals via the peripheral bus controller
		Other	Reserved. If the CPU attempts to write a reserved value to the SRC bits, the DMA controller will send a bus error interrupt (BERRINT) request to the CPU.

Table 17. DMACSDP Bit Field Descriptions (Continued)

Bit	Field	Value	Description
1-0	DATATYPE		Data type bit. DATATYPE indicates how data is to be accessed at the source and at the destination of the channel. Note that the DMA controller uses **byte addresses** for its accesses; each byte in data space or I/O space has its own address. For information on how addresses are updated between element transfers, see the descriptions for the SRCAMODE bits and the DSTAMODE bits in DMACCR (page 42).
		00b	8-bit The DMA controller makes 8-bit accesses at the source and at the destination of the channel. The source and destination start addresses have no alignment constraint: Start address: XXXX XXXX XXXX XXXXb (X can be 0 or 1) If you choose the automatic post increment addressing mode at the source or the destination, the corresponding address is updated by an increment of 1 after each element transfer.
		01b	16-bit The DMA controller makes 16-bit accesses at the source and at the destination. The source and destination start addresses must each be on an even 2-byte boundary; the least significant bit (LSB) must be 0: Start address: XXXX XXXX XXXX XXX0b (X can be 0 or 1) If you choose the automatic post increment addressing mode at the source or the destination, the address is updated by an increment of 2 after each element transfer.
		10b	32-bit The DMA controller makes 32-bit accesses at the source and at the destination. The source and destination start addresses must be on an even 4-byte boundary; the 2 LSBs must be 0: Start address: XXXX XXXX XXXX XX00b (X can be 0 or 1) If you choose the automatic post increment addressing mode at the source or the destination, the address is updated by an increment of 4 after each element transfer.
		11b	Reserved. If the CPU attempts to write 11b to the DATATYPE bits, the DMA controller will send a bus error interrupt (BERRINT) request to the CPU.

15.6 Source Start Address Registers (DMACSSAL and DMACSSAU)

Each channel has two source start address registers, which are shown in Figure 17 and described in Table 18 and Table 19. For the first access to the source port of the channel, the DMA controller generates a byte address by concatenating the contents of the two I/O-mapped registers. DMACSSAU supplies the upper bits, and DMACSSAL supplies the lower bits:

Source start address = DMACSSAU:DMACSSAL

Notes:

1) You must load the source start address registers with a byte address. If you have a word address, shift it left by 1 before loading the registers.

2) If you have a 16-bit or 32-bit data type, the start address must be aligned properly. See the description for the DATATYPE bits of DMACSDP (page 55). If the CPU attempts to write an unaligned address, the DMA controller will send a bus error interrupt (BERRINT) request to the CPU.

3) It is the programmer's responsibility to ensure that the start address, element index and frame index will produce valid addresses within the range of the port. If an invalid address is generated, a time-out error will occur.

The destination start address is supplied by DMACDSAL and DMACDSAU, which are described in section 15.7.

Figure 17. Source Start Address Registers (DMACSSAL and DMACSSAU)

DMACSSAL

15 … 0
SSAL
R/W-0

DMACSSAU

15 … 0
SSAU
R/W-0

Legend: R = Read; W = Write; -*n* = Value after reset

Table 18. DMACSSAL Bit Field Description

Bit	Field	Value	Description
15-0	SSAL	0000h-FFFFh	Lower part of source start address (byte address)

Table 19. DMACSSAU Bit Field Description

Bit	Field	Value	Description
15-0	SSAU	0000h-00FFh	Upper part of source start address (byte address)
		0100h-FFFFh	Reserved (do not use these values)

15.7 Destination Start Address Registers (DMACDSAL and DMACDSAU)

Each channel has two destination start address registers, which are shown in Figure 18 and described in Table 20 and Table 21. For the first access to the destination port of the channel, the DMA controller generates a byte address by concatenating the contents of the two I/O-mapped registers. DMACDSAU supplies the upper bits, and DMACDSAL supplies the lower bits:

Destination start address = DMACDSAU:DMACDSAL

Notes:

1) You must load the destination start address registers with a byte address. If you have a word address, shift it left by 1 before loading the registers.

2) If you have a 16-bit or 32-bit data type, the start address must be aligned properly. See the description for the DATATYPE bits of DMACSDP (page 55). If the CPU attempts to write an unaligned address, the DMA controller will send a bus error interrupt (BERRINT) request to the CPU.

3) It is the programmer's responsibility to ensure that the start address, element index and frame index will produce valid addresses within the range of the port. If an invalid address is generated, a time-out error will occur.

The source start address is supplied by DMACSSAL and DMACSSAU, which are described in section 15.6.

Figure 18. Destination Start Address Registers (DMACDSAL and DMACDSAU)

DMACDSAL

15 … 0
DSAL
R/W-0

DMACDSAU

15 … 0
DSAU
R/W-0

Legend: R = Read; W = Write; -*n* = Value after reset

Table 20. DMACDSAL Bit Field Description

Bit	Field	Value	Description
15-0	DSAL	0000h-FFFFh	Lower part of destination start address (byte address)

Table 21. DMACDSAU Bit Field Description

Bit	Field	Value	Description
15-0	DSAU	0000h-00FFh	Upper part of destination start address (byte address)
		0100h-FFFFh	Reserved (do not use these values)

15.8 Element Number Register (DMACEN) and Frame Number Register (DMACFN)

Each channel has an element number register and a frame number register, (see Figure 19, Table 22, and Table 23). Load DMACFN with the number of frames you want in each block. Load DMACEN with the number of elements you want in each frame. You must have at least one frame and one element, and you can have as many as 65535 of each:

$1 \le \text{frame number} \le 65535$
$1 \le \text{element number} \le 65535$

If the CPU attempts to write 0 to DMACEN or DMACFN, the DMA controller will send a bus error interrupt (BERRINT) request to the CPU.

Figure 19. Element Number Register (DMACEN) and Frame Number Register (DMACFN)

DMACEN

15 … 0
ELEMENTNUM
R/W-0001h

DMACFN

15 … 0
FRAMENUM
R/W-0001h

Legend: R = Read; W = Write; -*n* = Value after reset

Table 22. DMACEN Bit Field Description

Bit	Field	Value	Description
15-0	ELEMENTNUM	0000h	Reserved. If the CPU attempts to write 0000h to this field, the DMA controller will send a bus error interrupt (BERRINT) request to the CPU.
		0001h-FFFFh	Number of elements per frame (1-65535)

Table 23. DMACFN Bit Field Description

Bit	Field	Value	Description
15-0	FRAMENUM	0000h	Reserved. If the CPU attempts to write 0000h to this field, the DMA controller will send a bus error interrupt (BERRINT) request to the CPU.
		0001h-FFFFh	Number of frames per block (1-65535)

15.9 Element Index Registers (DMACSEI, DMACDEI) and Frame Index Registers (DMACSFI, DMACDFI)

The single- or double-index addressing mode can be selected separately for the source and destination ports by using the SRCAMODE bits and the DSTAMODE bits, respectively, in DMACCR (page 42). To support these index addressing modes, each channel has two element index registers and two frame index registers. These four registers are are shown in Figure 20 and described in the tables that follow the figure.

The DMA controller uses the following index registers to control the source port:

❑ DMACSEI: Contains the desired element index for the source for single- or double-index addressing mode.

❑ DMACSFI: Contains the desired frame index for the source for the double-index addressing mode.

The DMA controller uses the following index registers to control the destination port:

❑ DMACDEI: Contains the desired element index for the destination for the single- or double-index addressing mode.

❑ DMACDFI: Contains the desired frame index for the destination for the double-index addressing mode.

The element and frame indexes are 16-bit signed numbers, providing the following range:

-32768 bytes = frame index = 32767 bytes
-32768 bytes = element index = 32767 bytes

After each transfer, the source and destination address registers contain the address for the last byte of the element that was transferred. For example, consider the case in which the DMA channel is reading a 32-bit element at byte address 0x2000. The source address will be 0x2003 after the element is read because the DMA channel will read a total of four bytes. If the DMA channel reads a 16-bit element, the source address would be 0x2001 after the element read because only two bytes are read. For a byte read, the source address would stay at 0x2000 after the byte read.

When the single index mode is used, the element index is added to the source or destination address at the end of each element transfer. The modified address will then be used at the beginning of the next element transfer.

When the double index mode is used for the source or the destination address, the element index is added to the source or destination address at the end of each element transferred as described above, except for the last element in the frame. For the last element in the frame, the frame index is added to the source or destination address instead of the element index. For example, if the last element in the frame starts at byte address 0x801E where the data type is 16-bit and the frame index is set to 0x0003, the DMA will move the first byte (0x801E) then the second byte (0x801F) of the element. The frame index will then be added to the 0x801F to create the address for the first byte next element to be moved (0x801F + 0x0003 = 0x8022).

The element index that is added to the source or destination address must produce an aligned address according to the data type selected in the DATATYPE field of DMACSDP. For this reason only certain values are valid for the element index.

Valid values for the element index are:

- [4 x N] + 1 (where N = -2, -1, 0, 1, 2...) if the data type is 32-bit
- [2 x N] + 1 (where N = -2, -1, 0, 1, 2...) if the data type is 16-bit
- Any value if the data type is 8-bit

As with the element index, the frame index must produce an aligned address according to the data type selected in the DATATYPE field of DMACSDP.

Valid values for the frame index are:

- [4 x N] + 1 (where N = -2, -1, 0, 1, 2...) if the data type is 32-bit
- [2 x N] + 1 (where N = -2, -1, 0, 1, 2...) if the data type is 16-bit
- Any value if the data type is 8-bit

It is the programmer’s responsibility to ensure that the start address, element index, and frame index will produce valid addresses within the range of the port. If an invalid address is generated, a time-out error will occur.

If the CPU attempts to write an element or a frame index that would cause an unaligned address, the DMA controller will send a bus error interrupt (BERRINT) request to the CPU. This occurs even if address indexing is not used.

Figure 20. Element Index Registers (DMACSEI, DMACDEI) and Frame Index Registers (DMACSFI, DMACDFI)

DMACSEI

15 … 0
ELEMENTNDX
R/W-0

DMACSFI

15 … 0
FRAMENDX
R/W-0

DMACDEI

15 … 0
ELEMENTNDX
R/W-0

DMACDFI

15 … 0
FRAMENDX
R/W-0

Legend: R = Read; W = Write; -*n* = Value after reset

Table 24. DMACSEI Bit Field Description

Bit	Field	Value	Description
15-0	ELEMENTNDX	-32768 to 32767	Source element index (in bytes)

Table 25. DMACSFI Bit Field Description

Bit	Field	Value	Description
15-0	FRAMENDX	-32768 to 32767	Source frame index (in bytes)

Table 26. DMACDEI Bit Field Description

Bit	Field	Value	Description
15-0	ELEMENTNDX	-32768 to 32767	Destination element index (in bytes)

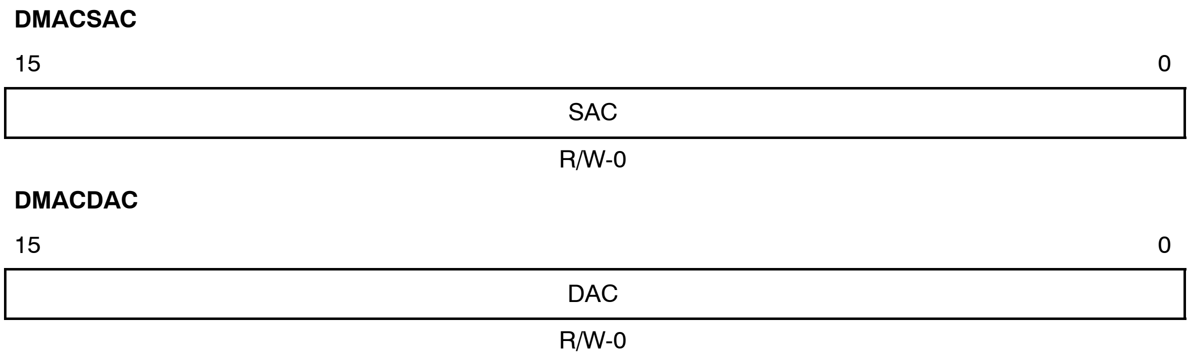
Table 27. DMACDFI Bit Field Description

Bit	Field	Value	Description
15-0	FRAMENDX	-32768 to 32767	Destination frame index (in bytes)

15.10 Source Address Counter (DMACSAC) and Destination Address Counter (DMACDAC)

The progress of each DMA channel can be monitored by reading the source and destination address counters (DMACSAC and DMACDAC). DMACSAC shows the low 16 bits of the current source address. DMACDAC shows the low 16 bits of the current destination address.

Figure 21. Channel Source Address Counter (DMACSAC) and Channel Destination Address Counter (DMACDAC)



Legend: R = Read; W = Write; -*n* = Value after reset

Table 28. DMACSAC Bit Field Description

Bit	Field	Value	Description
15-0	SAC	0000h-FFFFh	Current channel source address

Table 29. DMACDAC Bit Field Description

Bit	Field	Value	Description
15-0	DAC	0000h-FFFFh	Current channel destination address

This page is intentionally left blank.

Revision History

Table 30. Document Revision History

Page	Additions/Modifications/Deletions
14	Added Section 3.

Index