TMS320VC5503/5507/5509/5510 DSP Direct Memory Access (DMA) Controller Reference Guide

Literature Number: SPRU587E
January 2007

Texas Instruments

This page is intentionally left blank.

Preface

Read This First

About This Manual

This manual describes the features and operation of the direct memory access (DMA) controller that is available on TMS320VC5503, TMS320VC5507, TMS320VC5509 and TMS320VC5510 digital signal processors (DSPs) in the TMS320C55x™ (C55x™) DSP generation.

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.

TMS320VC5503 Fixed-Point Digital Signal Processor Data Manual (literature number SPRS245) describes the features of the TMS320VC5503 fixed-point DSP and provides signal descriptions, pinouts, electrical specifications, and timings for the device.

TMS320VC5507 Fixed-Point Digital Signal Processor Data Manual (literature number SPRS244) describes the features of the TMS320VC5507 fixed-point DSP and provides signal descriptions, pinouts, electrical specifications, and timings for the device.

TMS320VC5509 Fixed-Point Digital Signal Processor Data Manual (literature number SPRS163) describes the features of the TMS320VC5509 fixed-point DSP and provides signal descriptions, pinouts, electrical specifications, and timings for the device.

TMS320VC5509A Fixed-Point Digital Signal Processor Data Manual (literature number SPRS205) describes the features of the TMS320VC5509A fixed-point DSP and provides signal descriptions, pinouts, electrical specifications, and timings for the device.

TMS320VC5510 Fixed-Point Digital Signal Processor Data Manual (literature number SPRS076) describes the features of the TMS320VC5510 fixed-point DSP and provides signal descriptions, pinouts, electrical specifications, and timings for the device.

TMS320C55x Technical Overview (literature number SPRU393) introduces 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 Overview Reference Guide (literature number SPRU317) introduces 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.

This page is intentionally left blank.

Contents

Figures

This page is intentionally left blank.

Tables

This page is intentionally left blank.

Direct Memory Access (DMA) Controller

This document describes the features and operation of the DMA controller that is available on TMS320VC5503, TMS320VC5507, TMS320VC5509, TMS320VC5509A, and TMS320VC5510 DSPs. This DMA controller allows movement of data among internal memory, external memory, peripherals, and the host port interface (HPI) 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.
- ❑ Transfer data between the host port interface (HPI) and memory.

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, one for each data resource: internal dual-access RAM (DARAM), internal single-access RAM (SARAM), external memory, and peripherals.
- ❑ An auxiliary port to enable certain transfers between the host port interface (HPI) and memory.
- ❑ 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 24.
- ❑ 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 34.
- ❑ 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 36.

❑ 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. See *Reducing Power Consumed By the DMA Controller* on page 41.

To read about the registers used to program the DMA controller, see section 14 on page 42.

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 ports in the diagram are:

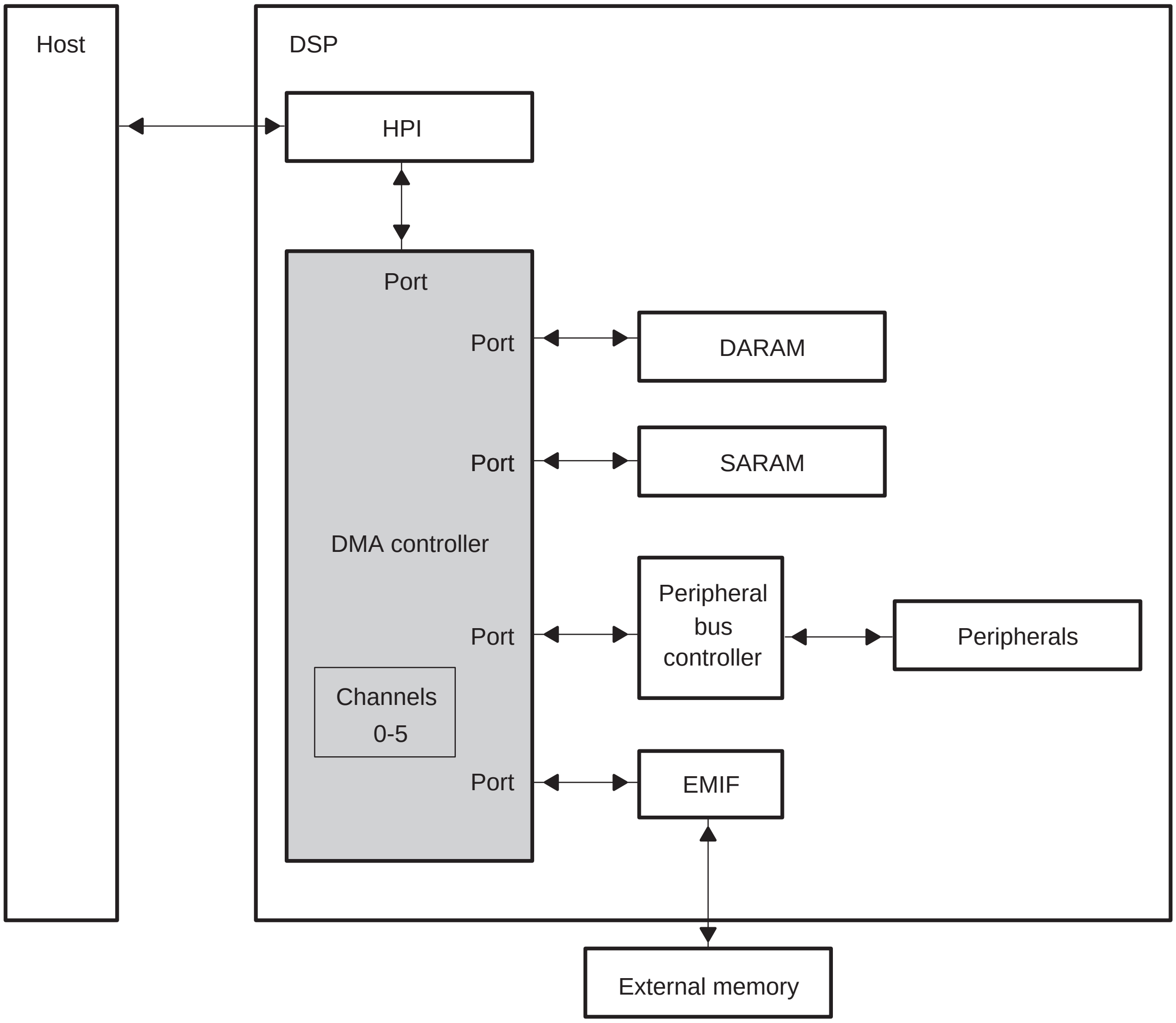
❑ Four standard ports. The DMA controller has a standard port for each of the following resources: internal dual-access RAM (DARAM), internal single-access RAM (SARAM), external memory, and peripherals. Data transfers among the standard ports occur in the six DMA channels. (The DMA channels are described on page 16.)

❑ Auxiliary port. A fifth port supports data transfers between memory and the host port interface (HPI). The HPI cannot access the peripheral port. If you want to transfer data from the HPI to the peripheral port, you must use data memory as a temporary buffer. Transfers between the HPI and the memory ports do not use a DMA channel.

  In TMS320VC5507/5509 DSPs, the HPI shares the auxiliary port with the USB module. The USB module is given the higher priority at the port.

It is possible for multiple channels (or for one or more channels and the HPI) to request access to the same standard port at the same time. To arbitrate simultaneous requests, the DMA controller has one programmable service chain that is used by each of the standard ports. For details on the service chain, see page 24.

Figure 1. Conceptual Block Diagram of the DMA Controller Connections



2 Channels and Port Accesses

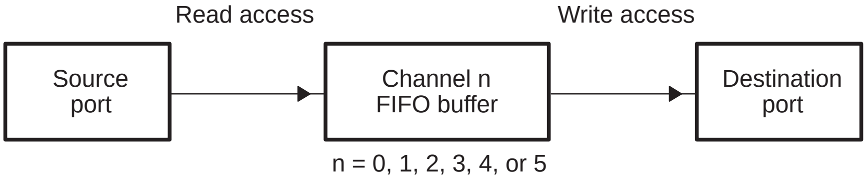
The DMA controller has six paths, called **channels**, to transfer data among the four standard ports (for DARAM, SARAM, external memory, and peripherals). Each channel reads data from one port (from the source) and writes data to that same port or another port (to the destination).

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.

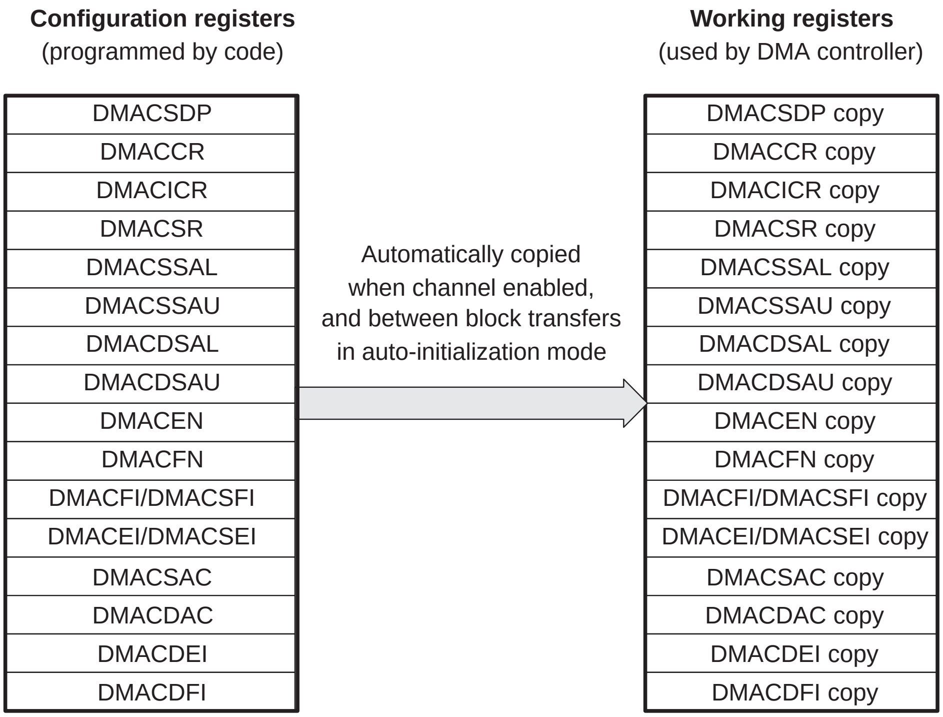
Figure 2. The Two Parts of a DMA Transfer



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 auto-initialization mode, see section 3. For more information about the DMA controller registers, see page 42.

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

Figure 3. Registers for Controlling the Context of a Channel



Configuration registers (programmed by code)	**Working registers** (used by DMA controller)
DMACSDP	DMACSDP copy
DMACCR	DMACCR copy
DMACICR	DMACICR copy
DMACSR	DMACSR copy
DMACSSAL	DMACSSAL copy
DMACSSAU	DMACSSAU copy
DMACDSAL	DMACDSAL copy
DMACDSAU	DMACDSAU copy
DMACEN	DMACEN copy
DMACFN	DMACFN copy
DMACFI/DMACSFI	DMACFI/DMACSFI copy
DMACEI/DMACSEI	DMACEI/DMACSEI copy
DMACSAC	DMACSAC copy
DMACDAC	DMACDAC copy
DMACDEI	DMACDEI copy
DMACDFI	DMACDFI copy

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 1 describes AUTOINIT, REPEAT, and ENDPROG. For a complete description of DMACCR, see section 14.4.

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 1. 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.

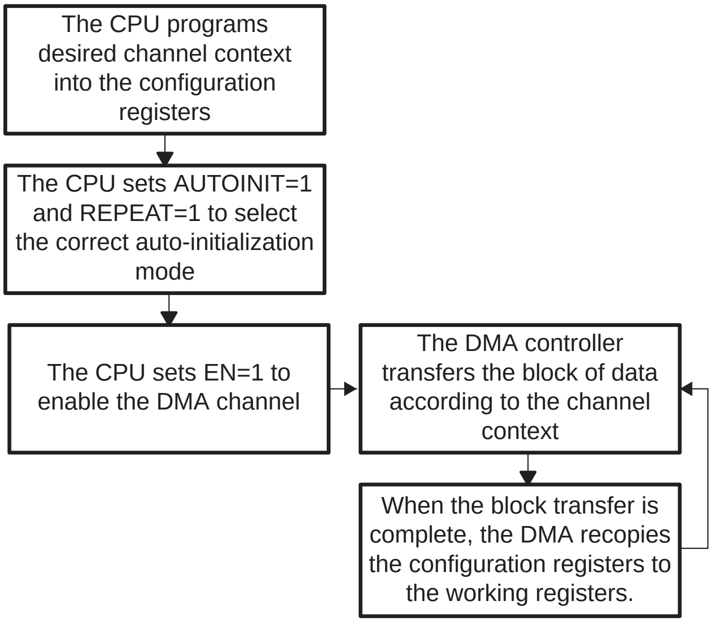
Table 1. Channel Control Register (DMACCR) Field Descriptions (Continued)

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.

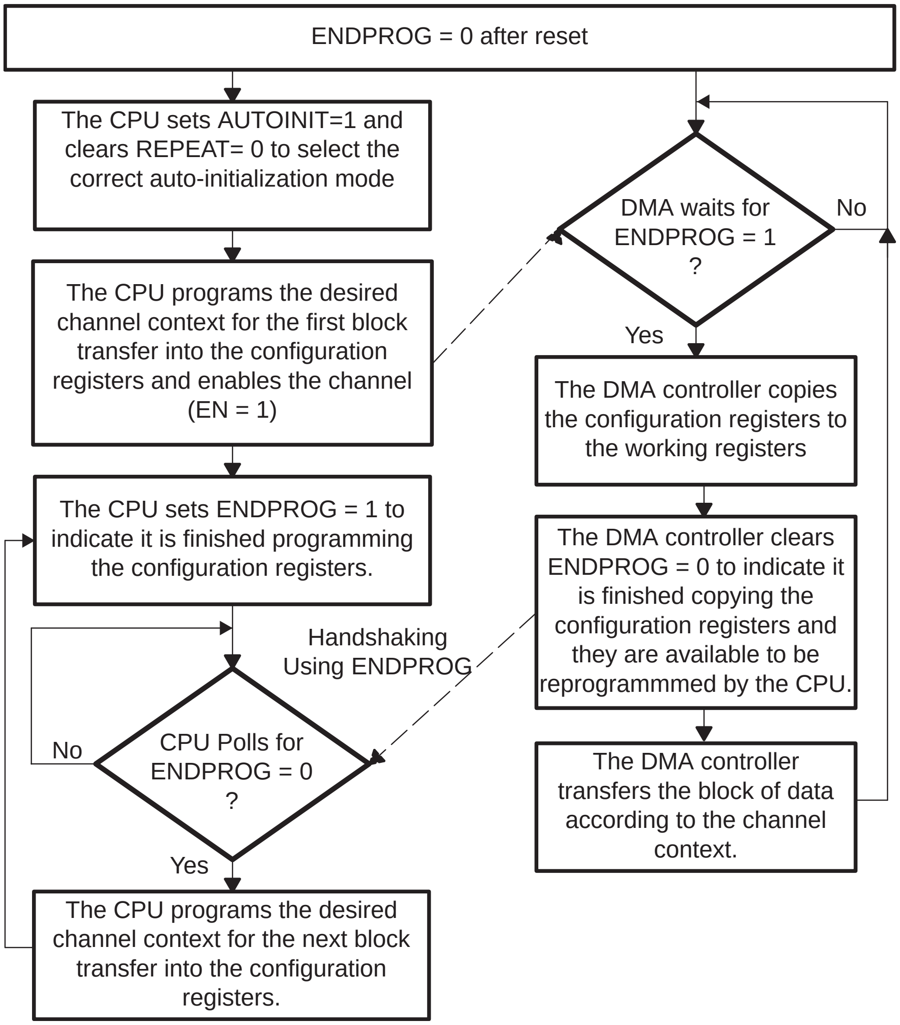
Figure 5. Auto-initialization Sequence with Unchanging Context (REPEAT = 1)


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 recopies the configuration registers to the working registers.

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)

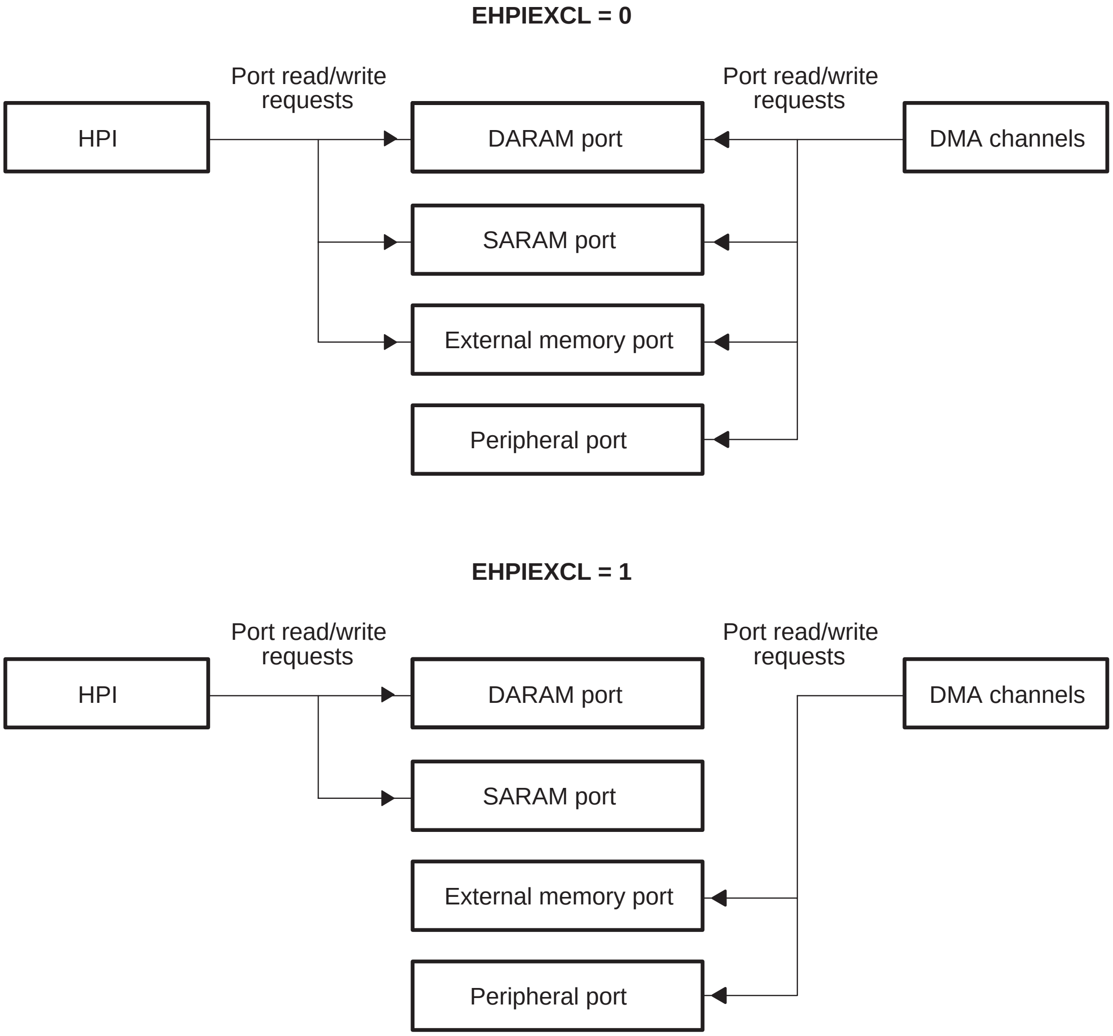

ENDPROG = 0 after reset
The CPU sets AUTOINIT=1 and clears REPEAT= 0 to select the correct auto-initialization mode
The CPU programs the desired channel context for the first block transfer into the configuration registers and enables the channel (EN = 1)
The CPU sets ENDPROG = 1 to indicate it is finished 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 transfers the block of data according to the channel context.

4 HPI Access Configurations

As shown in Figure 7, the EHPIEXCL bit in DMAGCR determines the relationship between the HPI and the DMA channels:

- ❑ When EHPIEXCL = 0, the HPI shares memory with the channels.
- ❑ When EHPIEXCL = 1, the HPI cannot access external memory, but it can access internal RAM without interruptions from the channels. The DARAM port and the SARAM port operate as if all the channels were disconnected from the service chain. (The service chain is described in section 5.)

Figure 7. HPI Access Configurations

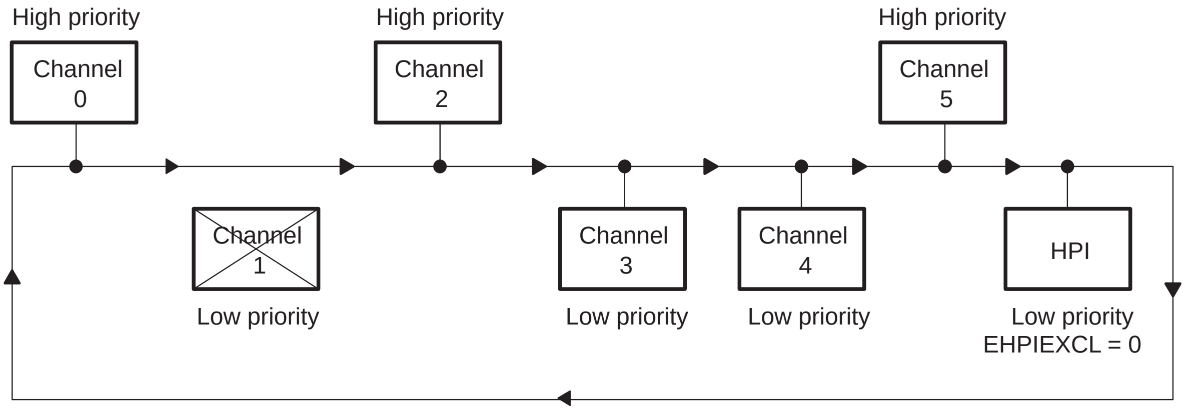


5 Service Chain

Each of the standard ports can arbitrate simultaneous access requests sent by the six DMA channels and the host port interface (HPI). Each of the standard 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 8. Important characteristics of the service chain are listed after the figure.

Section 5.1 contains an example that shows a service chain configuration applied to three ports.

Figure 8. One Possible Configuration for the Service Chains



- ❑ The channels and the HPI have a programmable priority level. Each channel has a PRIO bit in DMACCR for selecting a high priority or a low priority. You assign the HPI a high priority or low priority with the EHPIPRIO bit in DMAGCR. 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 and the HPI 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 and the HPI are low priority (in each of these channels, PRIO = 0, and for the HPI, EHPIPRIO = 0).

❑ The channels and the HPI have fixed positions in the service chain. The port checks the channels and the HPI in a repeating circular sequence: 0, 1, 2, 3, 4, 5, HPI, 0, 1, 2, 3, 4, 5, HPI, and so on. At each position in the service chain, the port checks whether the channel/HPI is ready and able to be serviced. If the channel is ready to be serviced and a higher priority request is not pending in another channel, 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 and the HPI in its repeating circular sequence, it will keep skipping channel 1 until the channel is reconnected.

❑ The HPI cannot access the peripheral port. The peripheral port operates as if the HPI is disconnected from the service chain.

❑ By writing a 1 to the EHPIEXCL bit in DMAGCR, you can give the HPI exclusive access to the DARAM and SARAM ports. Then the DARAM and SARAM ports operate as if only the HPI is connected to the service chain (as if none of the channels are connected, regardless of whether the channels are enabled). For more details, see *HPI Access Configurations* on page 23.

  In the figure, EHPIEXCL = 0. The HPI shares the RAM ports with the channels.

❑ 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.

5.1 Service Chain Example

Figure 9 shows a DMA service chain applied to the DARAM port, the external memory port, and the peripheral port. This 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).
- ❑ The HPI is sharing the internal memory with the channels (EHPIEXCL = 0 in DMAGCR) and is treated like a low-priority channel (EHPIPRIO = 0 in DMAGCR). Notice that the HPI is shown as disconnected in the peripheral port. This is because the HPI cannot access the peripheral port.

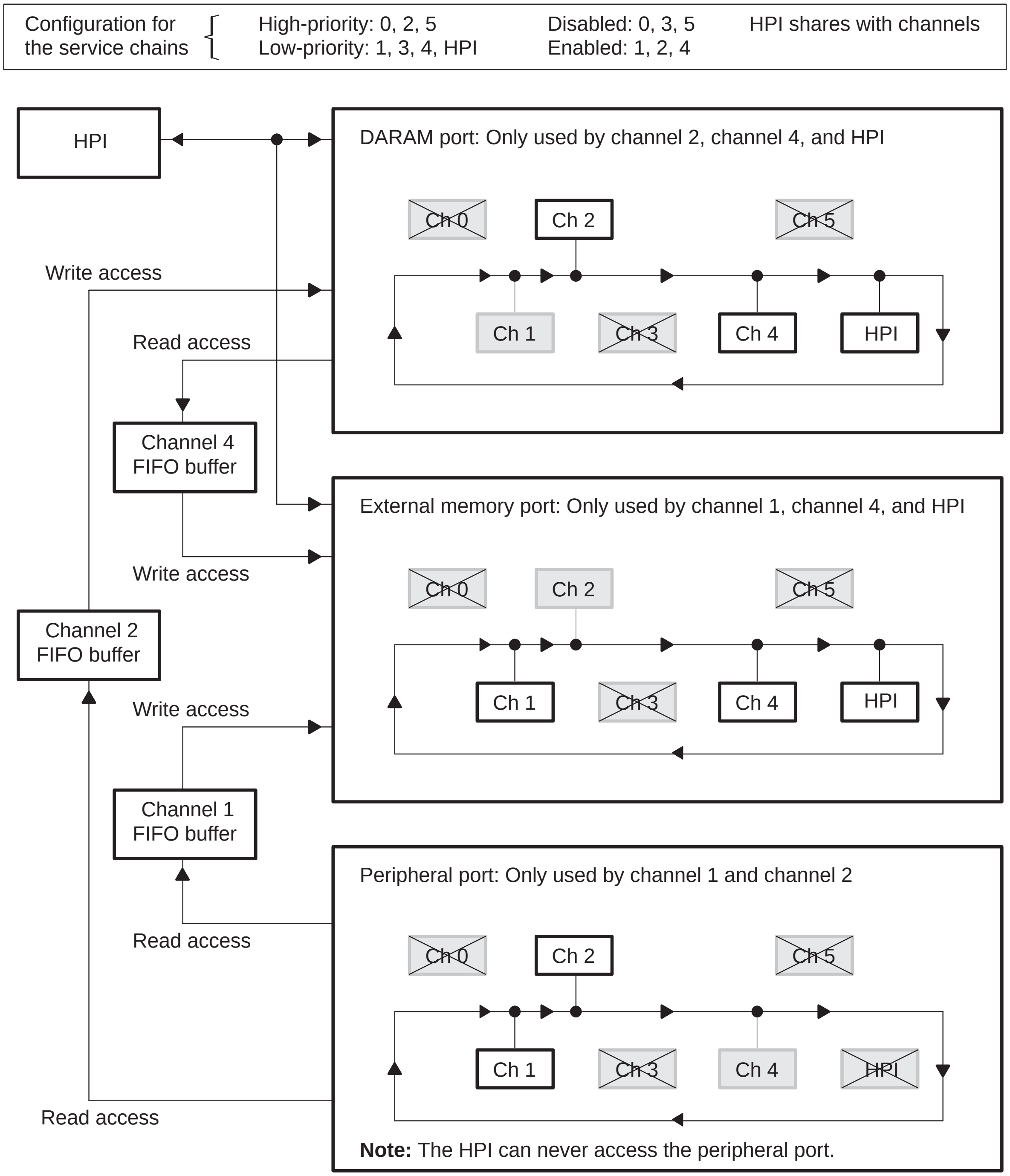
Table 2 summarizes the activity at the ports in Figure 9.

Table 2. Activity Shown in Figure 9

Port	This Port Arbitrates …
DARAM	Write access requests from channel 2 Read access requests from channel 4 Read or write access requests from the HPI
External memory	Write access requests from channel 1 Write access requests from channel 4 Read or write access request from the HPI
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 9. Service Chain Applied to Three DMA Ports



6 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 or the HPI 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 14 on page 42.

7 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 3. 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).

7.1 Start Address in Memory

Figure 10 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 of DMACSDP (page 62). 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.

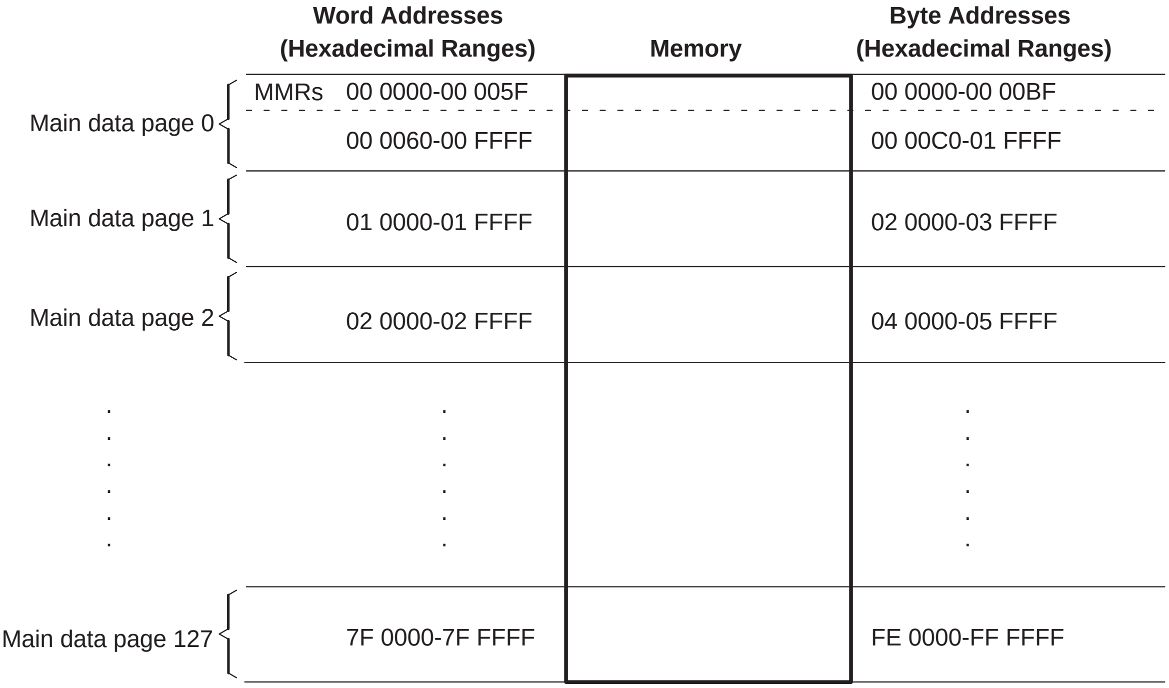
2) Load the 16 least significant bits (LSBs) of the byte address into DMACSSAL (for source) or DMACDSAL (for destination).

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 10. High-Level Memory Map for TMS320C55x DSPs



	Word Addresses (Hexadecimal Ranges)	Memory	Byte Addresses (Hexadecimal Ranges)
Main data page 0	MMRs 00 0000-00 005F		00 0000-00 00BF
	00 0060-00 FFFF		00 00C0-01 FFFF
Main data page 1	01 0000-01 FFFF		02 0000-03 FFFF
Main data page 2	02 0000-02 FFFF		04 0000-05 FFFF
. . . . . .	. . . . . .		. . . . . .
Main data page 127	7F 0000-7F FFFF		FE 0000-FF FFFF

7.2 Start Address in I/O Space

Figure 11 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 of DMACSDP (page 62). 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 11. 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

8 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 7). 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. At the end of an element transfer, the source address held by the DMA controller is the address of the last byte that was read from the source. Likewise, after a transfer, the destination address held by the DMA controller is the address of the last byte that was modified at the destination. Through software control, 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. If you choose a single-index or double-index addressing mode, you must load the appropriate index register or registers (see section 14.10 on page 67).

9 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 SARAM and 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 controller will perform four single accesses to move the burst of data. The peripheral port does not support burst capability; therefore, the DMA controller 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 byte.

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.

Burst mode is not supported when the source and destination are both configured to be the EMIF port.

10 Synchronizing Channel Activity

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, a read access request is sent to the source port and then a write access request is sent to the destination port. When all the bytes of the current element are transferred, the channel makes no more requests until the next occurrence of the synchronization event.

❑ Frame synchronization mode (FS = 1) requires one event to trigger an entire frame of elements. When the event occurs, the channel sends a read access request and a write access request for each element in the frame. When all the elements are transferred, the channel makes no more requests until the next occurrence of the event.

If you specify a synchronization event, the DMA access to the source and destination are handled as described in section 10.1. Once the request is received, it is handled according to the predefined position and the programmed priority of the channel in the DMA service chain (see page 26).

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.

If the DMA is configured to recognize a sync event (SYNC is something other than 00000b) and the sync event occurs before the channel is enabled, the sync event will be latched and serviced as soon as the channel is enabled. If it is preferable to ignore the sync events that occur before the channel is enabled, then the SYNC field should be set to 00000b while the channel is disabled.

10.1 DMA Channel Read Synchronization vs. Write Synchronization

When a DMA channel is configured for synchronization, the synchronization event is tied to the element read operation or the element write operation depending on the source and destination ports. There are three general cases (see Table 4):

- ❑ Case 1: Source port is peripheral; destination port is SARAM, DARAM, or EMIF.

  The channel waits for the synchronization event before reading from the peripheral port into the channel FIFO. Once the FIFO has filled, the DMA channel begins writing to the destination port to empty the FIFO (source synchronization).

- ❑ Case 2: Source port is SARAM, DARAM, or EMIF; destination is peripheral.

  As soon as the channel is enabled (i.e., the EN bit in the DMA Channel Control Register, DMACCR, is set) reads from the source port are performed to feed the channel FIFO. The FIFO writes to the peripheral port do not begin until the synchronization event is detected. When the channel is operating in frame-synchronization mode (DMACCR_FS = 1), several prereads may occur to the point of filling the FIFO while the channel is awaiting the synchronization event (destination synchronization).

- ❑ Case 3: Source port is SARAM, DARAM, or EMIF; destination port is SARAM, DARAM, or EMIF.

  The channel waits for the synchronization event before reading from the source port into the channel FIFO. Once the FIFO has filled, the DMA channel begins writing to the destination port to empty the FIFO (source synchronization).

Table 4. Read/Write Synchronization

Channel Synchronization Set by DMACCR sync[4:0] Not Equal to 00000	Source Port	Destination Port	Synchronization Event Triggers
No	Any port	Any port	No synchronization
Yes	Peripheral port	SARAM,DARAM, or EMIF port	Source read
Yes	SARAM, DARAM, or EMIF port.	Peripheral port	Destination write
Yes	SARAM,DARAM, or EMIF port	SARAM,DARAM, or EMIF port	Source read

10.2 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) when the DMA controller has completed the first read access (transfer from source port to channel buffer) after receiving synchronization.

10.3 Dropped Synchronization Events

If a synchronization event occurs in a channel before the DMA controller is done servicing the previous one in that channel (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, the DMA controller disables the channel (EN = 0 in DMACCR) and 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 36.

11 Monitoring Channel Activity

The DMA controller can send an interrupt to the CPU in response to the operational events listed in Table 5. 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 14.5 on page 54.) 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, no interrupt is sent and the status bit is not affected.

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 34.

Table 5. DMA Controller Operational Events and Their Associated Bits and Interrupts

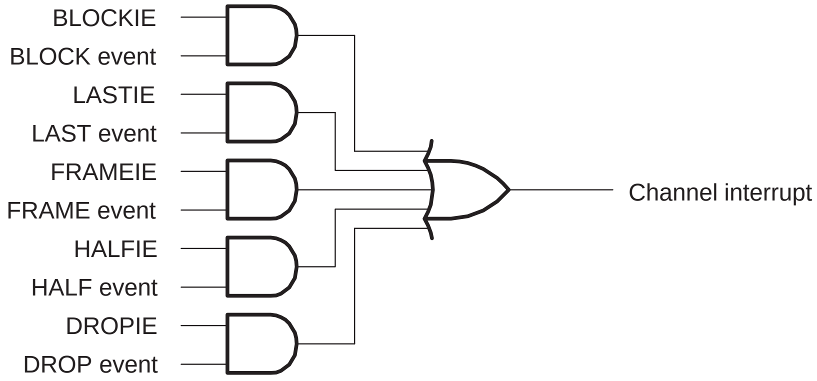
Operational Event	Interrupt Enable Bit	Status Bit	Associated Interrupt
Block transfer is complete	BLOCKIE	BLOCK	Channel interrupt
Last frame transfer has started	LASTIE	LAST	Channel interrupt
Frame transfer is complete	FRAMEIE	FRAME	Channel interrupt
First half of current frame has been transferred†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

11.1 Channel Interrupt

Each of the six channels has its own interrupt. As shown Figure 12, the channel interrupt is the logical OR of all the enabled operational events except the time-out event (the time-out event generates a bus-error interrupt request). You can choose any combination of these five 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. The bits in DMACSR are not automatically cleared. 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 12. 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:

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

When the current frame transfer is done or if a synchronization event is dropped (see 10.3 on page 36), 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 FRAME and DROP 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 14.5 on page 54.

11.2 Time-Out Error Conditions

A time-out error condition exists when a memory access has been stalled for too many cycles. Each of the four standard ports of the DMA controller is supported by hardware to detect a time-out error:

- ❑ DARAM port: A time-out counter in the DARAM port keeps track of how many cycles have passed since a request was made to access the DARAM. When the counter reaches its time-out value of 255 CPU clock cycles, the DARAM port generates an internal time-out signal. This counter can be enabled or disabled by writing to the DARAM time-out counter enable bit (DTCE in DMAGTCR). A time-out error on the DARAM port can occur because the CPU is using the port and preventing access by the DMA controller, or because an address was specified that does not exist in DARAM on the DSP.

- ❑ SARAM port: A time-out counter in the SARAM port keeps track of how many cycles have passed since a request was made to access the SARAM. When the counter reaches its time-out value of 255 CPU clock cycles, the SARAM port generates a time-out signal. This counter can be enabled or disabled by writing to the SARAM time-out counter enable bit (STCE in DMAGTCR). A time-out error on the SARAM port can occur because the CPU is using the port and preventing access by the DMA controller, or because an address was specified that does not exist in SARAM on the DSP.

- External memory port: A time-out counter in the external memory interface (EMIF) keeps track of how many cycles the external ready pin (ARDY) has been sampled low. When the counter reaches the programmed time-out value, the EMIF sends a time-out signal to the DMA controller. The external memory map is divided into four memory spaces, each of which has a programmable time-out value of up to 255 cycles.
- Peripheral port: A time-out counter in the peripheral bus controller counts how many cycles have passed since a request was made to access a peripheral. When the counter reaches its time-out value of 127 CPU clock cycles, the peripheral bus controller sends a time-out signal to the DMA controller. A time-out error on the peripheral port can occur because an address was specified that does not exist in I/O space on the DSP.

In response to a time-out signal, the DMA controller disables the channel (EN = 0 in DMACCR); activity in the 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 the time-out signal to the CPU as an interrupt request. The interrupt request sets the bus-error interrupt (BERRINT) flag bit in the CPU. The CPU can respond to the interrupt request or ignore the interrupt request.

12 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 CPU clock 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 24.
- Competition from the host port interface (HPI). If the HPI is sharing internal RAM with the channels, the DMA controller allocates cycles to the HPI like it does to channels. If you give the HPI exclusive access to the internal RAM, no channels can access the internal RAM until you change the HPI access configuration (see section 4 on page 23).
- Competition from the CPU. If the DMA controller and the CPU request access to the same internal memory block in the same cycle and the memory block cannot service both requests at the same time, the CPU request has higher priority. The DMA request is serviced as soon as there are no pending CPU requests.
- 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 34.

The minimum (best-case) latency is determined by the ports used. On the SARAM and DARAM ports, the DMA controller can initiate one access per cycle, if the DMA controller is not competing with the CPU for access to the same memory block. SARAM memory can support one access per cycle per memory block from either the DMA or the CPU. A CPU access to the same SARAM block used by the DMA (in the same cycle) will cause stalls on the DMA access. DARAM memory can support two accesses per cycle per memory block from either the DMA or the CPU. More than two CPU accesses to the same DARAM block used by the DMA will cause stalls on the DMA access. The best-case transfer rate for channels using these 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, programmed timings, and any delays caused by the memory itself (such as control of the ARDY pin). The minimum latency for the peripheral port is approximately 5 cycles per access.

13 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 emulation breakpoints, and the effect of a DSP reset on the DMA controller.

13.1 Reducing Power Consumed By the DMA Controller

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.

13.2 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.

13.3 DMA Controller after DSP Reset

A DSP reset resets the DMA controller and the DMA configuration registers. Some of the registers are initialized after reset and some are not. The register definitions that follow indicate the effects of a reset on the register contents.

14 DMA Controller Registers

Table 6 lists the types of registers in the direct memory access (DMA) controller. There are three global control registers (DMAGCR, DMAGSCR, and DMAGTCR) that affect all channel activity. In addition, for each of the DMA channels, there are channel configuration registers. For the I/O address of each register, see the data manual for your TMS320C55x DSP.

Table 6. Registers of the DMA Controller

Register	Description	For Details, See ...
DMAGCR	Global control register (only one)	Page 44
DMAGSCR†	Global software compatibility register (only one)	Page 45
DMAGTCR†	Global time-out control register (only one)	Page 47
DMACCR	Channel control register (one for each channel)	Page 48
DMACICR	Interrupt control register (one for each channel)	Page 54
DMACSR	Status register (one for each channel)	Page 54
DMACSDP	Source and destination parameters register (one for each channel)	Page 58
DMACSSAL	Source start address (lower part) register (one for each channel)	Page 63
DMACSSAU	Source start address (upper part) register (one for each channel)	Page 63
DMACDSAL	Destination start address (lower part) register (one for each channel)	Page 65
DMACDSAU	Destination start address (upper part) register (one for each channel)	Page 65
DMACEN	Element number register (one for each channel)	Page 66
DMACFN	Frame number register (one for each channel)	Page 66

† This register and its associated function are not supported on the TMS320VC5509 DSP, but are supported on the TMS320VC5503/5507/5509A/5510 DSPs.

Table 6. Registers of the DMA Controller (Continued)

Register	Description	For Details, See ...
DMACEI/ DMACSEI	Element index register/ Source element index register (one for each channel)	Page 67
DMACFI/ DMACSFI	Frame index register/ Source frame index register (one for each channel)	Page 67
DMACDEI†	Destination element index register (one for each channel)	Page 67
DMACDFI†	Destination frame index register (one for each channel)	Page 67
DMACSAC†	Source address counter register (one for each channel)	Page 70
DMACDAC†	Destination address counter register (one for each channel)	Page 70

† This register and its associated function are not supported on the TMS320VC5509 DSP, but are supported on the TMS320VC5503/5507/5509A/5510 DSPs.

14.1 Global Control Register (DMAGCR)

The global control register (see Figure 13 and Table 7) is a 16-bit read/write register. Use this I/O-mapped register to set the emulation mode of the DMA controller (FREE) and to define how the DMA controller treats the host port interface (EHPIEXCL and EHPIPRIO).

Figure 13. Global Control Register (DMAGCR)

15–8
Reserved
R-0

7–4	3	2	1	0
Reserved	Reserved[†]	FREE	EHPIEXCL	EHPIPRIO
R-0	R/W-1	R/W-0	R/W-0	R/W-0

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

[†] Always write 1 to this reserved bit.

Table 7. Global Control Register (DMAGCR) Field Descriptions

Bit	Field	Value	Description
15-4	Reserved		These read-only bits return 0s when read.
3	Reserved	1	Always write 1 to this reserved bit.
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	EHPIEXCL		HPI exclusive access bit. EHPIEXCL determines whether the host port interface (HPI) has exclusive access to the internal RAM of the DSP. **Note:** Regardless of the value of EHPIEXCL, the HPI cannot access the peripheral port.
		0	The HPI shares the internal RAM with the DMA channels. The HPI can access any internal and external memory in its address reach.
		1	The HPI has exclusive access to the internal RAM. If any channels must access the DARAM port or the SARAM port, activity in these channels is suspended. In this HPI access configuration, the HPI can only access the DARAM port and the SARAM port. It cannot access the external memory port.

Table 7. Global Control Register (DMAGCR) Field Descriptions (Continued)

Bit	Field	Value	Description
0	EHPIPRIO		HPI priority bit. EHPIPRIO assigns the HPI a high or low priority level in the service chain of the DMA controller: **Note:** When the HPI has exclusive access to the DARAM and SARAM ports (EHPIEXCL = 1), the HPI priority is irrelevant at these ports because none of the DMA channels can access the DARAM and SARAM ports.
		0	Low priority level.
		1	High priority level.

14.2 Global Software Compatibility Register (DMAGSCR)

The global software compatibility register is a 16-bit read/write register used to control how the DMA controller gets the destination element index and the destination frame index. The original DMA controller design used the same element index register (DMACEI) for both source and destination, and the same frame index register (DMACFI) for both source and destination. Later designs were enhanced to allow separate source and destination indexes. In the enhanced mode:

- DMACEI is DMACSEI, the source element index register. DMACFI is DMACSFI, the source frame index register.
- The destination element index is stored in a separate destination element index register (DMACDFI) and the destination frame index is stored in a separate destination frame index register (DMACDFI).

DMAGSCR provides the ability to choose either the original method of indexing (to maintain software compatibility with code written for the original design) or the enhanced method of indexing.

DMAGSCR is summarized by Figure 14 and Table 8.

Note:

DMAGSCR and its associated function are not supported on the TMS320VC5509 DSP, but they are supported on the TMS320VC5503/5507/5509A/5510 DSPs.

Figure 14. Global Software Compatibility Register (DMAGSCR)

15–8
Reserved
R-0

7–1	0
Reserved	DINDXMD
R-0	R/W-0

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

Table 8. Global Software Compatibility Register (DMAGSCR) Field Descriptions

Bit	Field	Value	Description
15-1	Reserved		These read-only bits return 0s when read.
0	DINDXMD		Destination element and frame index mode bit. This bit determines which registers will be used to indicate the destination element and frame indexes.
		0	Compatibility mode. One element index for both the source and destination is stored in the channel source element index register (DMACSEI). One frame index for both the source and destination is stored in the channel source frame index register (DMACSEI).
		1	Enhanced mode. The source element index is stored in the channel source element index register (DMACSEI). The destination element index is stored in the channel destination element index register (DMACDEI). The source frame index is stored in the channel source frame index register (DMACSFI). The destination frame index is stored in the channel destination frame index register (DMACDFI).

14.3 Global Time-Out Control Register (DMAGTCR)

The global time-out control register (see Figure 15 and Table 9) is a 16-bit read/write register used to enable or disable time-out counters on the SARAM and DARAM ports. If the time-out counters are disabled, the DMA controller will never generate a time-out error for these ports. For details about the time-out error conditions, see section 11.2 (page 38).

Note:

DMAGTCR and its associated function are not supported on the TMS320VC5509 DSP, but they are supported on the TMS320VC5503/5507/5509A/5510 DSPs.

Figure 15. Global Time-Out Control Register (DMAGTCR)

15 – 8
Reserved
R-0

7 – 2	1	0
Reserved	DTCE	STCE
R-0	R/W-0	R/W-0

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

Table 9. Global Time-Out Control Register (DMAGTCR) Field Descriptions

Bit	Field	Value	Description
15-2	Reserved		These read-only bits return 0s when read.
1	DTCE		DARAM time-out counter enable bit. This bit enables/disables the time-out counter used to monitor delays on DMA requests to the DARAM port.
		0	DARAM time-out counter disabled.
		1	DARAM time-out counter enabled.
0	STCE		SARAM time-out counter enable bit. This bit enables/disables the time-out counter used to monitor delays on DMA requests to the SARAM port.
		0	SARAM time-out counter disabled.
		1	SARAM time-out counter enabled.

14.4 Channel Control Register (DMACCR)

Each channel has a channel control register of the form shown in Figure 16. 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 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)

Table 10 describes the fields of this register.

Figure 16. Channel Control Register (DMACCR)

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

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

† Bit 10 must be kept 0 for proper operation of the DMA controller.

Table 10. Channel Control Register (DMACCR) Field Descriptions

Bit	Field	Value	Description
15-14	DSTAMODE		Destination addressing mode bits. DSTAMODE determines the addressing mode used by the DMA controller when it writes to the destination port of the channel. At the end of a transfer but before any incrementing, the destination address is the address of the last byte that was modified at the destination.
		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 10. Channel Control Register (DMACCR) Field Descriptions (Continued)

Bit	Field	Value	Description
13-12	SRCAMODE		Source addressing mode bits. SRCAMODE determines the addressing mode used by the DMA controller when it reads from the source port of the channel. At the end of a transfer but before any incrementing, the source address is the address of the last byte that was read from the source.
		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 10. Channel Control Register (DMACCR) 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	Reserved	0	This reserved bit must be kept 0. Make sure that whenever your program modifies DMACCR, it writes a 0 to bit 10.

Table 10. Channel Control Register (DMACCR) 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 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.
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	The 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	The channel is enabled. The channel can be serviced by the DMA controller at the next available time slot.

Table 10. Channel Control Register (DMACCR) 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 the data manual)	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 DMA is configured to recognize a sync event (SYNC is something other than 00000b) and the sync event occurs before the channel is enabled, the sync event will be latched and serviced as soon as the channel is enabled. If it is preferable to ignore the sync events that occur before the channel is enabled, then the SYNC field should be set to 00000b while the channel is disabled. The available SYNC events for each TMS320C55x DSP are documented in the device-specific data manuals.

14.5 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 17 and described in Table 11 and Table 12.

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. In addition the DMA controller can send a bus-error interrupt request to the CPU in response to a time-out error. The bus-error interrupt also has a set of flag and enable bits in the CPU.

To see which operational event or events have occurred in the DMA controller, your program can read DMACSR. The DMA controller sets one of the interrupt flag bits (bits 5-0) only if the operational event occurs and the associated interrupt enable bit is set in DMACICR. After your program reads DMACSR, all of its bits are cleared automatically.

The SYNC bit (bit 6) of DMACSR 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 17. Interrupt Control Register (DMACICR) and Status Register (DMACSR)

DMACICR

15							8
Reserved							
R-0							

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

DMACSR

15							8
Reserved							
R-0							

7	6	5	4	3	2	1	0
Reserved	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 DSP reset

Table 11. Interrupt Control Register (DMACICR) Field Descriptions

Bit	Field	Value	Description
15-6	Reserved		These read-only bits return 0s when read.
5	BLOCKIE		Whole block interrupt enable bit. BLOCKIE determines how the DMA controller responds when all of the current block has been transferred from the source port to the destination port.
		0	Do not record the event.
		1	Set the BLOCK bit and send the channel interrupt request to the CPU.
4	LASTIE		Last frame interrupt enable bit. LASTIE determines how the DMA controller responds when the DMA controller starts transferring the last frame from the source port to the destination port.
		0	Do not record the event.
		1	Set the LAST bit and send the channel interrupt request to the CPU.
3	FRAMEIE		Whole frame interrupt enable bit. FRAMEIE determines how the DMA controller responds when the all of the current frame has been transferred from the source port to the destination port.
		0	Do not record the event.
		1	Set the FRAME bit and send the channel interrupt request to the CPU.
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 from the source port 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 record the event.
		1	Set the HALF bit and send the channel interrupt request to the CPU.

Table 11. Interrupt Control Register (DMACICR) Field Descriptions (Continued)

Bit	Field	Value	Description
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 record the drop.
		1	Set the DROP bit and send the channel interrupt request to the CPU.
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 conditions are described in section 11.2 (page 38).
		0	Do not record the time-out error.
		1	Set the TIMEOUT bit and send the bus-error interrupt request to the CPU.

Table 12. Status Register (DMACSR) Field Descriptions

Bit	Field	Value	Description
15-7	Reserved		These read-only bits return 0s when read.
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. In response to the event, the synchronized channel submits an access request to its source port.
			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.

Table 12. Status Register (DMACSR) Field Descriptions (Continued)

Bit	Field	Value	Description
5	BLOCK		Whole block status bit. The DMA controller sets BLOCK only if BLOCKIE = 1 in DMACICR and all of the current block has been transferred from the source port 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.
4	LAST		Last frame status bit. The DMA controller sets LAST only if LASTIE = 1 in DMACICR and the DMA controller has started transferring the last frame from the source port to the destination port:
		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 only if FRAMEIE = 1 in DMACICR and all of the current frame has been transferred from the source port 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 only if HALFIE = 1 in DMACICR and the first half of the current frame has been transferred from the source port 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 12. Status Register (DMACSR) 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 is done 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.
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 conditions are described in section 11.2 (page 38).
		0	A time-out error has not occurred, or TIMEOUT has been cleared.
		1	A time-out error has occurred. A bus-error interrupt request has been sent to the CPU.

14.6 Source and Destination Parameters Register (DMACSDP)

Each channel has a source and destination parameters register of the form shown in Figure 18. 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). Table 13 describes the fields of this register.

Figure 18. Source and Destination Parameters Register (DMACSDP)

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

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

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

Table 13. Source and Destination Parameters Register (DMACSDP) Field Descriptions

Bit	Field	Value	Description
15-14	DSTBEN		Destination burst enable bits. 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. Burst is not supported when both the source and destination are configured to be the EMIF port (SRC = DST = XX10b).
		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 (do not use).
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. Table 14 (page 63) shows the instances where data packing is performed.

† On the 5503/5507/5509/5509A, when the source/destination is configured as a peripheral port, the DMA cannot read/write the Secure ROM Register (SROM), the USB Idle Control and Status Register (USBIDLECTL), the External Bus Selection register (EBSR), the timer registers, the EMIF registers, the PLL registers or the DMA registers.

‡ Refer to the device specific data manuals for the available DMA ports for each TMS320C55x DSP.

Table 13. Source and Destination Parameters Register (DMACSDP) Field Descriptions (Continued)

Bit	Field	Value	Description
12-9	DST[†‡]		Destination selection bits. DST selects which DMA port is the destination for data transfers in the channel (a bit shown as X can be 0 or 1):
		0000b	SARAM (single-access RAM inside the DSP).
		0001b	DARAM (dual-access RAM inside the DSP).
		0010b	External memory (via the external memory interface, EMIF).
		0011b	Peripherals (via the peripheral bus controller). Connection of a channel configured to have an 8-bit data type to the peripheral port is not supported.
		Others	Reserved.
8-7	SRCBEN		Source burst enable bits. 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. Burst is not supported when both the source and destination are configured to be the EMIF port (SRC = DST = XX10b).
		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 (do not use).

[†] On the 5503/5507/5509/5509A, when the source/destination is configured as a peripheral port, the DMA cannot read/write the Secure ROM Register (SROM), the USB Idle Control and Status Register (USBIDLECTL), the External Bus Selection register (EBSR), the timer registers, the EMIF registers, the PLL registers or the DMA registers.

[‡] Refer to the device specific data manuals for the available DMA ports for each TMS320C55x DSP.

Table 13. Source and Destination Parameters Register (DMACSDP) Field Descriptions (Continued)

Bit	Field	Value	Description
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. Table 14 (page 63) shows the instances where data packing is performed.
5-2	SRC†‡		Source selection bits. SRC selects which DMA port is the source for data transfers in the channel (a bit shown as X can be 0 or 1):
		0000b	SARAM (single-access RAM inside the DSP).
		0001b	DARAM (dual-access RAM inside the DSP).
		0010b	External memory (via the external memory interface, EMIF).
		0011b	Peripherals (via the peripheral bus controller). Connection of a channel configured to have an 8-bit data type to the peripheral port is not supported.
		Others	Reserved.

† On the 5503/5507/5509/5509A, when the source/destination is configured as a peripheral port, the DMA cannot read/write the Secure ROM Register (SROM), the USB Idle Control and Status Register (USBIDLECTL), the External Bus Selection register (EBSR), the timer registers, the EMIF registers, the PLL registers or the DMA registers.

‡ Refer to the device specific data manuals for the available DMA ports for each TMS320C55x DSP.

Table 13. Source and Destination Parameters Register (DMACSDP) Field Descriptions (Continued)

Bit	Field	Value	Description
1-0	DATA-TYPE		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 the DSTAMODE bits and the SRCAMODE bits of DMACCR (see pages 49 and 50).
		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. Connection of a channel configured as 8-bit data type to the peripheral port is not supported.
		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 (do not use).

[†] On the 5503/5507/5509/5509A, when the source/destination is configured as a peripheral port, the DMA cannot read/write the Secure ROM Register (SROM), the USB Idle Control and Status Register (USBIDLECTL), the External Bus Selection register (EBSR), the timer registers, the EMIF registers, the PLL registers or the DMA registers.

[‡] Refer to the device specific data manuals for the available DMA ports for each TMS320C55x DSP.

Table 14. Data Packing Performed by the DMA Controller

Data Type	Port Bus Size	Data Packing
8-bit	16-bit	Two data values packed into 16 bits
8-bit	32-bit	Four data values packed into 32 bits
16-bit	32-bit	Two data values packed into 32 bits

The SARAM, DARAM, and EMIF ports have 32-bit internal buses. The peripheral port has a 16-bit internal bus.

14.7 Source Start Address Registers (DMACSSAL and DMACSSAU)

Each channel has two source start address registers, which are shown in Figure 19 and described in Table 15 and Table 16. 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 62).

3) Make sure 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.

4) On the 5503/5507/5509/5509A, when the source/destination is configured as a peripheral port, the DMA cannot read/write the Secure ROM Register (SROM), the USB Idle Control and Status Register (USBIDLECTL), the External Bus Selection register (EBSR), the timer registers, the EMIF registers, the PLL registers or the DMA registers.

The destination start address is supplied by DMACDSAL and DMACDSAU, which are described in section 14.8.

Figure 19. Source Start Address Registers (DMACSSAL and DMACSSAU)

DMACSSAL

15	0
SSAL	
R/W-x	

DMACSSAU

15	0
SSAU	
R/W-x	

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

Table 15. Source Start Address Register - Lower Part (DMACSSAL) Field Description

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

Table 16. Source Start Address Register - Upper Part (DMACSSAU) 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).

14.8 Destination Start Address Registers (DMACDSAL and DMACDSAU)

Each channel has two destination start address registers, which are shown in Figure 20 and described in Table 17 and Table 18. 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 62).

3) Make sure 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.

4) On the 5503/5507/5509/5509A, when the source/destination is configured as a peripheral port, the DMA cannot read/write the Secure ROM Register (SROM), the USB Idle Control and Status Register (USBIDLECTL), the External Bus Selection register (EBSR), the timer registers, the EMIF registers, the PLL registers or the DMA registers.

The source start address is supplied by DMACSSAL and DMACSSAU, which are described in section 14.7.

Figure 20. Destination Start Address Registers (DMACDSAL and DMACDSAU)

DMACDSAL

15–0
DSAL
R/W-x

DMACDSAU

15–0
DSAU
R/W-x

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

Table 17. Destination Start Address Register − Lower Part (DMACDSAL) Field Description

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

Table 18. Destination Start Address Register - Upper Part (DMACDSAU) 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).

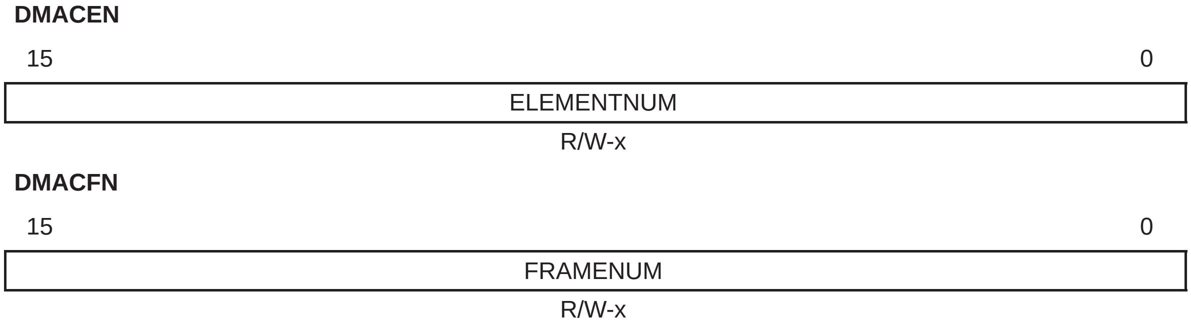
14.9 Element Number Register (DMACEN) and Frame Number Register (DMACFN)

Each channel has an element number register and a frame number register, (see Figure 21, Table 19, and Table 20). 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:

$1 \le$ frame number ≤ 65535
$1 \le$ element number ≤ 65535

DMACEN and DMACFN are uninitialized after a DSP reset.

Figure 21. Element Number Register (DMACEN) and Frame Number Register (DMACFN)



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

Table 19. Element Number Register (DMACEN) Field Description

Bit	Field	Value	Description
15-0	ELEMENTNUM	0000h	Reserved (do not use).
		0001h-FFFFh	Number of elements per frame (1-65535).

Table 20. Frame Number Register (DMACFN) Field Description

Bit	Field	Value	Description
15-0	FRAMENUM	0000h	Reserved (do not use).
		0001h-FFFFh	Number of frames per block (1-65535).

14.10 Element Index Registers (DMACEI/DMACSEI, DMACDEI) and Frame Index Registers (DMACFI/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, of DMACCR (see pages 49 and 50). To support the index addressing modes, there are four index registers: two source index registers (DMACSEI and DMACSFI) and two destination index registers (DMACDEI and DMACDFI). The way these registers are used depends on the destination index mode chosen with the DINDXMD bit of DMAGSCR.

When DINDXMD = 0 (the default forced by a DSP reset), a compatibility mode is selected. In the original DMA controller design, the source and the destination shared one element index register called DMACEI and one frame index register called DMACFI. When DINDXMD = 0, compatible behavior is enabled; DMACSEI is used as DMACEI, and DMACSFI is used as DMACFI. The destination index registers are not used.

When DINDXMD=1, an enhanced mode is selected. In this mode, the source index registers are used only for the source, and the destination index registers are used for the destination.

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

The element and/or frame index must produce an aligned address according to the data type selected in the DATATYPE field of DMACSDP. If the data type is 32-bit, the element/frame index must be (4 x N) + 1 (where N = -2, -1, 0, 1, 2,...). If the data type is 16-bit, the element/frame index must be (2 x N) + 1 (where N= -2, -1, 0, 1, 2,...). If the type is 8-bit, the element/frame can have any value.

If the CPU attempts to write an 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.

Notes:

1) DMACDEI, DMACDFI, and their associated functions are not supported on the TMS320VC5509 DSP, but they are supported on the TMS320VC5503/5507/5509A/5510 DSPs.

2) Make sure 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.

3) If you use an index addressing mode, make sure that all the addresses computed by the DMA controller will match the alignment constraint for the chosen data type. For more details, see the description for the DATATYPE bits of DMACSDP (see page 62).

The index registers are summarized by Figure 22 and the tables that follow the figure. All of the element index and frame index registers are uninitialized after a DSP reset.

Figure 22. Element Index Registers (DMACSEI, DMACDEI) and Frame Index Registers (DMACSFI, DMACDFI)

DMACEI / DMACSEI

15	0
ELEMENTNDX	
R/W-x	

DMACFI / DMACSFI

15	0
FRAMENDX	
R/W-x	

DMACDEI

15	0
ELEMENTNDX	
R/W-x	

DMACDFI

15	0
FRAMENDX	
R/W-x	

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

Table 21. Source Element Index Register (DMACSEI / DMACEI) Field Description

Bit	Field	Value	Description
15-0	ELEMENTNDX	-32768 to 32767	When DINDXMD = 0, DMACSEI is used as DMACEI; it contains the element index (in bytes) for both the source and the destination. When DINDXMD = 1, DMACSEI contains the source element index (in bytes).

Table 22. Source Frame Index Register (DMACSFI / DMACFI) Field Description

Bit	Field	Value	Description
15-0	FRAMENDX	-32768 to 32767	When DINDXMD = 0, DMACSFI is used as DMACFI; it contains the frame index (in bytes) for both the source and the destination. When DINDXMD = 1, DMACSFI contains the source frame index (in bytes).

Table 23. Destination Element Index Register (DMACDEI) Field Description

Bit	Field	Value	Description
15-0	ELEMENTNDX	-32768 to 32767	When DINDXMD = 1, DMACDEI contains the destination element index (in bytes).

Table 24. Destination Frame Index Register (DMACDFI) Field Description

Bit	Field	Value	Description
15-0	FRAMENDX	-32768 to 32767	When DINDXMD = 1, DMACDFI contains the destination frame index (in bytes).

14.11 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.

The address counters are summarized by Figure 23, Table 25, and Table 26. DMACSAC and DMACDAC are uninitialized after a DSP reset.

Note:

DMACSAC, DMACDAC, and their associated functions are not supported on the TMS320VC5509 DSP, but they are supported on the TMS320VC5503/5507/5509A/5510 DSPs.

Figure 23. Source Address Counter (DMACSAC) and Destination Address Counter (DMACDAC)

DMACSAC

15	0
SAC	
R/W-x	

DMACDAC

15	0
DAC	
R/W-x	

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

Table 25. Source Address Counter (DMACSAC) Field Description

Bit	Field	Value	Description
15-0	SAC	0000h-FFFFh	Low 16 bits of current source address.

Table 26. Destination Address Counter (DMACDAC) Field Description

Bit	Field	Value	Description
15-0	DAC	0000h-FFFFh	Low 16 bits of the current destination address.

This page is intentionally left blank.

Revision History

Page	Additions/Modifications/Deletions
21	Figure 5 − updated
22	Figure 6 − updated

This page is intentionally left blank.

Index

IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, modifications, enhancements, improvements, and other changes to its products and services at any time and to discontinue any product or service without notice. Customers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All products are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its hardware products to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by government requirements, testing of all parameters of each product is not necessarily performed.

TI assumes no liability for applications assistance or customer product design. Customers are responsible for their products and applications using TI components. To minimize the risks associated with customer products and applications, customers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any TI patent right, copyright, mask work right, or other TI intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information published by TI regarding third-party products or services does not constitute a license from TI to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of TI information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. Reproduction of this information with alteration is an unfair and deceptive business practice. TI is not responsible or liable for such altered documentation. Information of third parties may be subject to additional restrictions.

Resale of TI products or services with statements different from or beyond the parameters stated by TI for that product or service voids all express and any implied warranties for the associated TI product or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

TI products are not authorized for use in safety-critical applications (such as life support) where a failure of the TI product would reasonably be expected to cause severe personal injury or death, unless officers of the parties have executed an agreement specifically governing such use. Buyers represent that they have all necessary expertise in the safety and regulatory ramifications of their applications, and acknowledge and agree that they are solely responsible for all legal, regulatory and safety-related requirements concerning their products and any use of TI products in such safety-critical applications, notwithstanding any applications-related information or support that may be provided by TI. Further, Buyers must fully indemnify TI and its representatives against any damages arising out of the use of TI products in such safety-critical applications.

TI products are neither designed nor intended for use in military/aerospace applications or environments unless the TI products are specifically designated by TI as military-grade or "enhanced plastic." Only products designated by TI as military-grade meet military specifications. Buyers acknowledge and agree that any such use of TI products which TI has not designated as military-grade is solely at the Buyer's risk, and that they are solely responsible for compliance with all legal and regulatory requirements in connection with such use.

TI products are neither designed nor intended for use in automotive applications or environments unless the specific TI products are designated by TI as compliant with ISO/TS 16949 requirements. Buyers acknowledge and agree that, if they use any non-designated products in automotive applications, TI will not be responsible for any failure to meet such requirements.

Following are URLs where you can obtain information on other Texas Instruments products and application solutions:

Products		**Applications**	
Amplifiers	amplifier.ti.com	Audio	www.ti.com/audio
Data Converters	dataconverter.ti.com	Automotive	www.ti.com/automotive
DSP	dsp.ti.com	Broadband	www.ti.com/broadband
Clocks and Timers	www.ti.com/clocks	Digital Control	www.ti.com/digitalcontrol
Interface	interface.ti.com	Medical	www.ti.com/medical
Logic	logic.ti.com	Military	www.ti.com/military
Power Mgmt	power.ti.com	Optical Networking	www.ti.com/opticalnetwork
Microcontrollers	microcontroller.ti.com	Security	www.ti.com/security
RFID	www.ti-rfid.com	Telephony	www.ti.com/telephony
RF/IF and ZigBee® Solutions	www.ti.com/lprf	Video & Imaging	www.ti.com/video
		Wireless	www.ti.com/wireless

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