Section 31. DMA Controller

HIGHLIGHTS

This section of the manual contains the following topics:

31.1 INTRODUCTION

The Direct Memory Access (DMA) controller is a bus master module that is useful for data transfers between different peripherals without intervention from the CPU. The source and destination of a DMA transfer can be any of the memory-mapped modules included in the PIC32MX.For example, memory, or one of the Peripheral Bus (PBUS) devices such as SPI, UART, I^2C™ and so on.

Following are some of the key features of the DMA module:

- Depending on the device variant, up to eight identical channels are available, including the following:
  - Auto-Increment Source and Destination Address registers
  - Source and Destination Pointers
- Depending on the device variant, data transfers of up to 64 Kbytes are supported
- Automatic Word-Size Detection, featuring the following:
  - Transfer granularity down to byte level
  - Bytes need not be word-aligned at source and destination
- Fixed Priority Channel Arbitration
- Flexible DMA Channel Operating modes, including the following:
  - Manual (software) or automatic (interrupt) DMA requests
  - One-Shot or Auto-Repeat Block Transfer modes
  - Channel-to-channel chaining
- Flexible DMA Requests, featuring the following:
  - A DMA request can be selected from any of the peripheral interrupt sources
  - Each channel can select any interrupt as its DMA request source
  - A DMA transfer abort can be selected from any of the peripheral interrupt sources
  - Automatic transfer termination upon a data pattern match
- Multiple DMA Channel Status Interrupts, supplying the following:
  - DMA channel block transfer complete
  - Source empty or half empty
  - Destination full or half full
  - DMA transfer aborted due to an external event
  - Invalid DMA address generated
- DMA Debug Support Features, including the following:
  - Most recent address accessed by a DMA channel
  - Most recent DMA channel to transfer data
- CRC Generation Module, featuring the following:
  - CRC module can be assigned to any of the available channels
  - Data read from the source can be reordered on some device variants
  - CRC module is highly configurable

The following features are also available in the DMA controller:

- Unaligned Transfers
- Different Source and Destination Sizes
- Memory-to-Memory Transfers
- Memory-to-Peripheral Transfers
- Channel Auto-Enable
- Events Start/Stop
- Pattern Match Detection
- Channel Chaining
- CRC Calculation

31.1.1 DMA Operation

A DMA channel transfers data from a source to a destination without CPU intervention. The source and destination start addresses define the start address of the source and destination, respectively.

Both the source and destination have independently configurable sizes and the number of the transferred bytes is independent of the source and destination sizes.

A transfer is initiated either by software or by an interrupt request. The user can select any interrupt on the device to start a DMA transfer.

Upon transfer initiation, the DMA controller will perform a cell transfer and the channel remains enabled until a block transfer is complete. When a channel is disabled, further transfers will be prohibited until the channel is re-enabled.

The DMA channel uses separate pointers to keep track of the current word locations at the source and destination.

Interrupts can be generated when the Source/Destination Pointer is half of the source/destination size, or when the source/destination counter reaches the end of the source/destination.

A DMA transfer can be aborted by the software, by a pattern match or by an interrupt event. The transfer will also stop when an address error is detected.

Figure 31-1: Typical DMA Source to Destination Transfer Diagram

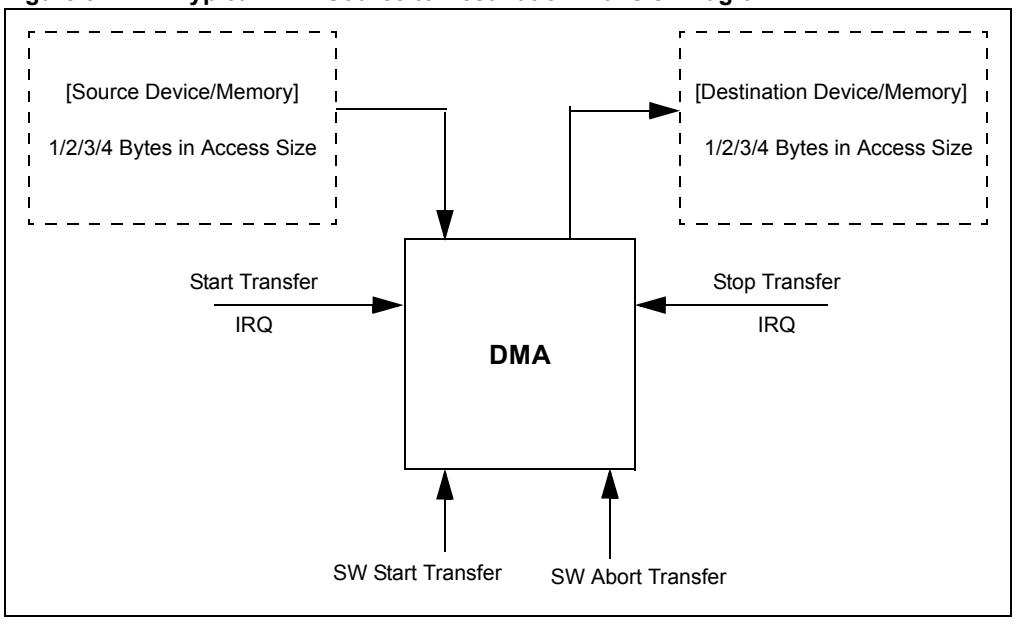


Figure 31-2: DMA Module Block Diagram

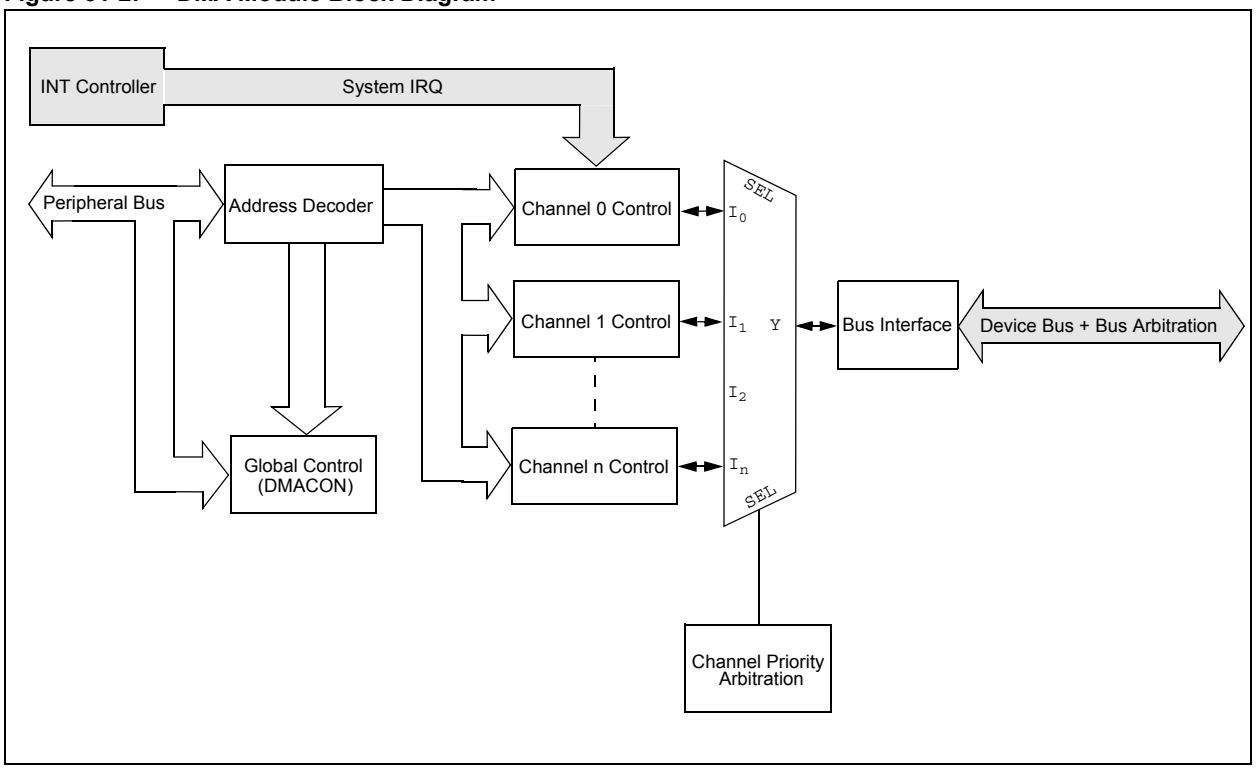
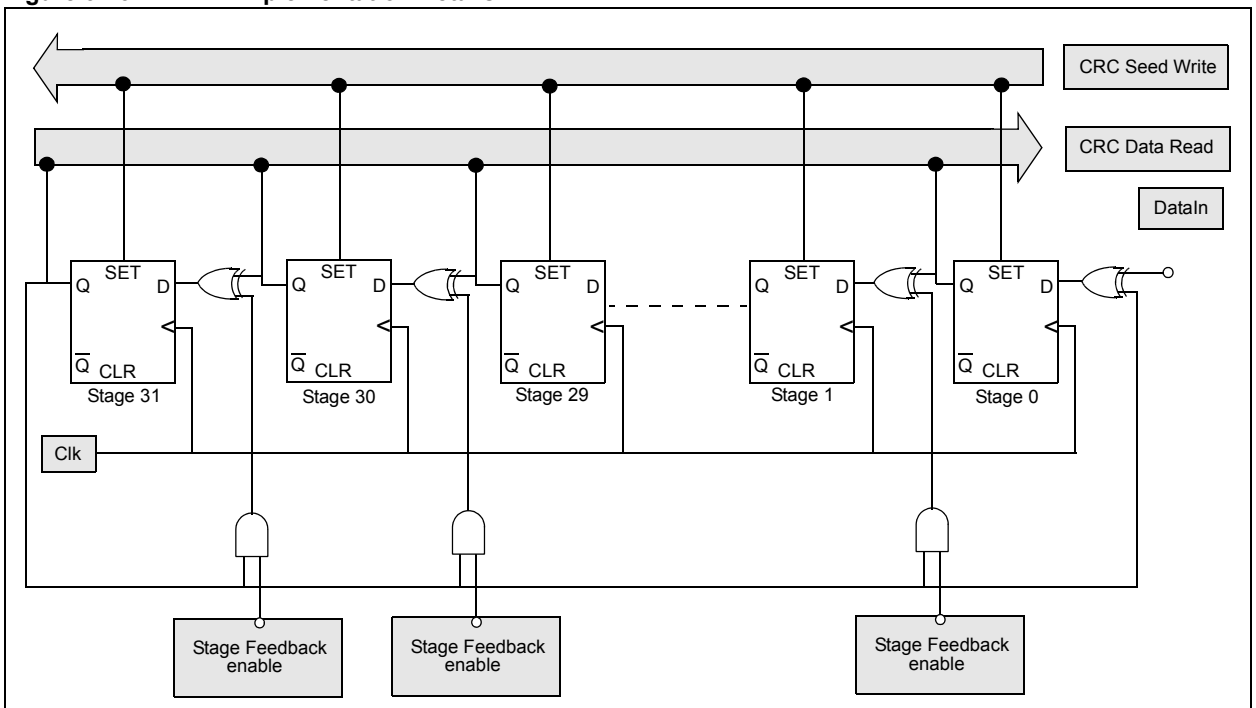


Figure 31-3: CRC Implementation Details



Note: Depending on the device variant, up to 32 stages (0 through 31) are available. Refer to the specific device data sheet for details.

31.2 STATUS AND CONTROL REGISTERS

Note: Each PIC32MX device variant may have one or more DMA channels. An ‘x’ used in the names of control/Status bits and registers denotes the particular channel. Refer to the specific device data sheets for more details.

The DMA module consists of the following Special Function Registers (SFRs):

- DMACON: Control Register for the DMA Controller
- DMASTAT: Status Register for the DMA Module
- DMAADDR: DMA Address Register
- DCRCCON: DMA CRC Control Register
- DCRCDATA: DMA CRC Data Register – The initial value of the CRC generator
- DCRCXOR: DMA CRC XOR Enable Register – Provides a description of the generator polynomial for CRC calculation
- DCHxCON: DMA Channel x Control Register
- DCHxECON: DMA Channel x Event Control Register
- DCHxINT: DMA Channel x Interrupt Control Register
- DCHxSSA: DMA Channel x Source Start Address Register
- DCHxDSA: DMA Channel x Destination Start Address Register
- DCHxSSIZ: DMA Channel x Source Size Register
- DCHxDSIZ: DMA Channel x Destination Size Register
- DCHxSPTR: DMA Channel x Source Pointer Register
- DCHxDPTR: DMA Channel x Destination Pointer Register
- DCHxCSIZ: DMA Channel x Cell-Size Register
- DCHxCPTR: DMA Channel x Cell Pointer Register
- DCHxDAT: DMA Channel x Pattern Data Register

Table 31-1 provides a brief summary of DMA-module-related registers. Corresponding registers appear after the summary, followed by a detailed description of each register.

Table 31-1: DMA Register Summary

Address Offset	Name	Bit Range	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
0x00	DMACON(1,2,3)	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	ON	FRZ	SIDL(4)	SUSPEND	BUSY(4)	—	—	—
		7:0	—	—	—	—	—	—	—	—
0x10	DMASTAT	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	—	—	—	—	—	—	—	—
		7:0	—	—	—	—	RDWR	DMACH<2:0>(5)		
0x20	DMAADDR	31:24	DMAADDR<31:24>							
		23:16	DMAADDR<23:16>							
		15:8	DMAADDR<15:8>							
		7:0	DMAADDR<7:0>							
0x30	DCRCCON(1,2,3)	31:24	—	—	BYTO1(4)	BYTO0(4)	WBO(4)	—	—	BITO(4)
		23:16	—	—	—	—	—	—	—	—
		15:8	—	—	—	PLEN<4:0>(5)				
		7:0	CRCEN	CRCAPP	CRCTYP(4)	—	—	CRCCH<2:0>(5)		
0x40	DCRCDATA(1,2,3)	31:24	DCRCDATA<31:24>(5)							
		23:16	DCRCDATA<23:16>(5)							
		15:8	DCRCDATA<15:8>							
		7:0	DCRCDATA<7:0>							
0x50	DCRCXOR(1,2,3)	31:24	DCRCXOR<31:24>(5)							
		23:16	DCRCXOR<23:16>(5)							
		15:8	DCRCXOR<15:8>							
		7:0	DCRCXOR<7:0>							
0x60	DCHxCON(1,2,3)	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	CHBUSY(4)	—	—	—	—	—	—	CHCHNS
		7:0	CHEN	CHAED	CHCHN	CHAEN	—	CHEDET	CHPRI<1:0>	
0x70	DCHxECON(1,2,3)	31:24	—	—	—	—	—	—	—	—
		23:16	CHAIRQ<7:0>							
		15:8	CHSIRQ<7:0>							
		7:0	CFORCE	CABORT	PATEN	SIRQEN	AIRQEN	—	—	—
0x80	DCHxINT(1,2,3)	31:24	—	—	—	—	—	—	—	—
		23:16	CHSDIE	CHSHIE	CHDDIE	CHDHIE	CHBCIE	CHCCIE	CHTAIE	CHERIE
		15:8	—	—	—	—	—	—	—	—
		7:0	CHSDIF	CHSHIF	CHDDIF	CHDHIF	CHBCIF	CHCCIF	CHTAIF	CHERIF
0x90	DCHxSSA(1,2,3)	31:24	CHSSA<31:24>							
		23:16	CHSSA<23:16>							
		15:8	CHSSA<15:8>							
		7:0	CHSSA<7:0>							

Legend: **Legend:** — = unimplemented, read as '0'. Address offset values are shown in hexadecimal.

Note 1: This register has an associated Clear register at an offset of 0x4 bytes. These registers have the same name with CLR appended to the end of the register name (e.g., DMACONCLR). Writing a '1' to any bit position in the Clear register will clear valid bits in the associated register. Reads from the Clear register should be ignored.

2: This register has an associated Set register at an offset of 0x8 bytes. These registers have the same name with SET appended to the end of the register name (e.g., DMACONSET). Writing a '1' to any bit position in the Set register will set valid bits in the associated register. Reads from the Set register should be ignored.

3: This register has an associated Invert register at an offset of 0xC bytes. These registers have the same name with INV appended to the end of the register name (e.g., DMACONINV). Writing a '1' to any bit position in the Invert register will invert valid bits in the associated register. Reads from the Invert register should be ignored.

4: This bit is not available on all devices. Refer to the specific device data sheet for details.

5: Depending on the device variant, not all bits are available. Refer to the specific device data sheet for details.

Table 31-1: DMA Register Summary (Continued)

Address Offset	Name	Bit Range	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
0xA0	DCHxDSA	31:24	CHDSA<31:24>							
		23:16	CHDSA<23:16>							
		15:8	CHDSA<15:8>							
		7:0	CHDSA<7:0>							
0xB0	DCHxSSIZ(1,2,3)	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	CHSSIZ<15:8>(5)							
		7:0	CHSSIZ<7:0>							
0xC0	DCHxDSIZ(1,2,3)	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	CHDSIZ<15:8>(5)							
		7:0	CHDSIZ<7:0>							
0xD0	DCHxSPTR	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	CHSPTR<15:8>(5)							
		7:0	CHSPTR<7:0>							
0xE0	DCHxDPTR	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	CHDPTR<15:8>(5)							
		7:0	CHDPTR<7:0>							
0xF0	DCHxCSIZ(1,2,3)	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	CHCSIZ<15:8>(5)							
		7:0	CHCSIZ<7:0>							
0x100	DCHxCPTR	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	CHCPTR<15:8>(5)							
		7:0	CHCPTR<7:0>							
0x110	DCHxDAT(1,2,3)	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	—	—	—	—	—	—	—	—
		7:0	CHPDAT<7:0>							

Legend: **Legend:** — = unimplemented, read as '0'. Address offset values are shown in hexadecimal.

Note 1: This register has an associated Clear register at an offset of 0x4 bytes. These registers have the same name with CLR appended to the end of the register name (e.g., DMACONCLR). Writing a '1' to any bit position in the Clear register will clear valid bits in the associated register. Reads from the Clear register should be ignored.

2: This register has an associated Set register at an offset of 0x8 bytes. These registers have the same name with SET appended to the end of the register name (e.g., DMACONSET). Writing a '1' to any bit position in the Set register will set valid bits in the associated register. Reads from the Set register should be ignored.

3: This register has an associated Invert register at an offset of 0xC bytes. These registers have the same name with INV appended to the end of the register name (e.g., DMACONINV). Writing a '1' to any bit position in the Invert register will invert valid bits in the associated register. Reads from the Invert register should be ignored.

4: This bit is not available on all devices. Refer to the specific device data sheet for details.

5: Depending on the device variant, not all bits are available. Refer to the specific device data sheet for details.

Register 31-1: DMACON: DMA Controller Control Register

r-x	r-x	r-x	r-x	r-x	r-x	r-x	r-x
—	—	—	—	—	—	—	—
bit 31							bit 24

r-x	r-x	r-x	r-x	r-x	r-x	r-x	r-x
—	—	—	—	—	—	—	—
bit 23							bit 16

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	r-x	r-x	r-x
ON	FRZ	SIDL(1)	SUSPEND	BUSY(1)	—	—	—
bit 15							bit 8

r-x	r-x	r-x	r-x	r-x	r-x	r-x	r-x
—	—	—	—	—	—	—	—
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 31-16 **Reserved:** Write ‘0’; ignore read

bit 15 **ON:** DMA On bit
1 = DMA module is enabled
0 = DMA module is disabled

Note: When using 1:1 PBCLK divisor, the user’s software should not read/write the peripheral’s SFRs in the SYSCLK cycle immediately following the instruction that clears the module’s ON bit.

bit 14 **FRZ:** DMA Freeze bit
1 = DMA is frozen during Debug mode
0 = DMA continues to run during Debug mode

Note: FRZ is writable in Debug Exception mode only, it is forced to ‘0’ in Normal mode.

bit 13 **SIDL:** Stop in Idle Mode bit(1)
1 = DMA transfers are frozen during Idle
0 = DMA transfers continue during Idle

bit 12 **SUSPEND:** DMA Suspend bit
1 = DMA transfers are suspended to allow CPU uninterrupted access to data bus
0 = DMA operates normally

bit 11 **BUSY:** DMA Module Busy bit(1)
1 = DMA module is active
0 = DMA module is disabled and not actively transferring data

bit 10-0 **Reserved:** Write ‘0’; ignore read

Note 1: This bit is not available on all devices. Refer to the specific device data sheet for details.

Register 31-2: DMASTAT: DMA Status Register(1)

r-x	r-x	r-x	r-x	r-x	r-x	r-x	r-x
—	—	—	—	—	—	—	—
bit 31							bit 24

r-x	r-x	r-x	r-x	r-x	r-x	r-x	r-x
—	—	—	—	—	—	—	—
bit 23							bit 16

r-x	r-x	r-x	r-x	r-x	r-x	r-x	r-x
—	—	—	—	—	—	—	—
bit 15							bit 8

r-x	r-x	r-x	r-x	R-0	R-0	R-0	R-0
—	—	—	—	RDWR	DMACH<2:0>(2)		
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 31-4 **Reserved:** Write ‘0’; ignore read

bit 3 **RDWR:** Read/Write Status bit
1 = Last DMA bus access was a read
0 = Last DMA bus access was a write

bit 2-0 **DMACH<2:0>:** DMA Channel bits(2)

Note 1: This register contains the value of the most recent active DMA channel.

2: Depending on the device variant, not all bits are available. Refer to the specific device data sheet for details.

Register 31-3: DMAADDR: DMA Address Register(1)

R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
DMAADDR<31:24>							
bit 31							bit 24

R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
DMAADDR<23:16>							
bit 23							bit 16

R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
DMAADDR<15:8>							
bit 15							bit 8

R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
DMAADDR<7:0>							
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 31-0 **DMAADDR<31:0>:** DMA Module Address bits

Note 1: This register contains the address of the most recent DMA access.

Register 31-4: DCRCCON: DMA CRC Control Register

r-x	r-x	R/W-0	R/W-0	R/W-0	r-x	r-x	R/W-0
—	—	BYTO<1:0>(1)		WBO(1,2)	—	—	BITO(1)
bit 31							bit 24

r-x	r-x	r-x	r-x	r-x	r-x	r-x	r-x
—	—	—	—	—	—	—	—
bit 23							bit 16

r-x	r-x	r-x	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	PLEN<4:0>(2)				
bit 15							bit 8

R/W-0	R/W-0	R/W-0	r-x	r-x	R/W-0	R/W-0	R/W-0
CRCEN	CRCAPP(2)	CRCTYP(1)	—	—	CRCCH<2:0>(1)		
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 31-30 **Reserved:** Write ‘0’; ignore read

bit 29-28 **BYTO<1:0>:** CRC Byte Order Selection bits(1)
- 11 = Endian byte swap on half-word boundaries (i.e., source half-word order with reverse source byte order per half-word)
- 10 = Swap half-words on word boundaries (i.e., reverse source half-word order with source byte order per half-word)
- 01 = Endian byte swap on word boundaries (i.e., reverse source byte order)
- 00 = No swapping (i.e., source byte order)

bit 27 **WBO:** CRC Write Byte Order Selection bit(1,2)
- 1 = Source data is written to the destination re-ordered as defined by BYTO<1:0>
- 0 = Source data is written to the destination unaltered

bit 26-25 **Reserved:** Write ‘0’; ignore read

bit 24 **BITO:** CRC Bit Order Selection bit(1)

When DCRCCON<CRCTYP> = 1 (CRC module is in IP Header mode):
- 1 = The IP header checksum is calculated Least Significant bit (LSb) first (i.e., reflected)
- 0 = The IP header checksum is calculated Most Significant bit (MSb) first (i.e., not reflected)

When DCRCCON<CRCTYP> = 0 (CRC module is in LFSR mode):
- 1 = The LFSR CRC is calculated Least Significant bit first (i.e., reflected)
- 0 = The LFSR CRC is calculated Most Significant bit first (i.e., not reflected)

bit 23-13 **Reserved:** Write ‘0’; ignore read

Note 1: Depending on the device variant, not all bits are available on all devices. Refer to the specific device data sheet for details.

2: When WBO = 1, unaligned transfers are not supported and the CRCAPP bit cannot be set.

Register 31-4: DCRCCON: DMA CRC Control Register (Continued)

bit 12-8 **PLEN<4:0>:** Polynomial Length bits(2)

When DCRCCON<CRCTYP> = 1 (CRC module is in IP Header mode):

The bits are unused.

When DCRCCON<CRCTYP> = 0 (CRC module is in LFSR mode):

Denotes the length of the polynomial -1.

bit 7 **CRCEN:** CRC Enable bit

1 = CRC module is enabled and channel transfers are routed through the CRC module
0 = CRC module is disabled and channel transfers proceed normally

bit 6 **CRCAPP:** CRC Append Mode bit(2)

1 = The DMA transfers data from the source into the CRC but NOT to the destination. When a block transfer completes the DMA writes the calculated CRC value to the location given by CHxDSA
0 = The DMA transfers data from the source through the CRC obeying WBO as it writes the data to the destination

bit 5 **CRCTYP:** CRC Type Selection bit(1)

1 = The CRC module will calculate an IP header checksum
0 = The CRC module will calculate a LFSR CRC

bit 4-3 **Reserved:** Write '0'; ignore read

bit 2-0 **CRCCH<2:0>:** CRC Channel Select bits(1)

111 = CRC is assigned to Channel 7
110 = CRC is assigned to Channel 6
101 = CRC is assigned to Channel 5
100 = CRC is assigned to Channel 4
011 = CRC is assigned to Channel 3
010 = CRC is assigned to Channel 2
001 = CRC is assigned to Channel 1
000 = CRC is assigned to Channel 0

Note 1: Depending on the device variant, not all bits are available on all devices. Refer to the specific device data sheet for details.

2: When WBO = 1, unaligned transfers are not supported and the CRCAPP bit cannot be set.

Register 31-5: DCRCDATA: DMA CRC Data Register

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
DCRCDATA<31:24>(1)							
bit 31							bit 24

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
DCRCDATA<23:16>(1)							
bit 23							bit 16

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
DCRCDATA<15:8>(1)							
bit 15							bit 8

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
DCRCDATA<7:0>(1)							
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 31-0 **DCRCDATA<31:0>:** CRC Data Register bits(1)

Writing to this register will seed the CRC generator. Reading from this register will return the current value of the CRC. Bits > PLEN will return ‘0’ on any read.

When DRCCON<CRCTYP> = 1 (CRC module is in IP Header mode):
Only the lower 16 bits contain IP header checksum information. The upper 16 bits are always ‘0’. Data written to this register is converted and read back in 1’s complement form (i.e., current IP header checksum value).

When DCRCCON<CRCTYP> = 0 (CRC module is in LFSR mode):
Bits greater than PLEN will return ‘0’ on any read.

Note 1: Depending on the device variant, not all bits are available. Refer to the specific device data sheet for details.

Register 31-6: DCRCXOR: DMA CRCXOR Enable Register

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
DCRCXOR<31:24>(1)							
bit 31							bit 24

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
DCRCXOR<23:16>(1)							
bit 23							bit 16

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
DCRCXOR<15:8>(1)							
bit 15							bit 8

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
DCRCXOR<7:0>(1)							
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 31-0 **DCRCXOR<31:0>:** CRC XOR Register bits(1)

When DRCCON<CRCTYP> = 1 (CRC module is in IP Header mode):
This register is unused.

When DCRCCON<CRCTYP> = 0 (CRC module is in LFSR mode):
1 = Enable the XOR input to the Shift register
0 = Disable the XOR input to the Shift register; data is shifted directly in from the previous stage in the register

Note 1: Depending on the device variant, not all bits are available. Refer to the specific device data sheet for details.

Register 31-7: DCHxCON: DMA Channel x Control Register

r-x	r-x	r-x	r-x	r-x	r-x	r-x	r-x
—	—	—	—	—	—	—	—
bit 31							bit 24

r-x	r-x	r-x	r-x	r-x	r-x	r-x	r-x
—	—	—	—	—	—	—	—
bit 23							bit 16

R/W-0	r-x	r-x	r-x	r-x	r-x	r-x	R/W-0
CHBUSY(1)	—	—	—	—	—	—	CHCHNS(2)
bit 15							bit 8

R/W-0	R/W-0	R/W-0	R/W-0	r-x	R-0	R/W-0	R/W-0
CHEN(3)	CHAED	CHCHN	CHAEN	—	CHEDET	CHPRI<1:0>	
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 31-16 **Reserved:** Write ‘0’; ignore read

bit 15 **CHBUSY:** Channel Busy bit(1)
- 1 = Channel is active or has been enabled
- 0 = Channel is inactive and has been disabled

bit 14-9 **Reserved:** Write ‘0’; ignore read

bit 8 **CHCHNS:** Chain Channel Selection bit(2)
- 1 = Chain to channel lower in natural priority (CH1 will be enabled by CH2 transfer complete)
- 0 = Chain to channel higher in natural priority (CH1 will be enabled by CH0 transfer complete)

bit 7 **CHEN:** Channel Enable bit(3)
- 1 = Channel is enabled
- 0 = Channel is disabled

bit 6 **CHAED:** Channel Allow Events If Disabled bit
- 1 = Channel start/abort events will be registered, even if the channel is disabled
- 0 = Channel start/abort events will be ignored if the channel is disabled

bit 5 **CHCHN:** Channel Chain Enable bit
- 1 = Allow channel to be chained
- 0 = Do not allow channel to be chained

bit 4 **CHAEN:** Channel Automatic Enable bit
- 1 = Channel is continuously enabled, and not automatically disabled after a block transfer is complete
- 0 = Channel is disabled on block transfer complete

bit 3 **Reserved:** Write ‘0’; ignore read

Note 1: This bit is not available on all devices. Refer to the specific device data sheet for details.

2: The chain selection bit takes effect when chaining is enabled (i.e., CHCHN = 1).

3: When the channel is suspended by clearing this bit, the user application should poll the CHBUSY bit (if available on the device variant) to see when the channel is suspended, as it may take some clock cycles to complete a current transaction before the channel is suspended.

Register 31-7: DCHxCON: DMA Channel x Control Register (Continued)

bit 2 **CHEDET:** Channel Event Detected bit
1 = An event has been detected
0 = No events have been detected

bit 1-0 **CHPRI<1:0>:** Channel Priority bits
11 = Channel has priority 3 (highest)
10 = Channel has priority 2
01 = Channel has priority 1
00 = Channel has priority 0

Note 1: This bit is not available on all devices. Refer to the specific device data sheet for details.

2: The chain selection bit takes effect when chaining is enabled (i.e., CHCHN = 1).

3: When the channel is suspended by clearing this bit, the user application should poll the CHBUSY bit (if available on the device variant) to see when the channel is suspended, as it may take some clock cycles to complete a current transaction before the channel is suspended.

Register 31-8: DCHxECON: DMA Channel x Event Control Register

r-x	r-x	r-x	r-x	r-x	r-x	r-x	r-x
—	—	—	—	—	—	—	—
bit 31							bit 24

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
CHAIRQ<7:0>							
bit 23							bit 16

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
CHSIRQ<7:0>							
bit 15							bit 8

S-0	S-0	R/W-0	R/W-0	R/W-0	r-x	r-x	r-x
CFORCE	CABORT	PATEN	SIRQEN	AIRQEN	—	—	—
bit 7							bit 0

Legend:	S = Settable bit		
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 31-24 **Reserved:** Write ‘0’; ignore read

bit 23-16 **CHAIRQ<7:0>:** IRQ that will abort Channel Transfer bits

11111111 = Interrupt 255 will abort any transfers in progress and set CHAIF flag
•
•
•
00000001 = Interrupt 1 will abort any transfers in progress and set CHAIF flag
00000000 = Interrupt 0 will abort any transfers in progress and set CHAIF flag

bit 15-8 **CHSIRQ<7:0>:** IRQ that will Start Channel Transfer bits

11111111 = Interrupt 255 will initiate a DMA transfer
•
•
•
00000001 = Interrupt 1 will initiate a DMA transfer
00000000 = Interrupt 0 will initiate a DMA transfer

bit 7 **CFORCE:** DMA Forced Transfer bit

1 = A DMA transfer is forced to begin when this bit is written to a ‘1’
0 = This bit always reads ‘0’

bit 6 **CABORT:** DMA Abort Transfer bit

1 = A DMA transfer is aborted when this bit is written to a ‘1’
0 = This bit always reads ‘0’

bit 5 **PATEN:** Channel Pattern Match Abort Enable bit

1 = Abort transfer and clear CHEN on pattern match
0 = Pattern match is disabled

bit 4 **SIRQEN:** Channel Start IRQ Enable bit

1 = Start channel cell transfer if an interrupt matching CHSIRQ occurs
0 = Interrupt number CHSIRQ is ignored and does not start a transfer

bit 3 **AIRQEN:** Channel Abort IRQ Enable bit

1 = Channel transfer is aborted if an interrupt matching CHAIRQ occurs
0 = Interrupt number CHAIRQ is ignored and does not terminate a transfer

bit 2-0 **Reserved:** Write ‘0’; ignore read

Register 31-9: DCHxINT: DMA Channel x Interrupt Control Register

r-x	r-x	r-x	r-x	r-x	r-x	r-x	r-x
—	—	—	—	—	—	—	—
bit 31							bit 24

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
CHSDIE	CHSHIE	CHDDIE	CHDHIE	CHBCIE	CHCCIE	CHTAIE	CHERIE
bit 23							bit 16

r-x	r-x	r-x	r-x	r-x	r-x	r-x	r-x
—	—	—	—	—	—	—	—
bit 15							bit 8

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
CHSDIF	CHSHIF	CHDDIF	CHDHIF	CHBCIF	CHCCIF	CHTAIF	CHERIF
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 31-24 **Reserved:** Write‘0’; ignore read

bit 23 **CHSDIE:** Channel Source Done Interrupt Enable bit
1 = Interrupt is enabled
0 = Interrupt is disabled

bit 22 **CHSHIE:** Channel Source Half Empty Interrupt Enable bit
1 = Interrupt is enabled
0 = Interrupt is disabled

bit 21 **CHDDIE:** Channel Destination Done Interrupt Enable bit
1 = Interrupt is enabled
0 = Interrupt is disabled

bit 20 **CHDHIE:** Channel Destination Half Full Interrupt Enable bit
1 = Interrupt is enabled
0 = Interrupt is disabled

bit 19 **CHBCIE:** Channel Block Transfer Complete Interrupt Enable bit
1 = Interrupt is enabled
0 = Interrupt is disabled

bit 18 **CHCCIE:** Channel Cell Transfer Complete Interrupt Enable bit
1 = Interrupt is enabled
0 = Interrupt is disabled

bit 17 **CHTAIE:** Channel Transfer Abort Interrupt Enable bit
1 = Interrupt is enabled
0 = Interrupt is disabled

bit 16 **CHERIE:** Channel Address Error Interrupt Enable bit
1 = Interrupt is enabled
0 = Interrupt is disabled

bit 15-8 **Reserved:** Write ‘0’; ignore read

bit 7 **CHSDIF:** Channel Source Done Interrupt Flag bit
1 = Channel Source Pointer has reached end of source (CHSPTR = CHSSIZ)
0 = No interrupt is pending

Register 31-9: DCHxINT: DMA Channel x Interrupt Control Register (Continued)

bit 6 **CHSHIF:** Channel Source Half Empty Interrupt Flag bit
1 = Channel Source Pointer has reached midpoint of source (CHSPTR = CHSSIZ/2)
0 = No interrupt is pending

bit 5 **CHDDIF:** Channel Destination Done Interrupt Flag bit
1 = Channel Destination Pointer has reached end of destination (CHDPTR = CHDSIZ)
0 = No interrupt is pending

bit 4 **CHDHIF:** Channel Destination Half Full Interrupt Flag bit
1 = Channel Destination Pointer has reached midpoint of destination (CHDPTR = CHDSIZ/2)
0 = No interrupt is pending

bit 3 **CHBCIF:** Channel Block Transfer Complete Interrupt Flag bit
1 = A block transfer has been completed (the larger of CHSSIZ/CHDSIZ bytes has been transferred), or a pattern match event occurs
0 = No interrupt is pending

bit 2 **CHCCIF:** Channel Cell Transfer Complete Interrupt Flag bit
1 = A cell transfer has been completed (CHCSIZ bytes have been transferred)
0 = No interrupt is pending

bit 1 **CHTAIF:** Channel Transfer Abort Interrupt Flag bit
1 = An interrupt matching CHAIRQ has been detected and the DMA transfer has been aborted
0 = No interrupt is pending

bit 0 **CHERIF:** Channel Address Error Interrupt Flag bit
1 = A channel address error has been detected
Either the source or the destination address is invalid.
0 = No interrupt is pending

Register 31-10: DCHxSSA: DMA Channel x Source Start Address Register

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
CHSSA<31:24>							
bit 31							bit 24

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
CHSSA<23:16>							
bit 23							bit 16

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
CHSSA<15:8>							
bit 15							bit 8

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
CHSSA<7:0>							
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 31-0 **CHSSA<31:0>** Channel Source Start Address bits
Channel source start address.

Note: This must be the physical address of the source.

Register 31-11: DCHxDSA: DMA Channel x Destination Start Address Register

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
CHDSA<31:24>							
bit 31							bit 24

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
CHDSA<23:16>							
bit 23							bit 16

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
CHDSA<15:8>							
bit 15							bit 8

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
CHDSA<7:0>							
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 31-0 **CHDSA<31:0>:** Channel Destination Start Address bits
Channel destination start address.

Note: This must be the physical address of the destination.

Register 31-12: DCHxSSIZ: DMA Channel x Source Size Register

r-x	r-x	r-x	r-x	r-x	r-x	r-x	r-x
—	—	—	—	—	—	—	—
bit 31							bit 24

r-x	r-x	r-x	r-x	r-x	r-x	r-x	r-x
—	—	—	—	—	—	—	—
bit 23							bit 16

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
CHSSIZ<15:8>(1)							
bit 15							bit 8

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
CHSSIZ<7:0>(1)							
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 31-16 **Reserved:** Write ‘0’; ignore read

bit 15-0 **CHSSIZ<15:0>:** Channel Source Size bits(1)

65335 =65,535 byte source size

•

•

•

2 = 2 byte source size

1 = 1 byte source size

0 = 65,536 byte source size

Note 1: Depending on the device variant, not all bits are available. Refer to the specific device data sheet for details.

Register 31-13: DCHxDSIZ: DMA Channel x Destination Size Register

r-x	r-x	r-x	r-x	r-x	r-x	r-x	r-x
—	—	—	—	—	—	—	—
bit 31							bit 24

r-x	r-x	r-x	r-x	r-x	r-x	r-x	r-x
—	—	—	—	—	—	—	—
bit 23							bit 16

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
CHDSIZ<15:8>(1)							
bit 15							bit 8

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
CHDSIZ<7:0>(1)							
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 31-16 **Reserved:** Write ‘0’; ignore read

bit 15-0 **CHDSIZ<15:0>:** Channel Destination Size bits(1)

65535 =65,535 byte destination size

•

•

•

2 = 2 byte destination size

1 = 1 byte destination size

0 = 65,536 byte destination size

Note 1: Depending on the device variant, not all bits are available. Refer to the specific device data sheet for details.

Register 31-14: DCHxSPTR: DMA Channel x Source Pointer Register(1)

r-x	r-x	r-x	r-x	r-x	r-x	r-x	r-x
—	—	—	—	—	—	—	—
bit 31							bit 24

r-x	r-x	r-x	r-x	r-x	r-x	r-x	r-x
—	—	—	—	—	—	—	—
bit 23							bit 16

R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
CHSPTR<15:8>(2)							
bit 15							bit 8

R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
CHSPTR<7:0>(2)							
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 31-16 **Reserved:** Write ‘0’; ignore read

bit 15-0 **CHSPTR<15:0>:** Channel Source Pointer bits(2)

65535 =Points to byte 65,535 of the source

•

•

•

1 = Points to byte 1 of the source

0 = Points to byte 0 of the source

Note 1: When in Pattern Detect mode, this register is reset on a pattern detect.

2: Depending on the device variant, not all bits are available. Refer to the specific device data sheet for details.

Register 31-15: DCHxDPTR: DMA Channel x Destination Pointer Register

r-x	r-x	r-x	r-x	r-x	r-x	r-x	r-x
—	—	—	—	—	—	—	—
bit 31							bit 24

r-x	r-x	r-x	r-x	r-x	r-x	r-x	r-x
—	—	—	—	—	—	—	—
bit 23							bit 16

R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
CHDPTR<15:8>(1)							
bit 15							bit 8

R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
CHDPTR<7:0>(1)							
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 31-16 **Reserved:** Write ‘0’; ignore read

bit 15-0 **CHDPTR<15:0>:** Channel Destination Pointer bits(1)

65535 = Points to byte 65,535 of the destination

•

•

•

1 = Points to byte 1 of the destination

0 = Points to byte 0 of the destination

Note 1: Depending on the device variant, not all bits are available. Refer to the specific device data sheet for details.

Register 31-16: DCHxCSIZ: DMA Channel x Cell-Size Register

r-x	r-x	r-x	r-x	r-x	r-x	r-x	r-x
—	—	—	—	—	—	—	—
bit 31							bit 24

r-x	r-x	r-x	r-x	r-x	r-x	r-x	r-x
—	—	—	—	—	—	—	—
bit 23							bit 16

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
CHCSIZ<15:8>(1)							
bit 15							bit 8

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
CHCSIZ<7:0>(1)							
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 31-16 **Reserved:** Write ‘0’; ignore read

bit 15-0 **CHCSIZ<15:0>:** Channel Cell-Size bits(1)

65535 = 65,535 bytes transferred on an event

•

•

•

2 = 2 bytes transferred on an event

1 = 1 byte transferred on an event

0 = 65,536 bytes transferred on an event

Note 1: Depending on the device variant, not all bits are available. Refer to the specific device data sheet for details.

Register 31-17: DCHxCPTR: DMA Channel x Cell Pointer Register(1)

r-x	r-x	r-x	r-x	r-x	r-x	r-x	r-x
—	—	—	—	—	—	—	—
bit 31							bit 24

r-x	r-x	r-x	r-x	r-x	r-x	r-x	r-x
—	—	—	—	—	—	—	—
bit 23							bit 16

R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
CHCPTR<15:8>(2)							
bit 15							bit 8

R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
CHCPTR<7:0>(2)							
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 31-16 **Reserved:** Write ‘0’; ignore read

bit 15-0 **CHCPTR<7:0>:** Channel Cell Progress Pointer bits(2)

65535 = 65,535 bytes have been transferred since the last event

•

•

•

1 = 1 byte has been transferred since the last event

0 = 0 byte have been transferred since the last event

Note 1: When in Pattern Detect mode, this register is reset on a pattern detect.

2: Depending on the device variant, not all bits are available. Refer to the specific device data sheet for details.

Register 31-18: DCHxDAT: DMA Channel x Pattern Data Register

r-x	r-x	r-x	r-x	r-x	r-x	r-x	r-x
—	—	—	—	—	—	—	—
bit 31							bit 24

r-x	r-x	r-x	r-x	r-x	r-x	r-x	r-x
—	—	—	—	—	—	—	—
bit 23							bit 16

r-x	r-x	r-x	r-x	r-x	r-x	r-x	r-x
—	—	—	—	—	—	—	—
bit 15							bit 8

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
CHPDAT<7:0>							
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 31-8 **Reserved:** Write ‘0’; ignore read

bit 7-0 **CHPDAT<7:0>:** Channel Data Register bits

Pattern Terminate mode:
Data to be matched must be stored in this register to allow terminate on match.

All other modes:
Unused.

31.3 MODES OF OPERATION

The DMA module offers the following operating modes:

- Basic Transfer Mode
- Pattern Match Termination Mode
- Channel Chaining Mode
- Channel Auto-Enable Mode
- Special Function Module (SFM) Mode: LFSR CRC, IP Header Checksum

Note that these operation modes are not mutually exclusive but can be simultaneously operational. For example, the DMA controller can perform CRC calculation using chained channels and terminating the transfer upon a pattern match.

31.3.1 DMA Controller Terminology

Event: Any system event that can initiate or abort a DMA transfer.

Transaction: A single word transfer (up to 4 bytes), comprised of read and write operations.

Cell Transfer: The number of bytes transferred when a DMA channel has a transfer initiated before waiting for another event (given by the DCHxCSIZ register). A cell transfer is comprised of one or more transactions.

Block Transfer: Defined as the number of bytes transferred when a channel is enabled. The number of bytes is the larger of either DCHxSSIZ or DCHxDSIZ. A block transfer is comprised of one or more cell transfers.

31.3.2 Basic Transfer Mode Operation

A DMA channel will transfer data from a source register to a destination register without CPU intervention. The Channel Source Start Address register (DCHxSSA) defines the physical start address of the source. The Channel Destination Start Address register (DCHxDSA) defines the physical start address of the destination. Both the source and destination are independently configurable using the DCHxSSIZ and DCHxDSIZ registers.

A cell transfer is initiated in one of two ways:

- Software can initiate a transfer by setting the channel CFORCE (DCHxECON<7>) bit
- Interrupt event occurs on the device that matches the CHSIRQ interrupt and SIRQEN = 1 (DCHxECON<4>). The user can select any interrupt on the device to start a DMA transfer

A DMA transfer will transfer DCHxCSIZ (cell transfer) bytes when a transfer is initiated (an event occurs). The channel remains enabled until the DMA channel has transferred the larger of DCHxSSIZ and DCHxDSIZ (i.e., block transfer is complete). If DCHxCSIZ is greater than the larger of DCHxSSIZ and DCHxDSIZ, then the larger of DCHxSSIZ and DCHxDSIZ bytes will be transferred. When the channel is disabled, further transfers will be prohibited until the channel is re-enabled (CHEN is set to '1').

Each channel keeps track of the number of words transferred from the source and destination using the pointers DCHxSPTR and DCHxDPTR. Interrupts are generated when the source or Destination Pointer is half of the size (DCHxSSIZ/2 or DCHxDSIZ/2), or when the source or destination counter reaches the end. These interrupts are CHSHIF (DCHxINT<6>), CHDHIF (DCHxINT<4>), CHSDIF (DCHxINT<7>) or CHDDIF (DCHxINT<5>), respectively.

A DMA transfer request can be reset by the following:

- Writing the CABORT bit (DCHxECON<6>)
- Pattern match occurs if pattern match is enabled as described in **Section 31.3.3 "Pattern Match Termination Mode Operation"**, provided that Channel Auto-Enable mode bit CHAEN (DCHxCON<4>), is not set
- Interrupt event occurs on the device that matches the CHAIRQ <7:0> bits (DCHxECON<23:16>) interrupt if enabled by AIRQEN (DCHxECON<3>)
- Detection of an address error
- Completion of a cell transfer
- A block transfer completes provided that Channel Auto-Enable mode (CHAEN) is not set

When a channel abort interrupt occurs, the Channel Transfer Abort Interrupt Flag CHTAIF (DCHxINT<1>) bit is set. This allows the user to detect and recover from an aborted DMA transfer. When a transfer is aborted, any transaction currently underway will be completed.

The Source and Destination Pointers are updated as a transfer progresses. These pointers are read-only. The pointers are reset under the following conditions:

- If the channel source address (DCHxSSA) is updated, the Source Pointer (DCHxSPTR) will be reset.
- Similar updates to the destination address (DCHxDSA) will cause the Destination Pointer (DCHxDPTR) to be reset.
- A channel transfer is aborted by writing the CABORT (DCHxECON<6>) bit.

Note:	Refer to Table 31-4 for detailed information about the channel event behavior.

Example 31-1: DMA Channel Initialization for Basic Transfer Mode Code Example

```
/*
The following code example illustrates the DMA channel 0 configuration for a data transfer.
*/
       IEC1CLR=0x00010000;           // disable DMA channel 0 interrupts
       IFS1CLR=0x00010000;           // clear existing DMA channel 0 interrupt flag

       DMACONSET=0x00008000;         // enable the DMA controller
       DCH0CON=0x3;                  // channel off, pri 3, no chaining

       CH0ECON=0;                    // no start or stop irq's, no pattern match

                                     // program the transfer
       DCH0SSA=0x1d010000;           // transfer source physical address
       DCH0DSA=0x1d020000;           // transfer destination physical address
       DCH0SSIZ=200;                 // source size 200 bytes
       DCH0DSIZ=200;                 // destination size 200 bytes
       DCH0CSIZ=200;                 // 200 bytes transferred per event

       DCH0INTCLR=0x00ff00ff;        // clear existing events, disable all interrupts
       DCH0CONSET=0x80;              // turn channel on

                                     // initiate a transfer
       DCH0ECONSET=0x00000080;       // set CFORCE to 1

                                     // do something else

                                     // poll to see that the transfer was done

       while(TRUE)
       {
              register int pollCnt;  // use a poll counter.
                                     // continuously polling the DMA controller in a tight
                                     // loop would affect the performance of the DMA transfer

              int dmaFlags=DCH0INT;
              if( (dmaFlags&0xb)
              {                      // one of CHERIF (DCHxINT<0>), CHTAIF (DCHxINT<1>)
                                     // or CHBCIF (DCHxINT<3>) flags set
                     break;          // transfer completed
              }
              pollCnt=100;           // use an adjusted value here
              while(pollCnt--);      // wait before reading again the DMA controller
       }

                                     // check the transfer completion result
```

31.3.2.1 Interrupt and Pointer Updates

The Source and Destination Pointers are updated after every transaction. Interrupts will also be set or cleared at this time. If a pointer passes the halfway point during a transaction, the interrupt will be updated accordingly.

Pointers are reset when any of the following occurs:

- On any device Reset
- When the DMA is turned off (ON bit (DMACON<15>) is '0'
- A block transfer completes, regardless of the state of CHAEN (DCHxCON<4>)
- A pattern match terminates a transfer, regardless of the state of CHAEN (DCHxCON<4>)
- The CABORT (DCHxECON<6>) flag is written
- Source or destination start addresses are updated

31.3.3 Pattern Match Termination Mode Operation

Pattern Match Termination mode allows the user to end a transfer if a byte of data written during a transaction matches a specific pattern, as defined by the DCHxDAT register. A pattern match is treated the same way as a block transfer complete, where the CHBCIF bit (DCHxINT<3>) is set and the CHEN bit (DCHxCON<7>) is cleared.

This feature is useful in applications where a variable data size is required and eases the set up of the DMA channel. UART is a good example of where this can be effectively used.

Assuming a system has a series of messages that are routinely transmitted to an external host and it has a maximum message size of 86 characters, the user would set the following parameters on the channel:

- DCHxSSIZ to 87 bytes:

  If something unexpected occurs the CPU program will be interrupted when the buffer overflows and can take the appropriate action.
- DCHxDSIZ set to 1-byte.
- The destination address is set to the UART TXREG.
- The DCHxDAT is set to 0x00, which will stop the transfer on a NULL character in any byte lane.
- The CHSIRQ (DCHxECON<15:8>) is set to the UART “transmit buffer empty” IRQ.
- The SIRQEN (DCHxECON<4>) is set to enable the channel to respond to the start interrupt event.
- The start address is set to the start address of the message to be transferred.
- The channel is enabled, CHEN = 1 (DCHxCON<7>).
- The user will then force a cell transfer through CFORCE (DCHxECON<7>) and the first byte transmission by the UART.
- Each time a byte is transmitted by the UART, the transmit buffer empty interrupt will initiate the following byte transfer from the source to the UART.
- When the DMA channel detects a NULL character in any of the byte lanes of the channel, the transaction will be completed and the channel disabled.

Pattern matching is independent of the byte lane of the source data. If ANY byte in the source buffer matches DCHxDAT, a pattern match is detected. The transaction will be completed and the data read from the source will be written to the destination.

Example 31-2: DMA Channel Initialization in Pattern Match Transfer Mode Code Example

```
/*
The following code example illustrates the DMA channel 0 configuration for data transfer with
pattern match enabled. Transfer from the UART1 a <CR> ended string, at most 200 characters long
*/

       IEC1CLR=0x00010000;           // disable DMA channel 0 interrupts
       IFS1CLR=0x00010000;           // clear any existing DMA channel 0 interrupt flag

       DMACONSET=0x00008000;         // enable the DMA controller
       DCH0CON=0x03;                 // channel off, priority 3, no chaining

       DCH0ECON=(27 <<8)| 0x30;      // start irq is UART1 RX, pattern match enabled
       DCH0DAT='\r';                 // pattern value, carriage return

                                     // program the transfer
       DCH0SSA=VirtToPhys(&U1RXREG); // transfer source physical address
       DCH0DSA=0x1d020000;           // transfer destination physical address
       DCH0SSIZ=1;                   // source size is 1 byte
       DCH0DSIZ=200;                 // destination size at most 200 bytes
       DCH0CSIZ=1;                   // one byte per UART transfer request

       DCH0INTCLR=0x00ff00ff;        // clear existing events, disable all interrupts
       DCH0INTSET=0x00090000;        // enable Block Complete and error interrupts

       IPC9CLR=0x0000001f;           // clear the DMA channel 0 priority and sub-priority
       IPC9SET=0x00000016;           // set IPL 5, sub-priority 2
       IEC1SET=0x00010000;           // enable DMA channel 0 interrupt

       DCH0CONSET=0x80;              // turn channel on


                                     // wait for the UART1 RX interrupt to initiate a
                                     // transfer

                                     // do something else

                                     // will get an interrupt when the transfer is done
                                     // or when an address error occurred
```

31.3.4 Channel Chaining Mode Operation

Channel chaining is an enhancement to the DMA channel operation. A channel (slave channel) can be chained to an adjacent channel (master channel). The slave channel will be enabled when a block transfer of the master channel completes, i.e., CHBCIF (DCHxINT<3>), is set.

At this point, any event on the slave channel will initiate a cell transfer. If the channel has an event pending, a cell transfer will begin immediately.

The master channel will set its interrupt flags normally, CHBCIF (CHxINT<3>) and has no knowledge of the “chain” status of the slave channel. The master channel is still able to cause interrupts at the end of a DMA transfer if one of the CHSDIE/CHDDIE/CHBCIE (DCHxINT<23/21/19>) bits is set.

In the channels natural priority order, channel 0 has the highest priority and channel 7 the lowest. The channel higher or lower in natural priority, that can enable a specific channel, is selected by CHCHNS (DCHxCON<8>), provided that channel chaining is enabled, CHCHN = 1 (DCHxCON<5>).

Note:	In some devices, channel 0 has the highest priority and channel 4 the lowest. Refer to the specific device data sheet for availability.

A feature of the DMA module is the ability to allow events while the channel is disabled using CHAED (DCHxCON<6>). This bit is particularly useful in Chained mode, in which the slave channel needs to be ready to start a transfer as soon as the channel is enabled by the master channel.

The following examples demonstrate situations in which chaining may be useful:

1. Transferring data in one peripheral (e.g., from UART1, DMA channel 0, at 9600 baud, to SRAM) to another peripheral (e.g., from SRAM to UART2, DMA channel 1, at 19200 baud).

   In this example, CHAED will be set in both channels; with UART2 setting the event detect, CHEDET (DCHxCON<2>), on channel 1 when the last byte has been transmitted. As soon as channel 0 completes a transfer, channel 1 is enabled and the data is transferred immediately.

2. A/D converter transfers data to one buffer (connected to channel 0).

   When the destination buffer 0 is full (block transfer completes), channel 1 is enabled and further conversions are transferred to buffer 1. In this case, CHAED will not be enabled. If it were, the last word transferred by channel 0 would be transferred a second time by channel 1 (because the A/D converter interrupt event would have set the event detect flag CHEDET in both channels).

Example 31-3: DMA Channel Initialization in Chaining Mode Code Example

```
/*
The following code example illustrates the DMA channel 0 configuration for data transfer with
pattern match enabled. DMA channel 0 transfer from the UART1 to a RAM buffer while DMA channel 1
transfers data from the RAM buffer to UART2. Transferred strings are at most 200 characters
long. Transfer on UART2 will start as soon as the UART1 transfer is completed.
*/

  unsigned char myBuff<200>;// transfer buffer

 IEC1CLR=0x00010000;         // disable DMA channel 0 interrupts
 IFS1CLR=0x00010000;         // clear any existing DMA channel 0 interrupt flag

 DMACONSET=0x00008000;       // enable the DMA controller

 DCH0CON=0x3;                // channel 0 off, priority 3, no chaining
 DCH1CON=0x62;               // channel 1 off, priority 2
                             // chain to higher priority
                             // (channel 0), enable events detection while disabled

 DCH0ECON=(27 <<8)| 0x30;    // start irq is UART1 RX, pattern enabled
 DCH1ECON=(42 <<8)| 0x30;    // start irq is UART1 TX, pattern enabled

 DCH0DAT=DCH1DAT='\r';       // pattern value, carriage return

                             // program channel 0 transfer
 DCH0SSA=VirtToPhys(&U1RXREG); // transfer source physical address
 DCH0DSA=VirtToPhys(myBuff); // transfer destination physical address
 DCH0SSIZ=1;                 // source size is 1 byte
 DCH0DSIZ=200;               // dst size at most 200 bytes
 DCH0CSIZ=1;                 // one byte per UART transfer request

                             // program channel 1 transfer
 DCH1SSA=VirtToPhys(myBuff); // transfer source physical address
 DCH1DSA=VirtToPhys(&U2TXREG); // transfer destination physical address
 DCH1SSIZ=200;               // source size at most 200 bytes
 DCH1DSIZ=0;                 // dst size is 1 byte
 DCH1CSIZ=1;                 // one byte per UART transfer request

 DCH0INTCLR=0x00ff00ff;      // DMA0: clear events, disable interrupts
 DCH1INTCLR=0x00ff00ff;      // DMA1: clear events, disable interrupts
 DCH1INTSET=0x00090000;      // DMA1: enable Block Complete and error interrupts


 IPC9CLR=0x00001f1f;         // clear the DMA channels 0 and 1 priority and
                             // sub-priority
 IPC9SET=0x00000b16;         // set IPL 5, sub-priority 2 for DMA channel 0
                             // set IPL 2, sub-priority 3 for DMA channel 1
 IEC1SET=0x00020000;         // enable DMA channel 1 interrupt

 DCH0CONSET=0x80;            // turn channel on

                             // do something else

                             // the UART1 RX interrupts will initiate the DMA channel 0 transfer
                             // once this transfer is complete, the DMA channel 1 will start
                             // upon DMA channel 1 transfer completion will get an interrupt

 while(!intCh1Ocurred);      // poll DMA channel 1 interrupt
```

31.3.5 Channel Auto-Enable Mode Operation

The channel auto-enable can be used to keep a channel active, even if a block transfer completes or pattern match occurs. This prevents the user from having to re-enable the channel each time a block transfer completes. To use this mode the user will configure the channel, setting the CHAEN (DCHxCON<4>) bit before enabling the channel, i.e., setting the CHEN bit (DCHxCON<7>). The channel will behave as normal except that normal termination of a transfer will not result in the channel being disabled.

Normal block transfer completion is defined as:

- Block Transfer Complete
- Pattern Match Detect

As before, the Channel Pointers will be reset. This mode is useful for applications that do repeated pattern matching.

Note:	CHAEN prevents the channel from being automatically disabled once it has been enabled. The channel will still have to be enabled by the software.

31.3.6 Suspending Transfers

The user application can immediately suspend the DMA module by writing the SUSPEND bit (DMACON<12>). This will immediately suspend the DMA controller from any further bus transactions.

Depending on the device variant, when the DMA module is suspended by setting the SUSPEND bit, the user application should poll the BUSY (DMACON<11>) bit to determine when the module is completely suspended following the completion of the current transaction.

Note:	The BUSY bit is not available on all device. Refer to the specific device data sheet for availability.

Individual channels may be suspended using the CHEN (DCHxCON<7>) bit. If a DMA transfer is in progress and the CHEN bit is cleared, the current transaction will be completed and further transactions on the channel will be suspended.

Depending on the device variant, when the channel is suspended by clearing the CHEN bit, the user application should poll the CHBUSY (DCHxCON<15>) bit to determine when the channel is completely suspended following completion of the current transaction.

Clearing the enable bit (CHEN) will not affect the Channel Pointers or the transaction counters. While a channel is suspended, the user can elect to continue to receive events (abort interrupts, etc.) by setting CHAED (DCHxCON<6>).

31.3.7 Resetting the Channel

The channel logic will be reset on any device Reset. The channel is also reset when the channel flag bit CABORT (DCHxECON<6>) is written. This will turn off channel flag bit CHEN = 0, clear the Source and Destination Pointers, and reset the event detector. When the CABORT bit is set, the current transaction in progress (if any) will complete before the channel is reset, but any remaining transactions will be aborted.

The user should modify the channel registers only while the channel is disabled (CHEN = 0). Modifying the Source and Destination registers will reset the corresponding pointer registers (DCHxSPTR or DCHxDPTR).

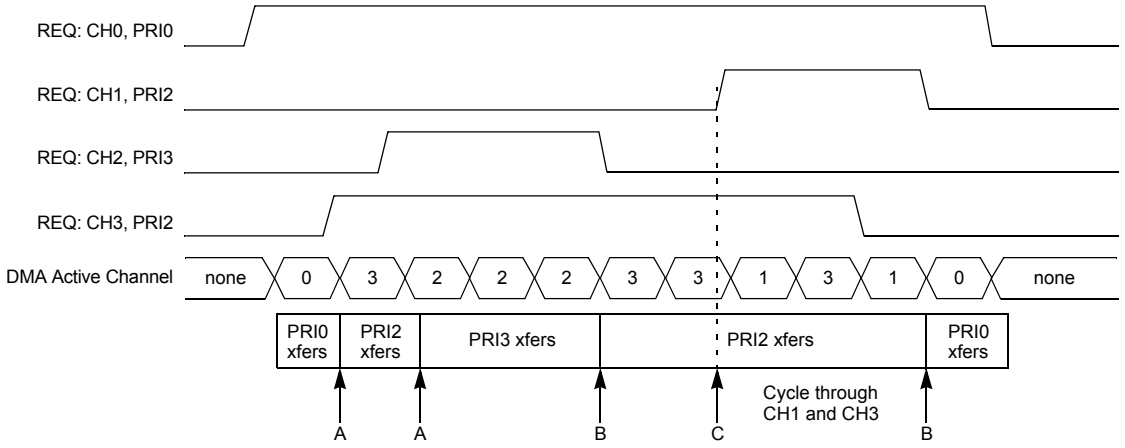
Note:	The channel size must be changed while the channel is disabled.

31.3.8 Channel Priority and Selection

The DMA controller has a natural priority associated with each of the channels. Channel 0 has the highest natural priority. Each channel has two priority bits, CHPRI<1:0> (DCHxCON<1:0>). These bits identify the channel's priority. When multiple channels have transfers pending, the next channel to transmit data will be selected as follows:

- Channels with the highest priority will complete all cell transfers before moving onto channels with a lower priority (see behavior, "PRI3 xfers", in Figure 31-4).
- If multiple channels have the same priority (identical CHPRI), the controller will cycle through all channels at that priority. Each channel with a cell transfer in progress at the highest priority will be allowed a single transaction of the active cell transfer before the controller allows a single transaction by the next channel at that priority level (see behavior, "PRI2 xfers" between markers "C" and "B", in Figure 31-4).
- If a channel with a higher priority requests a transfer while another channel of lower priority has a transaction in process, the transaction will complete before moving to the channel with the higher priority (see events at markers "A" in Figure 31-4).

Figure 31-4: Channel Priority Behavior



Transition Legend:

A – Higher priority transfer request; suspend current and transfer next.

B – All highest priority transfers complete; drop to channels at lower priority.

C – Cycle through all channels at the current priority.

31.3.9 Byte Alignment

The byte alignment feature of the DMA controller relieves the user from aligning the source and destination addresses.The read portion of a transaction will read the maximum number of bytes that are available to be read in a given word. For example, if the Source Pointer is N > 4 bytes from the source size, 4 bytes will be read if the Source Pointer points to byte 0, 3 bytes if the Source Pointer points to byte 1, etc. If the number of bytes remaining in the source is N < 4, only the first N bytes are read. This is important when the read includes registers that are updated on a read.

The Source Pointer and Destination Pointers are updated after every write, with the number of bytes that have been written. The user should note that in cases where a transfer is aborted, before a transaction is complete, the Source Pointer will not necessarily reflect the reads that have taken place.

An example of this behavior is given in Table 31-2. Example 1 demonstrates a simple transfer of 9 bytes between two large buffers, in which CHxSSA = 0x1000, CHxSSIZ = 100, CHxDSA = 0x43F9, CHxDSIZ = 100 and CHxCSIZ = 9.

Table 31-2: Source and Destination Pointer Updates – Example 1

Transaction	Operation	Source Pointer	Destination Pointer	Transfer Count/Size	Read Address	Write Address	Read Data(1)	Write Data(2)
1	Read	9	11	0/9	1009	xxxx	33_22_11_XX	XX_XX_XX_XX
1	Write1	9	11	0/9	1009	440A	33_22_11_XX	22_11_XX_XX
1	Ptr Update(3)	B	13	2/9	1009	440A	33_22_11_XX	XX_XX_XX_XX
1	Write2	B	13	2/9	1009	440C	33_22_11_XX	XX_XX_XX_33
1	Ptr Update(3)	C	14	3/9	1009	440C	33_22_11_XX	XX_XX_XX_XX
2	Read	C	14	3/9	100C	440C	77_66_55_44	XX_XX_XX_XX
2	Write1	C	14	3/9	100C	440D	77_66_55_44	66_55_44_XX
2	Ptr Update(3)	F	17	6/9	100C	440D	77_66_55_44	XX_XX_XX_XX
2	Write2	F	17	6/9	100C	4410	77_66_55_44	XX_XX_XX_77
2	Ptr Update(3)	10	18	7/9	100C	4410	77_66_55_44	XX_XX_XX_XX
3	Read	10	18	7/9	1010	4410	XX_XX_99_88	XX_XX_XX_XX
3	Write1	10	18	7/9	1010	4411	XX_XX_XX_88	XX_99_88_XX
3	Ptr Update(3)	12	1A	9/9	1010	4411	XX_XX_XX_88	XX_XX_XX_XX

Note 1: XX indicates that data read is discarded.
2: XX indicates that data that is NOT written.
3: Interrupts are updated when the pointers are updated as required.

Another example of this behavior is given in Table 31-3. Example 2 demonstrates worst-case bus utilization, i.e., unaligned buffers with destination buffer wrapping, in which CHxSSA = 0x1000, CHxSSIZ = 100, CHxDSA = 0x4402, CHxDSIZ = 4 and CHxCSIZ = 8.

Table 31-3: Source and Destination Pointer Updates – Example 2

Transaction	Operation	Source Pointer	Destination Pointer	Transfer Count/Size	Read Address	Write Address	Read Data(1)	Write Data(2)
1	Read	9	0	0/8	1009	xxxx	33_22_11_XX	XX_XX_XX_XX
1	Write1	9	0	0/8	1009	4402	33_22_11_XX	22_11_XX_XX
1	Ptr Update(3)	B	2	2/8	1009	4402	33_22_11_XX	XX_XX_XX_XX
1	Write2	B	2	2/8	1009	4404	33_22_11_XX	XX_XX_XX_33
1	Ptr Update(3)	C	3	3/8	1009	4404	33_22_11_XX	XX_XX_XX_XX
2	Read	C	3	3/8	100C	4404	77_66_55_44	XX_XX_XX_XX
2	Write1	C	3	3/8	100C	4405	77_66_55_44	XX_XX_44_XX
2	Ptr Update(3)	D	0	4/8	100C	4405	77_66_55_44	XX_XX_XX_XX
2	Write2	D	0	4/8	100C	4402	77_66_55_44	66_55_XX_XX
2	Ptr Update(3)	F	2	6/8	100C	4402	77_66_55_44	XX_XX_XX_XX
3	Write3	F	2	6/8	100F	4404	77_66_55_44	XX_XX_XX_77
3	Ptr Update(3)	10	3	7/8	100F	4404	77_66_55_44	XX_XX_XX_XX
3	Read	10	18	7/8	1010	4404	BB_AA_99_88	XX_XX_XX_XX
3	Write1	10	18	7/8	1010	4405	BB_AA_99_88	XX_XX_88_XX
3	Ptr Update(3)	11	1A	8/8	1010	4405	77_66_55_44	XX_XX_XX_XX

Note 1: XX indicates that data read is discarded.

2: XX indicates that data that is NOT written.

3: Interrupts are updated when the pointers are updated as required.

31.3.10 Channel Transfer Behavior

Once a channel has been enabled, CHEN = 1 (DCHxCON<7>), any event that starts a cell transfer will transfer the CHCSIZ (DCHxCSIZ) bytes of data. This will require one or more transactions. Once the cell transfer is complete the channel will return to an inactive state, and will wait for another channel start event to occur before starting another cell transfer.

When the larger of CHSSIZ (DCHxSSIZ) or CHDSIZ (DCHxDSIZ) bytes are transferred, a block transfer completes, the channel transfer will be halted and the channel will be disabled (i.e., CHEN set to '0' by hardware, and pointers are reset).

31.3.11 Channel Enable

Each channel has an enable bit CHEN, which can be used to enable or disable the channel in question. When this bit is set, the channel transfer requests are serviced by the DMA controller.

When CHEN is clear, the state of the channel is preserved (this allows the channel to be suspended once a transfer has begun).

CHEN will be cleared by hardware under the following conditions:

- A block transfer is complete, the pointer to the larger of the source or destination matches the size (only if CHAEN (DCHxCON<4>) is clear).
- A pattern match occurs in Pattern Match mode (only if CHAEN is clear).
- An abort interrupt occurs.
- The user writes the CABORT (DCHxECON<6>) flag.

31.3.12 Channel IRQ Detection

The DMA Controller maintains its own flags for detecting the start and abort IRQ in the system and is completely independent of the INT Controller and IES/IFS flags. The corresponding IRQ does not have to be enabled before a transfer can take place, nor cleared at the end of a DMA transfer.

Once the start or abort IRQ system events are triggered, they will be detected automatically by the DMA controller internal logic, without the need for user intervention.

31.3.13 Channel Event Transfer Initiation

A given channel transfer can be initiated by:

- Writing the CFORCE bit (DCHxECON<7>).
- An interrupt occurs that matches the value of CHSIRQ<7:0> (DCHxECON<15:8>) if it is enabled by SIRQEN (DCHxECON<4>).

Channel events are registered if the channel is enabled (CHEN = 1), or if “Allow Event If Disabled” is set, i.e., CHAED = 1 (DCHxCON<6>).

31.3.14 Channel Event Transfer Termination

Channel transfer is terminated in any of the following cases:

- A transfer is aborted as described in **Section 31.3.16 “Channel Abort”**.
- A cell transfer (CHCSIZ bytes (DCHxCSIZ transferred)) completes.
- The DMA has transferred the larger of CHSSIZ or CHDSIZ bytes (block transfer complete), the channel is disabled in hardware and must be re-enabled by user software before the channel will respond to channel events.
- A pattern match occurs if enabled.
- An abort interrupt, CHAIRQ<7:0> (DCHxECON<23:16>), occurs if abort interrupts are enabled by AIRQEN (DCHxECON<3>).
- An address error occurs.

An example of how to use the abort interrupt would be a transfer from a UART channel to the memory. While the UART Receive Data Available interrupt can be used to start the transfer, the UART Error interrupt can abort the transfer. This way, whenever an error occurs on the communication channel (a framing/parity error or even an overrun), the transfer is stopped and the user code gets control in an ISR (if the abort interrupt is enabled for the DMA controller).

A summary of the status flags affected by channel transfer initiation or termination is provided in Table 31-4. Channel abort events are allowed if the channel is enabled, CHEN = 1, or if the user elects to allow events while the channel is disabled, CHAED = 1.

Table 31-4: Channel Event Behavior

Event	Description and Function	Registers Affected
Events Initiating Transfers		
System Interrupt Matching CHSIRQ<7:0>(1,2)	The channel event detect will be set.	CHEDET = 1
Channel Chain Event	This will enable the channel if not already set. If an event detect is pending, a channel transfer will begin immediately.	CHEN = 1
User Writes the CFORCE Bit(1)	The channel event detect will be set.	CHEDET = 1
Events Terminating Transfers		
System Interrupt Matching CHAIRQ<7:0>(1,2)	The channel event detect will be reset and the channel turned off. The abort interrupt flag is set.	CHEDET = 0 CHEN = 0 CHAIF = 1
Pattern Match(1)	This occurs when any byte of data written in a transaction matches the data in CHPDAT. The channel event detect is reset. The channel is turned off if CHAEN = 0. This event is treated as a completed block transfer. Pointers are reset.	CHEDET = 0 CHEN = 0 CHBCIF = 1 CHSPTR = 0 CHDPTR = 0 CHCPTR = 0
Cell Transfer is Complete	This occurs when CHCSIZ bytes have been transferred. The transfer event detect is reset and the channel remains enabled pending the next event.	CHEDET = 0 CHCCIF = 1
Block Transfer is Complete	The channel event detect is reset. The channel is turned off if CHAEN = 0. This event is treated as a completed transfer. Pointers are reset.	CHEDET = 0 CHEN = 0 CHBCIF = 1 CHSPTR = 0 CHDPTR = 0 CHCPTR = 0
User Writes the CABORT bit	The channel is turned off and the channel event detect is reset. The pointers are reset.	CHEDET = 0 CHEN = 0 CHSPTR = 0 CHDPTR = 0 CHCPTR = 0
Address Error is Detected	The channel is turned off and the event detect is reset. The address error interrupt flag is set.	CHEDET = 0 CHEN = 0 CHERIF = 1

Note 1: Events are allowed only when the channel is enabled, or the user allows events while disabled (CHEN = 1 or CHAED = 1).

2: The DMA Controller maintains its own flags for detecting start and abort interrupt requests (IRQs) in the system, and is completely independent of the INT Controller IES/IFS flags. Once the start or abort IRQ system events are triggered, they will be detected automatically by the DMA controller internal logic, without the need for user intervention.

31.3.15 Channel Abort Interrupt

A channel can elect to abort a cell transfer if an interrupt event occurs. The interrupt is selected by the channel's abort IRQ, CHAIRQ<7:0> (DCHxECON<23:16>). Any one of the device interrupt events can cause a channel abort. An abort only occurs if enabled by AIRQEN (DCHxECON<3>).

If this occurs (often a timer time-out or a module error flag), the channel's status flags will indicate the external abort event on the channel in question by setting its CHTAIF bit (DCHxINT<1>). The Source and Destination Pointers are not reset, allowing the user to recover from the error.

31.3.16 Channel Abort

A channel transfer can be aborted by the user by writing the CABORT bit (DCHxECON<6>). When a transfer is aborted, the current bus transaction will be completed and any transactions that remain will be aborted. The CHEN (DCHxCON<7>) bit will be cleared. When the user writes the CABORT bit, the Source and Destination Pointers are reset.

31.3.17 Address Error

If the address (either source or destination) occurring during a transfer is an illegal address, the channel's address error interrupt flag CHERIF (DCHxINT<0>) will be set. The channel will be disabled, i.e., CHEN will be reset by hardware.

The channel status is unaffected to aid in the debug of the problem.

31.3.18 DMA Suspend

DMA transactions are suspended immediately if the SUSPEND bit (DMACON<12>) is set. The current read or write will be completed. If the suspend comes during the read portion of the transaction, the transaction will be suspended and the write will be put on hold. If the suspend comes during the write portion of the transaction, the write will complete and the pointers updated as normal. Any transactions that were in process will continue where they left off when the SUSPEND bit is cleared.

Depending on the device variant, when the DMA module is suspended by setting the SUSPEND bit, the user application should poll the BUSY (DMACON<11>) bit to determine when the module is completely suspended following the completion of the current transaction.

Note:	The BUSY bit is not available on all device. Refer to the specific device data sheet for availability.

Example 31-4: DMA Controller Suspension

```
/*
The following code example will suspend the DMA Controller.
*/
 DMACONSET=0x00001000;       // suspend the DMA controller

 while(!(DMACONbits.busy)); // wait for the transfer to be actually suspended

                            // let the CPU have complete control of the bus

 DMACONCLR=0x00001000;       // clear the suspend mode and let the DMA operate normally

                            // from now on, the CPU and DMA controller share the bus access
```

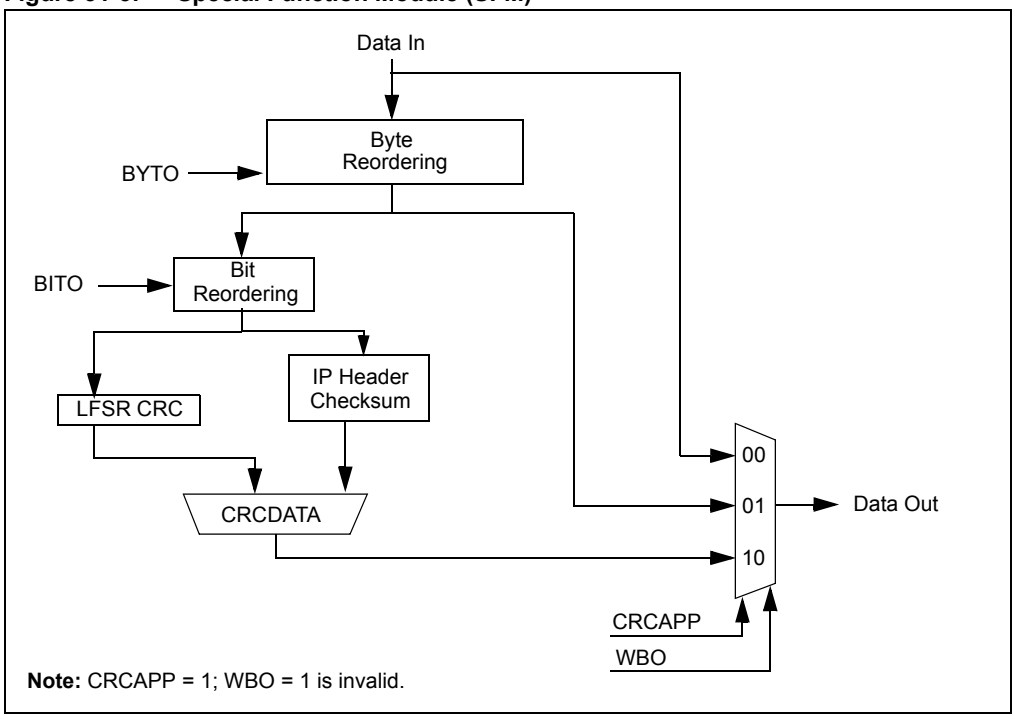
31.3.19 Special Function Module (SFM) Mode

The DMA module has one integrated Special Function module (SFM) shared by all channels.

As shown in Figure 31-5, the SFM has the following blocks:

- LFSR CRC
- IP Header Checksum
- Byte Reordering
- Bit Reordering

Figure 31-5: Special Function Module (SFM)



Note: CRCAPP = 1; WBO = 1 is invalid.

Depending on the device variant, the SFM is a highly configurable, 16-bit or 32-bit CRC generator. The SFM can be assigned to any available DMA channel by setting the CRCCH bits (DCRCCON) appropriately. The SFM is enabled by setting the CRCEN bit (DCRCCON<7>).

The data from the source can be optionally subjected to byte reordering using the WBO bit. The data is then optionally passed to the LFSR CRC or IP header checksum blocks based on the setting of the CRCTYP bit in the DCRCCON register as shown in Figure 31-5.

Further, the SFM modifies the behavior of the DMA channel associated with the SFM. The behavior of the channel is selected by the CRCAPP bit (DCRCCON<6>), resulting in the following two modes:

- Background Mode: CRC is calculated in the background, with normal DMA behavior maintained (see **Section 31.3.19.1 "CRC Background Mode (CRCAPP = 0)"**).
- Append Mode: Data read from the source is not written to the destination, but the CRC data is accumulated in the CRC data register. The accumulated CRC is written to the location given by DCHxDSA when a block transfer completes (see **Section 31.3.19.2 "CRC Append Mode (CRCAPP =** 1**)"**).

The order in which data is written to the destination can be selected using the WBO bit (DRCCON<27>). If the WBO bit is cleared, the writes to the destination are unaltered. If the WBO bit is set, the writes to the destination are reordered as defined by the CRC Byte Order Selection (BYTO<1:0>) bits (DRCCON<29:28>).

Note:	This feature is not available on all devices. Refer to the specific device data sheet for availability.

The SFM generator can be seeded by writing to the DCRCDATA register before enabling the channel.

Note that when in IP Header Checksum mode (CRCTYP = 1) data written, reads back as the one's complement form as this is the current value of the checksum.

The CRC value in DCRCDATA can be read at any time during the CRC generation but is only valid once the transfer completes.

31.3.19.1 CRC BACKGROUND MODE (CRCAPP = 0)

In this mode, the behavior of the DMA channel is maintained. The DMA reads the data from the source, passes it through the CRC module and writes it to the destination. Writes to the destination obey the WBO selection. In this mode, the calculated CRC is left in the DCRCDATA register at the end of the block transfer.

This mode can be used to calculate a CRC as data is moved from a source address to a destination address. The data source can be either a memory buffer or a FIFO in a peripheral. Likewise, the destination can be either a memory buffer or a FIFO. When the data transfer completes, the user can read the calculated CRC value and either append it to the transmitted data or verify the received CRC data.

Background mode potentially ties up the CRC module for extended periods of time. For instance, when assigned to a UART data stream, the SFM cannot be used by another channel until the UART data stream completes.

Example 31-5: DMA LFSR CRC Calculation in Background Mode Code Example

```
/*
The following code example illustrates a DMA calculation using the CRC background mode. Data is
transferred from a 200 bytes Flash buffer to a RAM buffer and the CRC is calculated while the
transfer takes place. */

 unsigned int blockCrc;            // CRC of the flash block

 IEC1CLR=0x00010000;               // disable DMA channel 0 interrupts
 IFS1CLR=0x00010000;               // clear any existing DMA channel 0 interrupt flag


 DMACONSET=0x00008000;             // enable the DMA controller

 DCRCDATA=0xffff;                  // seed the CRC generator
 DCRCXOR=0x1021;                   // Use the standard CCITT CRC 16 polynomial: X^16+X^12+X^5+1
 DCRCCON=0x0f80;                   // CRC enabled, polynomial length 16, background mode
                                   // CRC attached to the DMA channel 0.


 DCH0CON=0x03;                     // channel off, priority 3, no chaining
 DCH0ECON=0;                       // no start irqs, no match enabled

                                   // program channel transfer
 DCH0SSA=VirtToPhys(flashBuff);    // transfer source physical address
 DCH0DSA=VirtToPhys(ramBuff);      // transfer destination physical address
 DCH0SSIZ=200;                     // source size
 DCH0DSIZ=200;                     // destination size
 DCHOCSIZ=200;                     // 200 bytes per event

 DCH0INTCLR=0x00ff00ff;            // DMA0: clear events, disable interrupts

 DCH0CONSET=0x80;                  // channel 0 on

                                   // initiate a transfer
 DCH0ECONSET=0x00000080;           // set CFORCE to 1

                                   // do something else while the transfer takes place

                                   // poll to see that the transfer was done
 BOOL error=FALSE;
 while(TRUE)
 {
       register int pollCnt;       // don't poll in a tight loop
       int dmaFlags=DCH0INT;
       if( (dmaFlags& 0x3)
       {                           // CHERIF (DCHxINT<0>) or CHTAIF (DCHxINT<1> set
              error=TRUE;          // error or aborted...
              break;
       }
              else if (dmaFlags&0x8)
       {                           // CHBCIF (DCHxINT<3>) set
              break;               // transfer completed normally
       }
       pollCnt=100;                // use an adjusted value here
       while(pollCnt--);           // wait before polling again
 }

 if(!error)
 {
       blockCrc=DCRDATA;           // read the CRC of the transferred flash block
 }
 else
 {
                                   // process error
 }
```

31.3.19.2 CRC APPEND MODE (CRCAPP = 1)

In this mode, the DMA only feeds source data to the CRC module; it does not write source data to the destination address. However, when the block transfer completes or a pattern match occurs, the DMA writes the CRC value to the destination address.

This mode is best used when multiple peripherals are required to use the CRC generator. In this case, the input data is accumulated in a buffer on the device. Once the buffer is complete the CRC is generated on the buffer and used appropriately. Because the DMA does not need to wait for multiple events (typically interrupts) a block of data is passed through the CRC in fairly short order, allowing the CRC module to be assigned to a different channel, or redirected to a different block of data.

The following usage notes apply to CRC Append mode:

- Only the source buffer is viewed when considering whether a block transfer is complete, the destination address (DCHxDSA) is only used as the location to write the generated CRC value.
- The destination size (DCHxDSIZ) can be a maximum of 4.
  - If DCHxDSIZ is greater than 4, only 4 bytes are written
  - If DCHxDSIZ is less than 4, only DCHxDSIZ bytes of the CRC are written
  - PLEN has no effect on the number of CRC bytes or bits written
- After the write, the channel is disabled.
- Any abort (i.e., abort IRQ asserts) prevents the CRC value from being written
- Reordering is not supported in append mode if WBO is set to '0'.

Example 31-6: CRC Calculation in Append Mode Code Example

```
/*
The following code example illustrates a DMA calculation using the CRC append mode. The CRC of
a 256 bytes flash buffer is calculated without performing any data transfer. As soon as the CRC
calculation is completed the CRC value of the flash buffer is available in a local variable for
further use. */

 unsigned int blockCrc;           // CRC of the flash block

 IEC1CLR=0x00010000;              // disable DMA channel 0 interrupts
 IFS1CLR=0x00010000;              // clear any existing DMA channel 0 interrupt flag


 DMACONSET=0x00008000;            // enable the DMA controller

 DCRCDATA=0xffff;                 // seed the CRC generator
 DCRCXOR=0x1021;                  // Use the standard CCITT CRC 16 polynomial: X^16+X^12+X^5+1
 DCRCCON=0x0fc0;                  // CRC enabled, polynomial length 16, append mode
                                  // CRC attached to the DMA channel 0.


 DCH0CON=0x03;                    // channel off, priority 3, no chaining
 DCH0ECON=0;                      // no start irqs, no match enabled

                                  // program channel transfer
 DCH0SSA=VirtToPhys(flashBuff);   // transfer source physical address
 DCH0DSA=VirtToPhys(&blockCrc);   // transfer destination physical address
 DCH0SSIZ=200;                    // source size
 DCH0DSIZ=200;                    // dst size
 DCHOCSIZ=200;                    // 200 bytes transferred per event

 DCH0INTCLR=0x00ff00ff;           // DMA0: clear events, disable interrupts
 DCH1INTCLR=0x00ff00ff;           // DMA1: clear events, disable interrupts

 DCH0CONSET=0x80;                 // channel 0 on

                                  // initiate a transfer
 DCH0ECONSET=0x00000080;          // set CFORCE to 1

                                  // do something else while the CRC calculation takes place

                                  // poll to see that the transfer was done
 BOOL error=FALSE;
 while(TRUE)
 {
       register int pollCnt;      // don't poll in a tight loop
       int dmaFlags=DCH0INT;
       if( (dmaFlags& 0x3)
       {                          // CHERIF (DCHxINT<0>) or CHTAIF (DCHxINT<1> set
              error=TRUE;         // error or aborted...
              break;
       }
              else if (dmaFlags&0x8)
       {                          // CHBCIF (DCHxINT<3>) set
              break;              // transfer completed normally
       }
       pollCnt=100;               // use an adjusted value here
       while(pollCnt--);          // wait before polling again
 }

 if(error)
 {
                                  // process error
 }

                                  // the block CRC is available in the blockCrc variable
```

31.3.19.3 DATA ORDER

Data read from the source can be reordered to allow for variations in the byte order of the source data, such as endianness. The reordered source data is written to the channel destination when WBO = 1. The unaltered source data is written to the destination when WBO = 0.

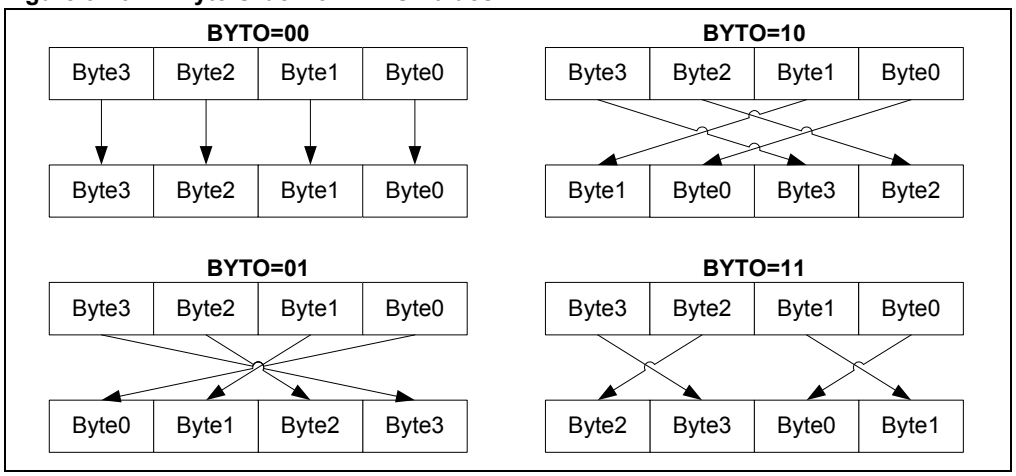
The CRC calculation takes place, even if the user does not utilize the result stored in DCRCDATA.

BYTO controls the byte order of the data being processed by the module. Figure 31-6 shows the different byte order settings and the effect on data reads. BYTO value of '01' is useful for reordering bytes within words. While BYTO values of '10' and '11' is useful for reordering bytes within half-words.

It is important to note that the data is reordered as it is read. This means that data that is not word-aligned may not be reordered correctly.

When using the LFSR CRC mode or IP Header Checksum mode of the SFM, the bit order can be changed by using the BITO bit.

Figure 31-6: Byte Order for BYTO Values



31.3.19.4 LFSR CRC

The CRC generator will take one system clock to process each byte of data read from the source. This implies that if 32-bits of data are read from the source, the CRC generation will take four system clocks to process the data.

When the CRYTYP bit is cleared, the SFM is set to LFSR CRC mode and calculates the LFSR CRC.

Note:	This feature is not available on all devices. Refer to the specific device data sheet for availability.

The implementation of the CRC module is software configurable. The terms of the polynomial and its length can be programmed using the DCRCXOR bits and the PLEN (DCRCCON) bits, respectively.

Example 31-7 and Example 31-8 show the polynomials for the 16-bit and 32-bit CRC.

Example 31-7: 16-bit CRC Polynomial

$$x^{16} + x^{12} + x^5 + 1$$

Example 31-8: 32-bit CRC Polynomial

$$x^{31} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$$

To program either polynomial into the CRC generator, the CRC register bits should be set as shown in the following table:

Table 31-5: Example CRC Setup

CRC Type	Bit Name	Bit Value
Devices with 16-bit CRC	PLEN<3:0>	'b1111'
	DCRCXOR<15:0>	'b0001 0000 0010 000'
Devices with 32-bit CRC	PLEN<4:0>	'b11111'
	DCRCXOR<31:0>	'b0000 0100 1100 0001 0001 1101 1011 0110'

The PLEN bits (DCRCCON) in the CRC generator are used to select which bit is used as the feedback point of the CRC. For a 16-bit CRC example, if PLEN<3:0> = 0x0110, then bit 6 of the Shift register is fed into the XOR gates of all bits set in the CRCXOR register.

The CRCXOR feedback points are specified using the DCRCXOR register. Setting the Nth bit in the DCRCXOR register will enable the input to the Nth bit of the CRC Shift register to be XORed with the (PLEN + 1)th bit of the CRC Shift register. Bit 0 of the CRC generator is always XORed.

31.3.19.5 CALCULATING THE IP HEADER CHECKSUM

When the CRCTYP bit is set, the SFM calculates the IP header checksum. Use the following procedure to calculate the IP header checksum:

1. Configure a channel to point to the IP header.
2. Configure CRCCON to enable the SFM and select the channel being used.
3. Set CRCTYP bit in the DCRCCON register, which selects IP Header checksum.
4. Set DCRCDATA to '0000'.
5. Start the transfer.
6. When the transfer completes, read the data out of the DCRCDATA register.

Note:	This feature is not available on all devices. Refer to the specific device data sheet for availability.

31.4 INTERRUPTS

The DMA device has the ability to generate interrupts reflecting the events that occur during the channel's data transfer:

- Error interrupts, signaled by each channel's CHERIF bit (DCHxINT<0>) and enabled using the CHERIE bit (DCHxINT<16>). This event occurs when there is an address error occurred during the channel transfer operation.
- Abort interrupts, signaled by each channel's CHTAIF bit (DCHxINT<1>) and enabled using the CHTAIE bit (DCHxINT<17>). This event occurs when a DMA channel transfer gets aborted because of a system event (interrupt) matching the CHAIRQ<7:0> (DCHxECON<23:16>) when the abort interrupt request is enabled, AIRQEN = 1 (DCHxECON<3>).
- Block complete interrupts, signaled by each channel's CHBCIF bit (DCHxINT<3>) and enabled using the CHBCIE bit (DCHxINT<19>). This event occurs when a DMA channel block transfer is completed.
- Cell complete interrupts, signaled by each channel's CHCCIF bit (DCHxINT<2>) and enabled using the CHCCIE bit (DCHxINT<18>). This event occurs when a DMA channel cell transfer is completed.
- Source Address Pointer activity interrupts: either when the Channel Source Pointer reached the end of the source, signaled by the CHSDIF bit (DCHxINT<7>) and enabled by CHSDIE bit (DCHxINT<23>), or when the Channel Source Pointer reached midpoint of the source, signaled by the CHSHIF bit (DCHxINT<6>) and enabled by the CHSHIE bit (DCHx-INT<22>).
- Destination Address Pointer activity interrupts: either when the Channel Destination Pointer reached the end of the destination, signaled by the CHDDIF bit (DCHxINT<5>) and enabled by the CHDDIE bit (DCHxINT<21>), or when the Channel Destination Pointer reached midpoint of the destination, signaled by the CHDHIF bit (DCHxINT<4>) and enabled by the CHDHIE bit (DCHxINT<20>).

All the interrupts belonging to a DMA channel map to the corresponding channel interrupt vector.

Note:	Not all DMA channels are available on all devices. Refer to the specific device data sheet for availability.

The corresponding DMA channels interrupt flags are:

- DMA0IF (IFS1<16>)
- DMA1IF (IFS1<17>)
- DMA2IF (IFS1<18>)
- DMA3IF (IFS1<19>)
- DMA4IF (IFS1<20>)
- DMA5IF (IFS1<21>)
- DMA6IF (IFS1<22>)
- DMA7IF (IFS1<23>)

All these interrupt flags must be cleared in software.

A DMA channel is enabled as a source of interrupts via the respective DMA interrupt enable bits:

- DMA0IE (IEC1<16>)
- DMA1IE (IEC1<17>)
- DMA2IE (IEC1<18>)
- DMA3IE (IEC1<19>)
- DMA4IE (IEC1<20>)
- DMA5IE (IEC1<21>)
- DMA6IE (IEC1<22>)
- DMA7IE (IEC1<23>)

The interrupt-priority-level bits and interrupt-subpriority-level bits must also be configured:

- DMA0IP<2:0> (IPC9<4:2>), DMA0IS<1:0> (IPC9<1:0>)
- DMA1IP<2:0> (IPC9<12:10>), DMA1IS<1:0> (IPC9<9:8>)
- DMA2IP<2:0> (IPC9<20:18>), DMA2IS<1:0> (IPC9<17:16>)
- DMA3IP<2:0> (IPC9<28:26>), DMA3IS<1:0> (IPC9<25:24>)
- DMA4IP<2:0> (IPC10<4:2>), DMA4IS<1:0> (IPC10<1:0>)
- DMA5IP<2:0> (IPC10<12:10>), DMA5IS<1:0> (IPC10<9:8>)
- DMA7IP<2:0> (IPC10<20:18>), DMA6IS<1:0> (IPC10<17:16>)
- DMA7IP<2:0> (IPC10<28:26>), DMA7IS<1:0> (IPC10<25:24>)

31.4.1 Interrupt Configuration

Each DMA channel internally has multiple interrupt flags (CHSDIF, CHSHIF, CHDDIF, CHDHIF, CHBCIF, CHCCIF, CHTAIF, CHERIF) and corresponding enable interrupt control bits (CHSDIE, CHSHIE, CHDDIE, CHDHIE, CHBCIE, CHCCIE, CHTAIE, CHERIE).

However, for the interrupt controller, there is just one dedicated interrupt flag bit per channel, DMAxIF, and the corresponding interrupt enable/mask bits, DMAxIE.

Note:	Depending on the device variant, up to 8 (i.e., 0-7) interrupt flags and interrupt enable/mask bits are available. Refer to the specific device data sheet for availability.

Therefore, note that all of the interrupt conditions for a specific DMA channel share just one interrupt vector. Each DMA channel can have its own priority level independent of other DMA channels.

Note that the DMAxIF bits will be set without regard to the state of the corresponding enable bits DMAxIE. The DMAxIF bits can be polled by software if desired.

The DMAxIE bits are used to define the behavior of the Vector Interrupt Controller or INT when a corresponding DMAxIF bit is set. When the corresponding DMAxIE bit is clear, the INT module does not generate a CPU interrupt for the event. If the DMAxIE bit is set, the INT module will generate an interrupt to the CPU when the corresponding DMAxIF bit is set (subject to the priority and subpriority as follows).

It is the responsibility of the user's software routine that services a particular interrupt to clear the appropriate interrupt flag bit before the service routine is complete.

The priority of each DMA channel can be set independently with the DMAxIP bits in the IPCx register. These priorities define the priority group to which the interrupt source will be assigned. The priority groups range from a value of 7 (the highest priority), to a value of 0, which does not generate an interrupt. An interrupt being serviced will be preempted by an interrupt in a higher priority group.

The subpriority bits allow setting the priority of an interrupt source within a priority group. The values of the subpriority range from 3 (the highest priority), to 0 the lowest priority. An interrupt with the same priority group but having a higher subpriority value will not preempt a lower subpriority interrupt that is in progress.

The priority group and subpriority bits allow more than one interrupt source to share the same priority and subpriority. If simultaneous interrupts occur in this configuration, the natural order of the interrupt sources within a Priority/subpriority group pair determine the interrupt generated. The natural priority is based on the vector numbers of the interrupt sources. The lower the vector number the higher the natural priority of the interrupt. Any interrupts that were overridden by natural order will then generate their respective interrupts based on Priority, subpriority and natural order after the interrupt flag for the current interrupt is cleared.

After an enabled interrupt is generated, the CPU will jump to the vector assigned to that interrupt. The vector number for the interrupt is the same as the natural order number. The CPU will then begin executing code at the vector address. The user's code at this vector address should perform any application-specific operations and clear the DMAxIF interrupt flags, and then exit. Refer to the vector address table details in **Section 8. "Interrupts"** (DS61108) in the "*PIC32MX Family Reference Manual*" for more information on interrupts.

Table 31-6: DMA Interrupt Vectors for Various Offsets with EBASE = 0x8000:0000

Interrupt	Vector/Natural Order	IRQ Number	Vector Address IntCtl.VS = 0x01	Vector Address IntCtl.VS = 0x02	Vector Address IntCtl.VS = 0x04	Vector Address IntCtl.VS = 0x08	Vector Address IntCtl.VS = 0x10
DMA0	36	48	8000 0680	8000 0B00	8000 1400	8000 2600	8000 4A00
DMA1	37	49	8000 06A0	8000 0B40	8000 1480	8000 2700	8000 4C00
DMA2	38	50	8000 06C0	8000 0B80	8000 1500	8000 2800	8000 4E00
DMA3	39	51	8000 06E0	8000 0BC0	8000 1580	8000 2900	8000 5000
DMA4(1)	40	52	8000 0700	8000 0C00	8000 1600	8000 2A00	8000 5200
DMA5(1)	41	53	8000 0720	8000 0C40	8000 1680	8000 2B00	8000 5400
DMA6(1)	42	54	8000 0740	8000 0C80	8000 1700	8000 2C00	8000 5600
DMA7(1)	43	55	8000 0760	8000 0CC0	8000 1780	8000 2D00	8000 5800

Note 1: These interrupts are not available on all devices. Refer to the specific device data sheet for availability.

Example 31-9: DMA Channel Initialization with Interrupts Enabled Code Example

```
/*
The following code example illustrates a DMA channel 0 interrupt configuration.
When the DMA channel 0 interrupt is generated, the CPU will jump to the vector assigned to
DMA0 interrupt.
*/

       IEC1CLR=0x00010000;         // disable DMA channel 0 interrupts
       IFS1CLR=0x00010000;         // clear any existing DMA channel 0 interrupt flag

       DMACONSET=0x00008000;       // enable the DMA controller
       DCH0CON=0x03;               // channel off, priority 3, no chaining

       DCH0ECON=0;                 // no start or stop irq's, no pattern match

                                   // program the transfer
       DCH0SSA=0x1d010000;         // transfer source physical address
       DCH0DSA=0x1d020000;         // transfer destination physical address
       DCH0SSIZ=200;               // source size 200 bytes
       DCH0DSIZ=200;               // destination size 200 bytes
       DCH0CSIZ=200;               // 200 bytes transferred per event

       DCH0INTCLR=0x00ff00ff;      // clear existing events, disable all interrupts
       DCH0INTSET=0x00090000;      // enable Block Complete and error interrupts

       IPC9CLR=0x0000001f;         // clear the DMA channel 0 priority and sub-priority
       IPC9SET=0x00000016;         // set IPL 5, sub-priority 2
       IEC1SET=0x00010000;         // enable DMA channel 0 interrupt

       DCH0CONSET=0x80;            // turn channel on
                                   // initiate a transfer
       DCH0ECONSET=0x00000080;     // set CFORCE to 1

                                   // do something else

                                   // will get an interrupt when the block transfer is done
                                   // or when error occurred
```

Example 31-10: DMA Channel 0 ISR Code Example

```
/*
The following code example demonstrates a simple Interrupt Service Routine for DMA channel 0
interrupts. The user's code at this vector should perform any application specific operations
and must clear the DMA0 interrupt flags before exiting.
*/

void __ISR(_DMA_0_VECTOR, ipl5) __DMA0Interrupt(void)
{
       int dmaFlags=DCH0INT&0xff;     // read the interrupt flags

       /*
       perform application specific operations in response to any interrupt flag set
       */

       DCH0INTCLR=0x000000ff;         // clear the DMA channel interrupt flags
       IFS1CLR = 0x00010000;          // Be sure to clear the DMA0 interrupt flags
                                      // before exiting the service routine.
}
```

Note: The DMA ISR code example shows MPLAB® C32 C compiler specific syntax. Refer to your compiler manual regarding support for ISRs.

31.5 OPERATION IN POWER-SAVING AND DEBUG MODES

31.5.1 DMA Operation in Idle Mode

When the device enters Idle mode, the system clock sources remain functional and the DMA module continues to operate.

On some variants, the SIDL bit (DMACON<13>) selects whether the module will stop or continue functioning on Idle.

If SIDL = 0, the module will continue operation in Idle mode and will have the clocks turned off. If SIDL = 1, the module will discontinue operation in Idle mode. The DMA module will turn off the clocks so that the power consumption is more efficient.

Note:	The DMA cannot be used by a peripheral that has its SIDL bit set to '1'.

31.5.2 DMA Operation in Sleep Mode

When the device enters Sleep mode, the system clock is disabled. No DMA activity can occur in this mode.

31.5.3 DMA Operation in Debug Mode

The FRZ bit (DMACON<14>) determines whether the DMA module will run or stop while the CPU is executing debug exception code (i.e., application is halted) in Debug mode. When FRZ = 0, the DMA module continues to run even when application is halted in Debug mode. When FRZ = 1 and the application is halted in Debug mode, the module will freeze its operations and make no changes to the state of the DMA module. The module will resume its operation after CPU resumes execution.

Note:	The FRZ bit is readable and writable only when the CPU is executing in Debug Exception mode. In all other modes, the FRZ bit reads as '0'. If FRZ bit is changed during Debug mode, the new value does not take effect until the current Debug Exception mode is exited and re-entered. During the Debug Exception mode, the FRZ bit reads the state of the peripheral when entering Debug mode.

31.6 EFFECTS OF VARIOUS RESETS

31.6.1 Device Reset

All DMA registers are forced to their Reset states upon a device Reset. When the asynchronous Reset input goes active, the DMA logic:

- Resets all fields in DMACON, DMASTAT, DMAADDR, DCRCCON, DCRCDATA, DCRCXOR
- Sets the appropriate values in each channel's register fields: DCHxCON, DCHxECON, DCHxINT, DCHxSSIZ, DCHxDSIZ, DCHxSPTR, DCHxDPTR, DCHxCSIZ, DCHxCPTR, DCHxDAT
- Registers DCHxSSA and DCHxDSA have random values after Reset
- Aborts any on-going data transfers

31.6.2 Power-On Reset

All DMA registers are forced to their Reset states upon a Power-on Reset.

31.6.3 Watchdog Timer Reset

All DMA registers are forced to their Reset states upon a Watchdog Timer Reset.

31.7 RELATED APPLICATION NOTES

This section lists application notes that are related to this section of the manual. These application notes may not be written specifically for the PIC32MX device family, but the concepts are pertinent and could be used with modification and possible limitations. The current application notes related to the Direct Memory Access (DMA) module are:

Title	Application Note #
No related application notes at this time	N/A

Note: Visit the Microchip web site (www.microchip.com) for additional application notes and code examples for the PIC32MX family of devices.

31.8 REVISION HISTORY

Revision A (October 2007)

This is the initial released version of this document.

Revision B (October 2007)

Updated document to remove Confidential status.

Revision C (April 2008)

Revised status to Preliminary; Revised U-0 to r-x; Revised Table 31-1; Revised Table 31-2 (DCHxCON, bit 3), deleted Note 1; Revised Registers 31-19, 31-39, 31-43, 31-47, 31-48, 31-49, 31-53; Revise Sections 31.3, 31.3.2; Revised Examples 31-1, 31-3, 31-4, 31-6, 31-7, 31-8; Delete Example 31-2 and renumber examples; Delete Section 31.3.3 and renumber sections; Revised Section 31.3.20.7.

Revision D (June 2008)

Revised Registers 31-58 to 31-60, Footnote; Revised Example 31-8; Change Reserved bits “Maintain as” to “Write”; Added Note to ON bit (DMACON Register).

Revision E (August 2009)

This revision introduces new bits and functionality that are only available on certain devices. The following details the resulting changes:

- DMA Register Summary (Table 31-1)
  - Added the BUSY, BYTO1, BYTO0, WBO, BITO, CRCTYP and CHBUSY bits
  - Removed references to the IEC1, IPC9 and IFS1 registers
  - Added the Address Offset column to the DMA Register Summary
  - Added Notes 1, 2 and 3, which describe the Clear, Set and Invert registers
  - Added Notes 4 and 5 regarding the availability of certain bits and ranges of bits depending on the device variant
- Added Notes describing the Clear, Set and Invert registers to the following registers:
  - DMACON (Register 31-1)
  - DMASTAT (Register 31-2)
  - DMAADDR (Register 31-3)
  - DCRCCON (Register 31-4)
  - DCRCDATA (Register 31-5)
  - DCRCXOR (Register 31-6)
  - DCHxCON (Register 31-7)
  - DCHxECON (Register 31-8)
  - DCHxINT (Register 31-9)
  - DCHxSSA (Register 31-10)
  - DCHxDSA (Register 31-11)
  - DCHxSSIZ (Register 31-12)
  - DCHxDSIZ (Register 31-13)
  - DCHSPTR (Register 31-14)
  - DCHxDPTR (Register 31-15)
  - DCHxCSIZ (Register 31-16)
  - DCHxCPTR (Register 31-17)
  - DCHxDAT (Register 31-18)
- Removed these registers: IFS1, IEC1 and IPC9
- Added the BUSY bit (DMACON<11>) and Note 1 regarding availability of the SIDL and BUSY bits to Register 31-1

Revision E (August 2009) (Continued)

- Updated the DMACH bit (DMASTAT<2:0>) and added Note 2 regarding the availability of all bits in Register 31-2
- Added the BYTO1, BYTO0, WBO, BITO and CRCTYP bits, updated bits PLEN<4:0> and CRCCH<2:0>, and added Notes 1 and 2 to Register 31-4
- Updated DCRCDATA bits and added Note 1 to Register 31-5
- Updated DCRCXOR bits and added Note 1 to Register 31-6
- Added CHBUSY bit (DCHxCON<15>) and added Note 1 to Register 31-7
- Updated DCHxSSIZ bits and added Note 1 to Register 31-12
- Updated DCHxDSIZ bits and added Note 1 to Register 31-13
- Updated DCHxSPTR bits and added Note 2 to Register 31-14
- Updated DCHxDPTR bits and added Note 1 to Register 31-15
- Updated DCHxCSIZ bits and added Note 1 to Register 31-16
- Updated DCHxCPTR bits and added Note 2 to Register 31-17
- Updated the lowest priority channel number and added a related note to the fourth paragraph in **Section 31.3.4 “Channel Chaining Mode Operation”**
- Added information on suspending the DMA module and a related note to **Section 31.3.6 “Suspending Transfers”** and **Section 31.3.18 “DMA Suspend”**
- Updated **Section 31.3.19 “Special Function Module (SFM) Mode”** to differentiate between the 16-bit and 32-bit CRC
- Added **Section 31.3.19.5 “Calculating the IP Header Checksum”**
- Added DMA channel interrupt flags, enable bits and priority-level bits to **Section 31.4 “Interrupts”**
- Added DMA interrupt vectors (DMA4-DMA7) to Table 31-6
- Updated **Section 31.5.1 “DMA Operation in Idle Mode”**

NOTES: