Direct Memory Access Controller (DMA)

HIGHLIGHTS

This section of the manual contains the following major topics:

Note: This family reference manual section is meant to serve as a complement to device data sheets. Depending on the device variant, this manual section may not apply to all dsPIC33/PIC24 family devices.

Please consult the note at the beginning of the **"Direct Memory Access (DMA) Controller"** chapter in the current device data sheet to check whether this document supports the device you are using.

Device data sheets and family reference manual sections are available for download from the Microchip Worldwide Website at: http://www.microchip.com

1.0 INTRODUCTION

The Direct Memory Access (DMA) Controller is designed to service high-data-throughput peripherals operating on the SFR bus, allowing them to access data memory directly and eliminating the need for CPU-intensive management. By allowing these data-intensive peripherals to share their own data path, the main data bus is also off-loaded, resulting in additional power savings.

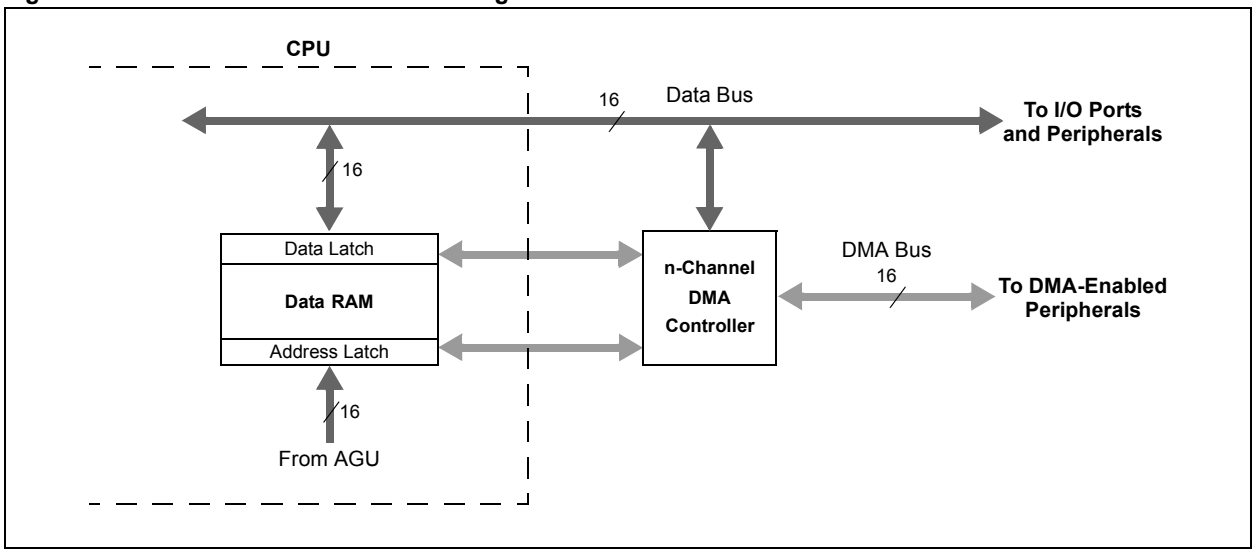
The DMA Controller has these features:

- Multiple Independent and Independently-Programmable Channels
- Concurrent Operation with the CPU (no DMA-caused Wait states)
- DMA Bus Arbitration
- Five Programmable Address Modes
- Four Programmable Transfer Modes
- Four Flexible Internal Data Transfer Modes
- Byte or Word Support for Data Transfer
- 16-Bit Source and Destination Address Register for Each Channel, Dynamically Updated and Reloadable
- 16-Bit Transaction Count Register, Dynamically Updated and Reloadable
- Upper and Lower Address Limit Registers
- Counter Half-Full Level Interrupt
- Software-Triggered Transfer
- Null Write Mode for Symmetric Buffer Operations

1.1 Organization

Conceptually, the DMA Controller functions both as a peripheral and a direct extension of the CPU. It is located on the microcontroller data bus between the CPU and DMA-enabled peripherals, with direct access to SRAM (Figure 1-1). This partitions the SFR bus into two buses, allowing the DMA Controller access to the DMA-capable peripherals located on the new DMA SFR bus. This also lowers bus loading for less power consumption per access.

Figure 1-1: DMA Functional Block Diagram



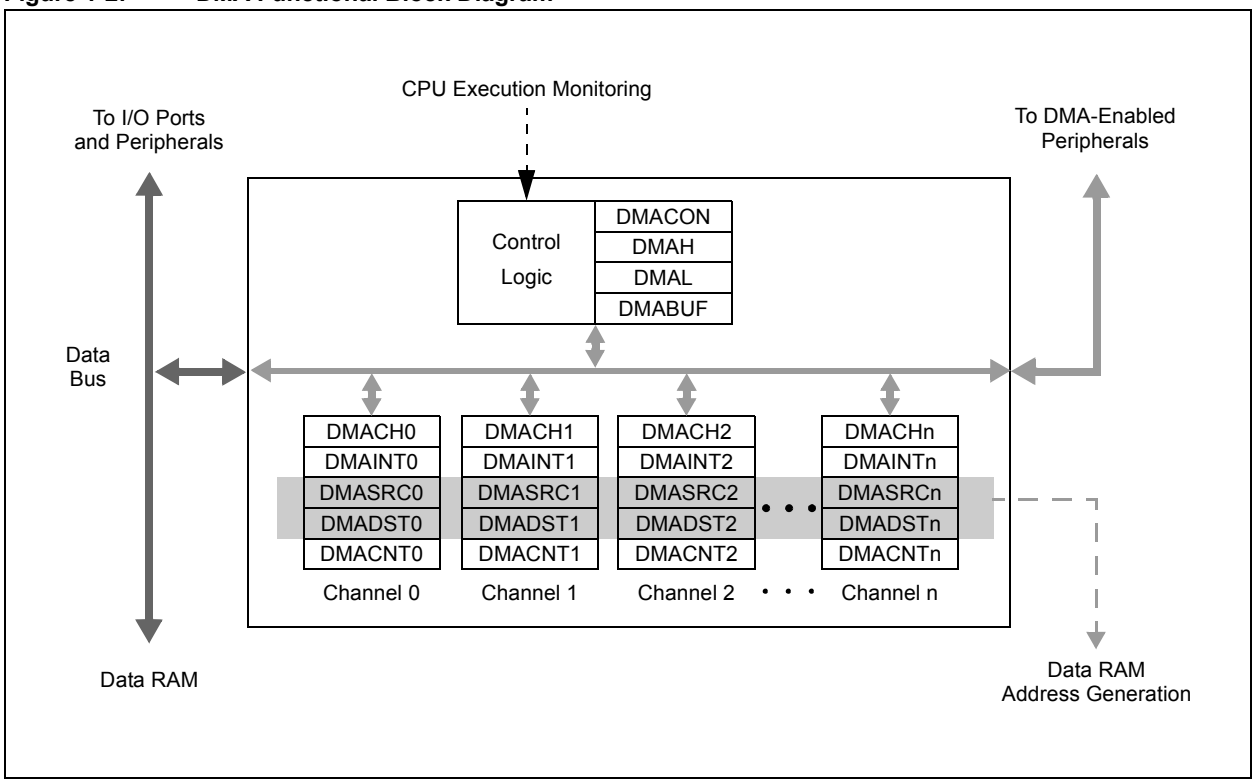
The controller serves as a master device on the DMA SFR bus, controlling data flow from DMA-capable peripherals. An arbiter monitors CPU instruction processing, allowing it to be aware of when the CPU requires access to peripherals on the DMA bus, and automatically relinquishing control to the CPU as needed. When the CPU is servicing peripherals that are not on the DMA bus, the DMA Controller is free to service peripherals on the DMA bus while the CPU is performing its operations. In this way, the effective bandwidth for handling data is increased; at the same time, DMA operations can proceed without causing a processor Stall. When the CPU and DMA are accessing the SFR, the CPU gets priority over the DMA and the DMA will have to wait to access the SFR until the CPU completes the task. Because of the monitoring of CPU execution, the DMA Controller is essentially transparent to the user.

On some devices, the priority of the DMA can be set above that of the CPU using the MSTRPR register. Refer to the device-specific data sheet for availability.

The DMA Controller itself is composed of multiple independent DMA channel controllers, or simply channels (Figure 1-2). Each channel can be independently programmed to transfer data between different areas of the data RAM, move data between single or multiple addresses, use a wide range of hardware triggers to initiate transfers and conduct programmed transactions once or many times. Multiple channels may even be programmed to work together in order to carry out more complex data transfers without CPU intervention. The number of channels present depends on the specific device family; refer to the device data sheet for more information.

The top-level controller sets the boundary addresses for all DMA operations, regardless of the channel. It also arbitrates data bus access between the channels based on a user-selectable priority scheme and determines how DMA will operate in power-saving modes.

Figure 1-2: DMA Functional Block Diagram



2.0 REGISTERS

The DMA Controller uses a number of registers to control its operation. The number of registers depends on the number of channels implemented for a particular device.

There are always four module-level registers (one control and three buffer/address):

- DMACON: DMA Engine Control Register
- DMAH and DMAL: DMA High and Low Address Limit Registers
- DMABUF: DMA Data Buffer Register

Each of the DMA channels implements five registers (two control and three buffer/address):

- DMACHn: DMA Channel n Control Register
- DMAINTn: DMA Channel n Interrupt Control Register
- DMASRCn: DMA Channel n Source Address Register
- DMADSTn: DMA Channel n Destination Address Register
- DMACNTn: DMA Channel n Count Register

A summary of the registers associated with the DMA Controller is provided in Table 3-1.

Note:	During a DMA operation, if the DMA is disabled by clearing the DMAEN bit (DMACON[15] = 0), it is recommended to clear all the used DMA channels before enabling the DMA again (DMAEN DMACON[15] = 1).

3.0 REGISTER MAP

Table 3-1: DMA Register Map

Register Name	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	All Resets
DMACON	DMAEN	—	DMASIDL(2)	—	—	—	—	—	—	—	—	—	—	—	—	PRSSEL	0000
DMABUF	DMA Data Buffer																0000
DMAL	DMA Low Address Limit																0000
DMAH	DMA High Address Limit																0000
DMACH0	—	—	—	—	—	NULLW	RELOAD	CHREQ	SAMODE[1:0]		DAMODE[1:0]		TRMODE[1:0]		SIZE	CHEN	0000
DMAINT0	DBUFWF	CHSEL[6:0]							HIGHIF	LOWIF	DONEIF	HALFIF	OVRUNIF	—	—	HALFEN	0000
DMASRC0	DMA Channel 0 Source Address																0000
DMADST0	DMA Channel 0 Destination Address																0000
DMACNT0	DMA Channel 0 Count																0001
DMACH1	—	—	—	—	—	NULLW	RELOAD	CHREQ	SAMODE[1:0]		DAMODE[1:0]		TRMODE[1:0]		SIZE	CHEN	0000
DMAINT1	DBUFWF	CHSEL[6:0]							HIGHIF	LOWIF	DONEIF	HALFIF	OVRUNIF	—	—	HALFEN	0000
DMASRC1	DMA Channel 1 Source Address																0000
DMADST1	DMA Channel 1 Destination Address																0000
DMACNT1	DMA Channel 1 Count																0001
• • •	• • •																• • •
DMACHn(1)	—	—	—	—	—	NULLW	RELOAD	CHREQ	SAMODE[1:0]		DAMODE[1:0]		TRMODE[1:0]		SIZE	CHEN	0000
DMAINTn(1)	DBUFWF	CHSEL[6:0]							HIGHIF	LOWIF	DONEIF	HALFIF	OVRUNIF	—	—	HALFEN	0000
DMASRCn(1)	DMA Channel n Source Address																0000
DMADSTn(1)	DMA Channel n Destination Address																0000
DMACNTn(1)	DMA Channel n Count																0001

Legend: x = unknown value on Reset; — = unimplemented, read as '0'. Reset values are shown in hexadecimal.

Note 1: The number of available DMA channels is device-dependent. Refer to the specific device data sheet for the exact number of DMA channels implemented.

2: Not available on all devices. Refer to the device-specific data sheet for availability.

Register 3-1: DMACON: DMA Engine Control Register

R/W-0	U-0	R/W-0	U-0	U-0	U-0	U-0	U-0
DMAEN	—	DMASIDL(1)	—	—	—	—	—
bit 15							bit 8

U-0	U-0	U-0	U-0	U-0	U-0	U-0	R/W-0
—	—	—	—	—	—	—	PRSSEL
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 15 **DMAEN:** DMA Module Enable bit
1 = Enables module
0 = Disables module and terminates all active DMA operation(s)

bit 14 **Unimplemented:** Read as '0'

bit 13 **DMASIDL:** DMA Stop in Idle bit(1)
1 = DMA continues to run in Idle mode
0 = DMA is disabled in Idle mode

bit 12-1 **Unimplemented:** Read as '0'

bit 0 **PRSSEL:** Channel Priority Scheme Selection bit
1 = Round-robin scheme
0 = Fixed priority scheme

Note 1: Not available on all devices. Refer to the device-specific data sheet for availability.

Register 3-2: DMABUF: DMA Data Buffer Register

R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
DBUF[15:8](1)							
bit 15							bit 8

R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
DBUF[7:0](1)							
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 15-0 **DBUF[15:0]:** Internal Data Buffer bits(1)
These bits reflect the content of the internal data buffer the DMAC macro uses to hold the data being moved from SADDR[15:0] to DADDR[15:0].

Note 1: When operating on byte rather than word data, the byte read data gets stored in both the upper and lower byte of the DBUF[15:0] bits to accommodate byte moves between even and odd memory locations.

Register 3-3: DMAL: DMA Low Address Limit 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
LADDR[15:8]							
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
LADDR[7:0]							
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as ‘0’	
-n = Value at POR	‘1’ = Bit is set	‘0’ = Bit is cleared	x = Bit is unknown

bit 15-0 **LADDR[15:0]:** Low Limit Address bits

These bits indicate the lower address location below which the DMAC module initiates a transaction, which results in the associated channel interrupt; LOWIF is set and CHEN is clear.

Register 3-4: DMAH: DMA High Address Limit 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
HADDR[15:8]							
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
HADDR[7:0]							
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as ‘0’	
-n = Value at POR	‘1’ = Bit is set	‘0’ = Bit is cleared	x = Bit is unknown

bit 15-0 **HADDR[15:0]:** High Limit Address bits

These bits indicate the upper address location beyond which the DMAC module initiates a transaction, which results in the associated channel interrupt; HIGHIF is set and CHEN is clear.

Register 3-5: DMACHn: DMA Channel n Control Register

U-0	U-0	U-0	r-0	U-0	R/W-0	R/W-0	R/W-0
—	—	—	—	—	NULLW	RELOAD(1)	CHREQ(3)
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
SAMODE[1:0]		DAMODE[1:0]		TRMODE[1:0]		SIZE	CHEN
bit 7							bit 0

Legend:	r = Reserved bit		
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 15-13 **Unimplemented:** Read as '0'

bit 12 **Reserved:** Maintain as '0'

bit 11 **Unimplemented:** Read as '0'

bit 10 **NULLW:** Null Write Mode bit

1 = A dummy write is initiated to DMASRCn for every write to DMADSTn
0 = No dummy write is initiated

bit 9 **RELOAD:** Address and Count Reload bit(1)

1 = DMASRCn, DMADSTn and DMACNTn registers are reloaded to their previous values upon the start of the next operation
0 = DMASRCn, DMADSTn and DMACNTn are not reloaded on the start of the next operation(2)

bit 8 **CHREQ:** DMA Channel Software Request bit(3)

1 = A DMA request is initiated by software; automatically cleared upon completion of a DMA transfer
0 = No DMA request is pending

bit 7-6 **SAMODE[1:0]:** Source Address Mode Selection bits

11 = DMASRCn is used in Peripheral Indirect Addressing and remains unchanged
10 = DMASRCn is decremented based on the SIZE bit after a transfer completion
01 = DMASRCn is incremented based on the SIZE bit after a transfer completion
00 = DMASRCn remains unchanged after a transfer completion

bit 5-4 **DAMODE[1:0]:** Destination Address Mode Selection bits

11 = DMADSTn is used in Peripheral Indirect Addressing and remains unchanged
10 = DMADSTn is decremented based on the SIZE bit after a transfer completion
01 = DMADSTn is incremented based on the SIZE bit after a transfer completion
00 = DMADSTn remains unchanged after a transfer completion

bit 3-2 **TRMODE[1:0]:** Transfer Mode Selection bits

11 = Repeated Continuous
10 = Continuous
01 = Repeated One-Shot
00 = One-Shot

bit 1 **SIZE:** Data Size Selection bit

1 = Byte (8-bit)
0 = Word (16-bit)

bit 0 **CHEN:** DMA Channel Enable bit

1 = The corresponding channel is enabled
0 = The corresponding channel is disabled

Note 1: Only the original DMACNTn is required to be stored to recover the original DMASRCn and DMADSTn.

2: DMASRCn, DMADSTn and DMACNTn are always reloaded in Repeated mode transfers (DMACHn[2] = 1), regardless of the state of the RELOAD bit.

3: The number of transfers executed while CHREQ is set depends on the configuration of TRMODE[1:0].

Register 3-6: DMAINTn: DMA Channel n Interrupt Control Register

R-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
DBUFWF(1)	CHSEL[6:0](3)						
bit 15							bit 8

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	U-0	U-0	R/W-0
HIGHIF(1,2)	LOWIF(1,2)	DONEIF(1)	HALFIF(1)	OVRUNIF(1)	—	—	HALFEN
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 15 **DBUFWF:** Buffered Data Write Flag bit(1)
- 1 = The content of the DMA buffer has not been written to the location specified in DMADSTn or DMASRCn in Null Write mode
- 0 = The content of the DMA buffer has been written to the location specified in DMADSTn or DMASRCn in Null Write mode

bit 14-8 **CHSEL[6:0]:** DMA Channel Trigger Selection bits(3)

These bits select one of 64 possible DMA triggers to be connected to the channel's input. Generally, these are the device-level interrupts; the list of triggers corresponds to the reverse order natural priority interrupt list. Refer to the specific device data sheet for the mapping of bit values to peripherals.

bit 7 **HIGHIF:** DMA High Address Limit Interrupt Flag bit(1,2)
- 1 = The DMA channel has attempted to access an address higher than DMAH or the upper limit of the data RAM space
- 0 = The DMA channel has not invoked the high address limit interrupt

bit 6 **LOWIF:** DMA Low Address Limit Interrupt Flag bit(1,2)
- 1 = The DMA channel has attempted to access the DMA SFR address lower than DMAL, but above the SFR range (07FFh)
- 0 = The DMA channel has not invoked the low address limit interrupt

bit 5 **DONEIF:** DMA Complete Operation Interrupt Flag bit(1)

If CHEN = 1:
- 1 = The previous DMA session has ended with completion
- 0 = The current DMA session has not yet completed

If CHEN = 0:
- 1 = The previous DMA session has ended with completion
- 0 = The previous DMA session has ended without completion

bit 4 **HALFIF:** DMA 50% Watermark Level Interrupt Flag bit(1)
- 1 = DMACNTn has reached the halfway point to 0000h
- 0 = DMACNTn has not reached the halfway point

bit 3 **OVRUNIF:** DMA Channel Overrun Flag bit(1)
- 1 = The DMA channel is triggered while it is still completing the operation based the previous trigger
- 0 = The overrun condition has not occurred

bit 2-1 **Unimplemented:** Read as '0'

bit 0 **HALFEN:** Halfway Completion Watermark bit
- 1 = Interrupts are invoked when DMACNTn has reached its halfway point and at completion
- 0 = An interrupt is invoked only at the completion of the transfer

Note 1: Setting these flags in software does not generate an interrupt.

2: Testing for address limit violations (DMASRCn or DMADSTn is either greater than DMAH or less than DMAL) is NOT done before the actual access.

3: Not available on all devices. Refer to the device-specific data sheet for availability.

Register 3-7: DMASRCn: DMA Channel n Source Address 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
SADDR[15:8]							
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
SADDR[7:0]							
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 15-0 **SADDR[15:0]:** Source Address bits

These bits indicate the address location from which the DMAC module initiates the read operation. The SADDR[15:0] bits are dynamically updated based on the SAMODE[1:0] and RELOAD bits.

Register 3-8: DMADSTn: DMA Channel n Destination Address Register(1,2,3,4)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
DADDR[15:8]							
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
DADDR[7:0]							
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 15-0 **DADDR[15:0]:** Destination Address bits

These bits indicate the address location to which the DMAC module initiates the write operation. The DADDR[15:0] bits are dynamically updated based on the DAMODE[1:0] and RELOAD bits.

Note 1: For SFR, the DMAC module operates over the entire the address range, except SFR_ADDRL[10:0] through SFR_ADDRH[10:0]. If the DADDR[15:0] bits fall within SFR_ADDRL[10:0]-SFR_ADDRH[10:0], no bus cycles will be formed.

2: For SRAM, the DMAC module operates over the entire range of the available memory.

3: In Byte mode, DADDR[15:0] bits indicate a byte address when SIZE = 1 and a 16-bit word address when SIZE = 0 (DADDR[0] is ignored).

4: In Microcode mode, DADDR[15:0] bits indicate a 32-bit word address (SADDR[1:0] and SIZE are ignored).

Register 3-9: DMACNTn: DMA Channel n Count 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
CNT[15:8]							
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
CNT[7:0]							
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as ‘0’	
-n = Value at POR	‘1’ = Bit is set	‘0’ = Bit is cleared	x = Bit is unknown

bit 15-0 **CNT[15:0]:** Count bits

These bits indicate the quantity of the remaining data to be transferred by the DMAC module based on the SIZE bit. CNT[15:0] bits are dynamically updated and automatically reloaded in Repeated One-Shot and Repeated Continuous operations. In One-Shot and Continuous operations, CNT[15:0] bits are reloaded based on the RELOAD bit.

4.0 DATA TRANSFER OPTIONS

The DMA Controller transfers data from a source address (DMASRCn) to a destination address (DMADSTn) upon the receipt of a hardware trigger. The transfer occurs as a two-step process: a read from a transfer of data from the source address to the DMA buffer, DMABUF, followed immediately by a write from DMABUF to the destination address. The controller then determines if the operation on this channel has been completed. The active DMA channel may assert an interrupt to indicate the end of a transfer and/or the status of a transfer in progress.

Each channel of the DMA Controller can be independently programmed to move data between the data RAM and peripherals (i.e., the SFR area), between peripherals or between areas of data RAM. Transactions can be single occurrences, repeated single occurrences or continuous, based on an event trigger and/or the channel transaction counter. The source and destination address registers can be independently programmed to increment, decrement or remain unchanged during transactions.

4.1 Data Size

The DMA Controller can handle both byte and word (16-bit) transactions. Each DMA channel is individually configurable for the data size to be used with the SIZE bit (DMACHn[1]). By default (SIZE = 0), the channel is configured for word-sized transactions.

Since data RAM address space is both word-oriented and byte-addressable, byte transfers are accommodated through bit 0 of the address; when bit 0 is '0', the lower byte is addressed, while the upper byte is addressed when bit 0 is '1'.

By default, DMASRCn and DMADSTn maintain bit 0 as '0' to maintain word-aligned addresses and increment address from bit 1. When the byte data size is selected (SIZE = 1), DMASRCn and DMADSTn are incremented or decremented through bit 0.

DMACNTn is also decremented by 1 through bit 0, regardless of the data size. When DMACNTn is being used to track the number of transfers, the SIZE bit also indicates whether bytes or words are being counted.

4.2 Trigger Sources

Each DMA channel can select from up to 128 hardware triggers to initiate a DMA transfer. The trigger sources are generally the device-level interrupts from peripheral modules, as well as the external interrupts and interrupt-on-change sources. The CHSEL[x:0] bits (DMAINTn[x:8]) select which interrupt (and thus module) is used as a trigger for a particular DMA channel. Refer to the specific device data sheet for a specific list of available triggers. The CHSELx bits may be changed, at any time, in order to select another module to service. However, it is not recommended to make a change while the associated DMA channel is in operation.

A DMA channel can be configured to service any memory-mapped peripheral, regardless of the trigger's origin. This is because the trigger (interrupt) source is independent of the DMA source and destination addresses. For example, a DMA channel configured to respond to the INT0 interrupt could be used to move data into or out of a UART FIFO. In most cases, it makes more sense to use a peripheral's own interrupt before performing a data transfer. However, there are also many cases where it is desirable to use one peripheral interrupt to perform a DMA operation on another peripheral, perhaps even another DMA channel. Examples of such operations are provided in **Section 7.0 "Examples of DMA Operations"**.

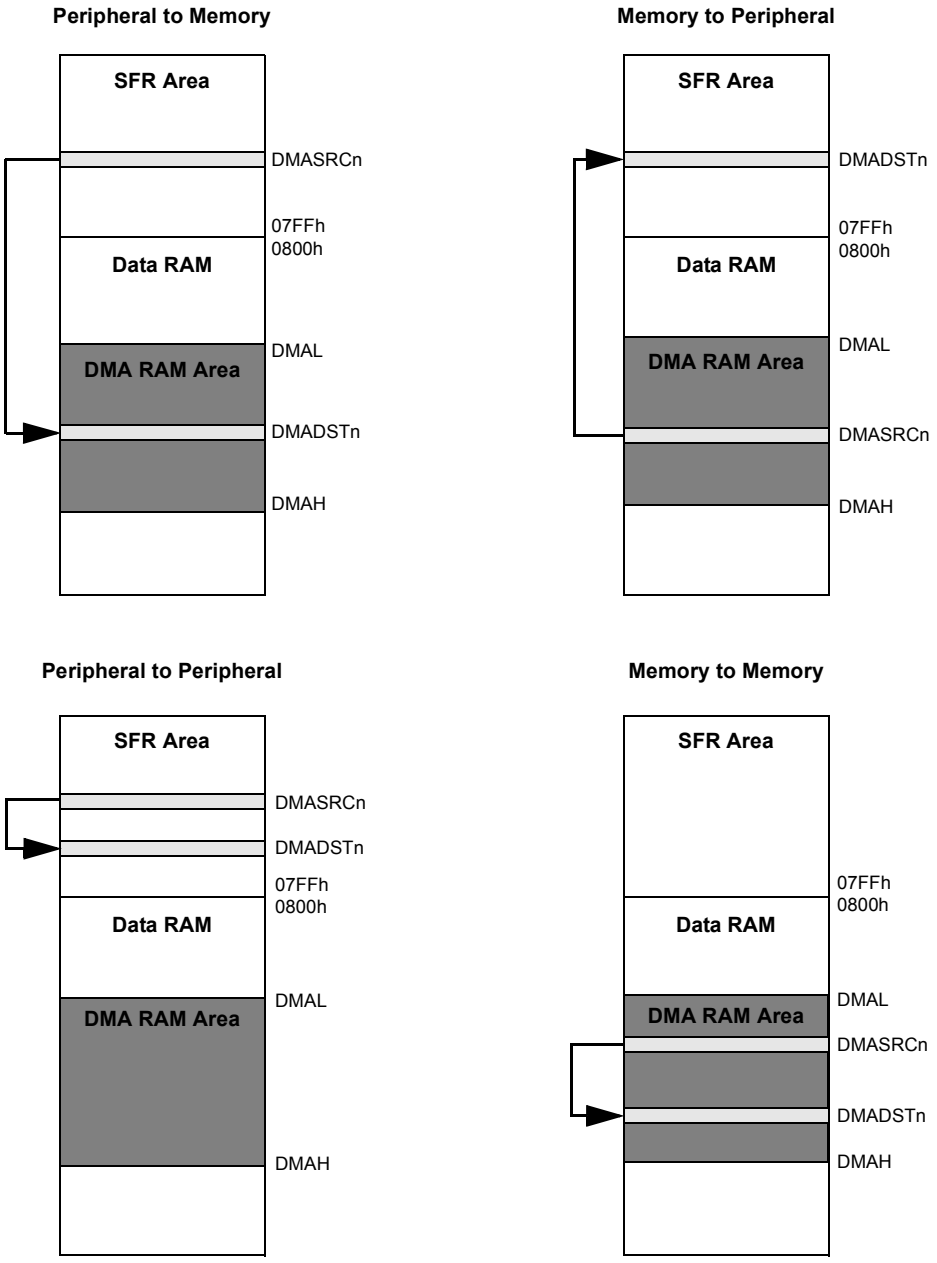
In addition, a DMA channel may be triggered in software by setting the CHREQ bit (DMACHn[8]). This allows for an application to use the DMA Controller to move data directly, without having to wait for a hardware interrupt. CHREQ is also set when a hardware trigger occurs.

4.3 Types of Data Transfers

All DMA transactions occur solely within the data RAM address space. In the least restricted case, all data RAM addresses are available to the DMA Controller; this includes the entire SFR space and (by extension) all peripherals. As defined by the source and destination, there are four types of DMA data transfers (Figure 4-1.):

- Peripheral to Memory (Receive)
- Memory to Peripheral (Transmit)
- Memory to Memory
- Peripheral to Peripheral

Figure 4-1: Types of DMA Data Transfers



Note: Relative sizes of memory areas are not shown to scale.

4.3.1 PERIPHERAL TO MEMORY (RECEIVE)

If a source address register is programmed with an SFR address while the destination register contains a data RAM address, the controller will read from the peripheral module being serviced and write the retrieved data content to the specified location in data RAM. This access variable is most suitable for the peripheral modules configured to receive data, such as a UART or SPI module.

4.3.2 MEMORY TO PERIPHERAL (TRANSMIT)

If the source address register is programmed with a data RAM address, and the destination address register contains a peripheral (SFR) address, the controller is forced to read from the RAM and write to the SFR when triggered. This makes this type of data flow most suitable to support the peripheral modules configured to transmit data, such as a serial communication module.

4.3.3 PERIPHERAL TO PERIPHERAL

Data can also be moved between two peripherals by programming the selected pair of source and destination address registers with the SFR addresses of the data buffers for the peripherals. In this case, data RAM memory bandwidth is not required.

4.3.4 MEMORY TO MEMORY

If data relocation within RAM is required, the source and destination address registers for any channel can simply be programmed with the desired RAM memory locations. Obviously, this type of access does not require access to any peripheral; however, the trigger from any of the peripherals can be used to initiate the transfer.

4.3.5 DMAH AND DMAL REGISTERS

While the 16-bit DMASRCn and DMADSTn registers allow access to the entire data RAM space, there may be circumstances where it is desirable to limit DMA operations to a much narrower range. This may be required for many reasons; for example, to protect program variables or a software stack.

The DMAH and DMAL registers allow the user to set the upper and lower address limits for DMA operations in the data RAM space above the SFR space (i.e., addresses typically greater than 0800h for PIC24 devices and 1000h for dsPIC33 devices). All DMA channels are restricted to the address range set by DMAH and DMAL. Any DMA operations that attempt to access addresses above DMAH will cause a HIGHIF interrupt and terminate the transaction in progress. Similarly, operations that attempt to access below DMAL, but above the end of SFR space, will cause a LOWIF interrupt and terminate the transaction in progress.

4.3.6 BUFFER DATA WRITE BIT

The DBUFWF bit (DMAINTn[15]) indicates whether buffered data in DMABUF has been stored into the specified destination location. It serves as a protection against data loss due to unexpected termination of the active DMA operation. For example, if the user decides to stop the DMA operation that happens to be between load and store, the buffered data in DMABUF will not reach its destination. The user can examine this bit to see if the buffered data still needs to be stored at the specified destination location.

Table 4-1 summarizes the behavior of DBUFWF in various modes of operation.

Table 4-1: Interpretation of the DBUFWF Bit Status

DBUFWF Status	Operation Status		
	Repeated One Shot	Repeated Continuous	Null Write
1	After Loading DMABUF		
0	After Writing to [DMADSTn]		After Writing to [DMASRCn]

4.4 Data Transfer Modes

Data transfers are also defined by how the transaction is structured: the number of data transfers that can occur per trigger event, how events are counted and if the event repeats. The DMA Controller defines four transfer modes, which encompass all of these features:

- One-Shot
- Repeated One-Shot
- Continuous
- Repeated Continuous

The transfer mode is defined by the TRMODEx bits (DMACHn[3:2]). In addition, the RELOAD bit (DMACHn[9]) can modify the behavior of some modes.

4.4.1 COMMON TRANSFER MODE SEQUENCE

Regardless of the transfer mode, all DMA transfers follow the same basic sequence:

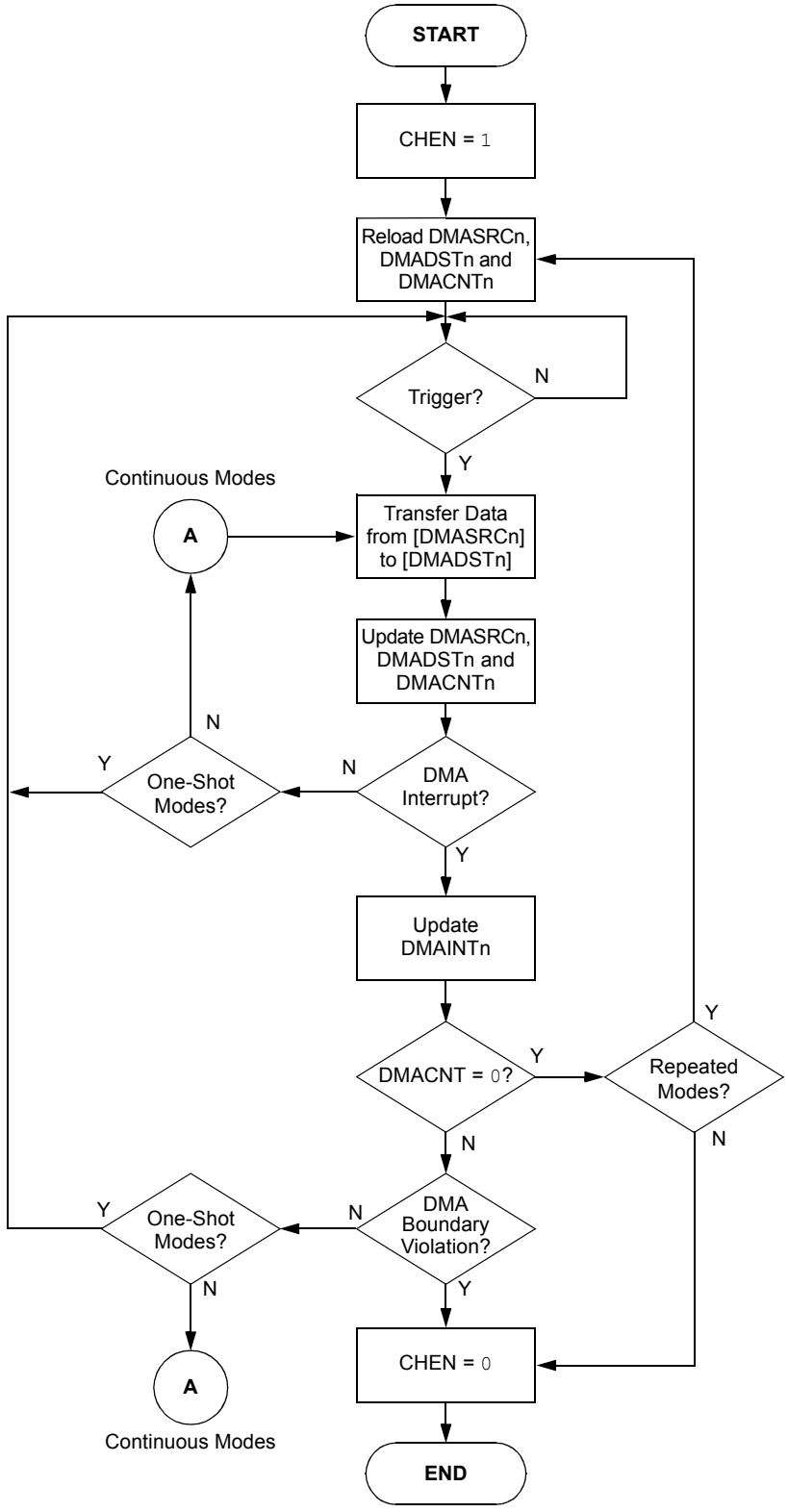
1. Upon the receipt of a DMA trigger or the setting of the CHREQ bit (DMACHn[8]), data is loaded into DMABUF from the location addressed by DMASRCn, then stored in the location addressed by DMADSTn.
2. Following the transaction, DMASRCn and DMADSTn are updated appropriately; one or both may be incremented or decremented, depending on the channel's configuration (see **Section 4.5 "Addressing Modes"** for additional information). At the same time, DMACNTn is decremented by one.
3. The module tests for any DMA interrupt conditions. If an interrupt condition has occurred, the DMAINTn register flags are updated accordingly:
   a) If a DMA interrupt has occurred, all modes continue to Step 4.
   b) If an interrupt has not occurred, all modes return to Step 1. In One-Shot mode, the controller waits for the next trigger. In Continuous mode, the controller repeats the cycle continuously until a DMA interrupt occurs.
4. If DMACNTn has decremented to zero:
   a) The values of DMASRCn, DMADSTn and DMACNTn are reloaded and the sequence repeats from Step 1 (all Repeated modes).
   b) The CHEN bit (DMACHn[0]) is cleared and the channel is disabled (One-Shot and Continuous modes).
5. If DMACNTn has not decremented to zero, the controller checks for a memory address boundary violation of DMAL or DMAH:
   a) If one of the boundaries has been crossed, the CHEN bit is cleared and the channel is disabled.
   b) If there is no boundary violation, the controller returns to Step 1. For both One-Shot modes, the controller waits for the next trigger. For both Continuous modes, the controller proceeds to performing the next data transfer.

The four data transfer modes differ in the number of data transfers than can take place with a single trigger and how the DMACNT register behaves. The common logic flow for all data transfer modes is illustrated in the flow chart in Figure 4-2. The differences between the modes are summarized in Table 4-2.

Table 4-2: Comparison of DMA Data Transfer Modes

Transfer Mode	Transfers per Trigger	DMACNTn Behavior	
		Decrements On	At 0000h
One-Shot	Single	Trigger	Disable channel
Repeated One-Shot		Transfer	Reload and repeat
Continuous	Multiple	Trigger	Disable channel
Repeated Continuous		Transfer	Reload and repeat

Figure 4-2: Common Logic Flow for Data Transfer Modes



Legend:
One-Shot Modes: One-Shot and Repeated One-Shot
Continuous Modes: Continuous and Repeated Continuous
Repeated Modes: Repeated One-Shot and Repeated Continuous

4.4.2 ONE-SHOT MODE

In One-Shot mode (TRMODE[1:0] = 00), a single transfer (from DMASRCn to DMADSTn) is performed for each trigger event. By default, the Reset value of DMACNTn is 0001h. When a single One-Shot transfer occurs, DMACNTn is decremented to 0000h; this disables the channel and requires the channel to be re-enabled to perform the next transaction. Of course, it is also possible to store a larger value in DMACNTn and then conduct a defined number of One-Shot transfers.

Example 4-1 shows a typical code sequence for a One-Shot mode transfer.

Example 4-1: Code for One Shot-Transfer (Memory to Memory)

```
unsigned short int Array1[100];
unsigned short int Array2[100];
int i;
int main()
{
    for (i=0;i<100;i++)
    {
        Array1[i]=i+1;                      //fill with i+1
        Array2[i]=0;                        //fill with 0
    }

    DMACONbits.DMAEN=1;
    DMACONbits.PRSSEL=1;
    DMAH=0x5000;                            //set lower and upper address limit
    DMAL=0x850;
    DMASRC0=(unsigned short int)& Array1;   //load the source address
    DMADST0=(unsigned short int)& Array2;   //load destination address
    DMACNT0=4;                              //Four Transfer to be done
    DMACH0=0;
    DMACH0bits.SAMODE=1;                    //Source address increment mode
    DMACH0bits.DAMODE=1;                    //Destination address increment mode
    DMACH0bits.TRMODE=0;                    //One-Shot Transfer mode
    DMACH0bits.CHEN=1;                      //Enable channel

    IFS0bits.DMA0IF=0;
    DMACH0bits.CHREQ=1;                     //First trigger
    while(DMACH0bits.CHREQ);

    DMACH0bits.CHREQ=1;                     //Second trigger
    while(DMACH0bits.CHREQ);

    DMACH0bits.CHREQ=1;                     //Third trigger
    while(DMACH0bits.CHREQ);                //HALFIF=1 since DMACNT is
                                            //at halfway point

    DMACH0bits.CHREQ=1;                     //Fourth trigger DMACNT=0
                                            //and transfer complete
    while(DMACH0bits.CHREQ);

    while(!IFS0bits.DMA0IF);                //Transfer Complete
                                            //DMA0IF=1, DONEIF=1, CHEN=0;
    IFS0bits.DMA0IF=0;
    while(1);
}
```

4.4.3 REPEATED ONE-SHOT MODE

In Repeated One-Shot mode (TRMODE[1:0] = 01), single transfers occur repeatedly as long as triggers are being provided or CHREQ is set. Each time a trigger occurs or CHREQ is set, DMACNTn is decremented. In this case, however, the channel is not disabled when DMACNTn reaches 0000h. Instead, the original value of DMACNTn is reloaded; if RELOAD = 1, the original values of DMASRCn and DMADSTn are also reloaded. The entire cycle then starts again on the next trigger. To end the sequence, the channel must be disabled by clearing the CHEN bit in software.

Example 4-2 shows a typical code sequence for a Repeated One-Shot mode transfer.

Example 4-2: Code for Repeated One-Shot Transfer (Memory to Memory, RELOAD = 1)

```
unsigned short int Array1[50];
unsigned short int Array2[50];
int i;
void test(void);

int  main()
{
     for (i=0;i<50;i++)
     {
        Array1[i]=i+1;                    //fill with i+1
        Array2[i]=0;                      //fill with 0
     }

     DMACONbits.DMAEN=1;
     DMACONbits.PRSSEL=1;
     DMAH=0x5000;                         //set lower and upper address limit
     DMAL=0x850;
     DMASRC0=(unsigned short int)& Array1;  //load the source address
     DMADST0=(unsigned short int)& Array2;  //load destination address
     DMACNT0=4;
     DMACH0=0;
     DMACH0bits.SAMODE=1;                 //Source address increment mode
     DMACH0bits.DAMODE=1;                 //Destination address increment mode
     DMACH0bits.TRMODE=1;                 //Transfer mode Repeat One-Shot
     DMACH0bits.RELOAD=1;                 //Reload Source/Destination address
     DMACH0bits.CHEN=1;                   //Channel enable

     IFS0bits.DMA0IF=0;
     while(1)
     {
        DMACH0bits.CHREQ=1;               //First trigger
        while(DMACH0bits.CHREQ);

        DMACH0bits.CHREQ=1;               //Second trigger
        while(DMACH0bits.CHREQ)

        DMACH0bits.CHREQ=1;               //Third trigger
        while(DMACH0bits.CHREQ);          //HALFIF=1 since DMACNT is in
                                          //half way point
        DMACH0bits.CHREQ=1;               //Fourth trigger DMACNT=0
                                          //and transfer complete
        while(DMACH0bits.CHREQ);          //DMACNT reloaded to 4, DMA0IF=1,
                                          //Since RELOAD=1, DMASRC0/DMADST0
                                          //are reloaded

        while(!IFS0bits.DMA0IF);
        DMAINT0bits.DONEIF=0;             //Clear DONEIF and HALIF flag
        DMAINT0bits.HALFIF=0;
        IFS0bits.DMA0IF=0;
     }
}
```

4.4.4 CONTINUOUS MODE

In Continuous mode (TRMODE[1:0] = 10), a single trigger starts a sequence of back-to-back transfers; these continue with each transfer decrementing DMACNTn until it reaches 0000h. At this point, as in One-Shot mode, the channel is disabled.

One-Shot and Continuous modes are similar in that each mode performs a certain number of transfers for one time. The difference is that One-Shot mode requires a trigger for each transfer, while Continuous mode allows many transfers for each trigger. In addition, DMACNTn is controlled by the number of individual transactions, not the number of triggers.

Example 4-3 shows a typical code sequence for a Continuous mode transfer.

Example 4-3: Code for Continuous Transfer

```
unsigned short int Array1[100];
unsigned short int Array2[100];
int  i;
void test(void);

int  main()
{
     for (i=0;i<100;i++)
     {
        Array1[i]=i+1;                      //fill with i+1
        Array2[i]=0;                        //fill with 0
     }

     DMACONbits.DMAEN=1;
     DMACONbits.PRSSEL=1;
     DMAH=0x5000;                           //set lower and upper address limit
     DMAL=0x850;
     DMASRC0=(unsigned short int)& Array1;  // load the source address
     DMADST0=(unsigned short int)& Array2;  // load destination address
     DMACNT0=100;
     DMACH0=0;
     DMACH0bits.SAMODE=1;                   //Source address increment mode
     DMACH0bits.DAMODE=1;                   //Destination address increment mode
     DMACH0bits.TRMODE=2;                   //Transfer mode Continous
     DMACH0bits.CHEN=1;                     //Channel enable

     IFS0bits.DMA0IF=0;
     DMACH0bits.CHREQ=1;                    //Enable the transfter by software trigger

     while(!DMAINT0bits.HALFIF);            //HALFIF=1 is set when
                                            //DMACNT0 reaches halfway
     while(!IFS0bits.DMA0IF);               //DONEIF=1;CHEN=0,DMA0IF=1
                                            //and 100 (DMACNTx=100)transfer complete
                                            //with one trigger
     Nop();
     Nop();
     Nop();
     IFS0bits.DMA0IF=0;
     while(1);
}
```

4.4.5 REPEATED CONTINUOUS MODE

Repeated Continuous mode (TRMODE[1:0] = 11) can be thought of as a combination of Continuous and Repeated One-Shot modes; data transfers keep occurring as long as triggers are provided and multiple transfers can occur with each trigger. As in Continuous mode, each transfer decrements DMACNTn. When it reaches 0000h, the address and count registers are reloaded, and the process is repeated.

Like Continuous mode, ending the sequence requires disabling the channel, either by disabling the trigger source or clearing the CHEN bit in software.

Example 4-4 shows a typical code sequence for a Repeated Continuous mode transfer.

Example 4-4: Code for Continuous Transfer (Memory to Memory, RELOAD = 1)

```
unsigned short int Array1[100];
unsigned short int Array2[100];
int i;

int main()
{
    ANSA=0;
    TRISA=0;
    LATA=0x70;

    for (i=0;i<100;i++)
    {
        Array1[i]=i+1;                      //fill with i+1
        Array2[i]=0;                        //fill with 0
    }

    DMACONbits.DMAEN=1;
    DMACONbits.PRSSEL=1;
    DMAH=0x5000;                            //set lower and upper address limit
    DMAL=0x850;
    DMASRC0=(unsigned short int)& Array1;   // load the source address
    DMADST0=(unsigned short int)& Array2;   // load destination address
    DMACNT0=100;                            // 100 Transaction per trigger
    DMACH0=0;
    DMACH0bits.SAMODE=1;                    //Source address increment mode
    DMACH0bits.DAMODE=1;                    //Destination address increment mode
    DMACH0bits.TRMODE=3;                    //Transfer mode Repeat continous
    DMACH0bits.RELOAD=1;                    //Reload Source and Destination Address
    DMACH0bits.CHEN=1;                      //Channel enable

    IFS0bits.DMA0IF=0;
    while(1)
    {
        DMACH0bits.CHREQ=1;                 //TIGGER
        while(!IFS0bits.DMA0IF);
        for (i=0;i<100;i++)                 //Clear the Destination Memory
        {
            Array2[i]=0;                    //fill with 0
        }
        IFS0bits.DMA0IF=0;
        PORTAbits.RA0=0;
        DMAINT0bits.DONEIF=0;
        DMAINT0bits.HALFIF=0;
    }
}
```

4.4.6 ADDRESS AND COUNT RELOAD

Although the Repeated modes explicitly include it, all of the Transfer modes allow the automatic re-use of the initial source and destination addresses, and transaction counts for multiple operations. Setting the RELOAD bit (DMACHn[9]) allows the values of DMASRCn, DMADSTn and DMACNTn to be restored for the next DMA operation. This causes the registers to be reloaded in One-Shot and Continuous modes, after a transfer operation is complete, and the channel is re-enabled. Address and transaction count reloading is automatic in Repeated One-Shot and Repeated Continuous modes.

DMACNTn also has its value reloaded after it has been decremented to 0000h, regardless of the setting of the RELOAD or TRMODEx bits. The only exception is if the channel is stopped in mid-operation and restarted later.

Table 4-3 shows the effect of RELOAD on DMASRCn, DMADSTn and DMACNTn for the Data Transfer modes.

Table 4-3: RELOAD Bit and Data Transfer Modes

RELOAD bit	DMACHn[2]	DMASRCn	DMADSTn	DMACNTn
1	x	Reloaded	Reloaded	Reloaded
0	0	Not Reloaded	Not Reloaded	Reloaded(1)
0	0	Not Reloaded	Not Reloaded	Not Reloaded

Note 1: The reload only happens after DMACNTn has decremented to zero. No reload occurs if the channel is stopped and later resumed.

4.5 Addressing Modes

Following each transfer, the DMASRCn and DMADSTn registers may be automatically updated by the channel. This potentially allows the channel to move data between multiple locations without the need for user intervention. Automatic address updating is controlled by the SAMODE[1:0] and DAMODE[1:0] bits (DMACHn[7:6] and [5:4]).

The combination of the different address update options (fixed, increment, decrement or external index) provide five supported addressing modes:

- Fixed to Fixed
- Fixed to Block
- Block to Fixed
- Block to Block
- Peripheral Indirect Addressing (select devices only)

Table 4-4 shows the address modes and the various SAMODEx and DAMODEx combinations.

Table 4-4: Configurations for DMA Addressing Modes

Mode	SAMODE[1:0]	DAMODE[1:0]
Fixed to Fixed	00	00
Fixed to Block (Address Increment)(1)	00	01
Fixed to Block (Address Decrement)(1)	00	10
Block to Fixed (Address Increment)(1)	01	00
Block to Fixed (Address Decrement)(1)	10	00
Block to Block (Address Increment)(1)	01	01
Block to Block (Address Decrement)(1)	10	10
Peripheral Index Addressing	11	11

Note 1: The increment and decrement of the address will be based on the SIZE bit value.

4.5.1 FIXED TO FIXED

Fixed to Fixed mode is set by configuring the SAMODE[1:0] and DAMODE[1:0] bits to '00'. In this mode, the source and destination address remains the same after each transaction. This mode is suited for One-Shot mode transfers of a single byte, or word of data, between two fixed addresses.

4.5.2 FIXED TO BLOCK

In Fixed to Block mode, the source address remains unchanged throughout the transfer, but the destination address is incremented or decremented (depending on the DAMODEx setting). This works well for receiving data from the single-word buffer of a serial communication peripheral, and filling a block of addresses designated as a buffer.

Example 4-5 (page 24) shows sample code for Fixed to Block Addressing. The source address (UART receive) interrupts when the UART buffer is full with four bytes; the UART Receive Interrupt Flag (U2RIF) will trigger the transfer.

4.5.3 BLOCK TO FIXED

In Block to Fixed mode, the source address is incremented or decremented throughout the transfer (depending on the SAMODEx setting) while the destination address remains unchanged. This is well-suited for moving a packet of data to be transmitted into the single-word transmit buffer of a serial communication peripheral.

4.5.4 BLOCK TO BLOCK

In Block to Block mode, both the source and destination addresses increment or decrement (depending on the SAMODEx and DAMODEx setting) throughout the transfer. This mode is useful for copying a block of data from one part of the data RAM to another.

Example 4-5: Code for Fixed to Block Continuous Transfer (Peripheral to Memory)

```
void UartInit(void);
unsigned char Array2[100];
int  i;
void test(void);

int  main()
{
     UartInit();
     for (i=0;i<100;i++)
     {
        Array2[i]=0;                        //fill with 0
     }
     DMACONbits.DMAEN=1;
     DMACONbits.PRSSEL=1;
     DMAH=0x5000;                           //set lower and upper address limit
     DMAL=0x850;
     DMASRC0=(unsigned int)&U2RXREG;
     DMADST0=(unsigned short int)& Array2;  // load destination address
     DMACNT0=4;                             //When the Uart buffer is full 4,
                                            //do a interrupt and transfer 4 bytes
     DMACH0=0;
     DMACH0bits.BYTE=1;
     DMACH0bits.SAMODE=0;                   //Source address increment mode,
                                            //do not increment
     DMACH0bits.DAMODE=1;                   //Destination address increment mode,

                                            //increment 1
     DMACH0bits.TRMODE=2;                   //Transfer mode Continous
     DMAINT0bits.CHSEL=33;                  //Trigger on UART2 Receive
     DMACH0bits.CHEN=1;                     //Channel enable
     IFS0bits.DMA0IF=0;
     while(!IFS0bits.DMA0IF);               //DONEIF=1;CHEN=0,DMA0IF=1 and
                                            //transfer complete with one trigger
     IFS0bits.DMA0IF=0;
     while(1);
}
void UartInit(void)
{
     TRISFbits.TRISF5=0;
     TRISFbits.TRISF4=1;
     __builtin_write_OSCCONL(OSCCON & 0xBF);
     RPINR19bits.U2RXR = 10;                //RF4 U2RX
     RPOR8bits.RP17R = 5;                   //RF4 T2TX
     __builtin_write_OSCCONL(OSCCON | 0x40);
     U2BRG = 100;                           //BAUDRATEREG2;
     U2MODE = 0;
     U2MODEbits.BRGH = 1;
     U2STA = 0;
     U2STAbits.URXISEL=3;                   //Interrupt after 4 transfers
     U2MODEbits.UARTEN = 1;
     U2MODEbits.PDSEL1= 0;
     U2MODEbits.PDSEL0= 0;
     U2STAbits.UTXEN = 1;
     IFS1bits.U2RXIF = 0;
}
```

4.5.5 PERIPHERAL INDIRECT ADDRESSING

Peripheral Indirect Addressing (PIA) is a special Auto-Incrementing mode for the transfer of data, to and from, a multilevel peripheral buffer. This mode is only available for specific peripherals designed with its use in mind. Refer to the specific device data sheet to verify if it possesses PIA-capable peripherals.

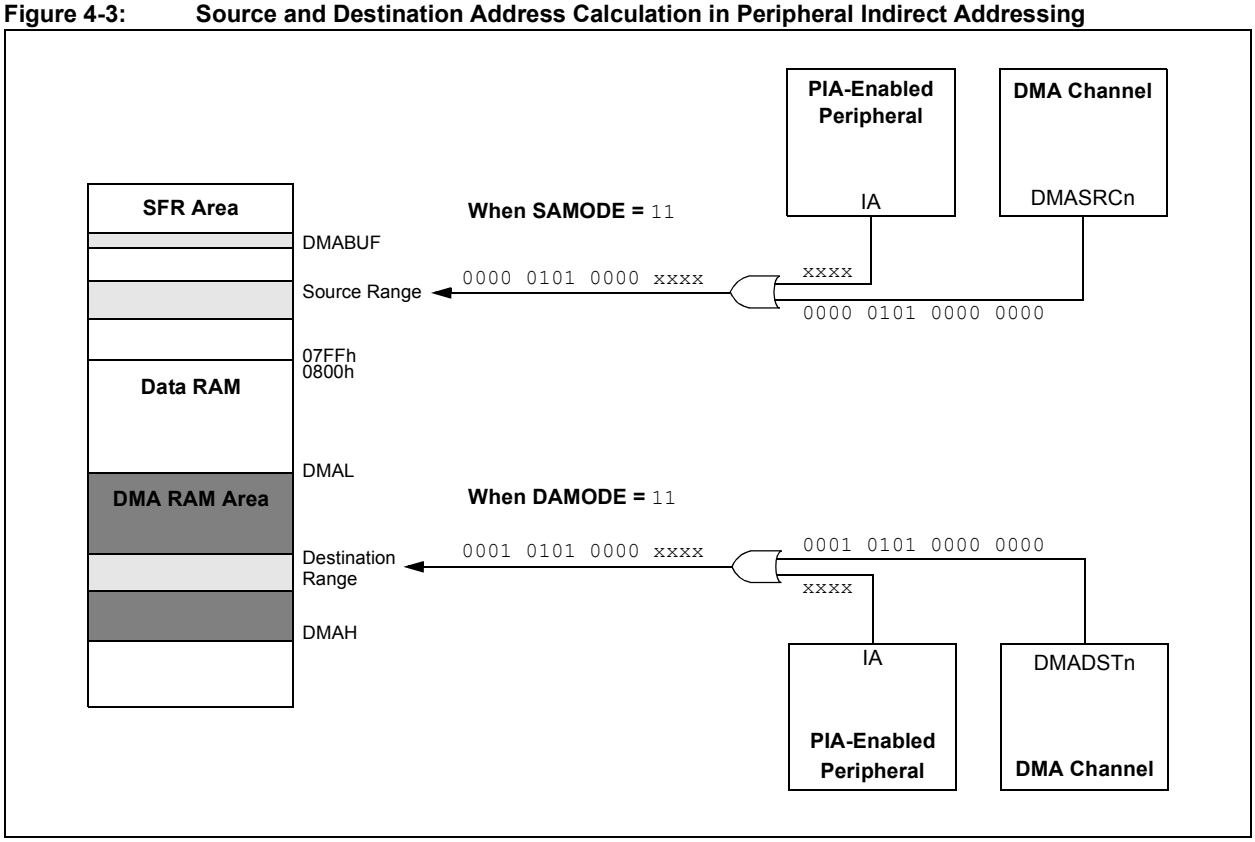
PIA mode is enabled when either SAMODEx or DAMODEx are set to '11'. When selected, the PIA-enabled peripheral generates a short Indirect Address (IA) (size defined by the peripheral) to the DMA channel. The IA is logically ORed with either the contents of DMASRCn and/or DMADSTn to define a specific address for the peripheral inside the DMA address space. If SAMODE[1:0] = 11, DMASRCn is used as the base address for the actual source address in the transfer. Similarly, if DAMODE[1:0] = 11, DMADSTn is used as the base destination address.

The peripheral manages the IA, incrementing or decrementing it as the peripheral directs. This generates a limited size, and generally circular, buffer within the DMA address space. For example, a 4-bit IA provided by the peripheral will be able to access a 16-byte buffer, starting at the address defined by the contents of DMASRCn or DMADSTn. Buffers, therefore, cannot cross the boundaries by the DMASRCn or DMADSTn register and the maximum PIA value, and will always wrap accordingly.

Note that the actual indirect source and destination addresses to be read from and written to are not stored in DMASRCn or DMADSTn, or in any other register. To ensure the full range of address values, the user must be sure the Least Significant bits (LSb) of the source or destination registers corresponding to the PIA are always '0's.

Figure 4-3 shows how PIA mode uses DMASRCn and DMADSTn to create the actual source and destination addresses. In this instance, DMASRCn contains a base source address of 0500h, while DMADSTn contains a base address of 1500h. Note that these addresses are arbitrary; in real applications, PIA mode permits any type of data transfer between peripherals and memory.

Figure 4-3: Source and Destination Address Calculation in Peripheral Indirect Addressing



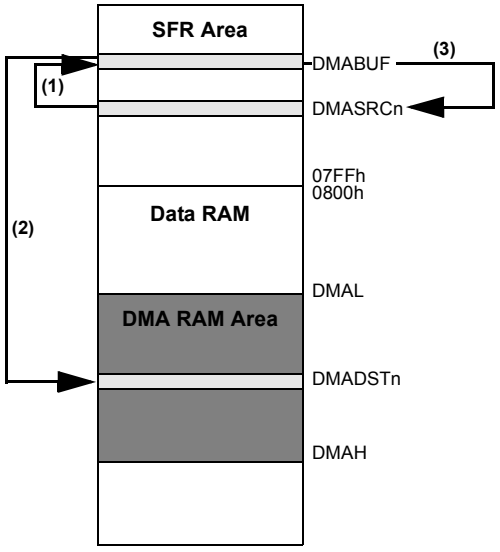
4.6 Null Write Mode

By default, DMA transfers operate in only one direction: from the source address to the destination address. However, some communication protocols require symmetrical buffer accesses; that is, for every read operation performed on a buffer, there must be an accompanying write operation. An example of this requirement occurs with the SPI module operating in Master mode.

The Null Write mode is designed to satisfy this requirement. This mode works by transferring data from the address in DMASRCn to the address in DMADSTn, like any other DMA operation. Once this is done, however, the transferred data that is still stored in DMABUF is written back to the address specified by DMASRCn. The write-back occurs before the DMA proceeds to its next transfer. A typical example of this is shown in Figure 4-4.

Null Write mode is enabled by setting the NULLW bit (DMACHn[10]).

Figure 4-4: Null Write Mode Transfer (Peripheral to Memory Example)



1. Data at address designated by DMASRCn is transferred into DMABUF.
2. Data in DMABUF is copied to address designated by DMADSTn.
3. The same data in DMABUF is also written back to the address designated by DMASRCn.

5.0 CHANNEL PRIORITY AND PRIORITY SCHEMES

While DMA channels can function independently to service different peripherals at the same time, they are still limited by the presence of a single DMA data bus and a single data channel to RAM. When two or more channels request the DMA Controller to handle a data transfer at the same time, the controller arbitrates the requests and decides which channel receives priority. The controller uses two defined arbitration schemes to assign channel priority: Fixed and Round-Robin. The PRSSEL bit (DMACON[0]) determines the scheme to be used.

In the Round-Robin scheme (DMACON[0] = 1), the controller starts by giving Channel 0 preference in any data transfer conflicts. For each successive transfer conflict, the next higher channel receives preference, continuing as a cycle through all the channels. If the channel that has priority does not make a request at that time, it is skipped for the next channel in the cycle.

As an example, if Channels 0, 1 and 2 all simultaneously request a data transfer, Channel 0 is serviced; Channels 1 and 2 are then serviced in that order. During the next service request, any request from Channel 1 will receive preference; Channel 2 will receive preference in the following round. Any subsequent transfer requests from Channel 0 will be ignored until all the other channels have received priority once. Typical examples of Round-Robin arbitration are shown in Table 5-1.

Table 5-1: Examples of Channel Access Using Round-Robin Priority Scheme

Requesting DMA Channel(s)				Channel Granted Priority
0	1	2	3	
				None
	X			CH1
X	X	X		CH2
X	X			CH0
X	X			CH1
X	X		X	CH3
X	X			CH0

In contrast, the Fixed scheme (DMACON[0] = 0) always gives priority to the lowest requesting channel number. Using the previous example, if there are several sequential transfer requests involving Channel 0, Channel 0 will always receive preference over other channels. The Fixed priority scheme is the default. Typical examples are shown in Table 5-2.

Table 5-2: Examples of Channel Access Using Fixed Priority Scheme

Requesting DMA Channel(s)				Channel Granted Priority
0	1	2	3	
				None
	X			CH1
X	X	X		CH0
X	X			CH0
	X			CH1
	X		X	CH1
			X	CH3

6.0 DMA INTERRUPTS

Each DMA channel has its own set of five interrupt flags, used to indicate a range of conditions during and following data transfers. Setting any of these flags with an interrupt event causes the device-level DMA Channel Interrupt Flag (DMAnIF) to be set. With one exception, these flags are always enabled and not configurable. The DMAnIE bits, located in the IECx Interrupt registers, will determine if a device-level interrupt is actually generated.

Since any of the DMA channel's individual event flags can trigger a device-level interrupt for the channel, the user must include a method within the ISR to determine which flag triggered the interrupt.

6.1 DMA Completion Interrupt

The DONEIF bit (DMAINTn[5]) indicates the completion status of the last DMA operation. It is automatically set when DMACNTn decrements to 0000h in the course of a One-Shot or Continuous mode DMA transaction.

By also examining the corresponding CHEN bit (DMACHn[0]), it is possible to gain additional information on the status of the previous and current transactions.The possible interpretations are shown in Table 6-1.

Note that DONEIF remains cleared (= 0) when any Repeated Transfer modes are being used. This is because the address registers and transaction counters automatically reload, and the transaction automatically repeats when DMACNTn decrements to 0000h. Repeated mode transfers must be terminated in software by clearing CHREQ.

Table 6-1: DMA Transaction Status

Bit Status		DMA Transaction Status
DONEIF	CHEN	
0	0	Previous transaction ended without completion.
0	1	Current transaction is not yet complete.
1	0	Previous transaction ended without completion.
1	1	Previous transaction ended with completion.

6.2 DMA Halfway Point Interrupt

The HALFIF interrupt flag (DMAINTn[4]) is an optional interrupt that indicates that the DMACNTn register is at the halfway point between its original programmed value and 0000h. This can be used with the DONEIF interrupt to monitor the progress of the DMA transfer.

When enabled, HALFIF is set only when DMACNTn reaches the halfway mark, but not thereafter. This results in a non-persistent interrupt. In Repeated modes, the DMA Controller attempts to set HALFIF every time DMACNTn reaches the halfway point, whether or not the bit has been cleared. It is the user's responsibility to clear the bit after it has been set.

Unlike the other DMA interrupts, the HALFIF interrupt must be enabled. The HALFEN bit (DMAINTn[0]) enables the halfway point interrupt.

6.3 Overrun Interrupt

When a DMA channel receives a trigger while its CHREQ bit is already set (either by software or another hardware trigger), an overrun condition occurs. This condition indicates that the channel is being requested before its current transaction is finished. This implies that the active channel may not be able to keep up with the demands from the peripheral module being serviced, which may result in data loss. An overrun condition causes the OVRUNIF flag (DMAINTn[3]) to be set.

Note that the OVRUNIF flag being set does not cause the current DMA operation to terminate. Therefore, the channel for which OVRUNIF is set does not need to be the active channel.

Setting the priority scheme correctly also helps to avoid overrun errors. For example, if one of the channels operates more frequently, a fixed priority scheme with that as the channel will help to reduce overrun interrupts.

6.4 DMA Address Limit Interrupts

The HIGHIF and LOWIF flags (DMAINTn[7:6]) indicate that a DMA operation has crossed the data RAM address boundaries set by the DMAH and DMAL registers. The flag is set on any operation that attempts to read data from, or write data to, an address outside of the DMA boundaries. An address limit interrupt also immediately terminates any DMA transaction in progress.

7.0 EXAMPLES OF DMA OPERATIONS

7.1 Basic Setup

To set up a DMA channel for any data transfer:

1. Enable the DMA Controller (DMAEN = 1) and select an appropriate channel priority scheme by setting or clearing PRSSEL.
2. Program DMAH and DMAL with appropriate upper and lower address boundaries for data RAM operations.
3. Select the DMA channel to be used and disable its operation (CHEN = 0).
4. Program the appropriate source and destination addresses for the transaction into the channel's DMASRCn and DMADSTn registers. For PIA Addressing mode, use the base address value.
5. Program the DMACNTn register for the number of triggers per transfer (One-Shot or Continuous modes) or the number of words (bytes) to be transferred (Repeated modes).
6. Set or clear the SIZE bit to select the data size.
7. Program the TRMODEx bits to select the data transfer mode.
8. Program the SAMODEx and DAMODEx bits to select the addressing mode.
9. Enable the DMA channel by setting CHEN.
10. Enable the trigger source interrupt.

7.2 Standard Operation (Data Transfer)

A basic example of using the DMA Controller is moving a constant stream of data from a serial communication channel, such as a UART, and buffering it in a location in data RAM until the CPU can process it. In this example, DMA0 is used to service the UART. It is configured as follows:

- DMA0 is configured to use the UART's receive interrupt as a trigger.
- DMA0 is programmed to use a single source address; to auto-increment the destination address; and to use Repeated One-Shot Data Transfer mode.
- DMASRC0 is programmed with the address of the UART's receive buffer; DMADST0 is programmed with an address in data RAM.
- DMACNT0 is programmed with 0015h.

In this configuration, the sequence of events is as follows:

1. When the UART triggers a receive interrupt, DMA0 transfers the data from the buffer to a data RAM location.
2. After the transfer, the destination address is incremented.
3. After 16 interrupts, DMACNT0 is decremented to 0000h. Because this is a Repeated mode transfer, the original values of DMADST0 and DMACNT0 are reloaded, and the cycle repeats.

This process allows the CPU to perform other tasks than buffering incoming serial data and processes the data when it has the time. Because the DMA is overwriting the same 16 locations in memory, it is assumed that the CPU will be able to retrieve the fresh data first.

7.3 Nested Operation (Wait State Generation)

DMA channels may be nested, using one channel to trigger another in performing a data transfer. When one of the microcontroller's general purpose timers is included, it becomes possible to generate a fixed delay between a service request and the data transfer.

In this case, DMA0 and DMA1 are used to service a UART after a forced Wait state. DMA Channels 0 and 1 are preconfigured as follows:

- DMA Channel 0 is configured to use the UART's receive interrupt as a trigger.
- DMASRC0 is programmed with an address in data RAM; DMADST0 is programmed with the address of the T0CON register.
- DMA Channel 1 is configured to use Timer0's interrupt as a trigger.
- DMASRC1 is programmed with the address of the UART's receive buffer; DMADST0 is programmed for the address of a destination in data RAM.

The sequence of events is as follows:

1. When the UART sends an interrupt, DMA0 transfers data into Timer0's control register.
2. This causes Timer0 to count down once for a fixed interval (the Wait state), then generate an interrupt.
3. When Timer0 sends its interrupt, DMA1 is triggered, and transfers data from the UART to data RAM.

Note that in this case, neither DMA channel was servicing the module from which it received its trigger.

7.4 Cascaded Operation (SPI Duplex Servicing)

Another method is to cascade two DMA channels together, allowing one to perform part of a function and then trigger a second channel to perform the other part. A good example is an SPI module operating in Slave mode. Using two cascaded DMA channels allows automatic duplex operation, alternately receiving and sending data without the CPU's intervention.

DMA0 (the Read Channel) and DMA1 (the Write Channel) are configured as follows:

- DMA0 is configured to use the SPI's transfer interrupt as the trigger; for Repeated One-Shot mode transfers; for a fixed source address and a fixed destination address
- DMASCR0 is programmed with the address of SPIBUF, while DMADST0 is programmed with a destination address in data RAM
- DMACNT0 is programmed with 0001h (its default)
- DMA1 is configured to use the DMA0 interrupt as the trigger; for Repeated One-Shot mode transfers; and for fixed source and destination addresses
- DMASCR1 is programmed with a data source address in data RAM, while DMADST1 is programmed with the address of SPIBUF
- DMACNT1 is programmed with 0001h

The sequence of events is as follows:

1. When the SPI receives data, it causes an SPI transfer interrupt.
2. This triggers DMA0 to transfer the data from the SPI buffer into RAM; at the completion of the transfer, the DMA0 interrupt is triggered.
3. The DMA0 interrupt triggers DMA1 to move data out of the data RAM location into SPIBUF to be transmitted. The process ends at this point.

For simplicity, this example moves one word of data in and out of the SPI. By changing SAMODEx and DAMODEx for DMA0 and DMA1, respectively, and using different values for DMACNTn, it is also possible to create mutliword buffers for larger duplex transactions.

8.0 OPERATION DURING SLEEP AND IDLE MODES

Although the DMA Controller can be thought of as an extension of the CPU, it is treated as a peripheral when it comes to power-saving operations. Like other peripherals, the DMA Controller also uses Peripheral Module Disable (PMD) bits to further tailor its operation in low-power states.

8.1 Idle Mode

The DMA Controller does not support operation in Idle mode. Transactions in progress when Idle mode is invoked will be aborted.

If use of the DMA Controller is not optional, alternate strategies to reduce power consumption are available. For example, executing NOP instructions while the DMA Controller transfers data effectively powers down the program memory array, allowing for a significant power reduction. Other strategies may be available.

8.2 Sleep Mode

When the device enters Sleep mode, all clock sources to the module are shut down and stay at logic '0'. Any transfers in progress are aborted. The controller will not resume any partially completed transactions on exiting from Sleep mode.

Register contents are not affected by the device entering or leaving Sleep mode. It is recommended that DMA transactions be allowed to finish before entering Sleep mode.

8.3 Deep Sleep Modes

When the device enters any of the Deep Sleep modes, all clock sources to the DMA Controller are shut down. Any transfers in progress when a Deep Sleep mode is invoked are aborted. The controller will not resume any partially completed transactions on exiting from Deep Sleep mode.

In addition, register contents are affected when the device enters or exits Deep Sleep mode. The DMA Controller will need to be re-enabled.

It is recommended that DMA transactions be allowed to finish before entering any Deep Sleep mode.

8.4 V_{BAT} Modes

The DMA Controller does not function whenever there is a loss of V_{DD}; this includes V_{BAT} modes. Any transfers in progress when V_{DD} is lost are aborted.

In addition, register contents are affected whenever V_{DD} is lost. The DMA Controller will need to be re-enabled.

8.5 Peripheral Module Disable (PMD) Register

The Peripheral Module Disable (PMD) registers provide a method to disable DMA channels by stopping all clock sources supplied to that channel. On some devices, a single DMA PMD bit controls a block of up to four channels.

When all channels are disabled via their corresponding PMD control bits, the DMA Controller is in a minimum power consumption state. The module-level registers (DMACON, DMABUF, DMAH and DMAL) remain active. However, the control and status registers associated with any disabled channels will be disabled, so writes to those registers will have no effect and read values will be invalid.

9.0 EFFECTS OF A RESET

A device Reset forces all registers to their Reset state. This forces the DMA Controller and all channels to be turned off and any transfers in progress to be aborted. All buffer and address registers are initialized to 0000h.

10.0 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 dsPIC33/PIC24 families, but the concepts are pertinent, and could be used with modification and possible limitations. The current application notes related to the Direct Memory Access Controller (DMA) are:

Title	Application Note #
None.	

Note: Please visit the Microchip website (www.microchip.com) for additional Application Notes and code examples for the dsPIC33/PIC24 families of devices.

11.0 REVISION HISTORY

Revision A (February 2011)

Original version of this chapter.

Revision B (October 2012)

Adds code examples Example 1-1 through Example 1-4 to demonstrate each of the transfer modes.

Adds Example 1-5 to demonstrate Fixed to Block Addressing.

Other minor typographic corrections.

Revision C (January 2019)

Updates **Section 1.0 "Introduction"**, **Section 4.0 "Data Transfer Options"**, **Section 4.2 "Trigger Sources"** and **Section 4.3.5 "DMAH and DMAL Registers"**.

Updates Figure 1-1: DMA Functional Block Diagram.

Updates Register 3-6: DMAINTn: DMA Channel n Interrupt Control Register

Updates Table 3-1: DMA Register Map.

Moves the Register Summary table to the beginning of the Registers section.

Removes the Module Registers and Channel Registers sections.

Adds Register 3-2, Register 3-3, Register 3-4, Register 3-7, Register 3-8 and Register 3-9.

Other minor typographic corrections.

NOTES:

Note the following details of the code protection feature on Microchip devices:

- Microchip products meet the specification contained in their particular Microchip Data Sheet.
- Microchip believes that its family of products is one of the most secure families of its kind on the market today, when used in the intended manner and under normal conditions.
- There are dishonest and possibly illegal methods used to breach the code protection feature. All of these methods, to our knowledge, require using the Microchip products in a manner outside the operating specifications contained in Microchip's Data Sheets. Most likely, the person doing so is engaged in theft of intellectual property.
- Microchip is willing to work with the customer who is concerned about the integrity of their code.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of their code. Code protection does not mean that we are guaranteeing the product as "unbreakable."

Code protection is constantly evolving. We at Microchip are committed to continuously improving the code protection features of our products. Attempts to break Microchip's code protection feature may be a violation of the Digital Millennium Copyright Act. If such acts allow unauthorized access to your software or other copyrighted work, you may have a right to sue for relief under that Act.

Microchip received ISO/TS-16949:2009 certification for its worldwide headquarters, design and wafer fabrication facilities in Chandler and Tempe, Arizona; Gresham, Oregon and design centers in California and India. The Company's quality system processes and procedures are for its PIC® MCUs and dsPIC® DSCs, KEELOQ® code hopping devices, Serial EEPROMs, microperipherals, nonvolatile memory and analog products. In addition, Microchip's quality system for the design and manufacture of development systems is ISO 9001:2000 certified.

QUALITY MANAGEMENT SYSTEM
CERTIFIED BY DNV
═ ISO/TS 16949 ═

Trademarks

ISBN: 978-1-5224-4091-8

Worldwide Sales and Service

AMERICAS

Corporate Office
2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 480-792-7200
Fax: 480-792-7277
Technical Support:
http://www.microchip.com/support
Web Address:
www.microchip.com

Atlanta
Duluth, GA
Tel: 678-957-9614
Fax: 678-957-1455

Austin, TX
Tel: 512-257-3370

Boston
Westborough, MA
Tel: 774-760-0087
Fax: 774-760-0088

Chicago
Itasca, IL
Tel: 630-285-0071
Fax: 630-285-0075

Dallas
Addison, TX
Tel: 972-818-7423
Fax: 972-818-2924

Detroit
Novi, MI
Tel: 248-848-4000

Houston, TX
Tel: 281-894-5983

Indianapolis
Noblesville, IN
Tel: 317-773-8323
Fax: 317-773-5453
Tel: 317-536-2380

Los Angeles
Mission Viejo, CA
Tel: 949-462-9523
Fax: 949-462-9608
Tel: 951-273-7800

Raleigh, NC
Tel: 919-844-7510

New York, NY
Tel: 631-435-6000

San Jose, CA
Tel: 408-735-9110
Tel: 408-436-4270

Canada - Toronto
Tel: 905-695-1980
Fax: 905-695-2078

ASIA/PACIFIC

Australia - Sydney
Tel: 61-2-9868-6733

China - Beijing
Tel: 86-10-8569-7000

China - Chengdu
Tel: 86-28-8665-5511

China - Chongqing
Tel: 86-23-8980-9588

China - Dongguan
Tel: 86-769-8702-9880

China - Guangzhou
Tel: 86-20-8755-8029

China - Hangzhou
Tel: 86-571-8792-8115

China - Hong Kong SAR
Tel: 852-2943-5100

China - Nanjing
Tel: 86-25-8473-2460

China - Qingdao
Tel: 86-532-8502-7355

China - Shanghai
Tel: 86-21-3326-8000

China - Shenyang
Tel: 86-24-2334-2829

China - Shenzhen
Tel: 86-755-8864-2200

China - Suzhou
Tel: 86-186-6233-1526

China - Wuhan
Tel: 86-27-5980-5300

China - Xian
Tel: 86-29-8833-7252

China - Xiamen
Tel: 86-592-2388138

China - Zhuhai
Tel: 86-756-3210040

ASIA/PACIFIC

India - Bangalore
Tel: 91-80-3090-4444

India - New Delhi
Tel: 91-11-4160-8631

India - Pune
Tel: 91-20-4121-0141

Japan - Osaka
Tel: 81-6-6152-7160

Japan - Tokyo
Tel: 81-3-6880- 3770

Korea - Daegu
Tel: 82-53-744-4301

Korea - Seoul
Tel: 82-2-554-7200

Malaysia - Kuala Lumpur
Tel: 60-3-7651-7906

Malaysia - Penang
Tel: 60-4-227-8870

Philippines - Manila
Tel: 63-2-634-9065

Singapore
Tel: 65-6334-8870

Taiwan - Hsin Chu
Tel: 886-3-577-8366

Taiwan - Kaohsiung
Tel: 886-7-213-7830

Taiwan - Taipei
Tel: 886-2-2508-8600

Thailand - Bangkok
Tel: 66-2-694-1351

Vietnam - Ho Chi Minh
Tel: 84-28-5448-2100

EUROPE

Austria - Wels
Tel: 43-7242-2244-39
Fax: 43-7242-2244-393

Denmark - Copenhagen
Tel: 45-4450-2828
Fax: 45-4485-2829

Finland - Espoo
Tel: 358-9-4520-820

France - Paris
Tel: 33-1-69-53-63-20
Fax: 33-1-69-30-90-79

Germany - Garching
Tel: 49-8931-9700

Germany - Haan
Tel: 49-2129-3766400

Germany - Heilbronn
Tel: 49-7131-67-3636

Germany - Karlsruhe
Tel: 49-721-625370

Germany - Munich
Tel: 49-89-627-144-0
Fax: 49-89-627-144-44

Germany - Rosenheim
Tel: 49-8031-354-560

Israel - Ra'anana
Tel: 972-9-744-7705

Italy - Milan
Tel: 39-0331-742611
Fax: 39-0331-466781

Italy - Padova
Tel: 39-049-7625286

Netherlands - Drunen
Tel: 31-416-690399
Fax: 31-416-690340

Norway - Trondheim
Tel: 47-7288-4388

Poland - Warsaw
Tel: 48-22-3325737

Romania - Bucharest
Tel: 40-21-407-87-50

Spain - Madrid
Tel: 34-91-708-08-90
Fax: 34-91-708-08-91

Sweden - Gothenberg
Tel: 46-31-704-60-40

Sweden - Stockholm
Tel: 46-8-5090-4654

UK - Wokingham
Tel: 44-118-921-5800
Fax: 44-118-921-5820