Chapter 15 – Direct Memory Accessing; the 8237 DMA Chip

1. Explain DMA

Direct Memory Access (DMA) is a method of allowing data to be moved from one location to another in a computer without intervention from the central processor (CPU). It is also a fast way of transferring data within (and sometimes between) computers.

For example, DMA is often used to read data from a LAN adapter board and write it into a PC's memory (and vice versa). A DMA controller (often on a PC's motherboard) seizes the bus periodically (for example, once for every 16-bit transfer) to read data from the adapter, then seizes it again to write it to memory (therefore requiring two bus cycles per transfer).

A DMA channel is the combination of bus signals (to request use of the channel and to receive acknowledgment that use of the channel has been granted) and the counters that provide the addresses for the source and destination of the transfers. 16-bit ISA Bus PCs have eight DMA channels, though not all are available for use by add-on peripherals.

Most devices require a dedicated DMA channel (so the number of DMA channels that are available may limit the number of peripherals that can be installed).

A limitation of DMA is that the DMA writes must be to conventional PC memory since extended memory that is mapped to upper memory blocks is mapped only for CPU accesses (since the 80386 memory mapping is on-chip, not on the motherboard). Therefore device drivers that use DMA usually cannot be loaded into upper memory (unless the data buffers are in conventional memory or there is hardware support for *scatter/gather* to provide the memory mapping for the DMA controller). The following table summarizes the DMA channel assignments in an ISA bus PC.

ISA DMA Channel	Use	Comments
0	Was used for memory refresh on early PCs and is therefore not on the 8-bit ISA bus. Current PC dynamic memory (DRAM) is refreshed by a refresh circuit that does not use a DMA channel, and DMA channel 0 is on the 16-bit ISA bus.	Performs 8-bit transfers only. Maximum 64 kbytes per transfer. Only these four channels were supported by the original PC and PC/XT.
1	Available	
2	Floppy and hard drive controller.	
3	Hard disk controller in (now-ancient) XTs only. Usually available in current PCs.	
4	Cascade line to link controller for DMA channels 4 through 7 to controller for DMA channels 0 through 3. Not available for use.	Can perform 16-bit transfers. Maximum of 128 kbytes

		per transfer. These four channels were added when the 16-bit PC/AT bus was introduced.
5	Hard disk controller (in PS/2s only). Usually available on other current PCs.	
6	Available	
7	Available	

DMA channels 2 and 4 are not available for add-on peripherals (such as LAN adapters or sound boards), and the first four DMA channels are capable of transferring only 8 bits at a time (since these channels are compatible with the original PC's 8-bit bus). Therefore the best DMA channels to choose for add-on peripherals are 5, 6, or 7.

2. What is the advantage of using DMA?

Transfer of data is faster

3. What does the DMA need to know to transfer a block of data?

When the DMA needs to transfer a block of data from memory to an I/O device such as a disk, it must know:

a) the address of the beginning of the block (address of first byte of data), and

b) the number of bytes (count) it needs to transfer.

4. Since there is only one set of busses (data bus, address bus, and control bus), how do the DMA and CPU share the buses to transfer data?

When the DMA needs the buses to transfer data it will do the following: (see Figure 15-1)

a) The peripheral device (such as the disk controller) will request the service of DMA by pulling DREQ (DMA request) high.

b) The DMA will put a high on its HRQ (hold request), signaling the CPU through its HOLD pin that it needs to use the buses.

c) The CPU will finish the present bus cycle (not necessarily the present instruction) and respond to the DMA request by putting high on its HDLA (hold acknowledge), thus telling the 8237 DMA that it can go ahead and use the buses to perform its task. HOLD must remain active high as long as DMA is performing its task.

d) DMA will activate DACK (DMA acknowledge) which tells the peripheral device that it will start to transfer the data.

e) DMA starts to transfer the data from memory to peripheral by putting the address of the first byte of the block on the address bus and activating MEMR, thereby reading the byte from memory into the data bus; it then activates IOW to write it to the

peripheral. Then DMA decrements the counter and increments the address pointer and repeats this process until the count reaches zero and the task is finished.

f) After the DMA has finished its job it will deactivate HRQ, signaling the CPU that it can regain control over its buses.

g) While the DMA is using the buses to transfer data, the DMA is idle. By the same token, when the CPU is using the bus, the DMA is idle.

5. Can the DMA decode and execute instructions?

No. DMA can only transfer information

6. Explain the following regarding the 8237 chip (see Figure 15-8).

The 8237 DMA is known as a "fly-by" DMA controller. This means that the data being moved from one location to another does not pass through the DMA chip and is not stored in the DMA chip. Subsequently, the DMA can only transfer data between an I/O port and a memory address, but not between two I/O ports or two memory locations.

a) Number of channels

The 8237 contains four DMA channels that can be programmed independently and any one of the channels may be active at any moment. These channels are numbered 0, 1, 2 and 3. Starting with the PC/AT, IBM added a second 8237 chip, and numbered those channels 4, 5, 6 and 7.

b) Number of bytes that each channel can transfer

The original DMA controller (0, 1, 2 and 3) moves one byte in each transfer. The second DMA controller (4, 5, 6, and 7) moves 16-bits from two adjacent memory locations in each transfer, with the first byte always coming from an even-numbered address. The two controllers are identical components and the difference in transfer size is caused by the way the second controller is wired into the system.

c) What are the signals associated with each channel?

DREQ (DMA request) and DACK (DMA acknowledge). DREQ is an input to DMA coming from the peripheral device (such as the hard disk controller) and DACK is an output signal from the 8237 going to the peripheral device.

d) What terminals are used to connect to the CPU?

From the 8237 DMA, there is only one HOLD and one HDLA that are connected to HOLD and HDLA of the 80x86.

e) Since the 8237 has four channels, how does DMA decide which gets control of the system buses?

Four channels from four different devices can request use of the system buses, but DMA decides which gets control based on the way its priority register has been programmed. Every channel of the 8237 DMA must be initialized separately for the address of the data block and the count (the size of the block) before it can be used.

f) How is this initialization done?

This initialization involves writing into each channel:

i) The address of the first byte of the block of data that must be transferred (called the *base address*).

ii) The number of bytes to be transferred (called the *word count*).

g) How can each channel be enabled after initialization?

After initialization, each channel can be enabled and controlled with the use of a control word.

h) What is the function of address pins A0 – A3 and CS?

The 8237 has many modes of operation and they must be programmed into the chip's internal registers. To access these registers, the 8237 provides 4 address pins, A0 – A3, along with the CS (chip select).

i) How many ports are set aside for the 4 channels?

Since each channel needs separate addresses for the base address and the word count, a total of 8 ports is set aside for those alone.

j) Analyze Table 15-1a.

Table 15-1a shows the internal addresses of the 8237 registers for each channel.

7. Describe the 82374 Enhanced DMA Registers

The Intel 82374 EISA System Component (ESC) was introduced in early 1996 and includes a DMA controller that provides a superset of 8237 functionality as well as other PC-compatible core peripheral components in a single package. This chip is targeted at both EISA and PCI platforms, and provides modern DMA features like scatter-gather, ring buffers as well as direct access by the system DMA to all 32 bits of address space.

If these features are used, code should also be included to provide similar functionality in the previous 16 years worth of PC-compatible computers. For compatibility reasons, some of the 82374 registers must be programmed *after* programming the traditional 8237 registers for each transfer. Writing to a traditional 8237 register forces the contents of some of the 82374 enhanced registers to zero to provide backward software compatibility.

8. Problem 1. Find the port addresses for the base address and word count of each channel of the 8237 for Figure 15-2. A4 has an inverter and the inverter in Line A6 has been replaced by a wire. Hint: review example 15-1.

Using Table 15-1a the port addresses are:

Binary Address										
A7	**A6**	**A5**	**A4**	**A3**	**A2**	**A1**	**A0**	**Hex Address**	**Function**	**Read/ Write**
1	1	0	0	0	0	0	0	C0	CHAN0 memory address register	R/W
1	1	0	0	0	0	0	1	C1	CHAN0 count register	R/W
1	1	0	0	0	0	1	0	C2	CHAN1 memory address register	R/W
1	1	0	0	0	0	1	1	C3	CHAN1 count register	R/W
1	1	0	0	0	1	0	0	C4	CHAN2 memory address register	R/W

1	1	0	0	0	1	0	1	C5	CHAN2 count register	R/W
1	1	0	0	0	1	1	0	C6	CHAN3 memory address register	R/W
1	1	0	0	0	1	1	1	C7	CHAN3 count register	R/W

9. What are the two sets of information does the 8237 DMA need to program a channel?

a) the address of the first byte of data to be transferred, and

b) how many bytes of data are to be transferred.

10. How are the 16 bits of the memory address register sent to the 8-bit data bus of the 8237?

Since the memory address register of the 8237 is 16 bits and the data bus of the 8237 is 8 bits, one byte at a time, right after each other, is sent to the same port address. The same is true for the count register.

11. Review Example 15-2. What change(s) would you do if the DMA needs to transfer 4K (4096) byte block of data? The first memory location starts at 61500. Use the port addresses of Problem 1.

12. Explain the 8237's internal control registers and how these registers are accessed.

Although the 8237 has four channels and each channel must be programmed separately for the base address and count, there is only one set of control/command registers used by all channels. These registers are shown in table 15-1b. Review Example 15-3 to understand how to access those registers. The most essential registers will be described next.

Command register

- It is an 8-bit register that controls the operation of the 8237. See Figure 15-3.
- It must be programmed (written into) by the CPU.
- It is cleared by the RESET signal from the CPU or the master clear instruction of the DMA.

- The 8237 is capable of transferring data
  1. from a peripheral device to memory (reading from disk)
  2. from memory to a peripheral device (writing the file into disk)
  3. from memory to memory
- **Analyze Figure 15-3.**
  1. **D0**. Gives the option to use only channels 0 and 1 for transferring a block of data from memory to memory
  2. **D1**. It is used only when the memory-to-memory option is enabled and can be used to disable the memory incrementation/decrementation of channel 0 in order to write a fixed value into a block of memory.
  3. **D2**. It enables or disables DMA.
  4. **D3**. It gives the option to choose between the normal memory cycle of 4 clock pulses and compressed timing of 2 clock pulses per memory cycle.
  5. **D4**. It gives the option of using the four channels on fixed priority or rotating priority. If fixed priority is chosen, DREQ0 has the highest priority and DREQ3 has the lowest priority. In rotating mode, DREQ0 again has the highest priority and DREQ3 the lowest, but the system rotates through DREQ0, DREQ1, DREQ2, and DREQ3 in that order.
  6. **D5**. It allows time for the write signal to be extended for slow devices.
  7. **D6**. It gives the option of programming the activation level of DREQ. It can be an active-high or active-low signal.
  8. **D7**. It gives the option of programming the activation level of DACQ. It can be an active-high or active-low signal.
  9. The command byte is issued to this register through port address X8H (see Table 15-1b), where X is the combination provided to activate CS.
  10. Review examples 15-4 and 15-5.

Status register (See Figure 15-4)

- It is an 8-bit register that can only be read by the CPU through the same port address as the command register (port X8).
- This register is often referred to as RO (read only) in PC documentation.
- Bits **D0 – D3** are used to indicate if channels 0 – 3 have reached their TC (terminal count).
- TC is set high when the count register has been decremented to zero.
- Bits **D4 – D7** keep count of pending DMA requests. This information can be used by the CPU to see which channel has a pending DMA request.

Mode register (See Figure 15-5)

- It is an 8-bit register that can only be written to by the CPU through port address XBH (see Table 15-3).
- Bits **D0** and **D1** are used for channel selection.
- Bits **D2** and **D3** specify data transfer mode.
    1. In the write transfer option DMA transfers from an I/O device (such as a disk) to memory by activating IOR and MEMW.
    2. Reading from memory to an I/O is a read transfer and is achieved by activating MEMR and IOW.
    3. The verify transfer is called pseudo and is like a read or write except that it does not generate any control signals such as IOR, MEMR, etc.
- Bit **D4** is used for autoinitialization. If enabled, the memory address register and the count register are reloaded with their original values at the end of a DMA data transfer (when the count register becomes zero). In this way those registers are programmed only once and the original values are saved internally.
- Bit **D5** gives the option to increment or decrement the address.
- Bits **D6** and **D7** determine the way the 8237 is used: demand mode, block mode, single mode, and cascade mode.
- **Single mode:** A single byte (or word) is transferred. The DMA must release and re-acquire the bus for each additional byte. This is commonly-used by devices that cannot transfer the entire block of data immediately. The peripheral will request the DMA each time it is ready for another transfer. The standard PC-compatible floppy disk controller (NEC 765) only has a one-byte buffer, so it uses this mode.
- **Block/Demand mode:** Once the DMA acquires the system bus, an entire block of data is transferred, up to a maximum of 64K. If the peripheral needs additional time, it can assert the READY signal to suspend the transfer briefly. READY should not be used excessively, and for slow peripheral transfers, the Single Transfer Mode should be used instead.

  The difference between Block and Demand is that once a Block transfer is started, it runs until the transfer count reaches zero. DREQ only needs to be asserted until -DACK is asserted. Demand Mode will transfer one more byte until DREQ is de-asserted, at which point the DMA suspends the transfer and releases the bus back to the CPU. When DREQ is asserted later, the transfer resumes where it was suspended.

  Older hard disk controllers used Demand Mode until CPU speeds increased to the point that it was more efficient to transfer the data using the CPU, particularly if the memory locations used in the transfer were above the 16Meg mark.
- **Cascade mode:** This mechanism allows a DMA channel to request the bus, but then the attached peripheral device is responsible for placing the addressing information on the bus instead of the DMA. This is also used to implement a technique known as “Bus Mastering”.

When a DMA channel in Cascade Mode receives control of the bus, the DMA does not place addresses and I/O control signals on the bus like the DMA normally does when it is active. Instead, the DMA only asserts the -DACK signal for the active DMA channel.

At this point it is up to the peripheral connected to that DMA channel to provide address and bus control signals. The peripheral has complete control over the system bus, and can do reads and/or writes to any address below 16Meg. When the peripheral is finished with the bus, it de-asserts the DREQ line, and the DMA controller can then return control to the CPU or to some other DMA channel.

Cascade Mode can be used to chain multiple DMA controllers together, and this is exactly what DMA Channel 4 is used for in the PC architecture. When a peripheral requests the bus on DMA channels 0, 1, 2 or 3, the slave DMA controller asserts HLDREQ, but this wire is actually connected to DREQ4 on the primary DMA controller instead of to the CPU. The primary DMA controller, thinking it has work to do on Channel 4, requests the bus from the CPU using HLDREQ signal. Once the CPU grants the bus to the primary DMA controller, -DACK4 is asserted, and that wire is actually connected to the HLDA signal on the slave DMA controller. The slave DMA controller then transfers data for the DMA channel that requested it (0, 1, 2 or 3), or the slave DMA may grant the bus to a peripheral that wants to perform its own bus-mastering, such as a SCSI controller.

Because of this wiring arrangement, only DMA channels 0, 1, 2, 3, 5, 6 and 7 are usable with peripherals on PC/AT systems.

Bus Master Direct Memory Access

A method of transferring data between components of a computer system (such as a LAN adapter, the memory, and a disk controller). A faster data transfer technique (at least for larger transfers) than standard CPU-based DMA, for the following reasons:

- The peripheral (a LAN adapter, for example) writes from its memory directly to the PC's memory in one bus cycle (reducing the load on the bus), rather than the two-step process of the CPU's DMA controller first reading the data (from the adapter) and then writing it to the PC’s memory in a second bus cycle.
- Often, the adapter will do its transfer as the data are received from the LAN, so no, or little, on-board LAN adapter memory is required (this saves money).
- Uses much less CPU time than other methods. For example, programmed input-output (PIO) requires the CPU to first check for the availability of the data, then read the data, and then write the data. This requires bus and CPU time for both fetching the CPU's instructions and for reading and writing the data. Also, bus master DMA is faster than standard DMA, since the CPU does not even need to load the DMA registers (for example, with the source and destination addresses) to set up each transfer.
- Review Example 15-6.

Single Mask register (See Figure 15-6)

- It is an 8-bit register that can only be written to by the CPU through port address XAH (see Table 15-3).

- Bits **D0** and **D1** select the channel.
- Bit **D2** clears or sets the mask bit for that channel. It is through this register that the DREQ input of a specific channel can be masked (disabled) or unmasked (enabled).
- While the command register can be used to disable the whole DMA chip, the single mask register allows the programmer to disable or enable a specific channel.
- Review Example 15-7.

All Mask register (See Figure 15-7)

- This register is similar to the single mask register except that all 4 channels can be masked or unmasked with one write operation.
- This register can only be written to by the CPU through port address XFH (see Table 15-3).

Master clear/temporary register

- This register must only be written to by the CPU through port address XDH (see Table 15-3).
- The byte sent to this register does not matter since it simply clears the status, command, request, and mask registers and forces the DMA to the idle cycle. This is the same as activating then hardware RESET of the 8237.

Clear mask register

- This register can be written to by the CPU only through port address XEH (see Table 15-3).
- The bit patterns written to it do not matter. Its function is to clear the masks bits of all 4 channels, thereby enabling them to accept the DMA request through the DREQs.

12. Programming the DMA

- The DMA channel that is to be programmed should always be "masked" before loading any settings.
- This is because the hardware might unexpectedly assert the DREQ for that channel, and the DMA might respond, even though not all of the parameters have been loaded or updated.
- Once masked, the host must specify the direction of the transfer (memory-to-I/O or I/O-to-memory), what mode of DMA operation is to be used for the transfer (Single, Block, Demand, Cascade, etc), and finally the address and length of the transfer are loaded.
- The length that is loaded is one less than the amount you expect the DMA to transfer. The LSB and MSB of the address and length are written to the same 8-bit I/O port, so another port must be written to first to guarantee that the DMA accepts the first byte as the LSB and the second byte as the MSB of the length and address.
- Then, be sure to update the Page Register, which is external to the DMA and is accessed through a different set of I/O ports.

- Once all the settings are ready, the DMA channel can be un-masked. That DMA channel is now considered to be “armed”, and will respond when the DREQ line for that channel is asserted.

13. Describe Ultra DMA

Ultra DMA (UDMA, or, more accurately, Ultra DMA/33) is a protocol for transferring data between a hard disk drive through the computer’s data paths (or bus) to the computer’s random access memory (RAM). The Ultra DMA/33 protocol transfers data in burst mode at a rate of 33.3 MBps (megabytes per second), twice as fast as the previous Direct Memory Access (DMA) interface.

Ultra DMA was developed as a proposed industry standard by the Quantum Corporation, makers of hard disk drives, and Intel, makers of chipsets that support computer bus technology.

Ultra DMA support in your computer means that it will boot (start) and open new applications more quickly. It will also help users of graphics-intensive and other applications that require large amounts of access to data on the hard drive. Ultra DMA uses Cyclical Redundancy Checking (CRC), offering a new level of data protection.

Because the Ultra DMA protocol is designed to work with legacy application PIO and DMA protocols, it can be added to many existing computers by installing an Ultra DMA/33 Peripheral Component Interconnect adapter card. Ultra DMA uses the same 40-pin Integrated Drive Electronics interface cable as PIO and DMA.