Module 3

Embedded Systems I/O

Lesson 16

DMA

Instructional Objectives

After going through this lesson the student would learn

- The concept of Direct Memory Access
- When and where to use DMA?
- How to initiate an DMA cycle?
- What are the different steps of DMA?
- What is a typical DMA controller?

Pre-Requisite

Digital Electronics, Microprocessors

16(I) Introduction

Drect Memory Access (DMA) allows devices to transfer data without subjecting the processor a heavy overhead. Otherwise, the processor would have to copy each piece of data from the source to the destination. This is typically slower than copying normal blocks of memory since access to I/O devices over a peripheral bus is generally slower than normal system RAM. During this time the processor would be unavailable for any other tasks involving processor bus access. But it can continue to work on any work which does not require bus access. DMA transfers are essential for high performance embedded systems where large chunks of data need to be transferred from the input/output devices to or from the primary memory.

16(II) DMA Controller

A DMA controller is a device, usually peripheral to a CPU that is programmed to perform a sequence of data transfers on behalf of the CPU. A DMA controller can directly access memory and is used to transfer data from one memory location to another, or from an I/O device to memory and vice versa. A DMA controller manages several DMA channels, each of which can be programmed to perform a sequence of these DMA transfers. Devices, usually I/O peripherals, that acquire data that must be read (or devices that must output data and be written to) signal the DMA controller to perform a DMA transfer by asserting a hardware DMA request (DRQ) signal. A DMA request signal for each channel is routed to the DMA controller. This signal is monitored and responded to in much the same way that a processor handles interrupts. When the DMA controller sees a DMA request, it responds by performing one or many data transfers from that I/O device into system memory or vice versa. Channels must be enabled by the processor for the DMA controller to respond to DMA requests. The number of transfers performed, transfer modes used, and memory locations accessed depends on how the DMA channel is programmed. A DMA controller typically shares the system memory and I/O bus with the CPU and has both bus master and slave capability. Fig.16.1 shows the DMA controller architecture and how the DMA controller interacts with the CPU. In bus master mode, the DMA controller acquires the system bus (address, data, and control lines) from the CPU to perform the

DMA transfers. Because the CPU releases the system bus for the duration of the transfer, the process is sometimes referred to as cycle stealing.

In bus slave mode, the DMA controller is accessed by the CPU, which programs the DMA controller's internal registers to set up DMA transfers. The internal registers consist of source and destination address registers and transfer count registers for each DMA channel, as well as control and status registers for initiating, monitoring, and sustaining the operation of the DMA controller.

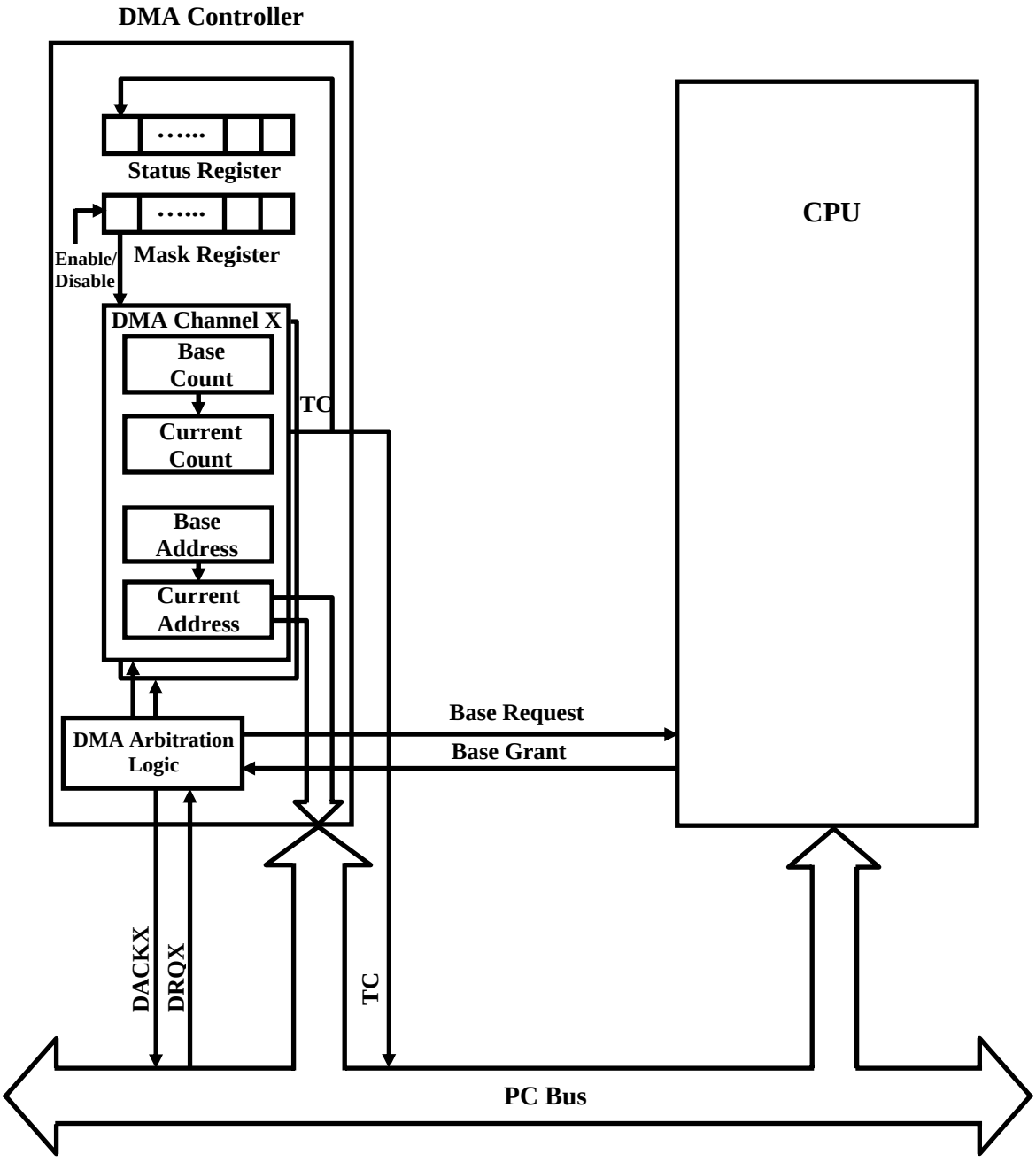

Fig. 16.1 The DMA controller architecture

DMA Transfer Types and Modes

DMA controllers vary as to the type of DMA transfers and the number of DMA channels they support. The two types of DMA transfers are flyby DMA transfers and fetch-and-deposit DMA transfers. The three common transfer modes are single, block, and demand transfer modes. These DMA transfer types and modes are described in the following paragraphs. The fastest DMA transfer type is referred to as a single-cycle, single-address, or flyby transfer. In a flyby DMA transfer, a single bus operation is used to accomplish the transfer, with data read from the source and written to the destination simultaneously. In flyby operation, the device requesting service asserts a DMA request on the appropriate channel request line of the DMA controller. The DMA controller responds by gaining control of the system bus from the CPU and then issuing the pre-programmed memory address. Simultaneously, the DMA controller sends a DMA acknowledge signal to the requesting device. This signal alerts the requesting device to drive the data onto the system data bus or to latch the data from the system bus, depending on the direction of the transfer. In other words, a flyby DMA transfer looks like a memory read or write cycle with the DMA controller supplying the address and the I/O device reading or writing the data. Because flyby DMA transfers involve a single memory cycle per data transfer, these transfers are very efficient. Fig.16.2 shows the flyby DMA transfer signal protocol.

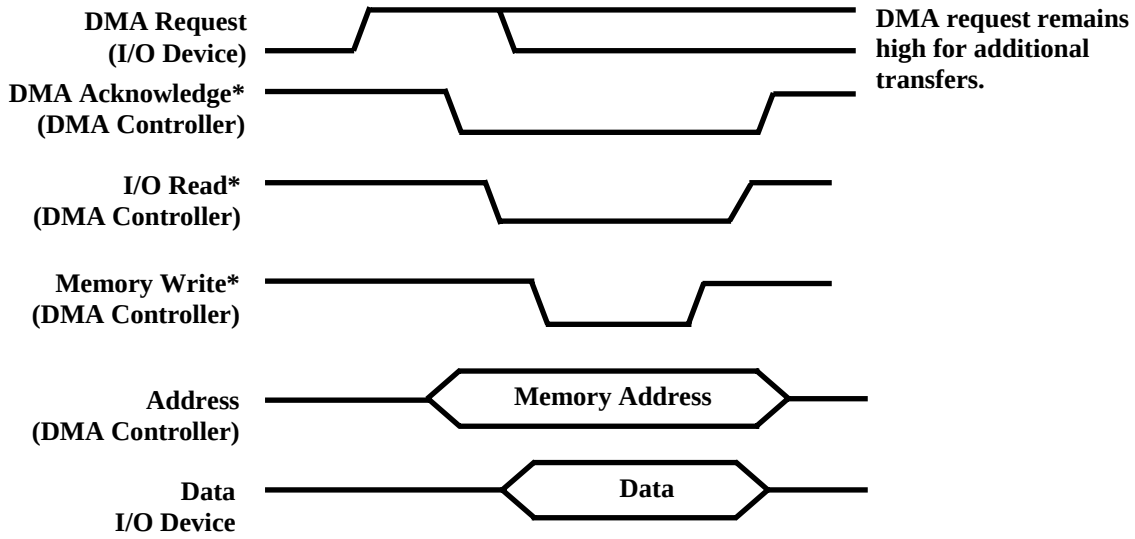

Fig. 16.2 Flyby DMA transfer

The second type of DMA transfer is referred to as a dual-cycle, dual-address, flow-through, or fetch-and-deposit DMA transfer. As these names imply, this type of transfer involves two memory or I/O cycles. The data being transferred is first read from the I/O device or memory into a temporary data register internal to the DMA controller. The data is then written to the memory or I/O device in the next cycle. Fig.16.3 shows the fetch-and-deposit DMA transfer signal protocol. Although inefficient because the DMA controller performs two cycles and thus retains the system bus longer, this type of transfer is useful for interfacing devices with different data bus sizes. For example, a DMA controller can perform two 16-bit read operations from one location followed by a 32-bit write operation to another location. A DMA controller supporting this type of transfer has two address registers per channel (source address and destination address) and bus-size registers, in addition to the usual transfer count and control registers.

Unlike the flyby operation, this type of DMA transfer is suitable for both memory-to-memory and I/O transfers.

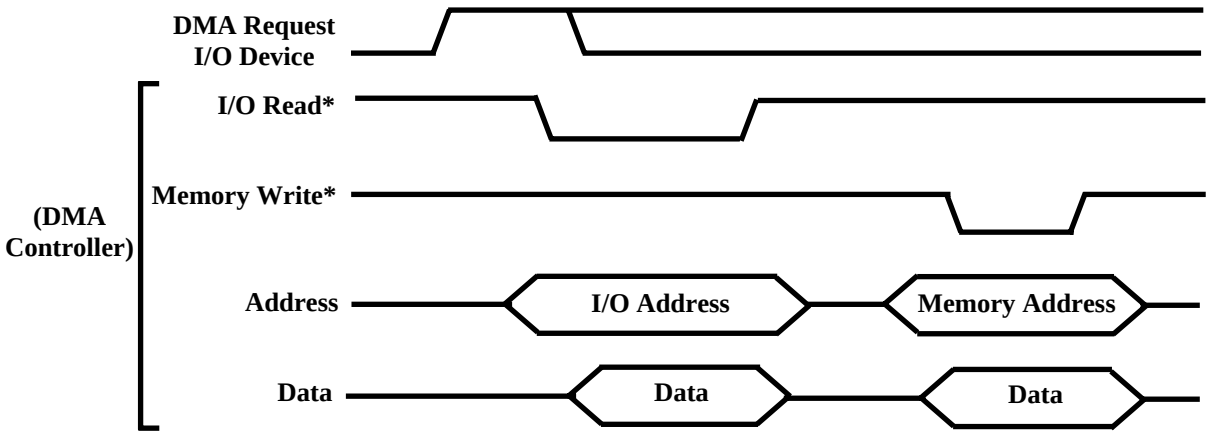

Fig. 16.3 Fetch-and-Deposit DMA Transfer

Single, block, and demand are the most common transfer modes. Single transfer mode transfers one data value for each DMA request assertion. This mode is the slowest method of transfer because it requires the DMA controller to arbitrate for the system bus with each transfer. This arbitration is not a major problem on a lightly loaded bus, but it can lead to latency problems when multiple devices are using the bus. Block and demand transfer modes increase system throughput by allowing the DMA controller to perform multiple DMA transfers when the DMA controller has gained the bus. For block mode transfers, the DMA controller performs the entire DMA sequence as specified by the transfer count register at the fastest possible rate in response to a single DMA request from the I/O device. For demand mode transfers, the DMA controller performs DMA transfers at the fastest possible rate as long as the I/O device asserts its DMA request. When the I/O device unasserts this DMA request, transfers are held off.

DMA Controller Operation

For each channel, the DMA controller saves the programmed address and count in the base registers and maintains copies of the information in the current address and current count registers, as shown in Fig.16.1. Each DMA channel is enabled and disabled via a DMA mask register. When DMA is started by writing to the base registers and enabling the DMA channel, the current registers are loaded from the base registers. With each DMA transfer, the value in the current address register is driven onto the address bus, and the current address register is automatically incremented or decremented. The current count register determines the number of transfers remaining and is automatically decremented after each transfer. When the value in the current count register goes from 0 to -1, a terminal count (TC) signal is generated, which signifies the completion of the DMA transfer sequence. This termination event is referred to as reaching terminal count. DMA controllers often generate a hardware TC pulse during the last cycle of a DMA transfer sequence. This signal can be monitored by the I/O devices participating in the DMA transfers. DMA controllers require reprogramming when a DMA channel reaches TC. Thus, DMA controllers require some CPU time, but far less than is required for the CPU to service device I/O interrupts. When a DMA channel reaches TC, the processor may need to reprogram the controller for additional DMA transfers. Some DMA controllers interrupt the

processor whenever a channel terminates. DMA controllers also have mechanisms for automatically reprogramming a DMA channel when the DMA transfer sequence completes. These mechanisms include auto initialization and buffer chaining. The auto initialization feature repeats the DMA transfer sequence by reloading the DMA channel's current registers from the base registers at the end of a DMA sequence and re-enabling the channel. Buffer chaining is useful for transferring blocks of data into noncontiguous buffer areas or for handling double-buffered data acquisition. With buffer chaining, a channel interrupts the CPU and is programmed with the next address and count parameters while DMA transfers are being performed on the current buffer. Some DMA controllers minimize CPU intervention further by having a chain address register that points to a chain control table in memory. The DMA controller then loads its own channel parameters from memory. Generally, the more sophisticated the DMA controller, the less servicing the CPU has to perform.

A DMA controller has one or more status registers that are read by the CPU to determine the state of each DMA channel. The status register typically indicates whether a DMA request is asserted on a channel and whether a channel has reached TC. Reading the status register often clears the terminal count information in the register, which leads to problems when multiple programs are trying to use different DMA channels.

Steps in a Typical DMA cycle

Device wishing to perform DMA asserts the processors bus request signal.

1. Processor completes the current bus cycle and then asserts the bus grant signal to the device.

2. The device then asserts the bus grant ack signal.

3. The processor senses in the change in the state of bus grant ack signal and starts listening to the data and address bus for DMA activity.

4. The DMA device performs the transfer from the source to destination address.

5. During these transfers, the processor monitors the addresses on the bus and checks if any location modified during DMA operations is cached in the processor. If the processor detects a cached address on the bus, it can take one of the two actions:

    - Processor invalidates the internal cache entry for the address involved in DMA write operation

    - Processor updates the internal cache when a DMA write is detected

6. Once the DMA operations have been completed, the device releases the bus by asserting the bus release signal.

7. Processor acknowledges the bus release and resumes its bus cycles from the point it left off.

16(III) 8237 DMA Controller

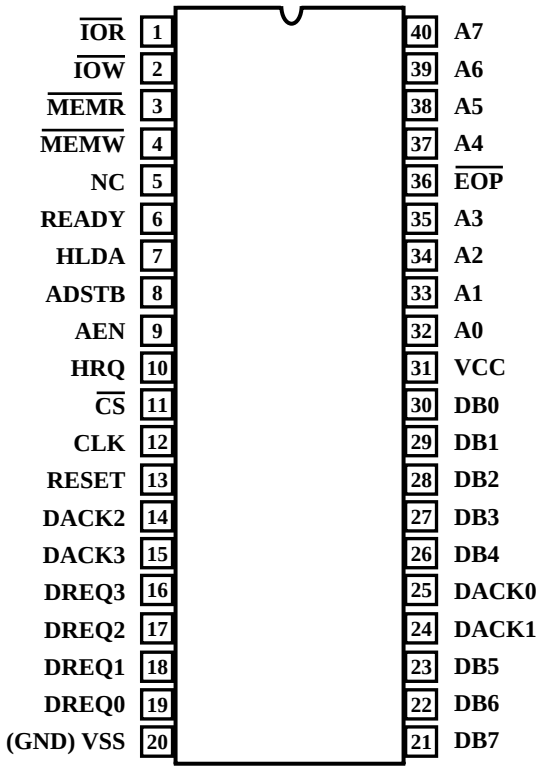

Fig. 16.4 The DMA pin-out

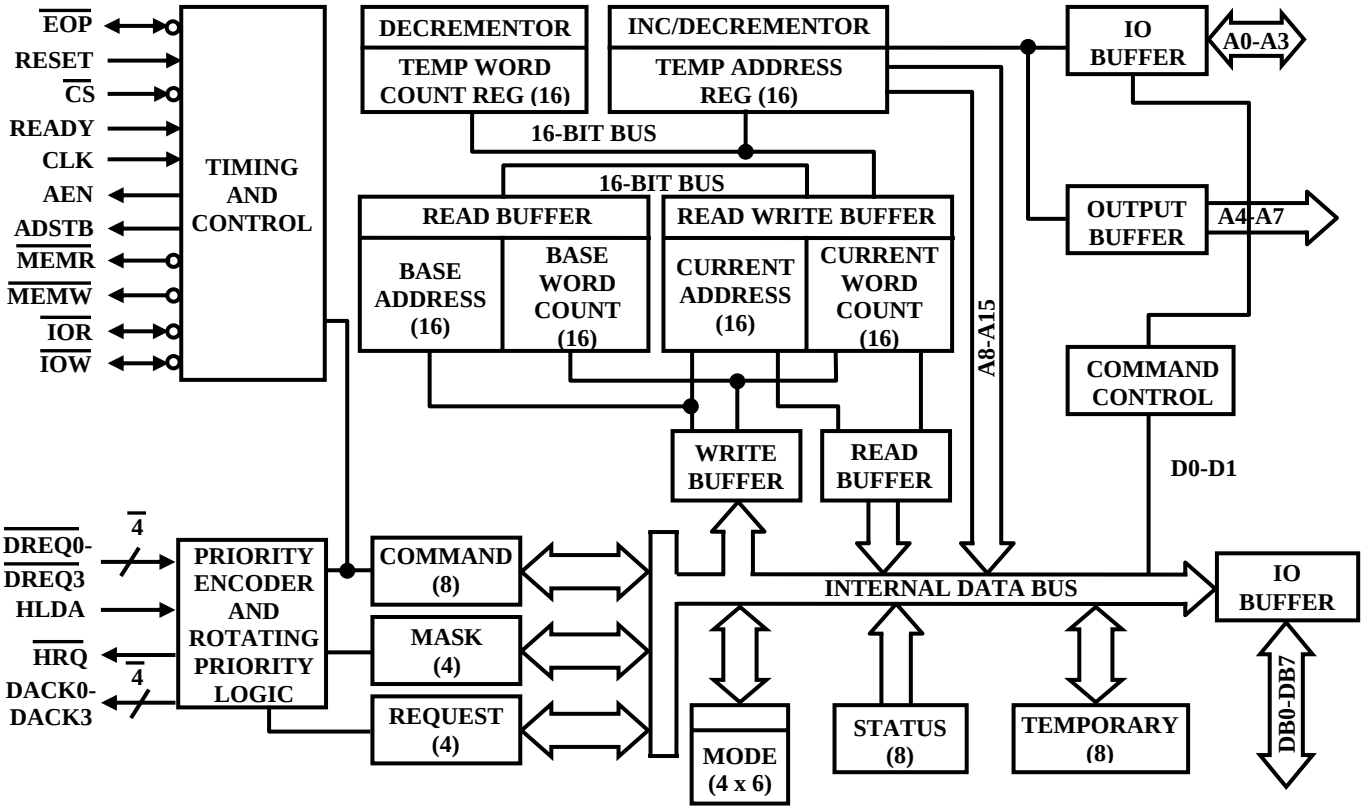

Fig. 16.5 The 8237 Architecture

Signal Description (Fig.16.4 and Fig.16.5)

VCC: is the +5V power supply pin

GND Ground

CLK: CLOCK INPUT: The Clock Input is used to generate the timing signals which control 82C37A operations.

CS: CHIP SELECT: Chip Select is an active low input used to enable the controller onto the data bus for CPU communications.

RESET: This is an active high input which clears the Command, Status, Request, and Temporary registers, the First/Last Flip-Flop, and the mode register counter. The Mask register is set to ignore requests. Following a Reset, the controller is in an idle cycle.

READY: This signal can be used to extend the memory read and write pulses from the 82C37A to accommodate slow memories or I/O devices.

HLDA: HOLD ACKNOWLEDGE: The active high Hold Acknowledge from the CPU indicates that it has relinquished control of the system busses.

DREQ0-DREQ3: DMA REQUEST: The DMA Request (DREQ) lines are individual asynchronous channel request inputs used by peripheral circuits to obtain DMA service. In Fixed Priority, DREQ0 has the highest priority and DREQ3 has the lowest priority. A request is generated by activating the DREQ line of a channel. DACK will acknowledge the recognition of a DREQ signal. Polarity of DREQ is programmable. RESET initializes these lines to active high. DREQ must be maintained until the corresponding DACK goes active. DREQ will not be recognized while the clock is stopped. Unused DREQ inputs should be pulled High or Low (inactive) and the corresponding mask bit set.

DB0-DB7: DATA BUS: The Data Bus lines are bidirectional three-state signals connected to the system data bus. The outputs are enabled in the Program condition during the I/O Read to output the contents of a register to the CPU. The outputs are disabled and the inputs are read during an I/O Write cycle when the CPU is programming the 82C37A control registers. During DMA cycles, the most significant 8-bits of the address are output onto the data bus to be strobed into an external latch by ADSTB. In memory-to-memory operations, data from the memory enters the 82C37A on the data bus during the read-from-memory transfer, then during the write-to-memory transfer, the data bus outputs write the data into the new memory location.

IOR: READ: I/O Read is a bidirectional active low three-state line. In the Idle cycle, it is an input control signal used by the CPU to read the control registers. In the Active cycle, it is an output control signal used by the 82C37A to access data from the peripheral during a DMA Write transfer.

IOW: WRITE: I/O Write is a bidirectional active low three-state line. In the Idle cycle, it is an input control signal used by the CPU to load information into the 82C37A. In the Active cycle, it is an output control signal used by the 82C37A to load data to the peripheral during a DMA Read transfer.

EOP: END OF PROCESS: End of Process (EOP) is an active low bidirectional signal. Information concerning the completion of DMA services is available at the bidirectional EOP pin. The 82C37A allows an external signal to terminate an active DMA service by pulling the EOP pin low. A pulse is generated by the 82C37A when terminal count (TC) for any channel is reached, except for channel 0 in memory-to-memory mode. During memory-to-memory

transfers, EOP will be output when the TC for channel 1 occurs. The EOP pin is driven by an open drain transistor on-chip, and requires an external pull-up resistor to VCC. When an EOP pulse occurs, whether internally or externally generated, the 82C37A will terminate the service, and if auto-initialize is enabled, the base registers will be written to the current registers of that channel. The mask bit and TC bit in the status word will be set for the currently active channel by EOP unless the channel is programmed for autoinitialize. In that case, the mask bit remains clear.

A0-A3: ADDRESS: The four least significant address lines are bidirectional three-state signals. In the Idle cycle, they are inputs and are used by the 82C37A to address the control register to be loaded or read. In the Active cycle, they are outputs and provide the lower 4-bits of the output address.

A4-A7: ADDRESS: The four most significant address lines are three-state outputs and provide 4-bits of address. These lines are enabled only during the DMA service.

HRQ: HOLD REQUEST: The Hold Request (HRQ) output is used to request control of the system bus. When a DREQ occurs and the corresponding mask bit is clear, or a software DMA request is made, the 82C37A issues HRQ. The HLDA signal then informs the controller when access to the system busses is permitted. For stand-alone operation where the 82C37A always controls the busses, HRQ may be tied to HLDA. This will result in one S0 state before the transfer.

DACK0-DACK3: DMA ACKNOWLEDGE: DMA acknowledge is used to notify the individual peripherals when one has been granted a DMA cycle. The sense of these lines is programmable. RESET initializes them to active low.

AEN: ADDRESS ENABLE: Address Enable enables the 8-bit latch containing the upper 8 address bits onto the system address bus. AEN can also be used to disable other system bus drivers during DMA transfers. AEN is active high.

ADSTB: ADDRESS STROBE: This is an active high signal used to control latching of the upper address byte. It will drive directly the strobe input of external transparent octal latches, such as the 82C82. During block operations, ADSTB will only be issued when the upper address byte must be updated, thus speeding operation through elimination of S1 states. ADSTB timing is referenced to the falling edge of the 82C37A clock.

MEMR: MEMORY READ: The Memory Read signal is an active low three-state output used to access data from the selected memory location during a DMA Read or a memory-to-memory transfer.

MEMW MEMORY WRITE: The Memory Write signal is an active low three-state output used to write data to the selected memory location during a DMA Write or a memory-to-memory transfer.

NC: NO CONNECT: Pin 5 is open and should not be tested for continuity.

Functional Description

The 82C37A direct memory access controller is designed to improve the data transfer rate in systems which must transfer data from an I/O device to memory, or move a block of memory to an I/O device. It will also perform memory-to-memory block moves, or fill a block of memory with data from a single location. Operating modes are provided to handle single byte transfers as

well as discontinuous data streams, which allows the 82C37A to control data movement with software transparency. The DMA controller is a state-driven address and control signal generator, which permits data to be transferred directly from an I/O device to memory or vice versa without ever being stored in a temporary register. This can greatly increase the data transfer rate for sequential operations, compared with processor move or repeated string instructions. Memory-to-memory operations require temporary internal storage of the data byte between generation of the source and destination addresses, so memory-to-memory transfers take place at less than half the rate of I/O operations, but still much faster than with central processor techniques. The block diagram of the 82C37A is shown in Fig.16.6. The timing and control block, priority block, and internal registers are the main components. The timing and control block derives internal timing from clock input, and generates external control signals. The Priority Encoder block resolves priority contention between DMA channels requesting service simultaneously.

DMA Operation

In a system, the 82C37A address and control outputs and data bus pins are basically connected in parallel with the system busses. An external latch is required for the upper address byte. While inactive, the controller's outputs are in a high impedance state. When activated by a DMA request and bus control is relinquished by the host, the 82C37A drives the busses and generates the control signals to perform the data transfer. The operation performed by activating one of the four DMA request inputs has previously been programmed into the controller via the Command, Mode, Address, and Word Count registers. For example, if a block of data is to be transferred from RAM to an I/O device, the starting address of the data is loaded into the 82C37A Current and Base Address registers for a particular channel, and the length of the block is loaded into the channel's Word Count register. The corresponding Mode register is programmed for a memory-to-I/O operation (read transfer), and various options are selected by the Command register and the other Mode register bits. The channel's mask bit is cleared to enable recognition of a DMA request (DREQ). The DREQ can either be a hardware signal or a software command. Once initiated, the block DMA transfer will proceed as the controller outputs the data address, simultaneous MEMR and IOW pulses, and selects an I/O device via the DMA acknowledge (DACK) outputs. The data byte flows directly from the RAM to the I/O device. After each byte is transferred, the address is automatically incremented (or decremented) and the word count is decremented. The operation is then repeated for the next byte. The controller stops transferring data when the Word Count register underflows, or an external EOP is applied.

To further understand 82C37A operation, the states generated by each clock cycle must be considered. The DMA controller operates in two major cycles, active and idle. After being programmed, the controller is normally idle until a DMA request occurs on an unmasked channel, or a software request is given. The 82C37A will then request control of the system busses and enter the active cycle. The active cycle is composed of several internal states, depending on what options have been selected and what type of operation has been requested. The 82C37A can assume seven separate states, each composed of one full clock period. State I (SI) is the idle state. It is entered when the 82C37A has no valid DMA requests pending, at the end of a transfer sequence, or when a Reset or Master Clear has occurred. While in SI, the DMA controller is inactive but may be in the Program Condition (being programmed by the processor). State 0 (S0) is the first state of a DMA service. The 82C37A has requested a hold but the processor has not yet returned an acknowledge. The 82C37A may still be programmed until it

has received HLDA from the CPU. An acknowledge from the CPU will signal the DMA transfer may begin. S1, S2, S3, and S4 are the working state of the DMA service. If more time is needed to complete a transfer than is available with normal timing, wait states (SW) can be inserted between S3 and S4 in normal transfers by the use of the Ready line on the 82C37A. For compressed transfers, wait states can be inserted between S2 and S4. Note that the data is transferred directly from the I/O device to memory (or vice versa) with IOR and MEMW (or MEMR and IOW) being active at the same time. The data is not read into or driven out of the 82C37A in I/O-to-memory or memory-to-I/O DMA transfers. Memory-to-memory transfers require a read-from and a write-to memory to complete each transfer. The states, which resemble the normal working states, use two-digit numbers for identification. Eight states are required for a single transfer. The first four states (S11, S12, S13, S14) are used for the read-from-memory half and the last four state (S21, S22, S23, S24) for the write-to-memory half of the transfer.

16(IV) Conclusion

This lesson has given an overview of DMA controller. The controllers are normally used in high-performance embedded systems where large bulks of data need to transferred from the input to the memory. One such system is a on-board Digital Signal Processor in a mobile telephone. Besides fast digital coding and decoding at times this processor is required to process the voice signals to improve the quality. This has to take place in real time. While the voice message is streaming in through the AD-converter it need to be transferred and windowed for filtering. DMA offers a great help here. For simpler systems DMA is not normally used.

The signals and functional architecture of a very familiar DMA controller(8237) used in personal computers has been discussed. For more detailed discussions the readers are requested to visit www.intel.com or any other manufactures and read the datasheet.

16(V) Questions and Answers

Q.1. Can you use 82C37A in embedded systems? Justify your answers

Ans: Only high performance systems where the power supply constraints are not stringent. The supply voltage is 5V and the current may reach up to 16 mA resulting in 80 mW of power consumption.

Q.2 Highlight on different modes of DMA data transfer. Which mode consumes the list power and which mode is the fastest?

Ans: *Refer to text*

Q.3. Draw the architecture of 8237 and explain the various parts.

Ans: *Refer to text*