8257- DMA Controller :-

DMA stands for Direct Memory Access.
It means transferring data directly b/w mly and I/O without the involvement of µP.

DMA transfers are faster compared to µP based transfers due to 2 reasons :

① Hardware based.

Normally data transfers are performed by µP and they are based on s/w and hence are also called Programmed I/O

These transfers are very slow as they involve fetching and decoding of instructions in a loop throughout the transfer process and in fact in most cases, more time is wasted in fetching and decoding of instructions than actually transferring data.

This is where DMA based transfers have a massive speed advantage.

DMA transfers are performed by a dedicated h/w called a DMAC (8257/8237). It is "hardwired" to perform DMA transfers and hence do not need instructions to transfer every byte. This saves a lot of time which would have been otherwise wasted in fetching and decoding instructions.

(2) Direct transfer

Normally data is transferred via µP. This means if a byte has to be transferred from m/y to I/O, it must first go from m/y to µP and then from µP to I/O.

This involves double the time as 2 machine cycles are needed: m/y read and I/O write.

In DMA transfers the byte is transferred directly from m/y to I/O and vice versa. It does not travel via the µP and hence does not take double the time.

Steps for performing a DMA transfer:

By default µP is the bus master.
Although the transfer is mainly carried out by the DMAC, the initialization is first done by the µP.

The following steps are required for the complete DMA transfer:

1. µP initializes the DMAC.
- This is done by giving the starting address and the no. of bytes to be transferred.

2. I/O device requests the DMAC.
- I/O device makes the DREQ signal = 1.

3. DMAC requests the µP for control of the system bus.
   - DMAC makes HOLD = 1

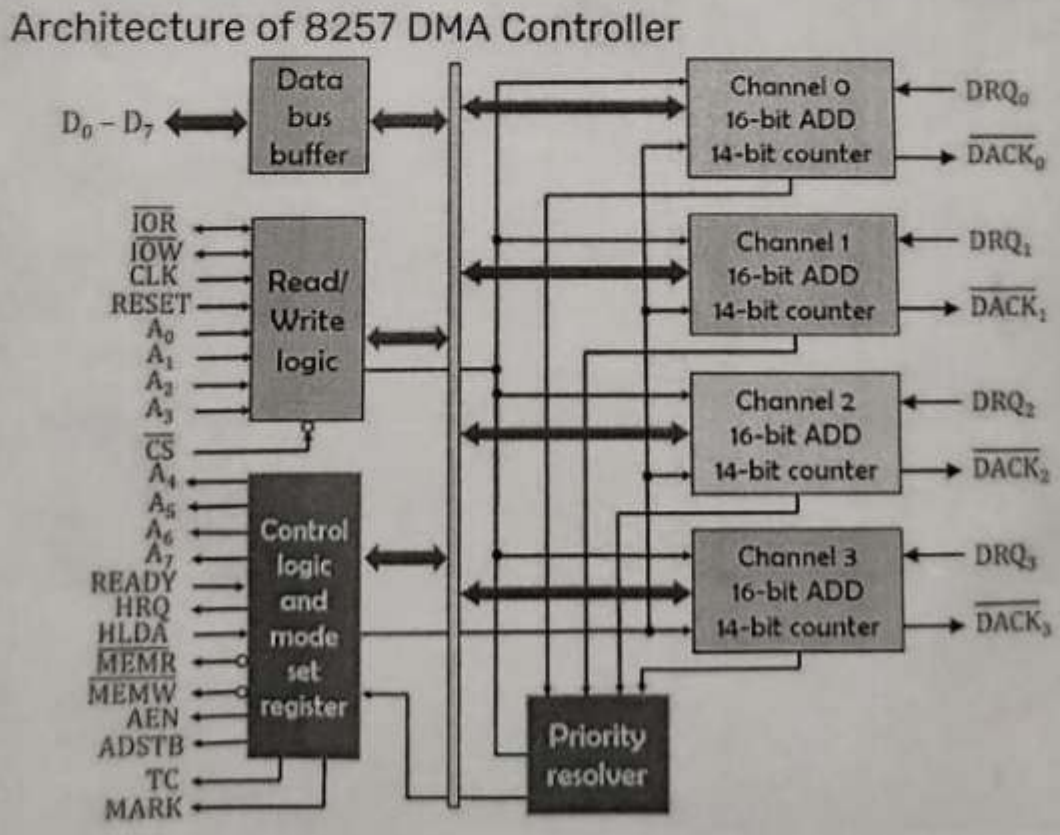
4. µP releases control of the system bus.
   - µP finishes the current m/c cycle and releases control of the sys bus.
   - µP informs the DMAC that the bus is released by making HLDA = 1.
   - Now µP enters HOLD state.

5. DMAC becomes the bus master
   - On getting HLDA from µP, DMAC becomes the bus master.
   - It informs the I/O device that DMA transfer is about to begin by activating the DACK signal.

6. DMA transfer begins.
   - DMAC transfers data b/w m/y and I/O, one byte in 1 cycle.
   - After every cycle, the Address Register is incremented and the count register is decremented.
   - This continues till the count reaches zero (Terminal Count)
   - Now the DMA transfer is completed.

7. DMA transfer ends
- DMAC releases control of the s/m bus.
- It makes HOLD = 0
- This makes µP come out of Hold state and once again becomes the bus master.
- µP takes control of the s/m bus and continues its operation

Block Diagram of 8257



Data Bus Buffer:

It is a bidirectional eight bit buffer which interfaces 8257 to the external data bus of the system.

In the slave mode, it is used to transfer data b/w µP and internal registers of 8257.

In master mode, it is used to send higher byte address (A_8 - A_{15}) on the data bus.

Read/Write Logic

It mainly provides the read and write signals as well as chip select signal.

When the µP is reading/programming one of the internal registers of 8257 (slave mode), the Read/Write logic accepts the I/O Read or I/O write signal,

Note: Read and write signals are connected to $\overline{IOR}$ and $\overline{IORW}$ of µP.

It decodes the least significant 4 bits (A_0-A_3) and either writes the contents of data bus into address register or places the contents of address register onto data bus.

During DMA cycles (master mode) the Read/Write logic generates the I/O read and memory write (DMA write cycle) or I/O write and memory read (DMA read cycle) signals which controls the data transfer b/w peripherals and mly.

DMA channels :

8257 has 4 identical channels labelled CH0 to CH3.
Four I/O devices can be connected, one each on these channels.
Each channel has 4 components :
a) Address Register b) Terminal Count Register c) DREQ and d) $\overline{DACK}$.

a) DMA address register (16 bit)

µP initializes this register with the starting address of the DMA transfer (ie the first mly location to be accessed).

Note :- DMAC can only produce a 16 bit address whereas the actual address is 20 bit. Hence the upper 4 bits of the address have to be still produced by the µP.

It is necessary to load a valid 16 bit mly address in the DMA address register before channel is enabled.

A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0

Format of DMA address register

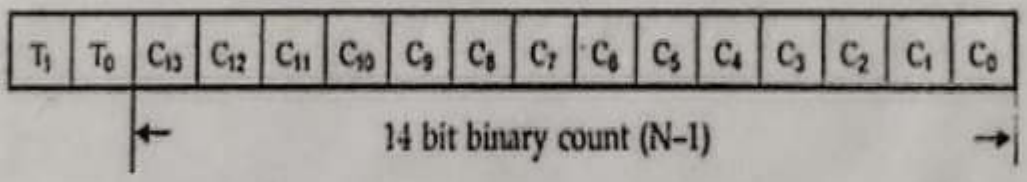
Thereafter after each byte is transferred the address gets incremented (or decremented depending on the mode selected by the programmer).

b) Terminal Count Register (16 bit)

The value loaded into the lower order 14 bits (C_{13}-C_0) of the TC register is the no. of DMA cycles minus one.
ie, µP initializes the register with 14 bit count (N-1) of the DMA transfer.

Thereafter as each byte is transferred the count gets decremented. This repeats till the count becomes 0, also called Terminal Count (TC).

The higher 2 bits are used to decode the type of DMA operation to be performed (DMA Verify, DMA Read, DMA write)



T_1	T_0	Type of operation
0	0	DMA Verify cycle
0	1	DMA Write cycle
1	0	DMA READ cycle
1	1	Illegal

It is necessary to load count for DMA cycles and operational code for valid DMA cycle in the TC register before channel is enabled.

DMA Read Cycle

On this cycle, DMAC becomes the bus master and it transfers data from memory to I/O.
Hence in every transfer, the signals produced are $\overline{MEMR}$ and $\overline{IOW}$

DMA write cycle:

On this cycle, when DMAC becomes the bus master, it transfers data from I/O to m/y. Hence in every transfer, the signals produced are $\overline{IOR}$ and $\overline{MEMW}$.

DMA verify cycle:

On this cycle, DMAC becomes the bus master, it doesn't really transfer any data. This is just used to verify the DMA process. The DMAC issues a HOLD, becomes bus master, issues acknowledgement to the I/O device and so on.

c) DREQ

I/O device gives DREQ signal to the DMAC to request a DMA transfer.

d) $\overline{DACK}$

It is given by DMAC to I/O device, indicating that a DMA transfer is being performed.

Priority Resolver:

Priority is needed when several DMA channels get request (DREQ) from I/O devices "simultaneously" for data transfer.

Priority resolver decides which channel will be 'serviced' first and which one will become "pending".

There are 2 priority schemes: → Fixed priority
→ Rotating Priority

Fixed Priority: This is the default mode.

- Channel 0 is the highest priority and Channel 3 is the lowest
- Fixed priority causes Domination.
  If channel 0 and 1 are keeping on requesting all the time, then channel 2 and 3 will starve and never get a chance.
- To avoid this rotating priority can be used.

Rotating Priority:

- Here once a channel is serviced it becomes the lowest priority.
- All channels below it rise up by one position in the priority order.
- As priorities move in a circular manner, it is called Rotating Priority.
- It gives every channel a fair chance of being higher priority and hence prevents Domination.

Before CH.0 is serviced.

Highest	CH.0 ← Active DMA request.
	CH.1
	CH.2
Lowest	CH.3

After CH.0 is serviced.

CH.1	Highest
CH.2	
CH.3	
CH.0	Lowest

Fixed Priority: This is the default mode.

- Channel 0 is the highest priority and Channel 3 is the lowest.
- Fixed priority causes Domination.
  If channel 0 and 1 are keeping on requesting all the time, then channel 2 and 3 will starve and never get a chance.
- To avoid this rotating priority can be used.

Rotating Priority:

- Here once a channel is serviced it becomes the lowest priority.
- All channels below it rise up by one position in the priority order.
- As priorities move in a circular manner, it is called Rotating Priority.
- It gives every channel a fair chance of being higher priority and hence prevents Domination.

Before CH.0 is serviced.

Highest	CH.0 ← Active DMA request.
	CH.1
	CH.2
Lowest	CH.3

After CH.0 is serviced.

CH.1	Highest
CH.2	
CH.3	
CH.0	Lowest

Control Logic and Mode Set Register:

It generates the internal control signals for DMAC.

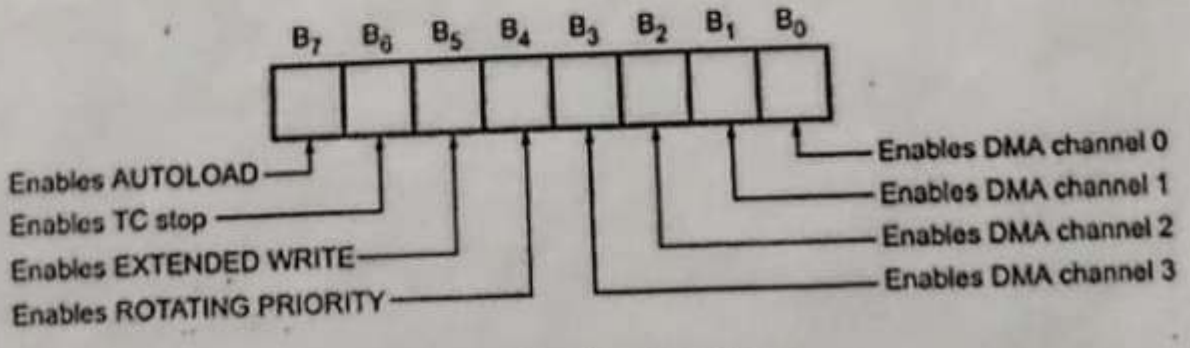
It has 2 registers

1. Mode Set Register - programmed by µP to configure 8257.
2. Status Register - read by µP to check which channels have reached terminal count condition and status of update flag.

Mode Set Register :-

The least significant 4 bits of Mode set Register allows to enable each of the 4 channels.

Most significant 4 bits allows 4 different options for 8257



Mode set register

Bit 0...3 Channel Enable

1: Enable respective DMA channel.

0: Disable

Bit 4 : Rotating Priority
1 : Rotating Priority
0 : Fixed Priority

Bit 5 : Extended Write
1 : Extended write
0 : Normal Write

Extended Write Mode : Here the write control signal gets activated one T-state in advance. This is similar to the Advanced write signals of 8086 Maximum Mode.

Bit 6 : TC Stop
1 : Enable TC stop mode
0 : Disable
If the TC stop bit is set, a channel is disabled, after the terminal count o/p goes high thus automatically preventing further DMA operations on that channel.

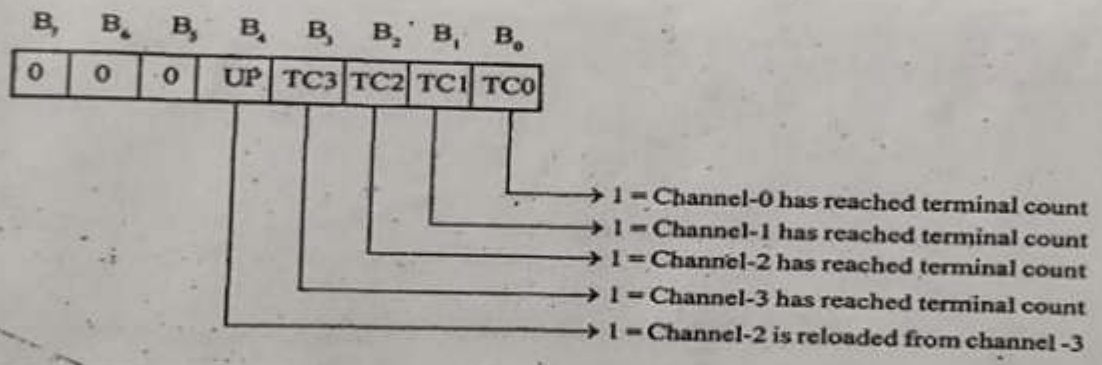
Bit 7 : Auto load
This mode is applicable only for channel 2. When this mode is enabled the address register and count of channel 2 are stored as backup in Channel 3 registers. This mode is used to do repetitive transfers.

Semester: IV

After every byte is transferred, channel 2 registers keep changing but channel 3 registers maintain the original values.

When channel 2 reaches TC, there is automatic reload of address and count information from channel 3 registers to channel 2 registers and the DMA transfer restarts.

Status Register:



The TC status bit if one indicates terminal count has been reached for that channel.

The TC bit remains set until the status register is read or 8257 is reset.

The update flag bit, if one indicates CPU that 8257 is executing update cycle.

In update cycle 8257 loads parameters in channel 3 to channel 2.