SYSC3601
Microprocessor Systems

Unit 8:
Direct Memory Access
(DMA)

Topics/Reading

1. DMA

2. The 8237 DMA Controller

Reading: Chapter 13, sections 1-2

http://www.freebsd.org/doc/en_US.ISO8859-1/books/developers-handbook/dma.html

Direct Memory Access (DMA)

- Provides direct access to memory for I/O devices while the µP is temporarily disabled.
- Data is directly transferred between memory and the I/O device.
- Transfer rates can approach 33-150 Mbyte/s
  - Limited only by DMA Controller speed and RAM speed
  - Limited to 1.6 Mbytes/s using 8237 & 8086/88
- Used for DRAM refresh, video displays, disk read/write.

Direct Memory Access (DMA) – Basic Operation

- 2 signals are used on the µP:
  - `HOLD` Input used to request a DMA action
  - `HLDA` Output acknowledgement, transfer busses to DMA controller
- 'DMA controller' is essentially a special purpose µP for conducting high speed data transfers.
- During DMA transfer, the µP is effectively disconnected from the busses.
- DMA transfers data between:
  - memory ↔ I/O
  - memory ↔ memory
    - Not used in more advanced µP (faster to use CPU)

Direct Memory Access (DMA) - Sequence

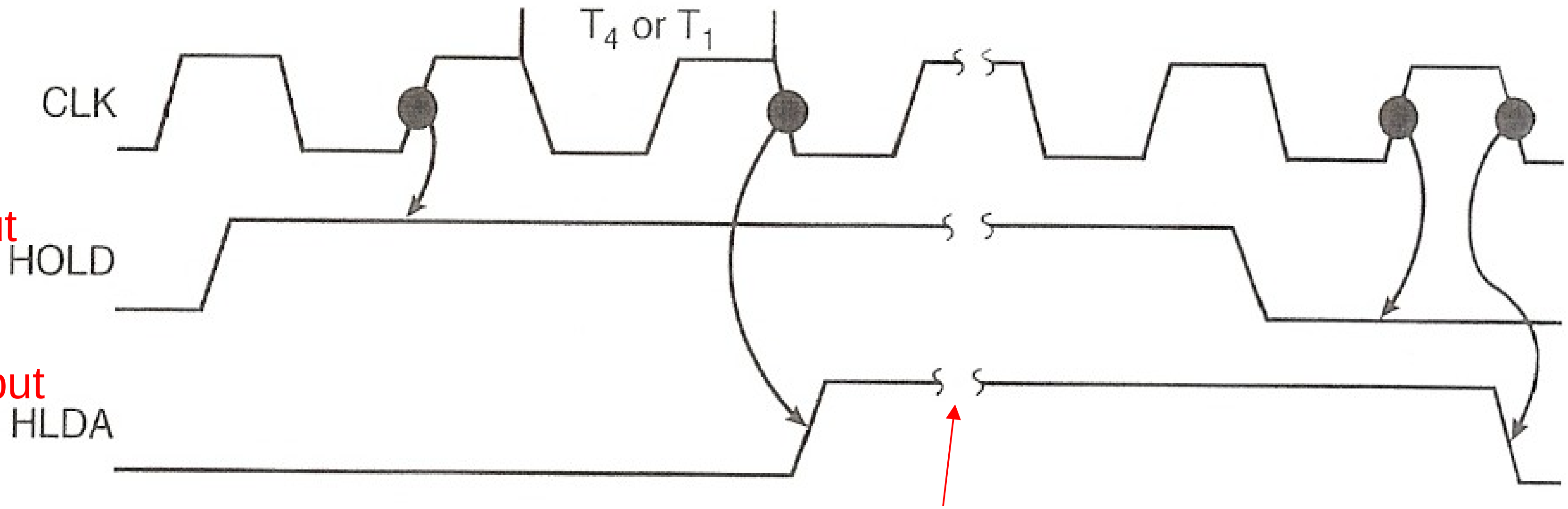
1. HOLD input raised by DMA controller
2. µP suspends execution of its program.
   - Note that HOLD has higher priority than INTR or NMI and can interrupt instructions mid-stream (like RESET, whereas INTR/NMI wait for instruction to complete)
3. µP places its address, data, and control busses in high-impedance state.
4. µP raises HLDA to signal device that it now has complete access to the busses.
5. DMA transfer takes place.
   - On more advanced µP, execution continues until system bus is required (from cache/pipeline)
6. Device signals µP that transfer is complete by dropping the HOLD signal.
7. µP drops HLDA.

Direct Memory Access (DMA) - Sequence

Why HLDA only after T1 or T4?

µP input

µP output



DMA transfer occurs…

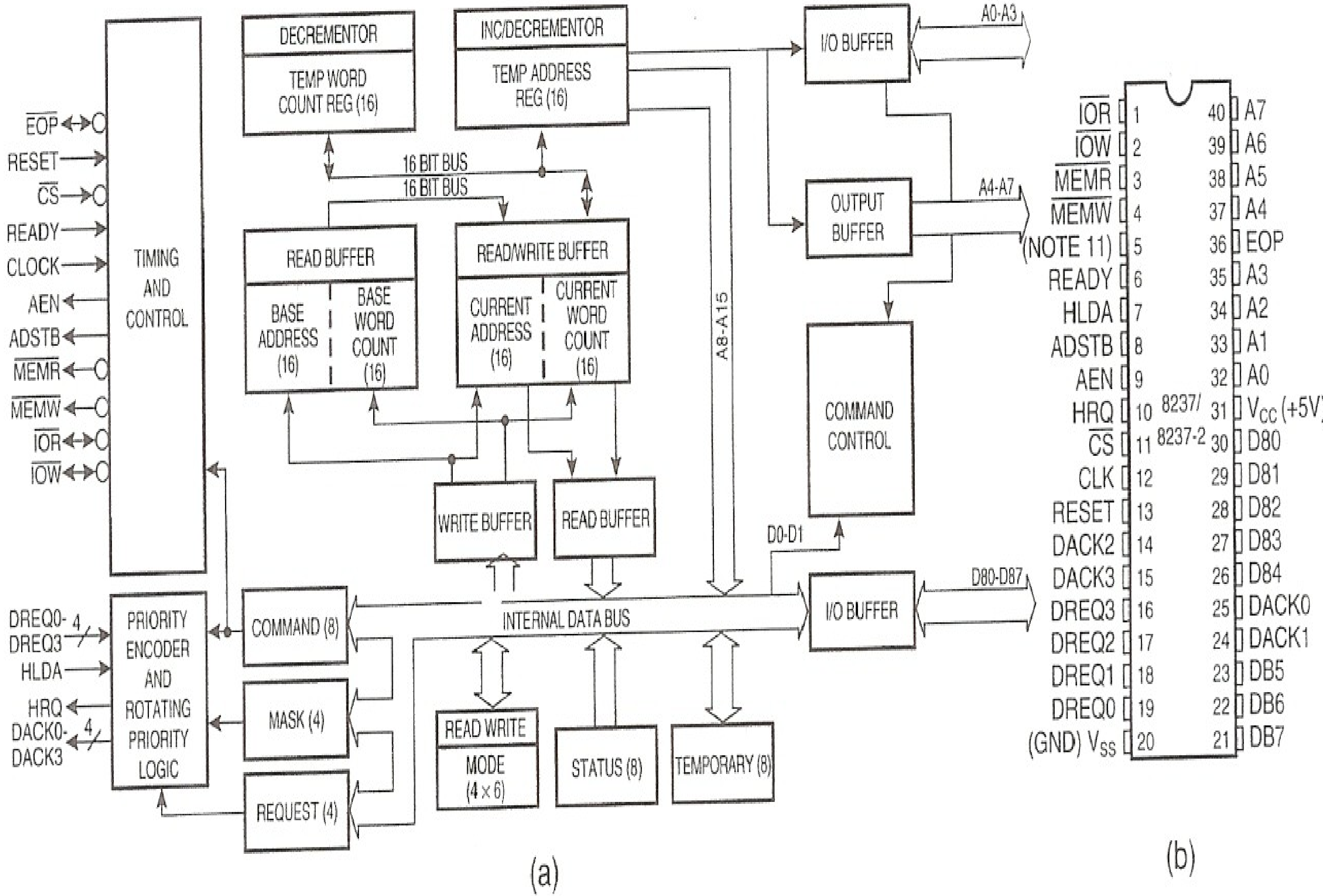
The 8237 DMA Controller

- A special purpose μP for conducting high speed data transfers.
- 4 DMA channels (0-3)
- Designed to manipulate the system busses in a way similar to the μP, but much faster.
  - Generates addresses and ALE
  - Generates bus control signals
    - $\overline{\texttt{MRDC}}$, $\overline{\texttt{MWRC}}$, $\overline{\texttt{IORC}}$, $\overline{\texttt{IOWC}}$
    - Note that these signals are not available on the 8086/88 in minimum mode – have to create using external circuitry.

The 8237 DMA Controller

- The µP programs the DMA controller.
  - How much data to transfer (size in bytes)
  - Where the data comes from (source)
  - Where the data goes (destination)
- Note: on the 8237, a *channel* is a path between the 8237 and the memory or I/O device.
  - Each channel includes a DREQ request line, a $\overline{\text{DACK}}$ acknowledge line, and internal registers for address, counter, configuration, etc.
  - For I/O, channel acknowledge signal bypasses address decoding and activates I/O chip directly.

Intel 8237

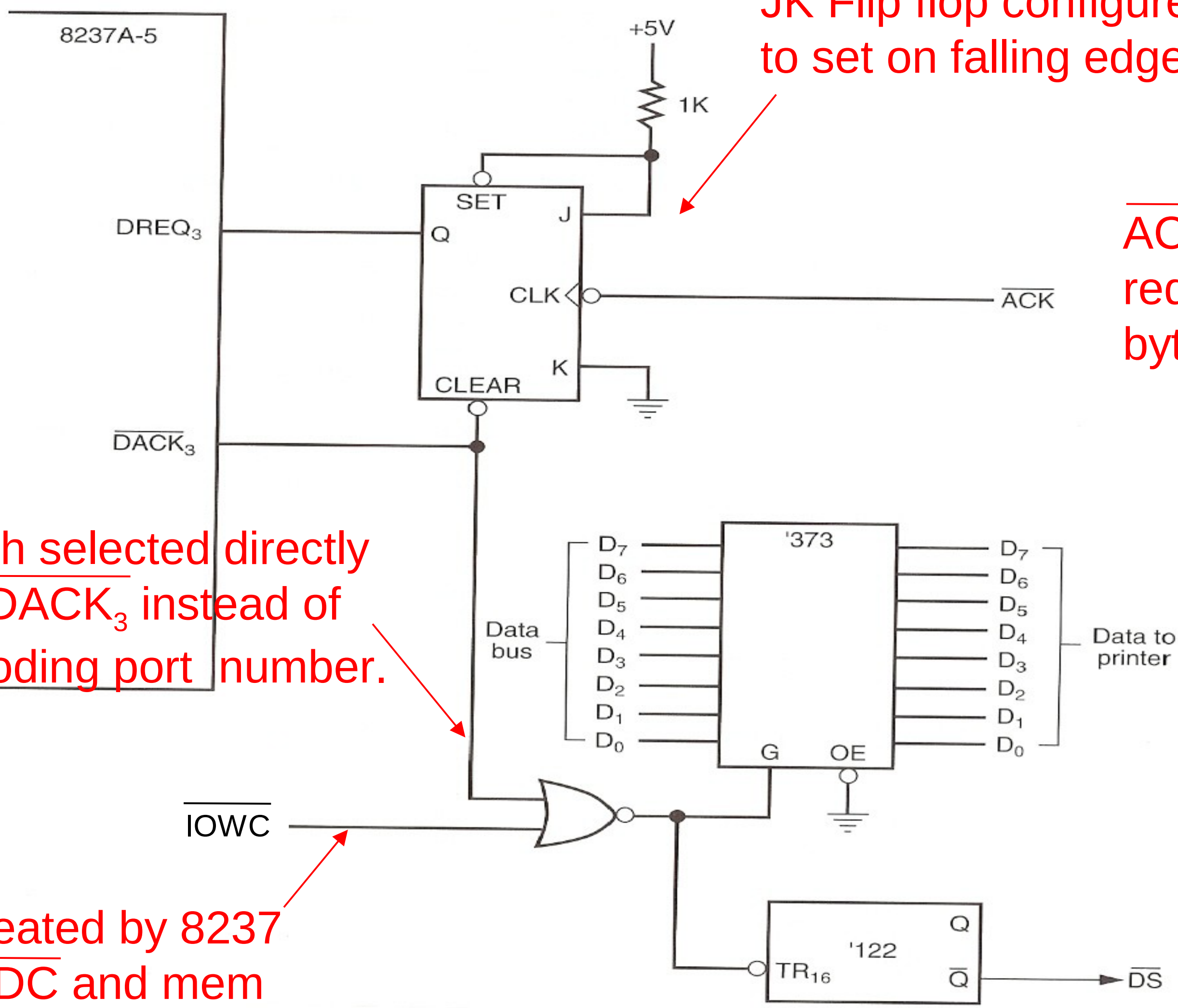


DECREMENTOR
TEMP WORD COUNT REG (16)
INC/DECREMENTOR
TEMP ADDRESS REG (16)
I/O BUFFER
A0-A3
OUTPUT BUFFER
A4-A7
16 BIT BUS
16 BIT BUS
EOP
RESET
CS
READY
CLOCK
AEN
ADSTB
MEMR
MEMW
IOR
IOW
TIMING AND CONTROL
READ BUFFER
BASE ADDRESS (16)
BASE WORD COUNT (16)
READ/WRITE BUFFER
CURRENT ADDRESS (16)
CURRENT WORD COUNT (16)
A8-A15
COMMAND CONTROL
WRITE BUFFER
READ BUFFER
D0-D1
D80-D87
I/O BUFFER
INTERNAL DATA BUS
DREQ0- DREQ3
4
HLDA
HRQ
DACK0- DACK3
4
PRIORITY ENCODER AND ROTATING PRIORITY LOGIC
COMMAND (8)
MASK (4)
REQUEST (4)
READ WRITE
MODE (4 × 6)
STATUS (8)
TEMPORARY (8)
(a)
IOR 1
IOW 2
MEMR 3
MEMW 4
(NOTE 11) 5
READY 6
HLDA 7
ADSTB 8
AEN 9
HRQ 10
CS 11
CLK 12
RESET 13
DACK2 14
DACK3 15
DREQ3 16
DREQ2 17
DREQ1 18
DREQ0 19
(GND) V_{SS} 20
8237/ 8237-2
40 A7
39 A6
38 A5
37 A4
36 EOP
35 A3
34 A2
33 A1
32 A0
31 V_{CC} (+5V)
30 D80
29 D81
28 D82
27 D83
26 D84
25 DACK0
24 DACK1
23 DB5
22 DB6
21 DB7
(b)

The 8237 DMA Controller

- Ex: Memory-to-I/O DMA transfers
  - Only need one channel (since only 1 address).
  - I/O device requests transfer, provides source memory address
  - $\overline{\texttt{DACK}}$ from DMA is used to enable the I/O device (rather than decoding port address)
  - Channel selects memory address
  - $\overline{\texttt{MRDC}}$ and $\overline{\texttt{IOWC}}$ are pulled low simultaneously
  - Data is transferred from memory directly to I/O device.

The 8237 DMA Controller – Printer example



JK Flip flop configured to set on falling edge of $\overline{ACK}$

$\overline{ACK}$ from printer requests another byte (char).

Latch selected directly via $\overline{DACK_3}$ instead of decoding port number.

$\overline{IOWC}$ created by 8237 as is $\overline{MRDC}$ and mem address in this example.

$\overline{DS}$ to printer tells that new byte (char) is ready.

The 8237 DMA Controller

- Ex: Memory-to-memory DMA transfers
  - Channel 0: source address
  - Channel 1: destination address
  - A byte is read from address accessed by channel 0 an saved within the 8237
  - The 8237 writes the data byte to the address selected by channel 1
  - The number of bytes transferred is determined by the channel 1 count register
  - Memory-to-memory transfers are actually faster using the CPU (16-bit or 32-bit transfers…)

- Transfer Modes

1. Single Mode: release `HOLD` after each byte is transferred.
   - Often used when I/O can only provide 1 byte at a time (e.g. floppy disk controller) or is very slow.
2. Demand: Tx/Rx as long as DREQ is asserted.
   - Can also be interrupted by external $\overline{\text{EOP}}$ signal.
3. Batch/Block Mode: automatic Tx of a block of data until counter reaches zero.
4. Cascade: Master/Slave arrangement of more than one 8237.

Use of DMA in multi-processor environment