1) DMA operation
a) List all the events that come from the operation of the DMA channel itself (not including events from software) that can reset any of the pointers -- DCHxCPTR, DCHxDPTR, DCHxSPTR.

Pointer registers are updated with each transfer and are reset when the associated portion of the transfer (i.e. cell, source, destination or block) is "done".

The DCHxCPTR is reset when the cell done which is when the DCHxCPTR exceeds the DCHxCSIZ register.

The DCHxSPTR is reset when the source is done, and the block is done. This occurs when the SSIZ > DSIZ.

The DCHxDPTR pointer is reset when the destination is done, and block is done which occurs when the DSIZ > SSIZ.

b) List the DMA channel events that come from the operation of the DMA channel itself (not including events from software) that stop a channel but that won't cause pointer reset.

System interrupt matching CHAIRQ<7:0>: The channel event detect will be reset and the channel turned off. The abort interrupt flag is set. Pointers are not reset.

Address Error is detected: The channel is turned off and the event detect is reset. The address error interrupt flag is set. Pointers are not reset.

c) Describe why the designers probably arranged for events in part b) not to cause pointer reset.

There are kept intact for troubleshooting purposes. Since it is an error that causes this, we would want to be able to trace it back to where it occurred.

d) Since the pointer registers are read only, list the ways we could get them reset in the case of an event that didn't cause automatic reset?

Device Reset.

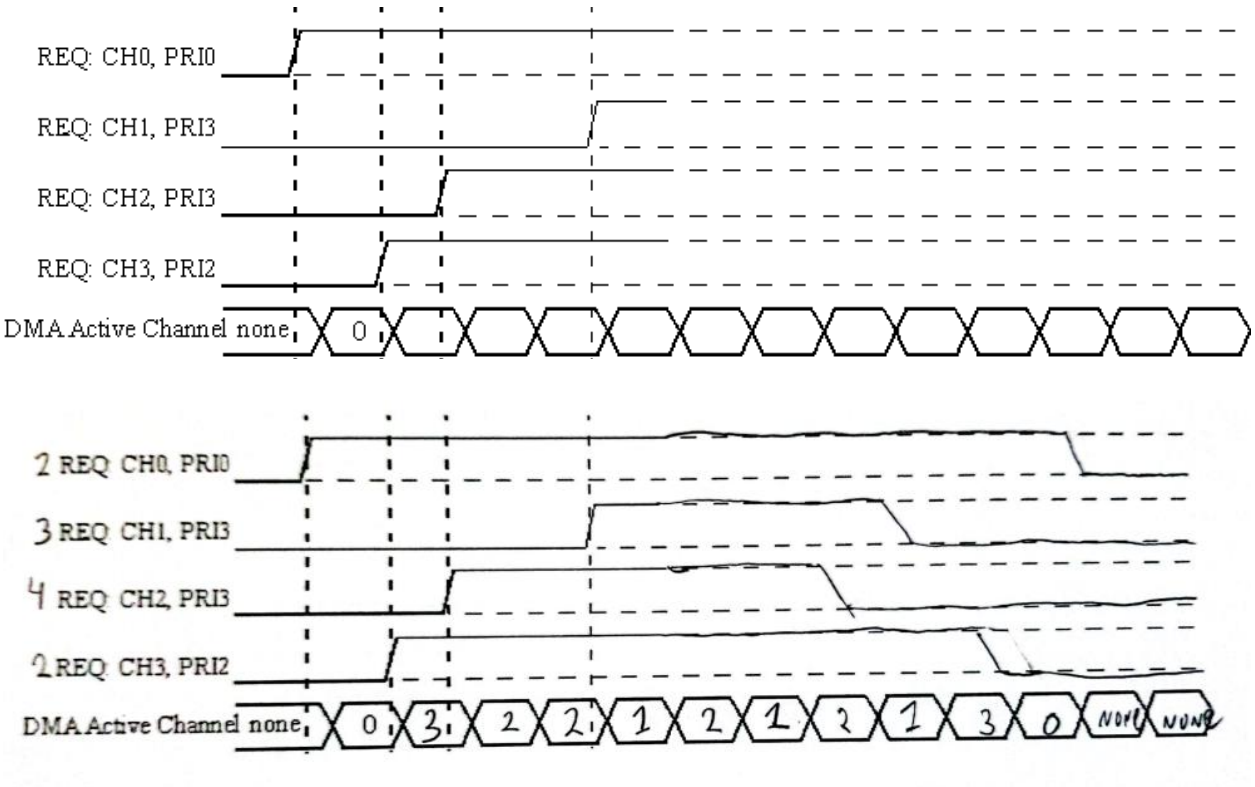
Turning the DMA off.

Source or destination start addresses are updated.

2) DMA configuration
A particular (hypothetical) device has registers containing **1** byte of data for each of the following parameters: the current temperature, the current pressure and the current humidity. These **3** bytes of information can be read from the memory map starting at virtual address 0xBF90.0010. (we could read all 3 bytes at once by reading a word (lw) from address 0xBF90.0010 or we could read the 3 bytes one byte at a time (lbu) from the sequential addresses 0xBF90.0010 through 0xBF90.0012.) We want to sample the temperature, pressure and humidity readings once per minute and store them in memory starting at virtual address 0x8000.0400 (using the least amount of memory possible). After an hour, we want to average them. We set up 32-bit timer 23 (timers 2 and 3 combined) to generate an interrupt every minute. Fill in the following blanks showing what settings we would need in order to be able to do the sampling with a DMA channel.

CSIZ ___**3**__ , DSIZ __**180**___ , SSIZ __**3**___ , SSA __**0x1F900010**_____ , DSA _**0x400**_____ , ECON.CHSIRQ __**0x14**__

3) Complete the following figure (like Figure 31-5) showing the order in which the DMA channels complete their transfers based on priority. Assume that Channel 0 needs 2, Channel 1 needs 3, Channel 2 needs 4 and that Channel 3 needs 2 transactions per request. Fill in the DMA Active Channel row, and show where each Channel's request drops.



4) Something like DMA chaining could be accomplished by having the ISR that responds to one channel's block complete interrupt enable another DMA channel. What advantage(s) does chaining provide over this approach?

It has the advantage of being handled by the DMA itself which reduces the amount of software overhead. It also allows for the DMA module to allow events while a channel is disabled so it can be ready to transfer as soon as the channel is enabled by the master channel.

5) How does the fact that the PIC32MZ UART has an 8 character receive buffer and an 8 character transmit buffer reduce the software overhead associated with multi-vector interrupt-based RS232 communication compared to a UART that only has a 1 character transmit and a 1 character receive buffer? Be specific -
a) list the sources of software overhead for multi-vector interrupts and

The prologue and epilogue plus all its instructions.

b) describe how the buffers can reduce it.

The buffers reduce this by only needing to do it once for 8 characters rather than 8 times.

6) DMA overhead and latency
a) How does DMA reduce software overhead over multi-vector interrupt driven I/O.

In the case of multi-vector interrupt with FIFOs it will incur the interrupt overhead once per k items. The DMA incurs interrupt overhead once per n items in a block and n can be > k.

b) How does DMA reduce latency over multi-vector interrupt driven I/O.? Be specific. List the sources of latency for Interrupt driven I/O and the sources of latency for DMA.

The DMA only needs to wait for the DMA hardware and to get access to the bus to do a transfer. Whereas a multi-vector interrupt driven I/O needs to do the following before a transfer.

Multi-vector interrupt driven I/O latency sources:

- **EIC/Core latency**
- **Prologue/Epilogue**
- **Need to complete current instruction or critical section**