**Assignment 1 HW:**

**Write a user space program to read the RTC, describe your findings:**

First we tried a simple C++ program that uses the “ /dev/rtc/” directory. As expected this worked like charm, although this was not the goal of the assignment.

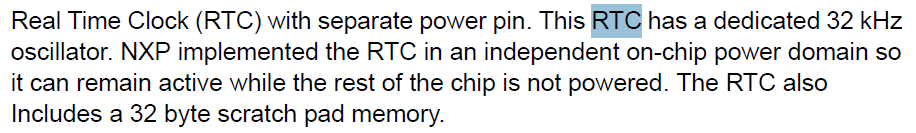
To read the RTC from user space we have to use the CMOS register. Some google results later we arrived at the following page for the CMOS RTC addresses : http://stanislavs.org/helppc/cmos\_ram.html and using this site we got some program examples: http://kernelx.weebly.com/cmos.html although this website was using DOS defined variables. Using the unix MANpages we found out that “ outportb == outb “ is for linux users.

After trying this for a long time and consulting the tutor and co-students it turned out that reading the CMOS is not the way to go, although we did learn some interesting facts, now we know we should read the registers and write them into a variable and try to print them.

To test this we are going to write a user space program that will write an **int**. We will print the memory address of this **int** and try to write to this memory address from a different user space program. The MMU (memory management unit) should not allow us to do this and give us a segmentation fault.

For example our first user space program created and reserved an **int** on the following memory address : 0xbfa835cc , when we tried to access this memory address or write on this address we received the segmentation fault we were waiting for.   
Now to confirm that it was not a program or code error, we tried reserving that same **int** on the memory and then altering it within the same user space program. This was allowed because we were only accessing our own segment of memory.

We tried this as well on the LCP3250, this had the same results as before. We could read the variable that we are assigned in our program, but when trying to read the reserved addresses on the board’s memory chip we encountered a segmentation fault.   
  
Now we wanted to try the same test with an address of a hardware register, for this we used the RTC UP counter address that was found on page 569 in the LCP data sheet. When we tried reading from this address on our laptop we got another segmentation fault this is because we have no idea if this hardware address is used or accessible or event points to the same register that we expect to find at that address.  
On the LCP3250 we were able to read the address successfully, but at first we got a negative value when reading the register. Then we made the variable an **unsigned long long** and were able to read the value successfully.   
  
Now for the reason that we are able to read this RTC register without the MMU interfering : We think a logical explanation is that the RTC is an entity that functions on its own. It has it owns power supply (0.9volt) and own SRAM (32bytes).

Partly confirming this thought is page 5 in the datasheet:  
  


*Image 1: page 5 LPC3250, RTC information*

After this we managed to at least verify that we can reach the memory but we still can’t read what is in it. The address that we now dereferenced was already mapped in the system page table so it was already linked to a physical address somehow. However as we are still operating from user space the value we request is most likely the mapping from virtual memory to the physical memory. So we requested the virtual address 0x40024000 which is in the standard user space range but it not yet claimed by another process. On our laptop this is the case because a lot of processes are running in the background.

**Assignment 2 : Write a kernel module that can read and write the hardware registers**

For this we started with reading the LKMPG chapter 6, so first we tried the helloWorld example.