mp4: Interrupt Driven I/O Driver

Assigned: 18 April 2017   **Due:** 9 May 2017

I. Objectives:

In this assignment you'll enhance a provided tutor program so that it can handle COM1 port I/O via a driver that uses the UART transmit and receive interrupts. You should carefully study the lecture on Introduction to MP4. See S&S Excerpts link and/or the UART data sheet for background technical information if you need it. Copy files from my mp4 to your mp4 directory.

II. The serial port interrupt:

Here is the new command in the supplied cmds.c. The code for the command itself is there, but it doesn't work yet because the rest of the functions are only stubs. Your job is to make it work as specified. None of this needs to be implemented for the UNIX version of tutor. You will only build and test the SAPC version.

spi <on|off> - stands for "serial port interrupt".

"spi on" enables interrupts on output to COM1. While the interrupts are enabled, the driver alternates between transmitting an application prompt to the user (TRANSMIT MODE) and receiving user entered data and passing it to the application (RECEIVE MODE).

TRANSMIT MODE
When a TX interrupt is detected, your program outputs the next character in the buffer received from the application. If that character is the NULL terminator of the string in the buffer, your program outputs the carriage return character (CR) instead of the NULL terminator character and executes a call to the callback function.

RECEIVE MODE
When an RX interrupt is detected, your program should echo each received character back to COM1, put the character into the application buffer, and check to see if it is the designated character for end of line (CR). If it is, you execute the application callback function passing the RX data line in the buffer as the argument.

"spi off" disables the interrupts.

The code for the "quit" command has also been enhanced

 q - Quit and go back to regular Tutor.

Disable any interrupts that have been left enabled. If you leave these interrupts enabled when you exit your tutor, you or the next student using your SAPC may get caught by unexpected interrupts occurring when the new downloaded code in the SAPC does not have interrupt handler code in the locations where the previously running version of tutor had them.

There are also a couple of new commands (timeon and timeoff) that you can study to see how a callback function works. The complete code for those commands is already in the cmds.c file and in the timepack.c file. You can use the timeon and timeoff commands to show that the timer and comport interrupts operate independently from each other and that the interrupt driven drivers for the timer chip and the UART don't interfere with the background PC-tutor code.

III. Software Architecture:

We will put the code for the COM1 port "driver" in its own source .c file so that tutor code doesn't need to know how it works. The API to the driver is provided in the comintspack.h file which is "included" in the tutor cmds.c code. See comintspack.h for the API to the COM1 port driver:

  void init_comints (int mode, void (*cb)(char *), char *buffer, int size);

        where mode: either TRANSMIT or RECEIVE.
              cb: the address of a call back function. You will provide a different callback function for transmit or receive mode.
          buffer: an array that contains a string to be transmitted or an empty array in which a string can be built from received characters depending on the mode value.
         size: the size of the array so that your code won't overflow it while reading and storing from the COM1 port in receive mode. There is no need to use this value in transmit mode. The null terminator in the array defines the end of the PROMPT string.

  void shutdown_comints (void);

   This function turns off both UART transmit and receive interrupts, and the IRQ4 Interrupt enable in the PIC.

IV. Expected Behavior:

See window1.txt and window2.txt for a run of my tutor.lnx with input on the COM1 line being passed through to the COM2 port. You will provide input on the COM1 port interacting with the SAPC as if it were a host computer and you were a user connected to a port (COM1) on that host.

V. Testing your COM1 port driver with the PC-tutor spi command

Download and run your tutor.lnx as you usually do ("mtip –f tutor.lnx", "~d", and "go 100100"). In a separate window, invoke the similar procedures as in setting up a remote gdb session. First, you find out the ip address for the vserver VM. Then you set up a ssh session with the vserver VM using putty or any SSH client.

> This is a letter l
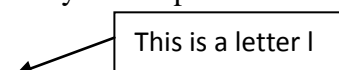
After logging into the vserver VM, issue the command "mtip –l /dev/ttyS0" where ttyS0 is the port number you are using for remote connection. However, because you are using it to provide input to your program, you will not be able to use it for remote gdb while debugging this program. With one window with mtip using COM2 and another with mtip attached to

COM1, you can test your program including its interrupt based driver.

When you enter the tutor command "spi on" on COM2, the program should print a prompt "Prompt:" to COM1. If it does not, enter one carriage return on COM1 to attract the attention of the driver. You should then get the prompt.

Each char you type in the second window is sent down the line to COM1 and each char sent to COM1 should be echoed back to the second window. After you hit enter, your ISR should call the callback function passing it the address of the buffer containing the input data. The call back function will just print it out in the first window. Don't forget to actually type something into the COM1 window when you want to test COM1 interrupts. The command mtip –l only sets up a communications channel to COM1. It doesn't send any test data for you.

After printing the received data on COM2, the program should output the prompt on COM1 again and be waiting for user input. If you enter the tutor command "spi off" on COM2, the program should stop printing prompts or accepting data entry on COM1.

You can continue to enter PC-tutor commands on COM2 port. They'll be processed normally. The interrupt driven COM1 port driver is multi-tasking its sequence of operations with the normal background operation of PC-tutor. In fact, you can also use the timeron command to start a timer interrupt that prints out (n) time ticks on the console interleaved with prompts from PC-tutor and data from the COM1 port. See the script files window1.txt and window2.txt or the lecture on Introduction to mp4 to see all of this happening at once.

VI: Turn in:

Write a discussion.txt describing how you tested your code and what interesting things you discovered while doing so. Include small portions of scripts showing output caused by the interrupt handlers and the callback functions to support your observations and your conclusions.

From your source directory (or your group's directory), capture and turn in hard copies of typescript files (system building and both COM2 and COM1 test windows). Since these typescripts files are in the vserver VM, you have to transfer them up to your CS341 mp4 course directory. The system building typescript file should show:

pwd
ls –l
cat comintspack.c
cat cmds.c
make clean
make tutor.lnx
cat typescript files showing test runs
cat discussion.txt

In the event that you are unable to correctly complete this assignment by the due date, submit what you have done and do not remove the work you were able to accomplish - partial credit is always better than none.