Linux driver assignments – 6
serial char device-driver – polling-mode

- you must pass a module parameter for the base address(starting address) of the serial port that you wish to test

- you must also pass as module parameters for other values – polling interval, kfifo size and any other that you feel is needed

- in the init() routine of your driver, allocate the device object structure and also call request_region() to lock the port and its respective addresses refer to the sample code(serial_class1.c) for the syntax and usage

- in the open() of your driver, initialize the UART controller in the normal mode (loop-back must be disabled) and as per settings used in the sample codes (serial_class1.c) and other pdfs provided for serial UART controller

Linux driver assignments – 6

serial char device driver – polling-mode .....continued

- you must also implement blocking/non-blocking operations in your read() and write() methods of your driver – the rules are same, as given on pages 151-152 in chapter 6 of your book
- you may use kfifo(s) in your driver to do intermediate buffering of data- one kfifo may be needed for Tx(transmission) and one for Rx(reception)
- you have to implement as many software -timer structures, corresponding routines and schedule them as needed – especially, the Rx timer-routine must be scheduled frequently such that Rx h/w buffer is not overrun - make necessary timing decisions as per your requirement
- for example, if you try to write data in the write method of the driver, you will dump as much data in the kfifo(Tx), schedule a software-timer structure/routine and return the total no. of bytes written to Tx kfifo – the timer handler will do the transfer of data stored in Tx kfifo to the device, when Tx Timer handler is executed - it may wake up the process that is blocked,if any, that tried to write when there is no space in Tx kfifo

Linux driver assignments – 6

serial char device driver – polling-mode .....continued

- for example, in the case of RX, there will be a software-timer structure/routine that will run periodically and copy data from Rx h/w buffer to the kfifo(Rx) and wake-up the process that is blocked, if any, that tried to read an empty Rx kfifo
- your write() method must only write to kfifo(Tx) – it is the responsibility of the Tx timer handler to write to the Tx h/w buffer, when Tx timer handler is executed and there is free space in the Tx h/w buffer
- in your read() method, if your kfifo(Rx) is empty, you must block
- in your read() method, if your kfifo(Rx) is not empty, start reading from the kfifo first as much as possible – return the value of total no. of bytes successfully read
- in your write() method, if Tx kfifo is full(no space), you must block the current process / thread
- in your write() method, if Tx kfifo is not full(there is space), you must write as much data as possible to Tx kfifo and return the value of no. bytes successfully written
- test your driver for different baud rates – lower to higher rates

---