Assignment 6 – driver part -
device driver for platform serial (UART) device(s)

- note that this is the driver part for serial UART device(s), which are being managed as platform device(s) – refer to Assignment 6 – device part document for registering serial UART devices, as platform devices

- in our driver module, we will require following methods and objects :
  - yyy_init() and yyy_exit() methods of the module
  - yyy_probe() and yyy_remove() methods to passed to our struct platform_driver {} object instance
  - yyy_open(), yyy_release(), yyy_write(), yyy_read() and yyy_isr() methods to be passed to our struct file_operations {} object instance
  - we need as many private objects, as the no. of serial UART device instances enumerated during yyy_probe() !! as per the device model, yyy_probe() will invoked for as many platform devices, whose names with that of the name stored in platform_driver{ }

Assignment 6 – driver part -
platform device driver for platform serial (UART) device(s).....

Note :read the entire problem statement two or three times,
before planning for implementation

- further parts of this assignment describe the design and implementation of the required methods as per the rules of device frame-work, interrupt frame-work and hw controller's characteristics

- you must pass as module parameters for driver/device parameter values – for instance, baud-rate and intermediate Tx/Rx kfifos sizes

- whenever your yyy_probe() method of your platform driver{ } is invoked for an enumerated device, collect resources of the specific device instance, allocate and initialize private device object structure and also call request_region() to lock the respective addresses - refer to the sample codes under platform_sample/ folder for the syntax and usage -

- you must do the reverse, in yyy_remove() method of your platform_driver{ }

Assignment 6 – platform device driver for platform serial (UART) device(s) ...

- following must be in yyy_open() of file_operations { } object :

  - disable hw interrupt events using IER and clear all status registers
  - initialize the UART controller in the normal mode(loop-back must be disabled) and as per settings used in the sample codes (serial_class.c) – you must also enable interrupt operation in h/w controller - ISR installation must be done before enabling hw interrupt event notifications from the hw controller

  - you must enable the appropriate bits in the IER register such that interrupt signal is asserted for Rx FIFO is non-empty (has data) condition , in yyy_open() method – you must not enable Tx FIFO empty condition interrupt, in yyy_open() method – Tx FIFO empty condition interrupt must be maintained as disabled, in yyy_open() method !!!

  - we may enable Tx FIFO empty condition interrupt in write() method of char driver file_operations{}, as needed and further, disable it in yyy_isr if needed

Assignment 6 – driver -part -
platform device driver for platform serial (UART) device(s) ...

- you must do the reverse of actions taken in yyy_open(), in yyy_release() method of your file_operations{ } - specifically, must disable hw interrupts, uninstall ISR and make sure that your device is quiet !!!

- you must do the following, in yyy_write() method of file_operations{} :
  - we must maintain a Tx kfifo for intermediate buffering per device instance and wait queue head per device instance, in the private object of the device instance
  - in your driver's write method, write as much data into the intermediate Tx kfifo and return the no. of bytes successfully written to the user-space;
  - if there is data in the Tx intermediate buffer of the device instance, Tx hw fifo empty interrupt event must be enabled – otherwise, Tx hw fifo empty interrupt for this device instance must be maintained as disabled
  - if there is no space in the intermediate Tx kfifo, block the current process/ thread, in the write wait queue of this device instance

Assignment 6 – driver part -
platform device driver for platform serial (UART) device(s) ...

- when will the data stored in intermediate Tx kfifo will be transferred to Tx h/w kfifo ?? when Tx h/w fifo is empty, UART controller will raise an interrupt event and our driver's/device's yyy_ISR will be executed – it is the responsibility of ISR to check LSR for Tx h/w buffer empty status and copy as much data as possible from Tx kfifo to Tx h/w buffer
- it is also the responsibility of Tx part of the yyy_ISR to disable Tx hw empty interrupt event notification, if there is no data left in the Tx intermediate kfifo
- you must implement blocking/non-blocking operations in your yyy_write() method of your driver – the rules are same, as given on pages 151-152 in chapter 6 of LDD/3
- in addition, you must also wake-up any blocked writer processes from ISR

Assignment 6 – driver part -

platform device driver for platform serial (UART) device(s) ...

- you must do the following, in yyy_read() method of file_operations{} :
  - whenever yyy_read() method is invoked and if your kfifo(Rx) is empty, current thread/ process will be blocked in the read wait queue of this Device instance
  - if your kfifo(Rx) is not empty, start reading from the kfifo as much as possible and return the no. of bytes copied to user-space buffer
  - whenever new data has arrived in Rx hw fifo, hw controller will generate an interrupt event – Rx part of the yyy_isr() method will copy as much data as possible into Rx intermediate kfifo – if there is no space in Rx intermediate kfifo, Rx part of yyy_isr() will drop data
  - it is the responsibility of yyy_isr() to wake up any processes that are blocked in read wait of device instance

- you must do the following, in yyy_isr() :
  - you must follow what is mentioned in the above descriptions of yyy_write() and yyy_read() methods
  - you must follow all the rules of isr coding of interrupt frame-work
  - you must follow the rules of your hw controller, which is better described in data-sheet(s)

h

Assignment 6 – driver part -
platform device driver for platform serial (UART) device(s) ...

- read this assignment statement twice or thrice - prepare your logic on a piece of paper

- next, proceed with coding your driver – start with yyy_init() and yyy_exit() routines; follow that with yyy_probe() and yyy_release() ; follow that with yyy_open() and yyy_release() ; follow that with yyy_write() and yyy_read() ; follow that with yyy_isr() coding

- write a test application similar to what you did for assignment 5, with appropriate changes

- test your driver, using your test application !!!

h

Assignment 6 – driver part -
platform device driver for platform serial (UART) device(s) ...

- read class note relevant to interrupt handling / ISRs and serial / UART driver / serial_class.c / serial_class1.c / samples under platform_sample/
- read the interrupt internals related slides/pdf
- read chapter 7 of LKD/3 for interrupt handling theory
- read chapter 9 of LDD/3 for hw controller access macros
- read data-sheet for serial/UART controller
- read UART primer slides/pdf
- read serial_hw_hints1.txt for recommended controller initialization steps
- read this assignment problem, whenever needed
