USART– Tutorial Exercise

In this tutorial you are asked to use the USART1 of the STM32F10x to implement a simple code that:

(1) Reads two numbers **base** and **exp**,
(2) Computes **base** to the power **exp**,
(3) Writes back the answer: **result**=(**base**)^{exp}.

You are expected to use the interrupt-driven method for managing the serial communication.

You have FOUR tasks in this tutorial:
Task 1: Launch the basic Interrupt-driven USART project and, during a debug session, track the operation of enqueuing and dequeuing.
Task 2: Compare this implementation to the Polling-drive USART project covered in the lecture. Can you think of an advantage of using the Interrupt method? Polling method?
Task 3: Write a C code that implements the function described above,
Task 4: Test your code in the simulation platform Keil.

Task #1

- Examine the USART setting after initialisation.
- Examine the buffer (or Queue) setting after initialisation.
- Examine the changes in the USART1_DR and the TX buffer while printf("….") is executing.
- Examine the changes in the USART1_DR and the RX buffer while getchar() is executing.
- Monitor the timing between two consecutive characters appearing in the USART window and interpret the result.

Task #2

- Which implementation is simpler?
- Which implementation is more efficient in terms of taking advantage of microprocessor time?
- Is there any microprocessor activity taking place between two interrupts?
- What happens in the polling-driven implementation between the transmission of two characters over serial communication?
- Which implementation do you recommend?

Task #3

```
int main (void) {
// initialise variables
// init RX / TX buffers
// STM32 setup
  printf ("Interrupt driven Serial I/O Example\r\n");
  printf ("Hello! If you give me two numbers a and b I will give you the value of a to
the power b. Let's play!\r\n");

  while (1) {    // Loop forever
//write your code here
  } // end while
} // end main
```

Task #4

Test the code in Keil.

- End of Tutorial Exercise -