Lab Session : 12

Creating Multiple Serial Port with RTOS

Name : Durvesh Naresh Patil
PRN : 2019BTEEN00035
Batch : EN-1
Sub. : RTOS - Lab

Theory :

Communication : Exchanging the data between two systems

Serial communication : Data is sent bit by bit i.e. one bit at a time.

In serial communication, the data is in the form of binary pulses. LPC2138148 has only one serial port. We need to use multiple serial port so here we are sending data via GPIO to the virtual terminal in the binary form.

Baud Rate : Two microcontrollers should have same baud rate for proper serial communication.

Baud rate is the number of bits transferred from receiver to sender per sec. Some std. baud rates are 1200, 2400, 4800, 9600, 57600.

Higher baud rate, more data transferred in less amount of time

Asynchronized data transfer : When data bits are not synchronized with clock line

For asynchronized data transfer, baud rate is important.

Code:

```
#include "config.h"
#include "stdlib.h"
#include <stdio.h>

#define TaskStkLengh   64                       //Define the Task0 stack length

OS_STK        TaskStk0 [TaskStkLengh];          //Define the Task stack
OS_STK        TaskStk1 [TaskStkLengh];          //Define the Task stack
OS_STK        TaskStk2 [TaskStkLengh];          //Define the Task stack
OS_STK        TaskStk3 [TaskStkLengh];          //Define the Task stack

void     Task0(void *pdata);
void     Task1(void *pdata);
void     Task2(void *pdata);
void     Task3(void *pdata);

char buffer[25];


int main (void)
{
        LED_init();
        TargetInit();
        OSInit ();

        OSTaskCreate (Task0,(void *)0, &TaskStk0[TaskStkLengh - 1], 6);
        OSTaskCreate (Task1,(void *)0, &TaskStk1[TaskStkLengh - 1], 7);
        OSTaskCreate (Task2,(void *)0, &TaskStk2[TaskStkLengh - 1], 8);
        OSTaskCreate (Task3,(void *)0, &TaskStk3[TaskStkLengh - 1], 9);

        OSStart();
        return 0;

}
```

```
/*
        ASCII Format: G T A 5
        Binary Format: 01000111 01010100 01000001 00110101
        Write Binary values of each character from LSB to MSB in Tasks
*/

void Task0      (void *pdata)
{
        pdata = pdata;
                                                        /* Dummy data */
        LED_on(0);
        OSTimeDly(10);

        while(1)
        {
                LED_off(0);     //Start Bit
                OSTimeDly(1);

                LED_on(0);              //1
                OSTimeDly(1);

                LED_on(0);              //1
                OSTimeDly(1);

                LED_on(0);              //1
                OSTimeDly(1);

                LED_off(0);             //0
                OSTimeDly(1);

                LED_off(0);             //0
                OSTimeDly(1);

                LED_off(0);             //0
```

```
                OSTimeDly(1);

                LED_on(0);              //1
                OSTimeDly(1);

                LED_off(0);             //0
                OSTimeDly(1);

                LED_on(0);              //Stop Bit
                OSTimeDly(1);

                OSTimeDly(10); //empty period



        }
}


void Task1      (void *pdata)
{
        pdata = pdata;
                                                /* Dummy data */
        LED_on(1);
        OSTimeDly(10);

        while(1)
        {
                LED_off(1);     //Start Bit
                OSTimeDly(1);

                LED_off(1);             //0
                OSTimeDly(1);

                LED_off(1);             //0
```

```
            OSTimeDly(1);

            LED_on(1);          //1
            OSTimeDly(1);

            LED_off(1);         //0
            OSTimeDly(1);

            LED_on(1);          //1
            OSTimeDly(1);

            LED_off(1);         //0
            OSTimeDly(1);

            LED_on(1);          //1
            OSTimeDly(1);

            LED_off(1);         //0
            OSTimeDly(1);

            LED_on(1);          //Stop Bit
            OSTimeDly(1);

            OSTimeDly(10); //empty period
        }
}

void Task2      (void *pdata)
{
        pdata = pdata;

        LED_on(2);
        OSTimeDly(10);

        while(1)
```

```
	{
		LED_off(2);	//Start Bit
		OSTimeDly(1);

		LED_on(2);		//1
		OSTimeDly(1);

		LED_off(2);		//0
		OSTimeDly(1);

		LED_off(2);		//0
		OSTimeDly(1);

		LED_off(2);		//0
		OSTimeDly(1);

		LED_off(2);		//0
		OSTimeDly(1);

		LED_off(2);		//0
		OSTimeDly(1);

		LED_on(2);		//1
		OSTimeDly(1);

		LED_off(2);		//0
		OSTimeDly(1);

		LED_on(2);		//Stop Bit
		OSTimeDly(1);

		OSTimeDly(10); //empty period
	}
}
```

```
void Task3      (void *pdata)
{
        pdata = pdata;

        LED_on(3);
        OSTimeDly(10);

        while(1)
        {
                LED_off(3);     //Start Bit
                OSTimeDly(1);

                LED_on(3);              //1
                OSTimeDly(1);

                LED_off(3);             //0
                OSTimeDly(1);

                LED_on(3);              //1
                OSTimeDly(1);

                LED_off(3);             //0
                OSTimeDly(1);

                LED_on(3);              //1
                OSTimeDly(1);

                LED_on(3);              //1
                OSTimeDly(1);

                LED_off(3);             //0
                OSTimeDly(1);
```

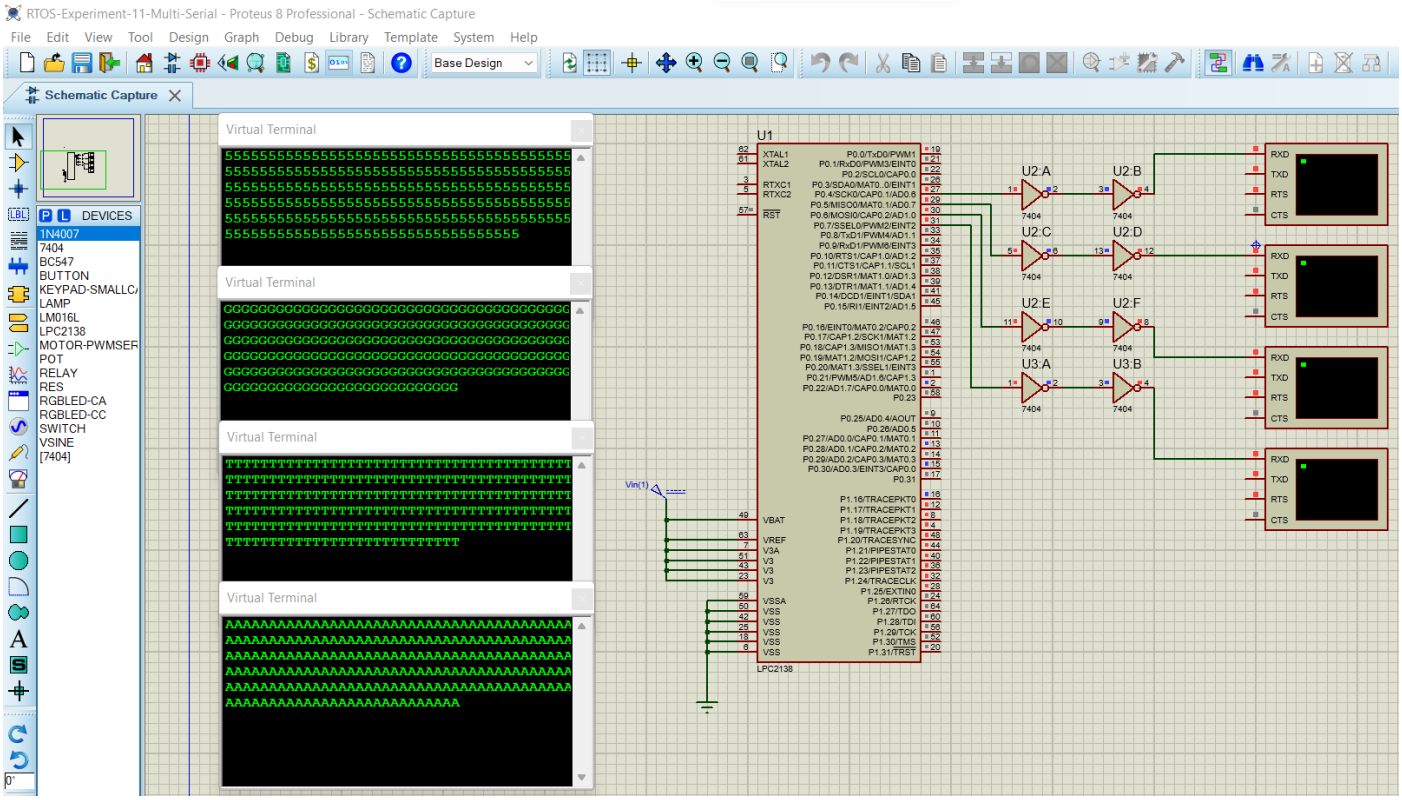
```
            LED_off(3);                 //0
            OSTimeDly(1);

            LED_on(3);                  //Stop Bit
            OSTimeDly(1);

            OSTimeDly(10); //empty period
        }
}
```

Observations:

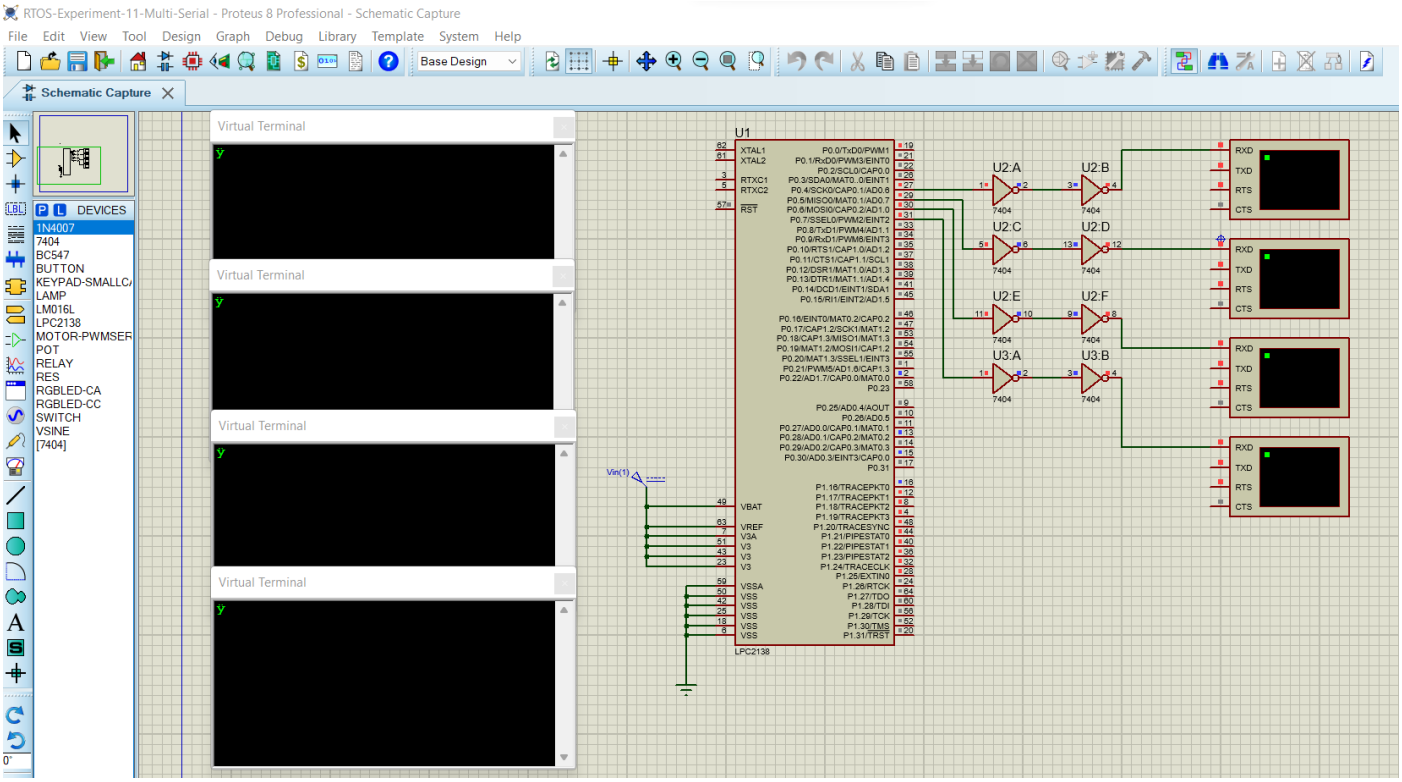
1) When baud rate is same (OS_TICKS_PER_SEC = 2400, Virtual Terminal = 2400)



Comments:

When baud rate is same, proper data got received at virtual terminal.

2) When baud rate is different (OS_TICKS_PER_SEC = 100, Virtual Terminal = 2400)



Comments:

When baud rate is not same, garbled data got received at virtual terminal.