Lab Session - 6

Using semaphore for event to task synchronization

Name : Durvesh Naresh Patil
PRN : 2019BTEEN00035
Batch : EN-1

7.2.1 Task0 with External Interrupt 1

Observations

1) Run the program & observe that, when the interrupt is given, the program reaches the breakpoint in ISR. Paste the screenshot

EventTaskSync.c

```c
}
//ISR of Ext int 1.
// p0.3 used as interrupt pin
char flag=0;
void isr_int1(void)__irq
{
    OSSemPost(pTask0EventSync);
        //clear the flag
    EXTINT = 0x00000002;
    VICVectAddr= 0x0;
}
//ISR of Ext int 2.
// p0.15 used EINT2
void isr_int2(void)__irq
{
    //write OSSempost for other semaphore
    //  OSSemPost(pTask1EventSync);
    EXTINT = 0x00000004;
    VICVectAddr= 0x0;
}
```

2) Write your comments.

⇒ When interrupt is given through port 0.3 the ISR associated with start executing i.e. program control goes to the isr_int1. semaphore is unavailable until the interrupt occurs. Task can do its job then semaphore value is incremented

3) If the program is run further, observe that it goes to Task0.

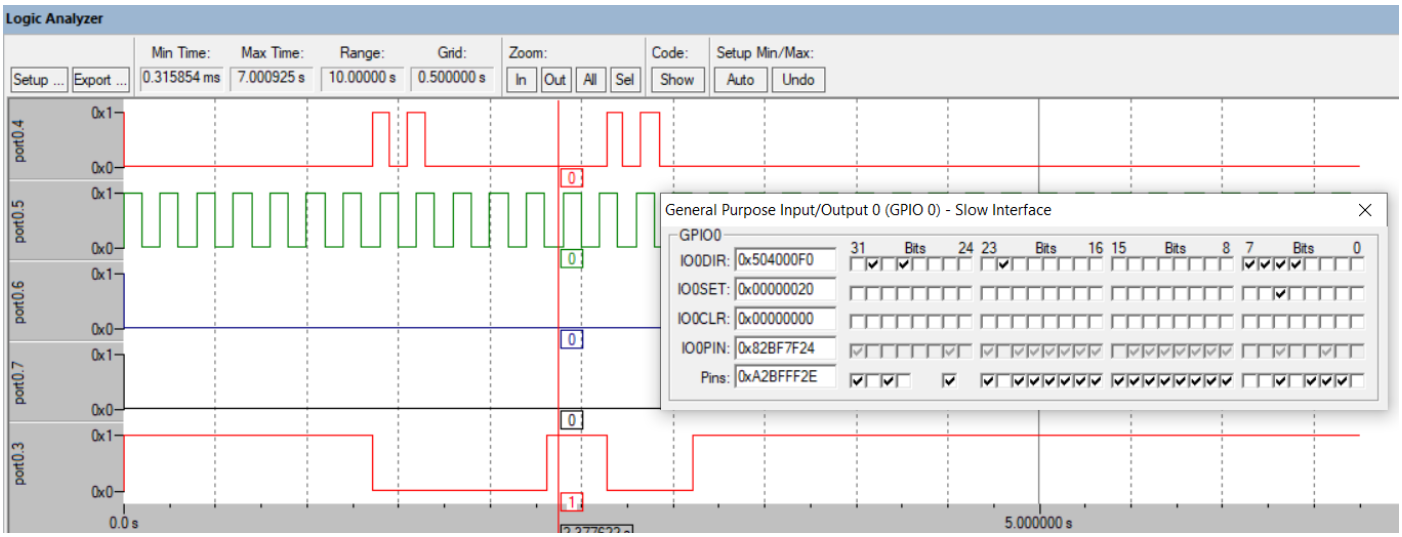
EventTaskSync.c

```c
    return 0;
}
void Task0  (void *pdata)
{
    pdata = pdata;                                  /* Dummy data */

    while(1)
    {
        OSSemPend(pTask0EventSync,0,&err);
        LED_on(0);  // All LEDs on
        OSTimeDly(4);


        LED_off(0); // All LEDs off
        OSTimeDly(4);
        LED_on(0);  // All LEDs on
        OSTimeDly(4);


        LED_off(0); // All LEDs off
```

4) Write your comments

⇒ Once the post function of semaphore executed, semaphore is available & then event can occur so the program goes to the Task0 after isr finish its job.

5. Show the interrupt pin on logic analyzer and take screenshot of interrupt and LED flashing together. Paste the screenshot.



6. Write your comments

Task1 is executing normally as expected and Task0 is executing only on the occurrence of the interrupt at port P0.3

7.2.2 Task1 with External Interrupt 2

Observations

1. Run the program and Observe that, when the interrupt is given, the program reaches the breakpoint in ISR. Paste the screenshot.

```
}
//ISR of Ext int 2.
// p0.15 used EINT2
void isr_int2(void)__irq
{
    //write OSSempost for other semaphore
    OSSemPost(pTasklEventSync);
    EXTINT = 0x00000004;
    VICVectAddr= 0x0;
}

// P0.3 used as ENT1 (PINSEL:11)
//P0.15 used as EINT2 (PINSEL:10)
```

2. Write your comments

As soon as we give interrupt using the port P0.15, the flow of the program will go from Task1 to the isr_int2 to perform that job first.

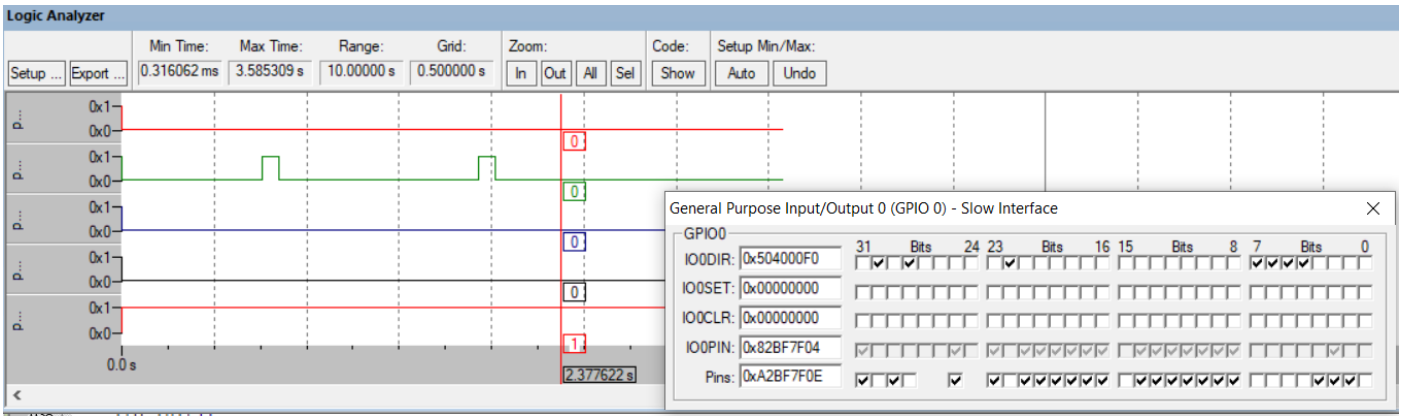
3. If the program is run further, observe that, it goes to Task0. Paste the screenshot.

```
}
void Task1  (void *pdata)
{
    pdata = pdata;

    while(1)
    {
        OSSemPend(pTask1EventSync,0,&err1);
        LED_on(1);  // All LEDs on


        OSTimeDly(4);
        LED_off(1); // All LEDs off

        OSTimeDly(4);
    }
}
```

4. Write your comments

As soon as the isr finishes its job, the control of the execution of the program will get back to the task in which it is called. In above example, when isr_int2 finishes its job, execution point will go back to the task1.

5. Show the interrupt pin on logic analyzer and take screenshot of interrupt and LED flashing together. Paste the screenshot.



6. Write your comments

Whenever interrupt will occur at port P0.3 then task0 will execute and when interrupt occur at the port P0.15 then task1 will execute.

7.3 Conclusion

1) What is the overall conclusion of this experiment?

⇒ (i) Whenever interrupt occurs program executes the isr first then executes the remaining job in that task

(ii) Using semaphore we can achieve event to task synchronization.

(iii) In isr semaphore will be released then only the remaining task execution take place

2) Write about what did you learn by performing this experiment

⇒ (i) Events / interrupts

(ii) How to synchronize events to tasks

(iii) How to use semaphore in real world.