Abhishek kumar simra
PRN:-220940130002

Program To get interrupt notification and delete task

This program objective is to create 3 tasks, then get interrupt from the user and send notification to other tasks.

SOLUTION:

```
xTaskCreate( task1, "Task1", 200, NULL, 0, &xTask1 );
xTaskCreate( task2, "Task2", 200, NULL, 1, &xTask2 );
xTaskCreate( task3, "Task3", 200, NULL, 2, &xTask3 );

vTaskStartScheduler();
```

here i have created created three tasks, task1 with lowest priority 0, task2 with medium priority 1, and task3 with highest priority 2 among all the three tasks.

Then the scheduler is declared to to start the tasks.

```
void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin)
{
        HAL_GPIO_TogglePin(GPIOD, GPIO_PIN_15);
        //vTaskResume(xTask1);

        vTaskNotifyGiveFromISR(xTask1,pdTRUE);

}
```

The interrupt call back function which gives notification to task1

The above code shows the external interrupt from the user
when the interrupt is received from from the button the a Notification is send to task1 using "vTaskNotifyGiveFromISR()", which takes task handler to be notified and the second is Highest priority is kept pdTrue which describes that the task which is best

Task1

```
void task1(void *ptr1){
        while(1){
                TickType_t delay =pdMS_TO_TICKS(100);
                ulTaskNotifyTake( pdTRUE, portMAX_DELAY );
                HAL_GPIO_WritePin(GPIOD, GPIO_PIN_12, GPIO_PIN_SET);

                xTaskNotifyGive( xTask2 );
                vTaskDelay(delay);

                HAL_GPIO_WritePin(GPIOD, GPIO_PIN_12, GPIO_PIN_RESET);
                vTaskDelete(NULL);

        }
}
```

->The task1 is the lowest priority task in the program
->in the starting a delay is declared of ticktype_t which stores the value of the pdMS_TO_TICKS()
->the pdMS_TO_TICKS is a macro used to convert the time specification in milliseconds to time specified in ticks
->After that i have used ulTaskNotifyTake which is a API function used to take the the notification from the sender who is giving the notifcation
->i have used pdTrue and portMAX_DELAY as the parameters of the ulTaskNotifyTake, the pdTRUE set a condition that if a notification is received, it will reset the API to zero and exit,
->the next parameter is the portMAX_DELAY which is a macro that create the API to wait for indefinitely for the notification, as i was not clear about how much time it will take to get notification, we can also set a definite time if clear about time.
->then I have used the vTaskNotifyGive() API, it is a API used to send Notification to a specified task in this case I have passed xTask2 as it was the task to be notified.
->then a delay was used to block the task .
->then the vTaskDelete API was use to delete the function and NULL was used as a parameter as we have to delete the same task, and if any other task is to be deleted its name is to be passed.

Task2

```
void task2(void *ptr2){
        while(1){
                TickType_t delay =pdMS_TO_TICKS(100);
                ulTaskNotifyTake( pdTRUE, portMAX_DELAY );
                HAL_GPIO_WritePin(GPIOD, GPIO_PIN_13, GPIO_PIN_SET);

                xTaskNotifyGive( xTask3);
                vTaskDelay(delay);

                HAL_GPIO_WritePin(GPIOD, GPIO_PIN_13, GPIO_PIN_RESET);
                vTaskDelete(NULL);

        }
}
```

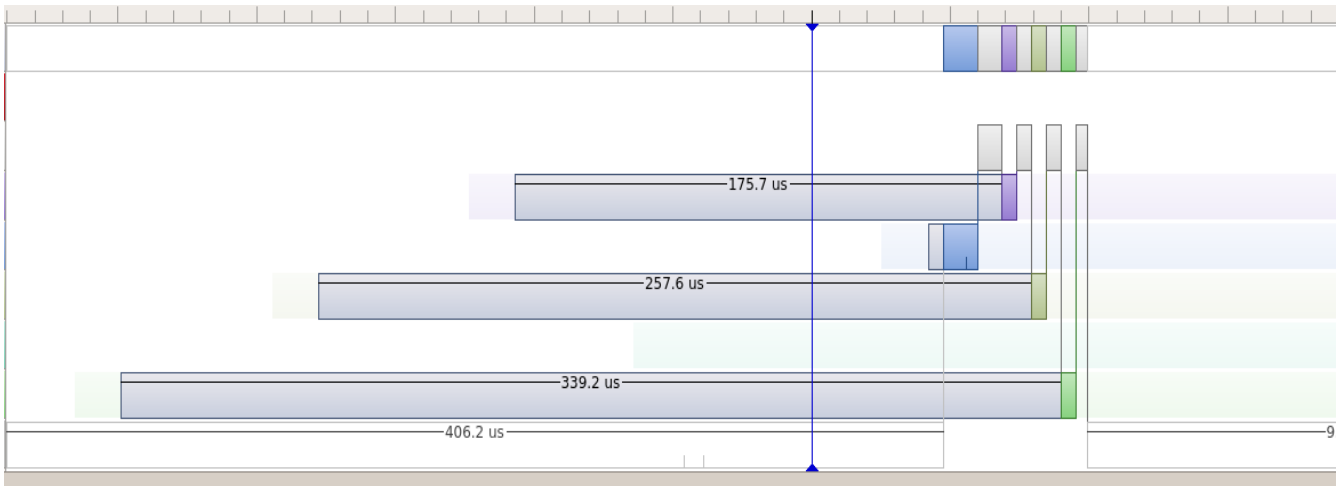
->The task2 is the medium priority task in the program
->in the starting a delay is declared of ticktype_t which stores the value of the pdMS_TO_TICKS()
->the pdMS_TO_TICKS is a macro used to convert the time specification in milliseconds to time specified in ticks
->After that i have used ulTaskNotifyTake which is a API function used to take the the notification from the sender who is giving the notifcation
->i have used pdTrue and portMAX_DELAY as the parameters of the ulTaskNotifyTake, the pdTRUE set a condition that if a notification is received, it will reset the API to zero and exit,
->the next parameter is the portMAX_DELAY which is a macro that create the API to wait for indefinitely for the notification, as i was not clear about how much time it will take to get notification, we can also set a definite time if clear about time.
->then I have used the vTaskNotifyGive() API, it is a API used to send Notification to a specified task in this case I have passed xTask3 as it was the task to be notified.
->then a delay was used to block the task .
->then the vTaskDelete API was use to delete the function and NULL was used as a parameter as we have to delete the same task, and if any other task is to be deleted its name is to be passed.

Task3

```
void task3(void *ptr2){
        while(1){
                TickType_t delay =pdMS_TO_TICKS(100);
                ulTaskNotifyTake( pdTRUE, portMAX_DELAY );
                HAL_GPIO_WritePin(GPIOD, GPIO_PIN_14, GPIO_PIN_SET);
                vTaskDelay(delay);
                HAL_GPIO_WritePin(GPIOD, GPIO_PIN_14, GPIO_PIN_RESET);

                vTaskDelete(NULL);
        }
}
```

->The task3 is the highest Priority task in the program
->in the starting a delay is declared of ticktype_t which stores the value
of the pdMS_TO_TICKS()
->the pdMS_TO_TICKS is a macro used to convert the time specification
in milliseconds to time specified in ticks
->After that i have used ulTaskNotifyTake which is a API function used to take
the the notification from the sender who is giving the notifcation
->i have used pdTrue and portMAX_DELAY as the parameters of the
ulTaskNotifyTake, the pdTRUE set a condition that if a notification is
received, it will reset the API to zero and exit,
->the next parameter is the portMAX_DELAY which is a macro that create the
API to wait for indefinitely for the notification, as i was not clear about how
much time it will take to get notification, we can also set a definite time
if clear about time.
->Then the vTaskDelete API was use to delete the function and NULL was used
as a parameter as we have to delete the same task, and if any other task
is to be deleted its name is to be passed.



1.Starting part of trace

25	0.000 401 077	Idle	Task Ready	Tmr Svc, runs after 5.161 us
26	0.000 406 238	Tmr Svc	Task Run	Runs for 12.708 us
27	0.000 414 839	Tmr Svc	vTaskDelayUntil	
28	0.000 418 946	Tmr Svc	Task Block	Reason=27
29	0.000 427 292	Task3	Task Run	Runs for 5.690 us
30	0.000 432 982	Task3	Task Block	Reason=27
31	0.000 438 202	Task2	Task Run	Runs for 5.369 us
32	0.000 443 571	Task2	Task Block	Reason=27
33	0.000 448 774	Task1	Task Run	Runs for 5.292 us
34	0.000 454 065	Task1	Task Block	Reason=27
35	0.000 458 327	Idle	System Idle	Idle for 139.924 ms
36	0.001 411 137	SysTick	ISR Enter	Runs for 3.625 us

->I have observed from the trace that all task got in the ready state and when the vTaskSchedular inwokes the SVC, the SVC get ready and run and then sends the controll to the schedular

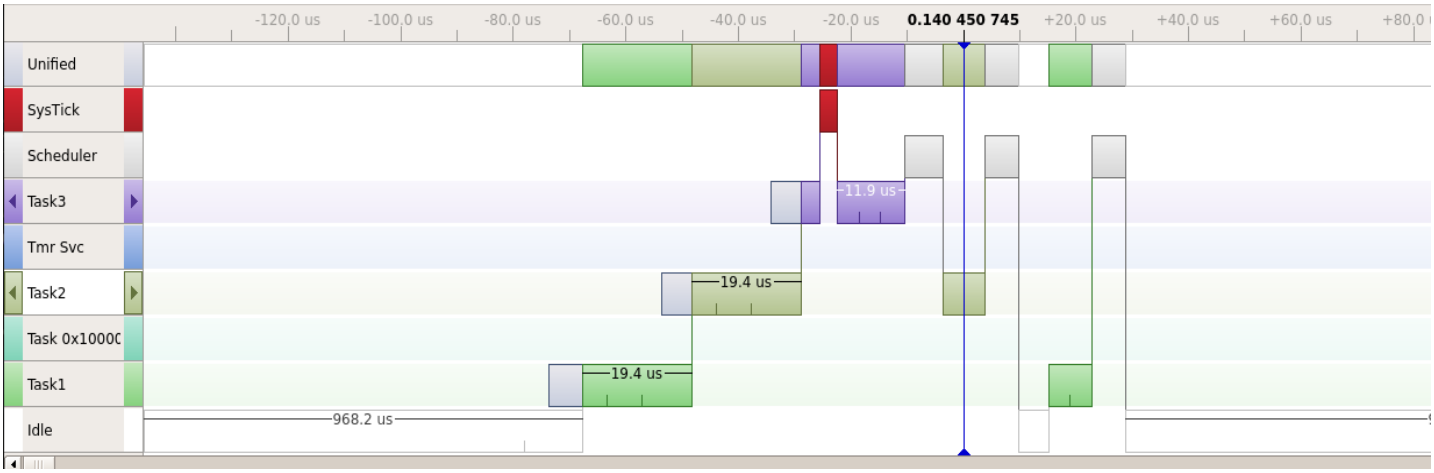
->the schedular schedules the highest priority task which in this case was task3

->the task3 start to run but gets blocked as it needs a notification to execute forward since it does not got notified it returns thecontroll to the schedular

->then the schedular schedules the next highest priority task which is task2

->the task2 gets the controll and start to run but get blocked as it needs a notification to execute forward, since no notification received the controll is send back to the schedular, then the schedular schedules the next highest priority task which is task1

->then task1 start to run and it also gets blocked as it also needs a notification to execute forward, since no notification is received it sends back the controller to the schedular, since there are no task left the system goes to the idel state



2.Second part of the trace(interrupt generated)

0.140 376 893	Idle	Task Ready		Task1, runs after 6.024 us
0.140 382 917	Task1	Task Run		Runs for 19.429 us
0.140 387 196	Task1	ulTaskNotifyTake		xClearCountOnExit=1 xTicksToWait=4294967295
0.140 393 321	Task1	xTaskGenericNotify	• Task2	xTaskToNotify=Task2 ulValue=0 eAction=2 pulPreviousNotificationValue=0
0.140 396 935	Task1	Task Ready		Task2, runs after 5.411 us
0.140 402 345	Task2	Task Run		Runs for 19.429 us
0.140 406 619	Task2	ulTaskNotifyTake		xClearCountOnExit=1 xTicksToWait=4294967295
0.140 412 738	Task2	xTaskGenericNotify	• Task3	xTaskToNotify=Task3 ulValue=0 eAction=2 pulPreviousNotificationValue=0
0.140 416 363	Task2	Task Ready		Task3, runs after 5.411 us
0.140 421 774	Task3	Task Run		Runs for 18.268 us
0.140 424 940	SysTick	ISR Enter		Runs for 3.119 us
0.140 428 060	SysTick	ISR Exit		Returns to Task3
0.140 431 982	Task3	ulTaskNotifyTake		xClearCountOnExit=1 xTicksToWait=4294967295
0.140 435 851	Task3	vTaskDelay		xTicksToDelay=100
0.140 440 042	Task3	Task Block		Delayed
0.140 446 839	Task2	Task Run		Runs for 7.815 us
0.140 450 589	Task2	vTaskDelay		xTicksToDelay=100
0.140 454 655	Task2	Task Block		Delayed
0.140 460 554	Idle	System Idle		Idle for 5.387 us
0.140 465 940	Task1	Task Run		Runs for 7.625 us
0.140 469 560	Task1	vTaskDelay		xTicksToDelay=100
0.140 473 565	Task1	Task Block		Delayed
0.140 479 613	Idle	System Idle		Idle for 99.948 ms

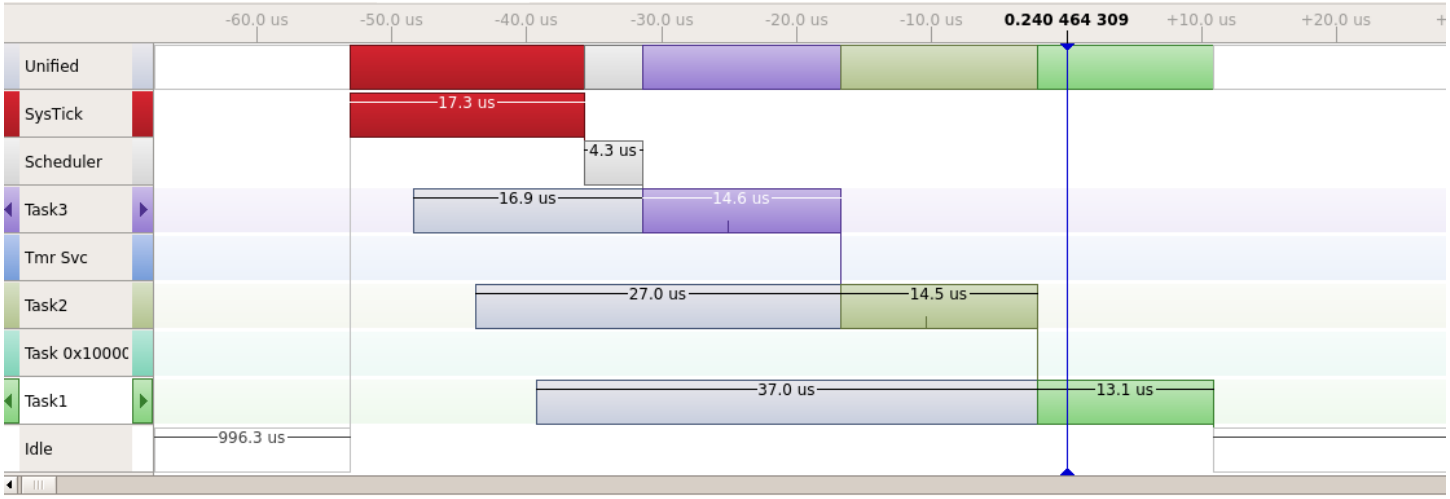
->In this part of the trace we can observe that the interrupt was generated and in the interrupt function we have declared the vTaksNotifyFromISR() API which send the notification from the interrupt block to the task block, since the notification was send to the task1 from interrupt, the task1 ulTaskNotifyTake accept the notification and executes further which was to give notification to Task2

->As soon the notification was given by task1, task2 gets ready and being higher priority task takes the controll from task1, and start execution and gives notification for task3

-> As soon the notification was generated by task2, task3 gets ready and being higher priority task takes the controll from task2, and start execution, then there is a delay so the task will go to the block state and return the controller back to task2

->the task2 is also having a delay it also goes to the block state and return back the controller to task1

->the task1 is also having a delay so it also goes to the block state



3.Third trace is the ending of task

0.240 428 440	SysTick	ISR Exit		Returns to Scheduler
0.240 432 833	Task3	Task Run		Runs for 14.649 us
0.240 439 131	Task3	vTaskDelete	Task3	xTaskToDelete=Task3
0.240 447 482	Task2	Task Run		Runs for 14.542 us
0.240 453 774	Task2	vTaskDelete	Task2	xTaskToDelete=Task2
0.240 462 024	Task1	Task Run		Runs for 13.173 us
0.240 468 149	Task1	vTaskDelete	Task1	xTaskToDelete=Task1
0.240 475 196	Idle	System Idle		
0.241 411 143	SysTick	ISR Enter		Runs for 3.673 us

->After the delay of task is completed the schedular again schedules the highest priority task
->task3 being the highest priority tasks start run and the API vTaskDelete(NULL) runs and deletes itself and then task 2 being the highest priority task left takes the controll
->then task2 start run and the API vTaskDelete(NULL) runs and deletes itself
->task1 being the highest priority task left takes the controll and start run and deletes it self.