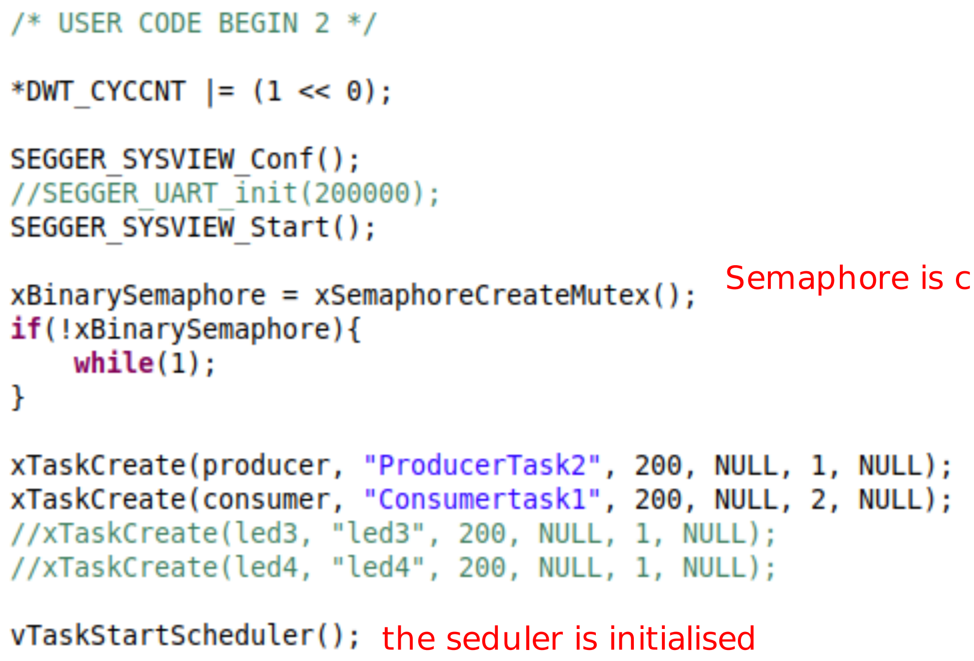
Abhishek kumar Simra
PRN-:220940130002

.

Problem
Producer Task PRIORITY 1
Consumer Task. PRIORITY 2
create a program to use Semaphore between the producer and consumer

SOLUTION:



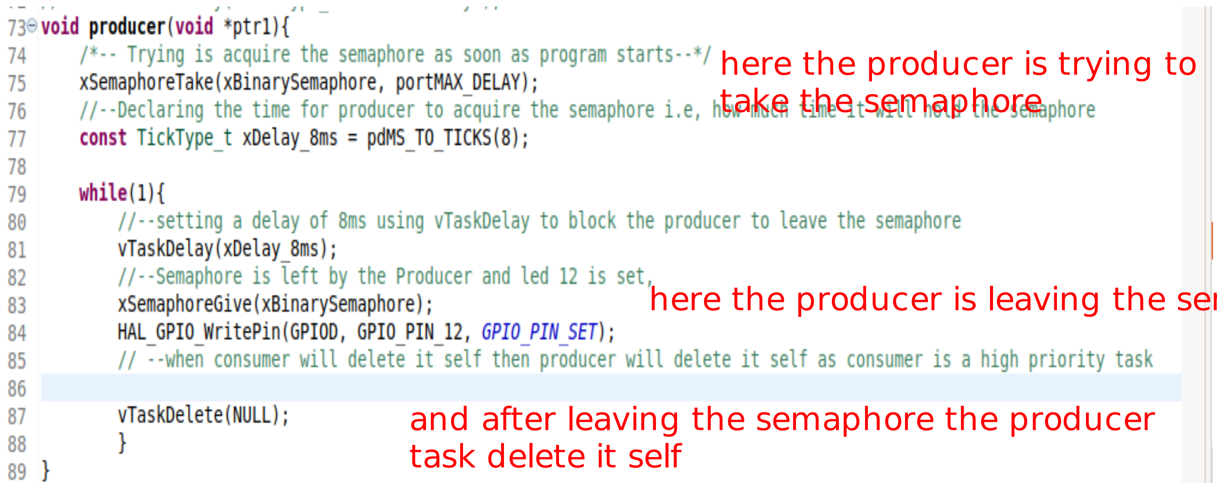
```
/* USER CODE BEGIN 2 */

*DWT_CYCCNT |= (1 << 0);

SEGGER_SYSVIEW_Conf();
//SEGGER_UART_init(200000);
SEGGER_SYSVIEW_Start();

xBinarySemaphore = xSemaphoreCreateMutex();
if(!xBinarySemaphore){
    while(1);
}

xTaskCreate(producer, "ProducerTask2", 200, NULL, 1, NULL);
xTaskCreate(consumer, "Consumertask1", 200, NULL, 2, NULL);
//xTaskCreate(led3, "led3", 200, NULL, 1, NULL);
//xTaskCreate(led4, "led4", 200, NULL, 1, NULL);

vTaskStartScheduler();
```

Semaphore is created using the mutex

Producer and Consumer task are being created with consumer having the higher priority

the seduler is initialised here to start the execution

The Producer task



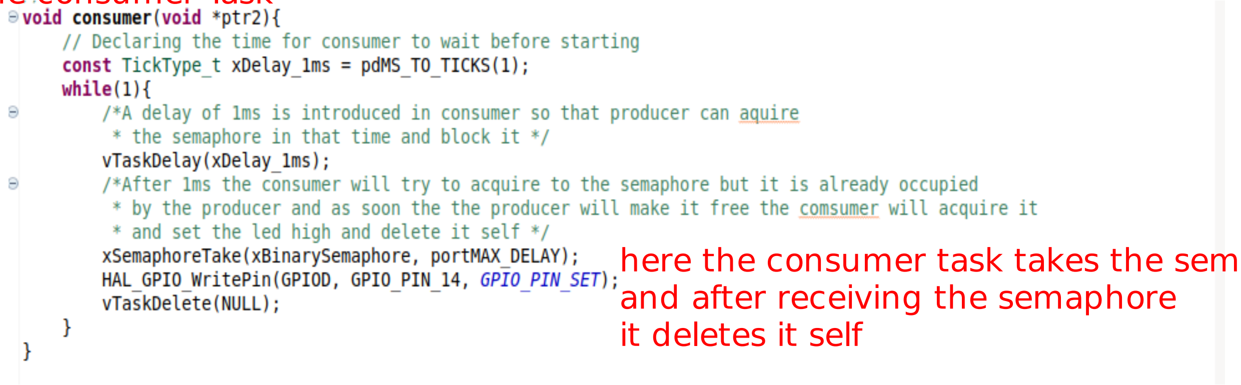
```
void producer(void *ptr1){
    /*-- Trying is acquire the semaphore as soon as program starts--*/
    xSemaphoreTake(xBinarySemaphore, portMAX_DELAY);
    //--Declaring the time for producer to acquire the semaphore i.e, how much time it will hold the semaphore
    const TickType_t xDelay_8ms = pdMS_TO_TICKS(8);

    while(1){
        //--setting a delay of 8ms using vTaskDelay to block the producer to leave the semaphore
        vTaskDelay(xDelay_8ms);
        //--Semaphore is left by the Producer and led 12 is set,
        xSemaphoreGive(xBinarySemaphore);
        HAL_GPIO_WritePin(GPIOD, GPIO_PIN_12, GPIO_PIN_SET);
        // --when consumer will delete it self then producer will delete it self as consumer is a high priority task

        vTaskDelete(NULL);
        }
}
```

here the producer is trying to take the semaphore

here the producer is leaving the semaphore

and after leaving the semaphore the producer task delete it self

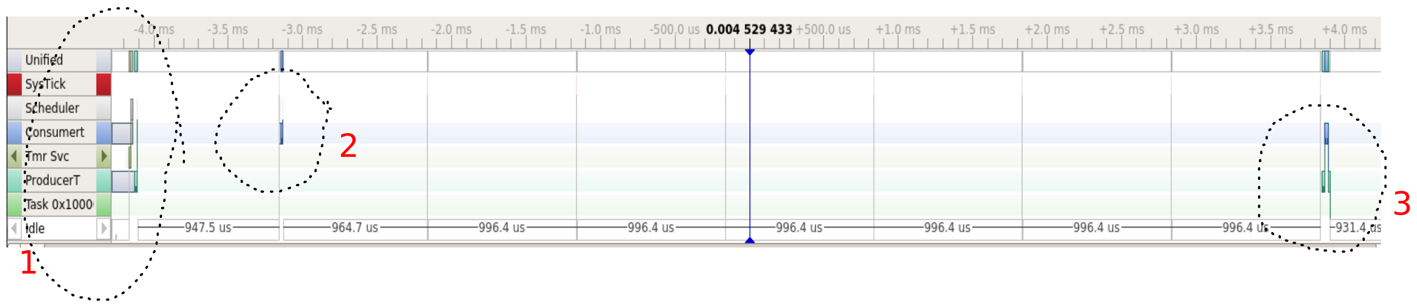
The consumer Task



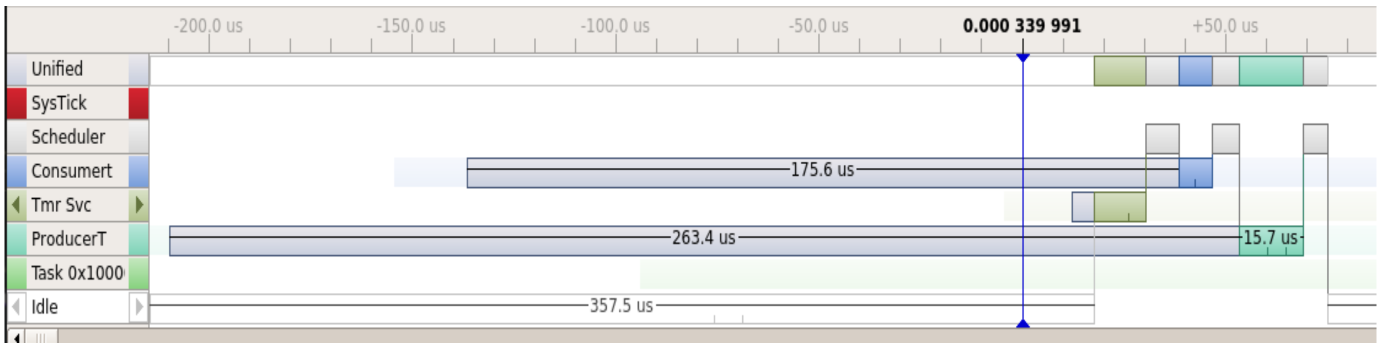
```
void consumer(void *ptr2){
    // Declaring the time for consumer to wait before starting
    const TickType_t xDelay_1ms = pdMS_TO_TICKS(1);
    while(1){
        /*A delay of 1ms is introduced in consumer so that producer can aquire
         * the semaphore in that time and block it */
        vTaskDelay(xDelay_1ms);
        /*After 1ms the consumer will try to acquire to the semaphore but it is already occupied
         * by the producer and as soon the the producer will make it free the comsumer will acquire it
         * and set the led high and delete it self */
        xSemaphoreTake(xBinarySemaphore, portMAX_DELAY);
        HAL_GPIO_WritePin(GPIOD, GPIO_PIN_14, GPIO_PIN_SET);
        vTaskDelete(NULL);
    }
}
```

here the consumer task takes the semaphore and after receiving the semaphore it deletes it self

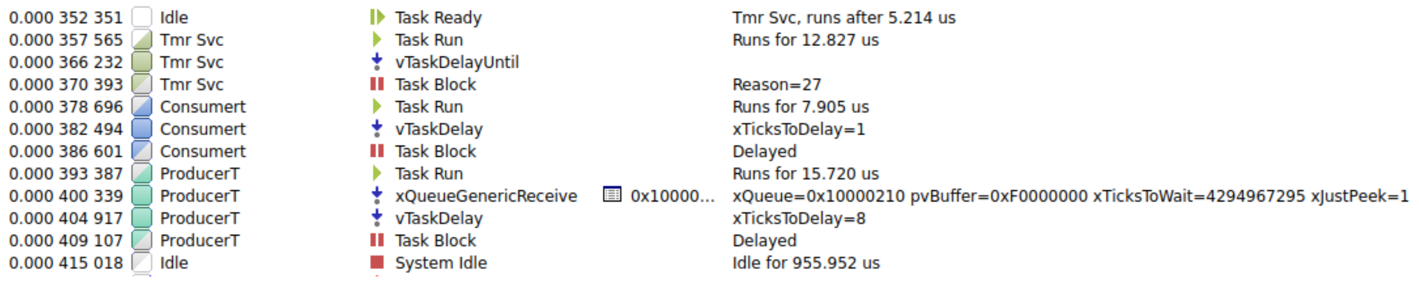
The complete trace of the whole process



1.



the enhanced trace of 1st part

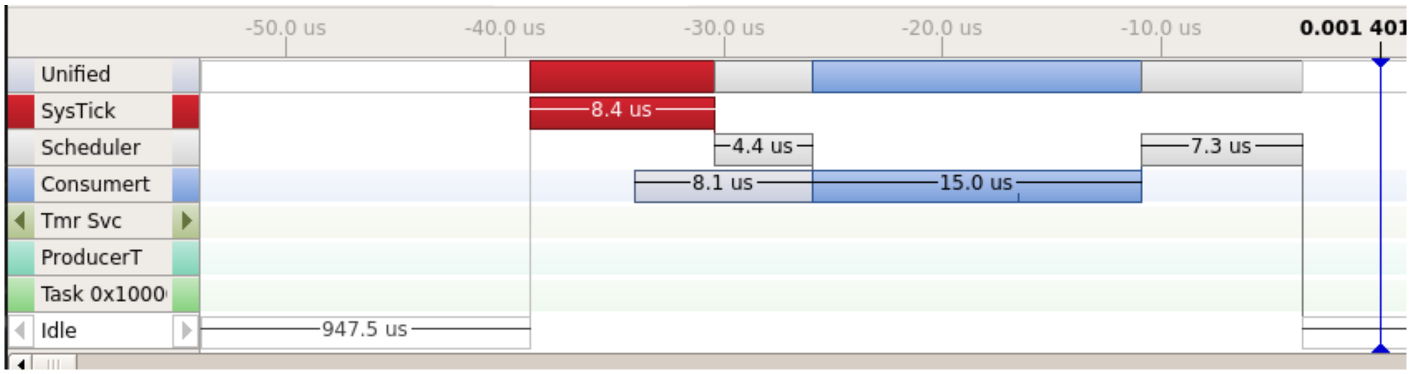


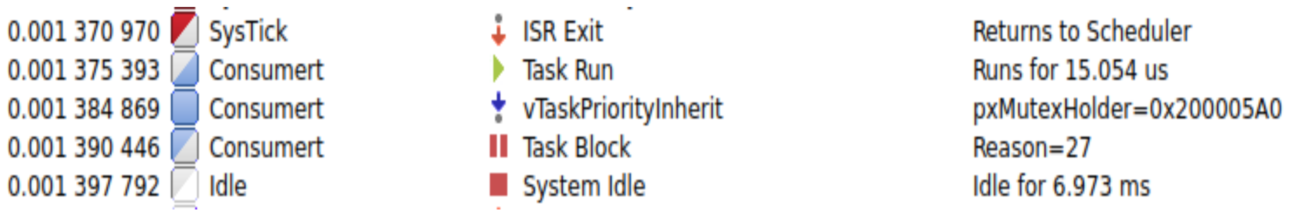
0.000 352 351	Idle	Task Ready		Tmr Svc, runs after 5.214 us
0.000 357 565	Tmr Svc	Task Run		Runs for 12.827 us
0.000 366 232	Tmr Svc	vTaskDelayUntil		
0.000 370 393	Tmr Svc	Task Block		Reason=27
0.000 378 696	Consumert	Task Run		Runs for 7.905 us
0.000 382 494	Consumert	vTaskDelay		xTicksToDelay=1
0.000 386 601	Consumert	Task Block		Delayed
0.000 393 387	ProducerT	Task Run		Runs for 15.720 us
0.000 400 339	ProducerT	xQueueGenericReceive	0x10000...	xQueue=0x10000210 pvBuffer=0xF0000000 xTicksToWait=4294967295 xJustPeek=1
0.000 404 917	ProducerT	vTaskDelay		xTicksToDelay=8
0.000 409 107	ProducerT	Task Block		Delayed
0.000 415 018	Idle	System Idle		Idle for 955.952 us

The above shows the starting of the trace hot it is initialisez

We can observe the how the tasks where initialised and how much time was used by each to do so
->at first we can see the producer task was ready
->then the consumer task also got ready
->as we have used sheduler it awokes the SVC
->when the SVC is initialised it initialised the execution
->The after SVC the schedular is invocked
->a the schedular sedules the highest priority task which is the consumer task
->when the consumer task is started there is a delay if 1ms in it so the task gets blocked for the tim being
->after the consumer task is blocked the controll is send back to the scheduler the schedules the next highest
priority task which in this case is the Producer task
->wehen the producer task is started it takes the semaphore for a defined time of 8ms
->so the Producer task goes to the blocked state

2.The inhanced trace of the second part

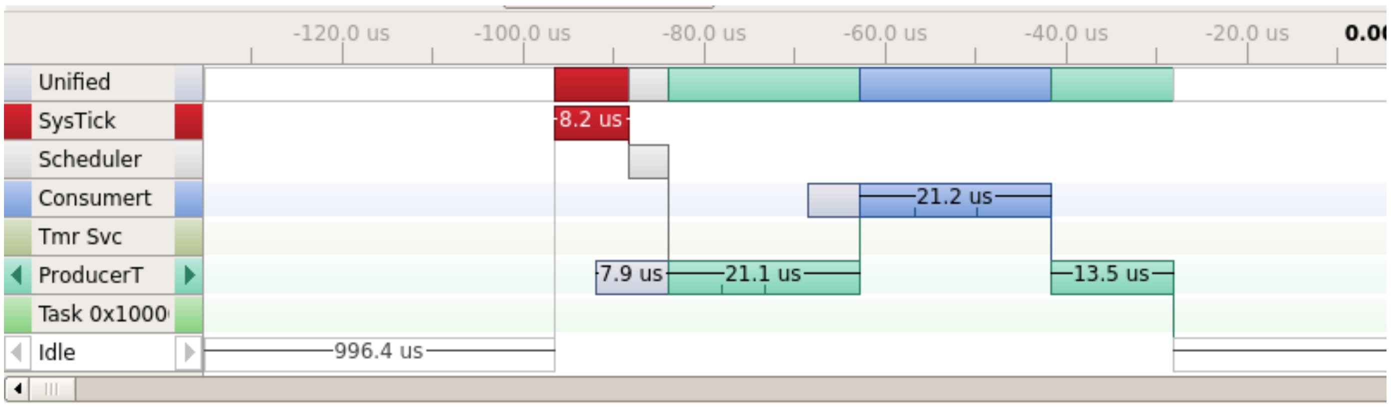


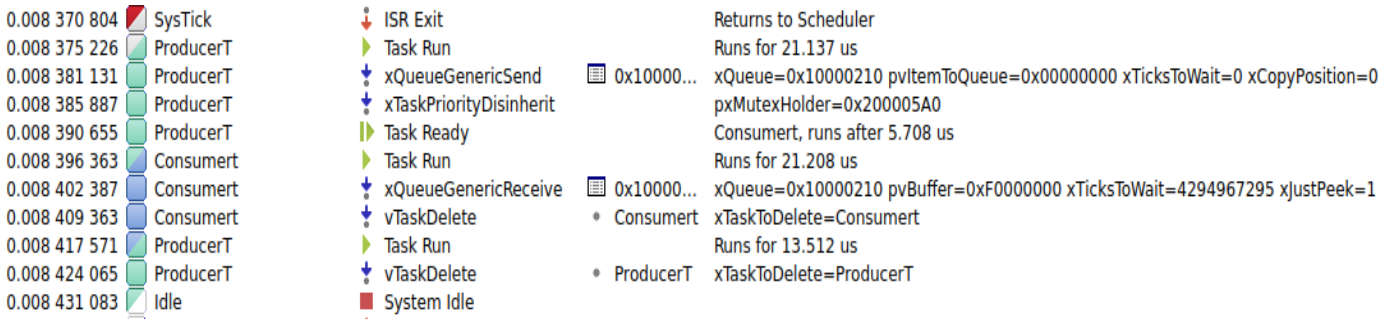


0.001 370 970	SysTick	ISR Exit	Returns to Scheduler
0.001 375 393	Consumert	Task Run	Runs for 15.054 us
0.001 384 869	Consumert	vTaskPriorityInherit	pxMutexHolder=0x200005A0
0.001 390 446	Consumert	Task Block	Reason=27
0.001 397 792	Idle	System Idle	Idle for 6.973 ms

->the consumer task wokes after 1ms and try to get semaphore but the semaphore is taken by the producer before
->so the consumer task again goes to the sleep state

3.The inhanced trace of the third part




```
0.008 370 804  SysTick    ISR Exit                                  Returns to Scheduler
0.008 375 226  ProducerT  Task Run                                  Runs for 21.137 us
0.008 381 131  ProducerT  xQueueGenericSend        0x10000...       xQueue=0x10000210 pvItemToQueue=0x00000000 xTicksToWait=0 xCopyPosition=0
0.008 385 887  ProducerT  xTaskPriorityDisinherit                   pxMutexHolder=0x200005A0
0.008 390 655  ProducerT  Task Ready                                Consumert, runs after 5.708 us
0.008 396 363  Consumert  Task Run                                  Runs for 21.208 us
0.008 402 387  Consumert  xQueueGenericReceive     0x10000...       xQueue=0x10000210 pvBuffer=0xF0000000 xTicksToWait=4294967295 xJustPeek=1
0.008 409 363  Consumert  vTaskDelete              Consumert        xTaskToDelete=Consumert
0.008 417 571  ProducerT  Task Run                                  Runs for 13.512 us
0.008 424 065  ProducerT  vTaskDelete              ProducerT        xTaskToDelete=ProducerT
0.008 431 083  Idle       System Idle
```

->After 8ms the producer task again wokes up and leaves the semaphore
->as soon the semaphore is left the consumer task gets activated and creates a interrupt to the producer
task execution as the consumer task is a higher priority task
->the consumer task ackuire the semaphore for a time and the leave it
after it leaves the semaphore
it deleates it self
->as the consumer task delete it self it controll is returned back to the producer
->then the producer also deleted itself

=>This All happened as per the PendSV instruction
the pendSV says that if a higher priority task comes or generates a interrupt
the current running task gets saved there only and the higher priority task start
and after the execution of higher priority task the controll returns back to the previous task