Software en Tiempo Real

Msc. Ing. Carlos Centeno
Ingeniería Electrónica
UTN FRC

Año 2023

Temario

- Administración de Tiempo.
- Sincronización con Eventos
  - Semaforos
  - Mailbox
  - Queues
  - Deadlock
- Cambio de Contexto
- Resumen de Aspectos Relevantes

Administración de tiempo

- Existen algunas funciones para el control del tiempo
  - OS_TimeDly()
  - OS_TimeDlyHMSM()
  - OS_TimeDlyResume()
  - OS_TimeGet()
  - OS_TimeSet()

OSTimeDly(INT 16U)

- Se pasa como parámetro la cantidad de TICKS que se desean.
- Cantidad de TICKS 1 a 65535.
- Se saca la tarea de la tabla de tareas ready.
- Se carga el valor de ticks en el TCB asociado a la tarea.
- ***Se llama al scheduler.***

- SE PUEDE resumir la tarea demorada en cualquier momento.

OSTimeDly(INT 16U)

```
void OSTimeDly (INT16U ticks)
{
        if (ticks > 0)
    {
        OS_ENTER_CRITICAL();
        if ((OSRdyTbl[OSTCBCur->OSTCBY] &= ~OSTCBCur->OSTCBBitX) == 0)
        {
                OSRdyGrp &= ~OSTCBCur->OSTCBBitY;
        }
        OSTCBCur->OSTCBDly = ticks;
        OS_EXIT_CRITICAL();
        OSSched();
    }
}
if ( operacion == 0)
OSRdyTbl[OSTCBCur->OSTCBY]  &=  ~OSTCBCur->OSTCBBitX
```

OSTimeDlyResume()

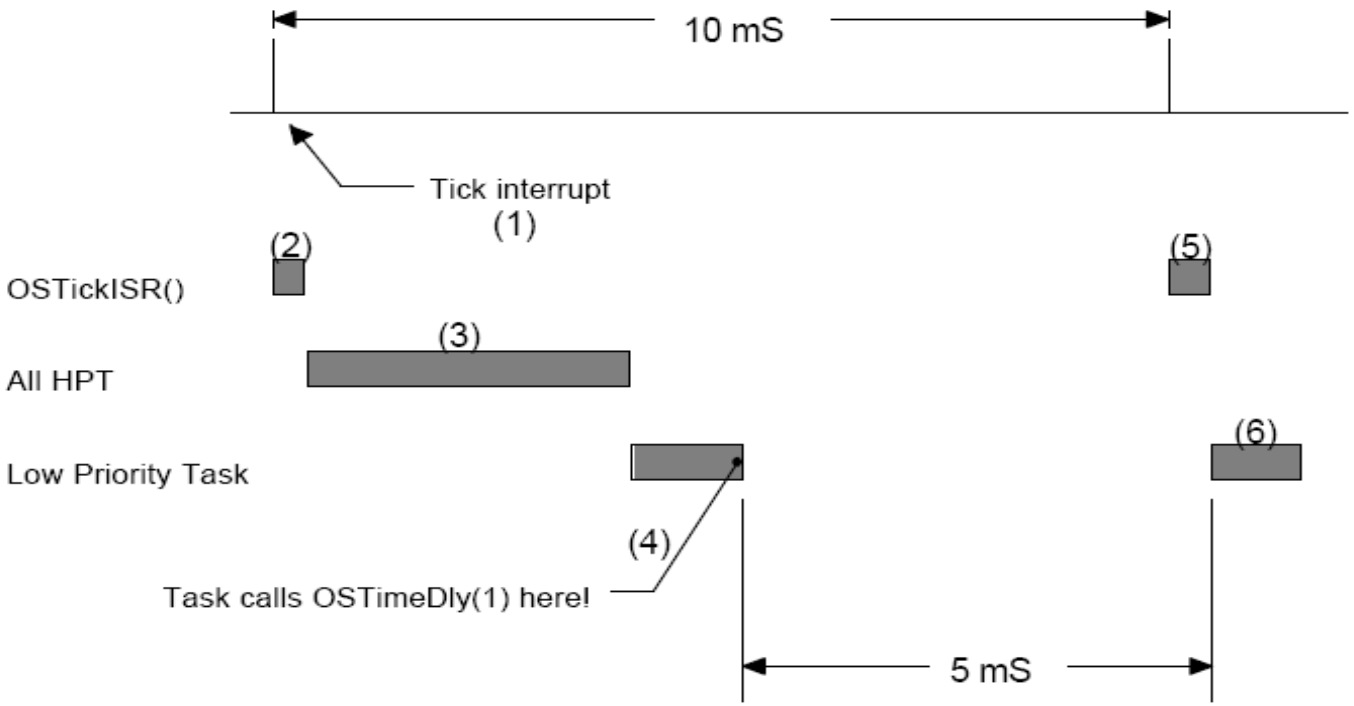
- Resume una tarea demorada
- La tarea pasa del estado WAITING al estado READY
- Se verifica que el valor de prioridad este definido dentro del intervalo
- Se verifica que la tarea exista
- Se verifica si la tarea esta esperando en el estado WAITING
- Se fuerza el tiempo de demora poniendo a CERO en el TCB correspondiente.
- Si la tarea no esta suspendida, se la pasa a estado READY.
- ***Se invoca al Scheduler();***

Resolución de Tiempos

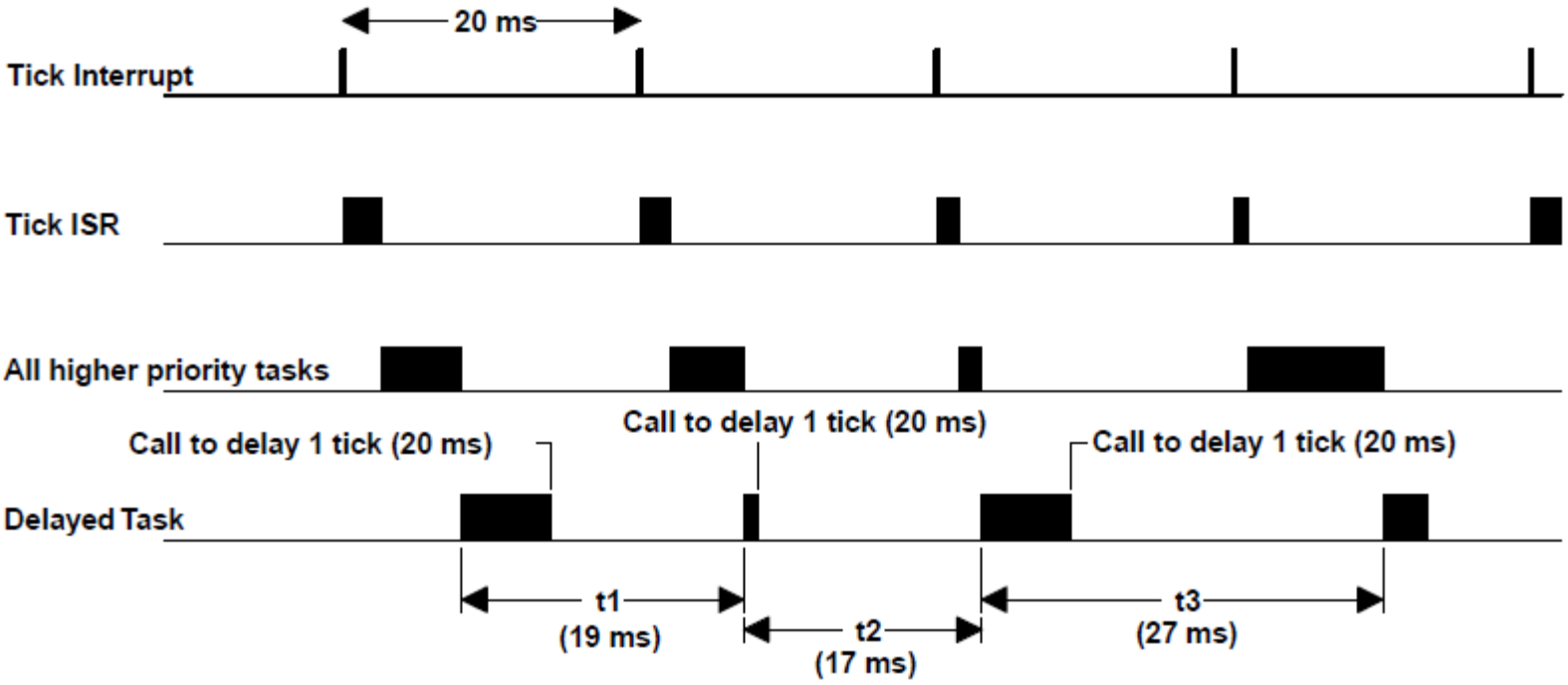
- Se debe tener en cuenta la sobrecarga del sistema y la asignación de prioridades cuando se hace uso de los servicios del kernel encargados de administrar una demora de tiempo.
- La tarea de MAS Alta prioridad tendrá la posibilidad de ejecutarse dentro del tiempo establecido.
- Las tareas de menor prioridad ocuparán el CPU pudiendo no ejecutarse dentro del tiempo del sistema.
  - Ejecución completa antes de que se produzca un tick del sistema.

Resolución de Tiempo

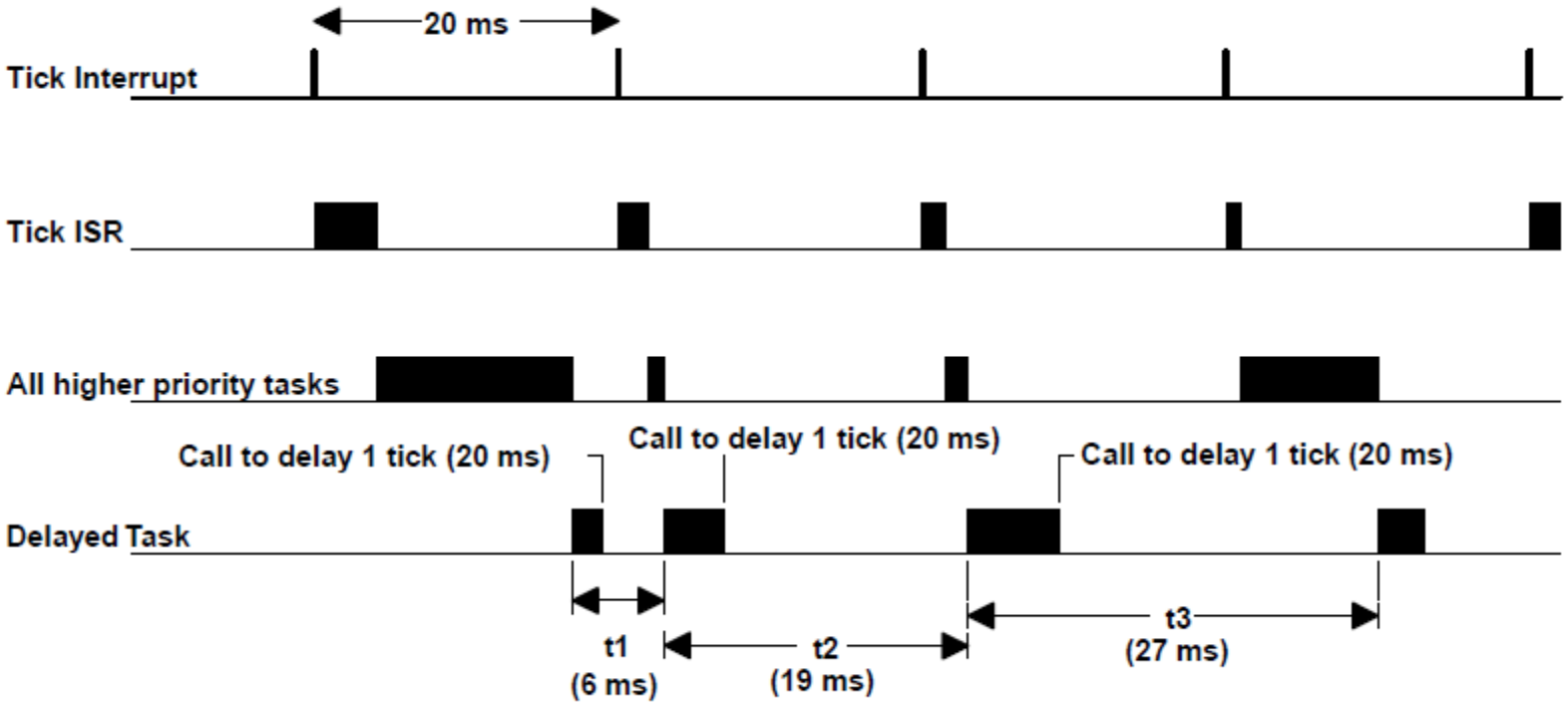
- La tarea de MAS Alta prioridad tendrá la posibilidad de ejecutarse dentro del tiempo establecido.



Resolución de Tiempo

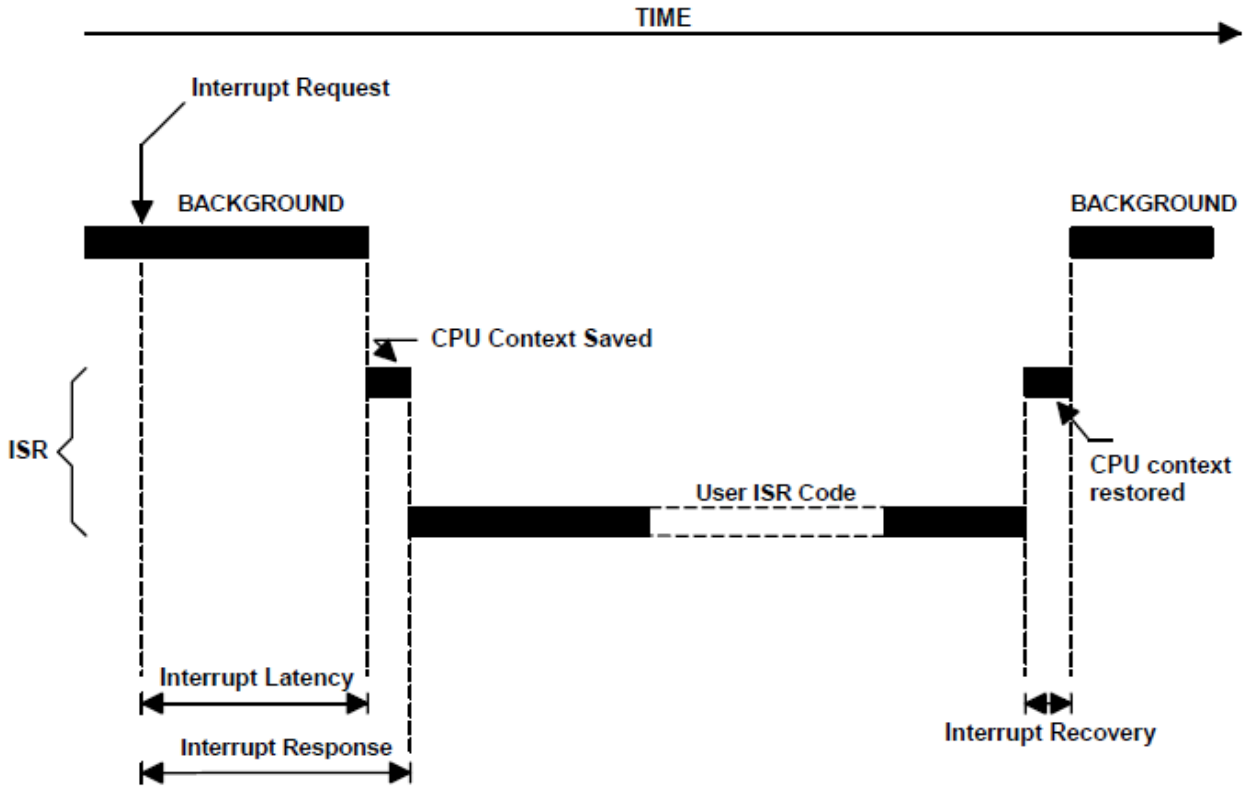


Resolución de Tiempo



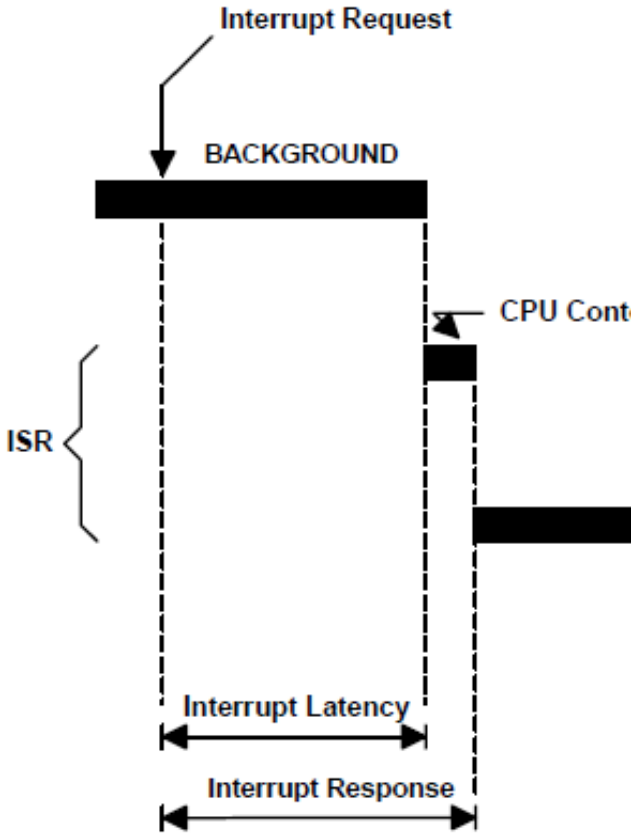
ISR – Tiempos de Respuesta

- Una de las especificaciones mas importantes tiene relación con el tiempo en que las interrupciones están deshabilitadas.



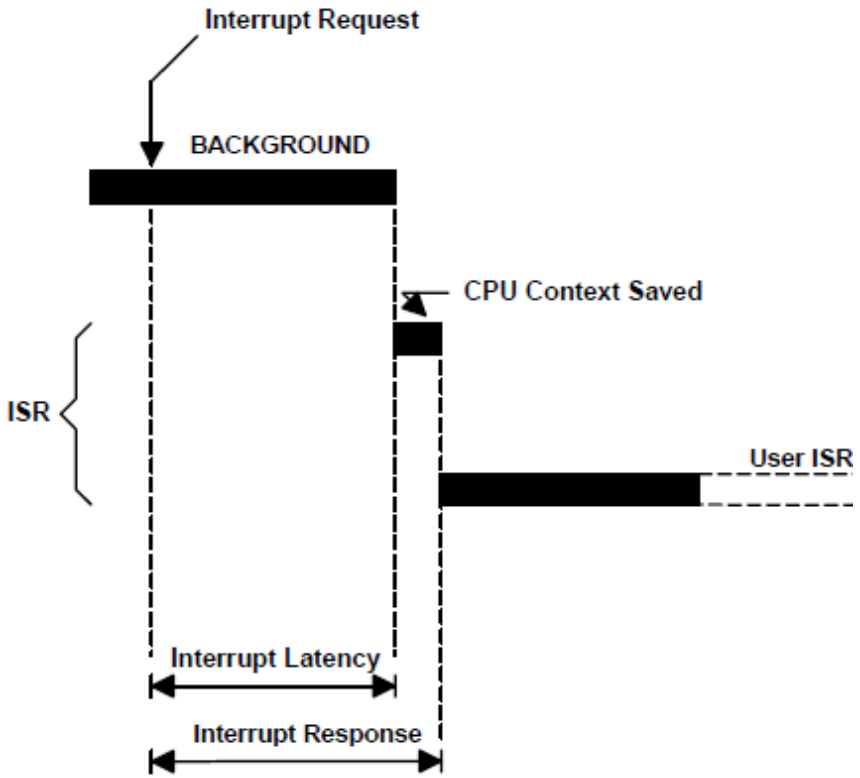
ISR – Latencia

- Es el tiempo durante el cual están deshabilitadas las interrupciones + mas el tiempo que se requiere para ejecutar la primer instrucción de la ISR.



ISR – Tiempo de Respuesta

- Es el que requiere el micro desde que recibe el pedido de interrupción hasta que ejecuta la primer instrucción del la ISR luego de salvar el contexto.
- Dependerá del tipo de sistema desarrollado, foreground/background o RTOS preemptive o nonpreemtive.

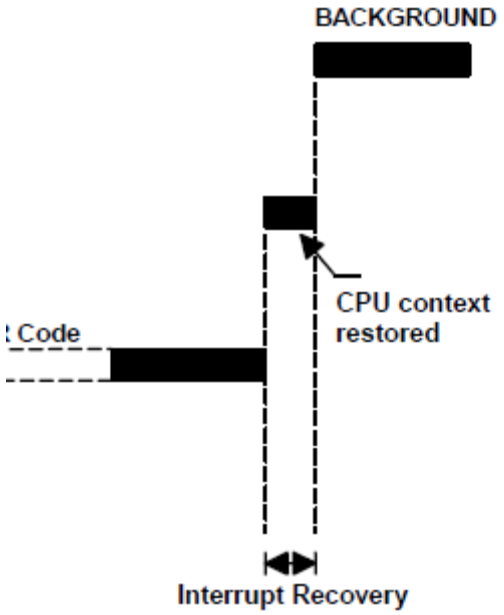


ISR – Tiempo de Respuesta

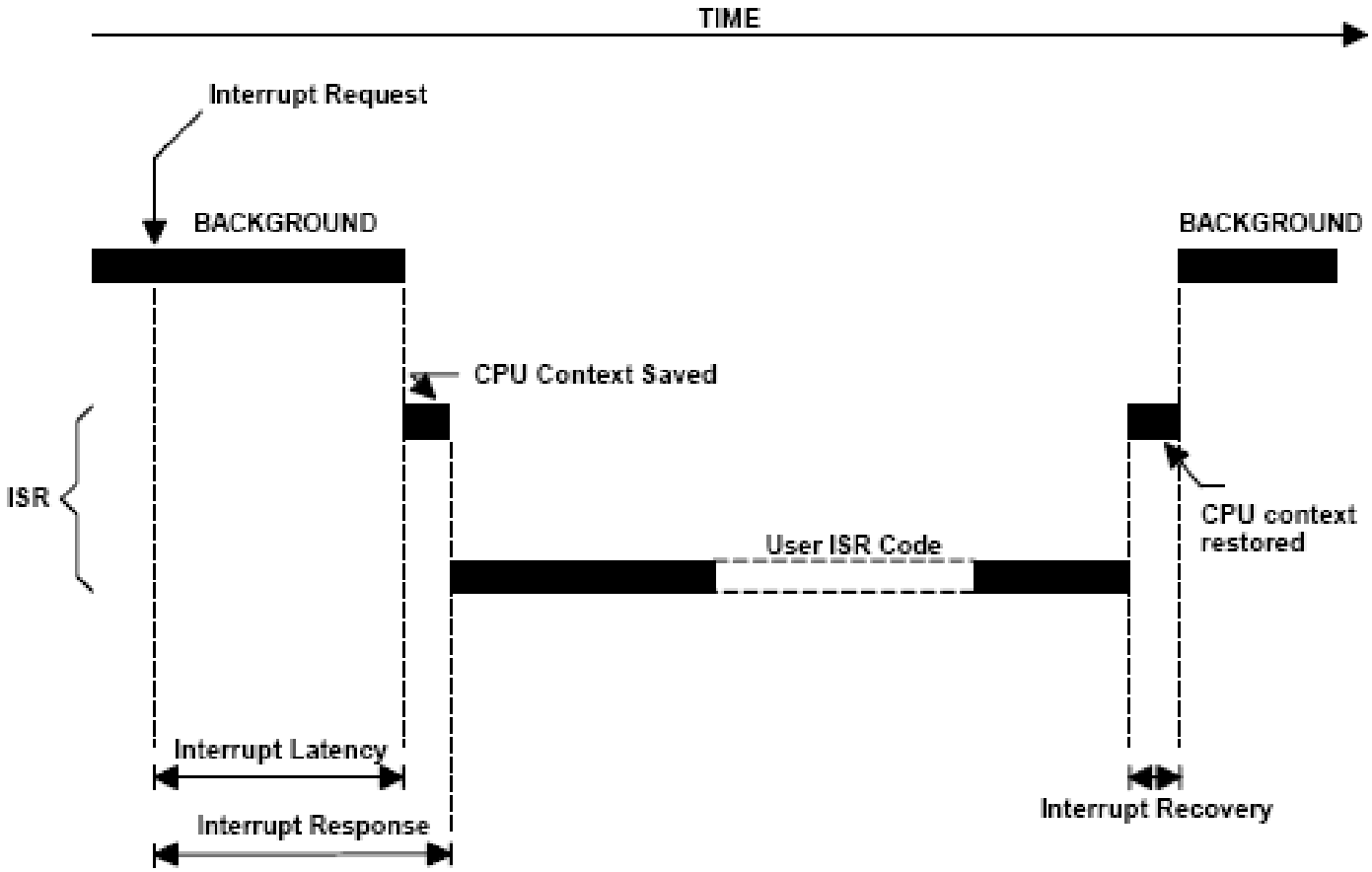
- Se debe considerar como tiempo de respuesta el peor caso como tiempo de respuesta del sistema.
  - Siempre responde en 50uSeg pero alguna vez en 250uS, entonces el tiempo se definirá como de 250uS.

ISR – Tiempo de Recuperación

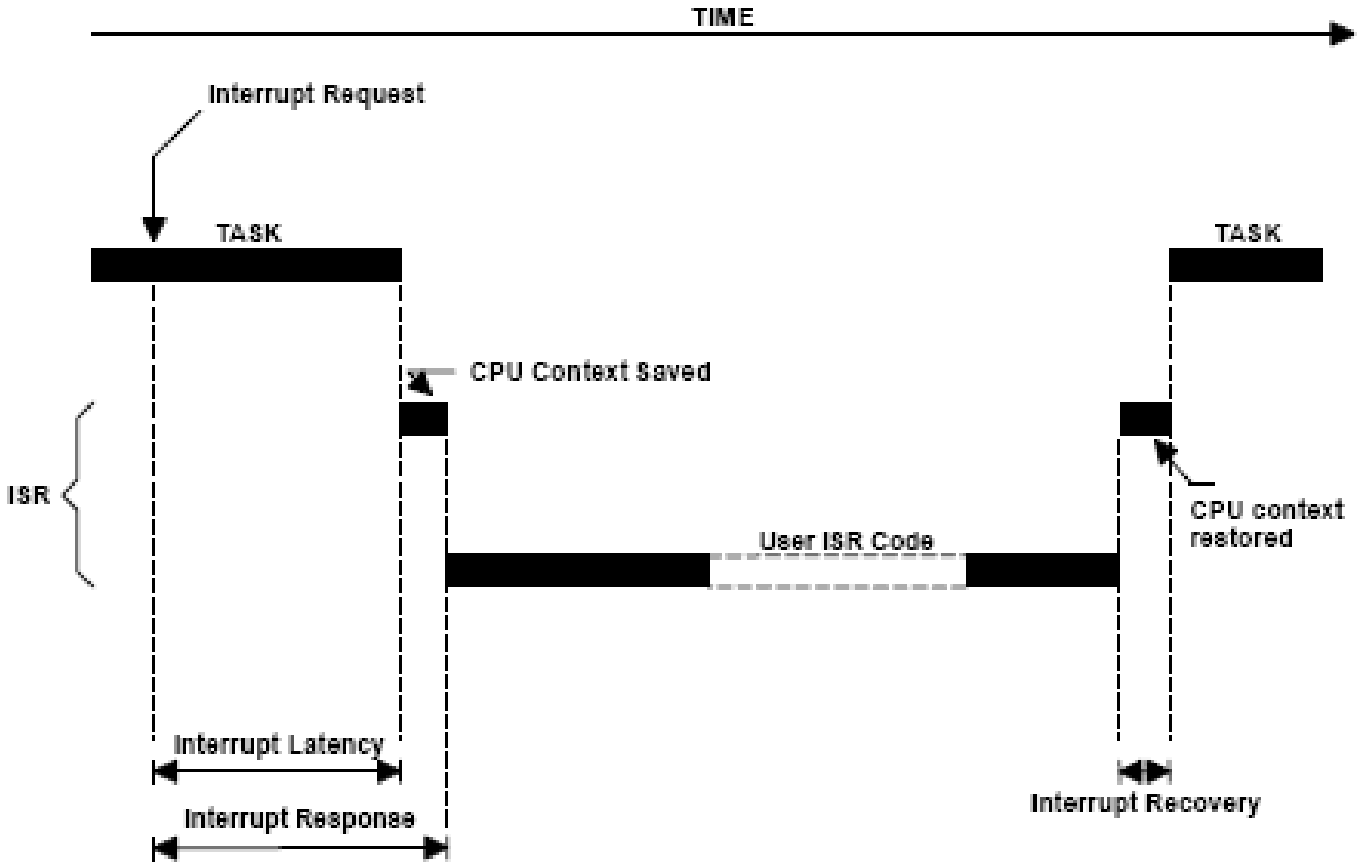
- Es el que se necesita para volver al punto de interrupción.
- Dependiendo del sistema desarrollado será mas o menos el tiempo requerido.
- Foreground/background
- RTOS non preemptive
- RTOS preemptive



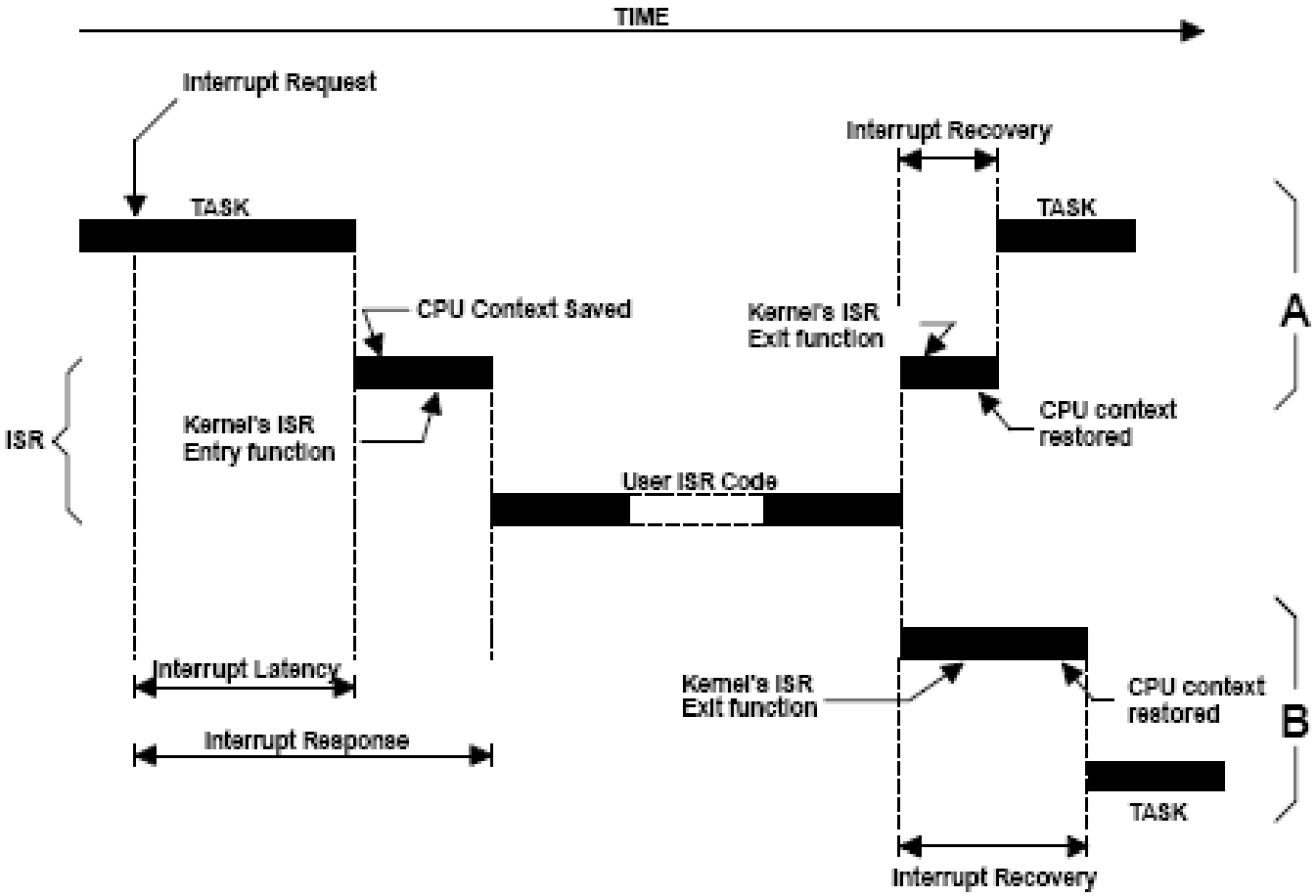
ISR – Tiempo Total

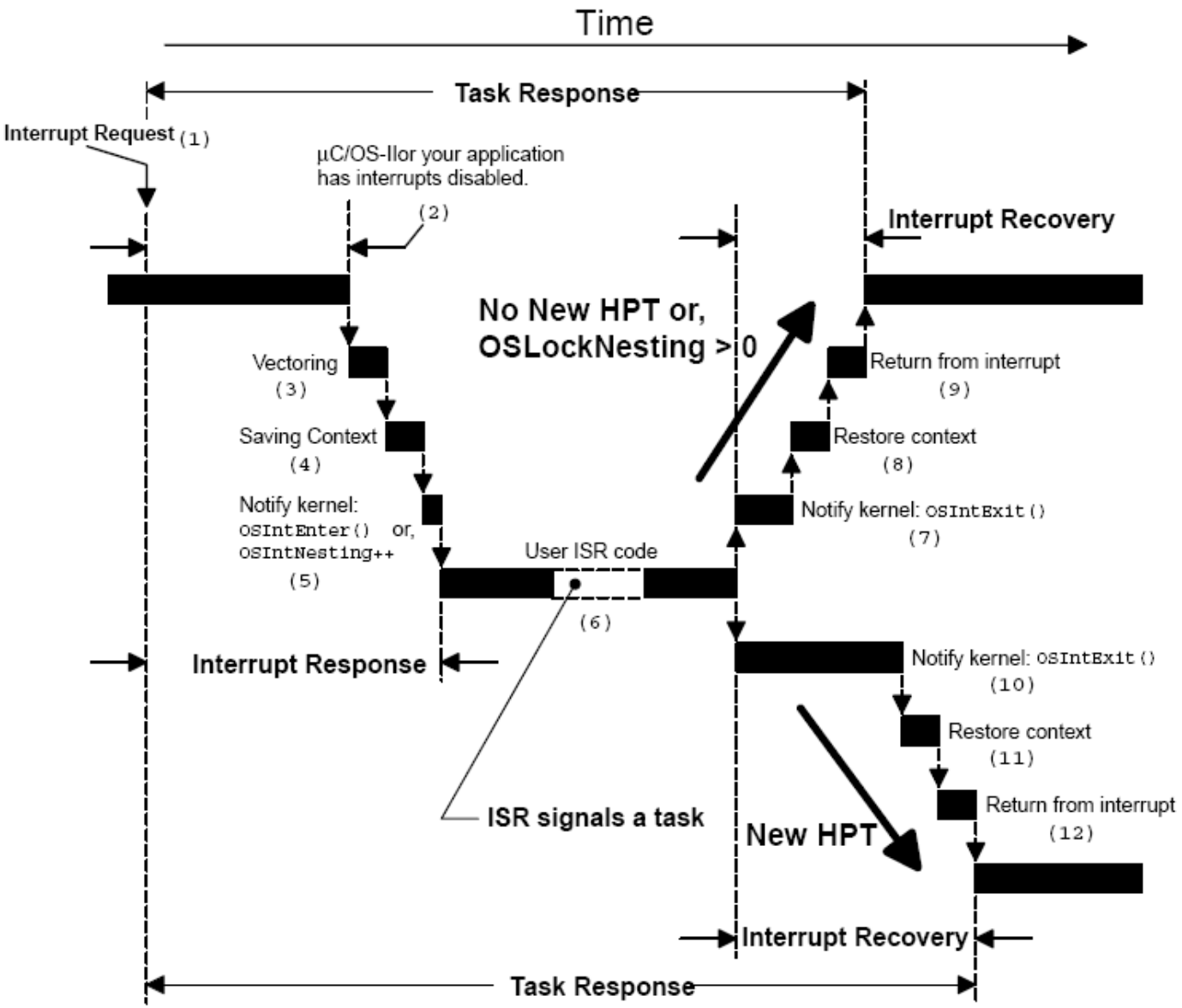


ISR – Tiempo Total



ISR – Tiempo Total




Time
Task Response
Interrupt Request (1)
µC/OS-IIor your application has interrupts disabled.
(2)
Interrupt Recovery
No New HPT or, OSLockNesting > 0
Vectoring (3)
Return from interrupt (9)
Saving Context (4)
Restore context (8)
Notify kernel: OSIntEnter() or, OSIntNesting++ (5)
Notify kernel: OSIntExit() (7)
User ISR code
(6)
Interrupt Response
Notify kernel: OSIntExit() (10)
Restore context (11)
Return from interrupt (12)
ISR signals a task
New HPT
Interrupt Recovery
Task Response

EVENTOS

Sincronización

- Se utilizan entidades denominadas EVENTOS.
- Requieren estructuras de control denominadas ECB.
- Precauciones.
  - Se debe definir correctamente como será la relación entre Tareas.
    - Evitar el “deadlock”.
    - Usar timeout para detectar esta instancia.

Sincronización

- Se utilizan entidades denominadas EVENTOS.
- Ejemplo: espera la llegada de datos vía ISR

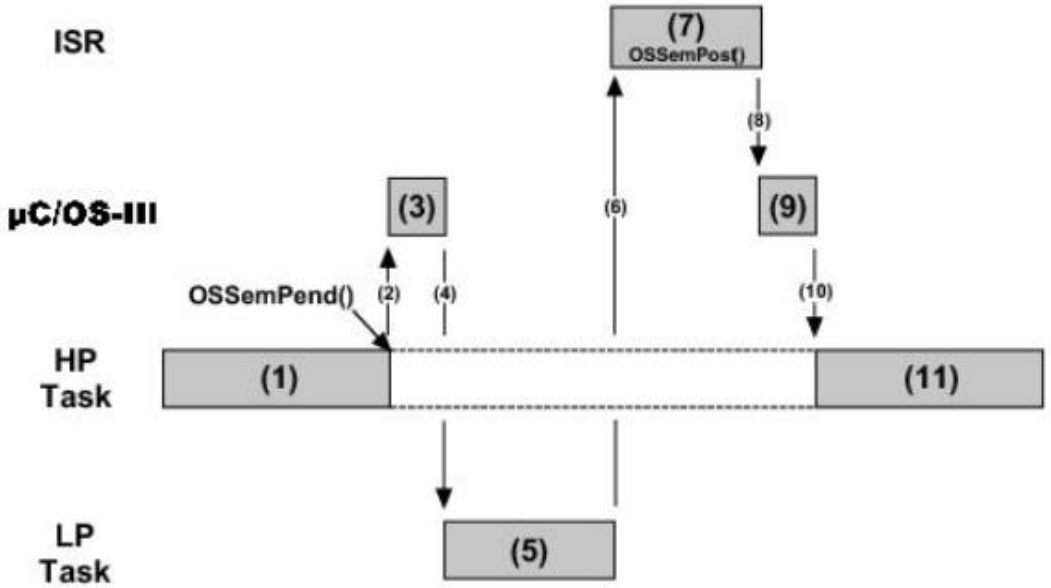
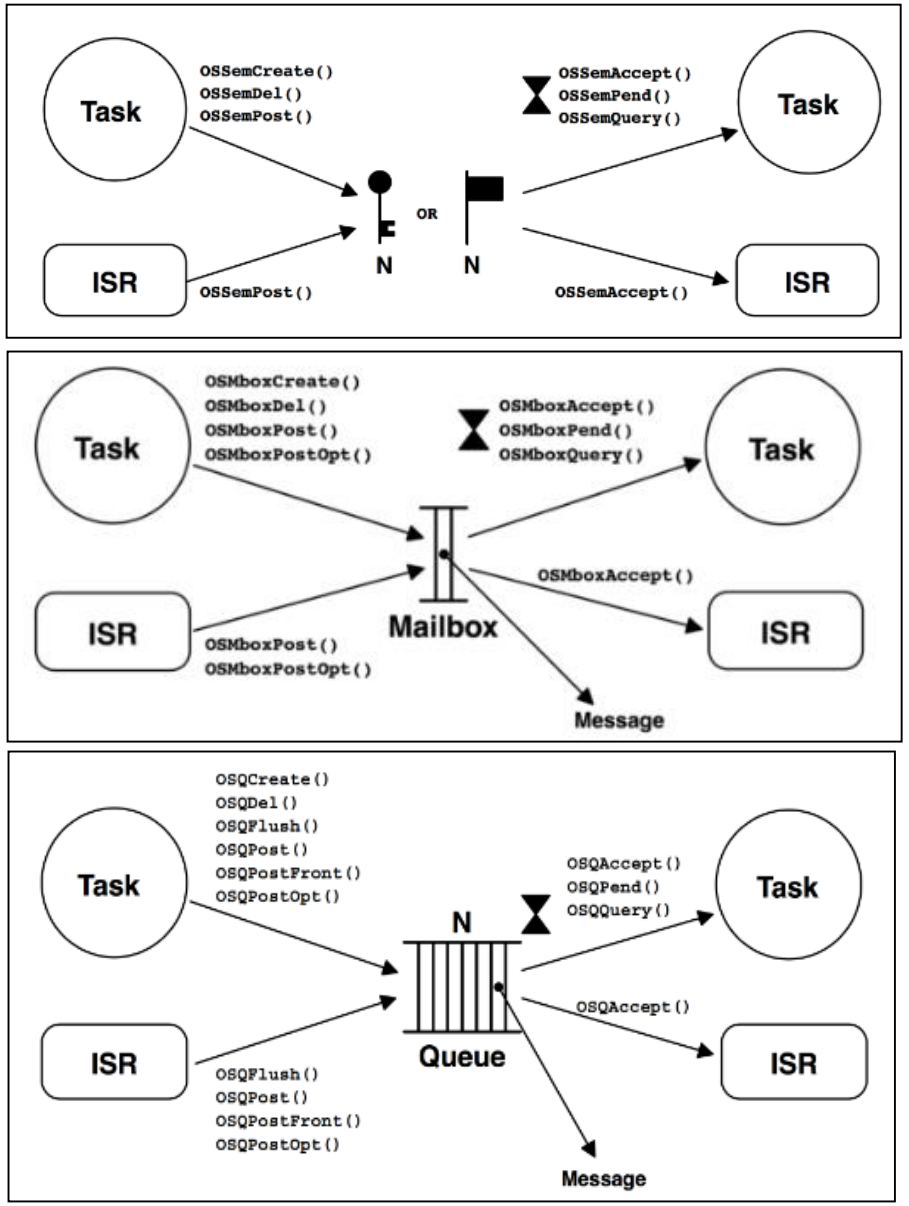


Figure - Unilateral Rendezvous, Timing Diagram

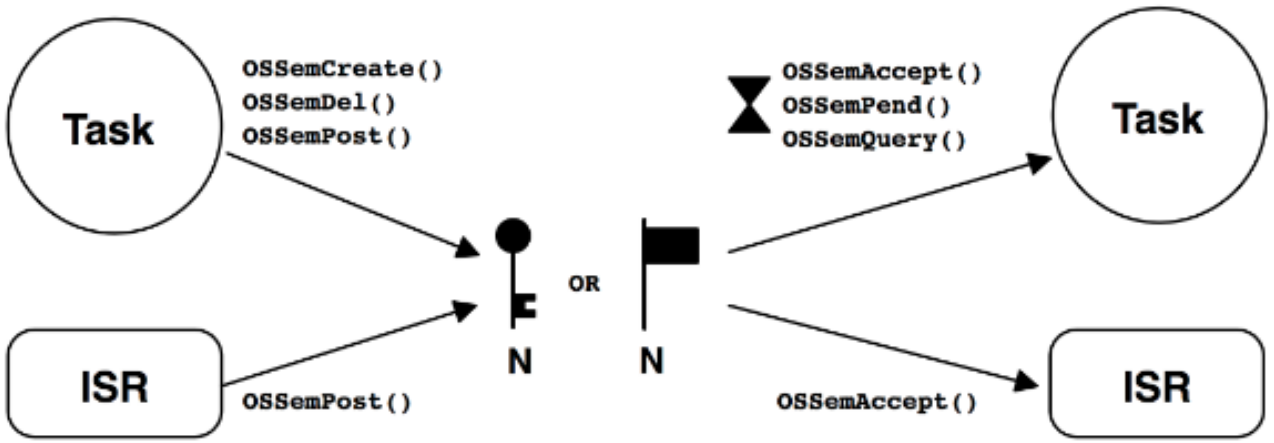
TIPOS de Eventos

- Semáforos
- Mailbox
- Queues



Eventos - Semáforos

- Semáforos Binarios
- Semáforos Contadores
- Semáforos Mutex
- Broadcasting



Eventos - Semáforos

- Es posible señalizar varias tareas con un único evento (semáforo) usando la opción OS_OPT_POST_ALL cuando se hace post.

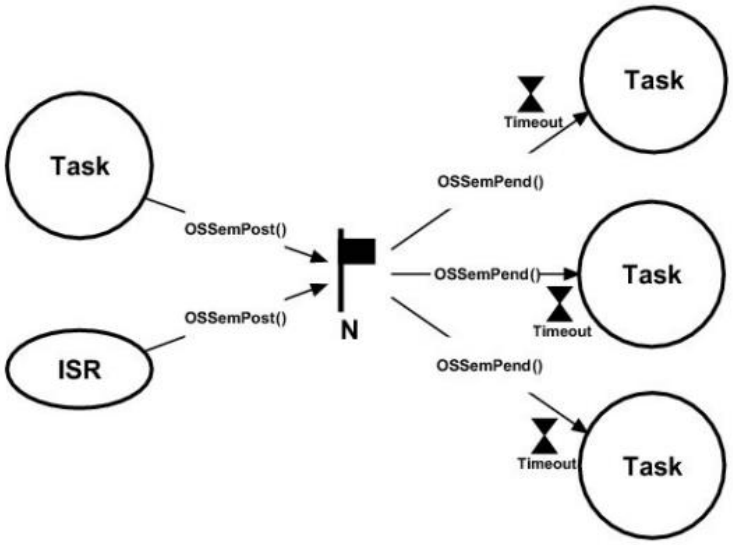
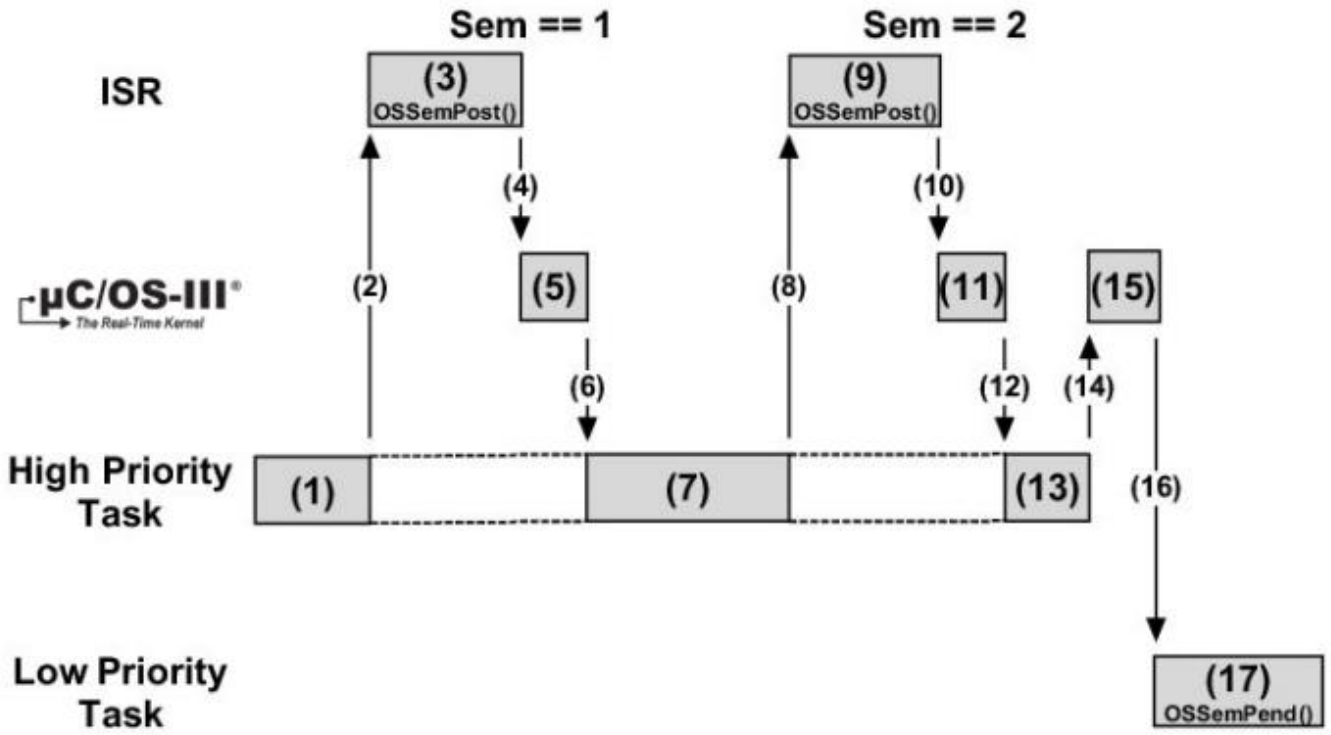


Figure - Multiple Tasks waiting on a Semaphore

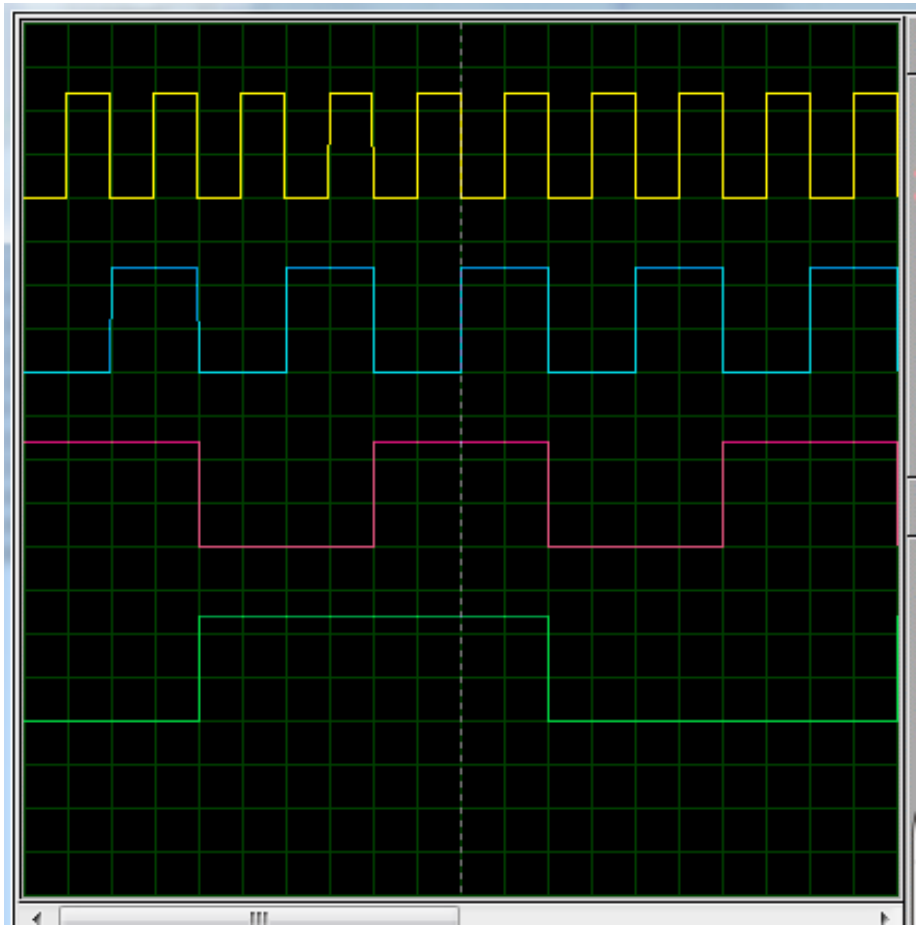
Eventos – Semáforo Contador

- Semáforos Contadores



Secuencia Temporal

- Posible Solución
- Sincronización con Semáforos Binarios.



Prioridad 1

```
//-----------------------------------
void Task1(void *pdata)
{
#if OS_CRITICAL_METHOD == 3
    OS_CPU_SR  cpu_sr;
#endif

    for(;;)
    {
        salidaLed_1 = ~salidaLed_1;
        if(salidaLed_1==0)
            OSSemPost(Semaforo1);
        OSTimeDly(10);
    }
}
```

Prioridad 2

```
for(;;)
{
    OSSemPend(Semaforo1,0,&err);
    salidaLed_2 = ~salidaLed_2;
    if(salidaLed_2==0)
        OSSemPost(Semaforo2);
    OSTimeDly(10);
}
```

Prioridad 3

```
for(;;)
{
    OSSemPend(Semaforo2,0,&err);
    salidaLed_3 = ~salidaLed_3;
    if(salidaLed_3==0)
        OSSemPost(Semaforo3);
    OSTimeDly(10);
}
```

Prioridad 4

```
INT8U *err;
INT16U timeOut=0;

for(;;)

    OSSemPend(Semaforo3,timeOut,err);
    salidaLed_4 = ~salidaLed_4;
    OSTimeDly(10);
}
```

Ejemplo 2

Eventos - Mailbox

- Entidad que permite transferir información entre Tareas.
- Se puede usar para Sincronizar dos Tareas.
- Se asemeja a una variable de tipo global.

Mailbox - Ejemplo

- Control de una salida Digital en base a un valor que resulta de un canal ADC.

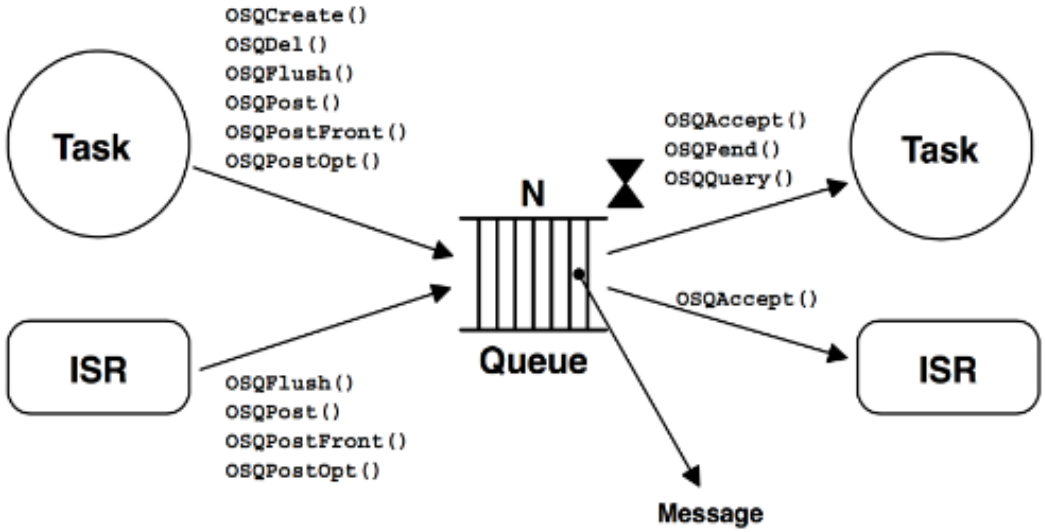
```
for(;;)
{
    OSTimeDly(5);
    uiValorADC = convertirADC();
    OSMboxPost(ADC_Mbox, (void *)&uiValorADC);
}
```

```
{
    rxmsg = OSMboxPend(ADC_Mbox, 0, err);
    inc_P = *rxmsg;

    if(*rxmsg >= 250)
        salidaRELE = 1;
    else
        salidaRELE = 0;

    OSTimeDly(1);
}
```

Ejemplo 5

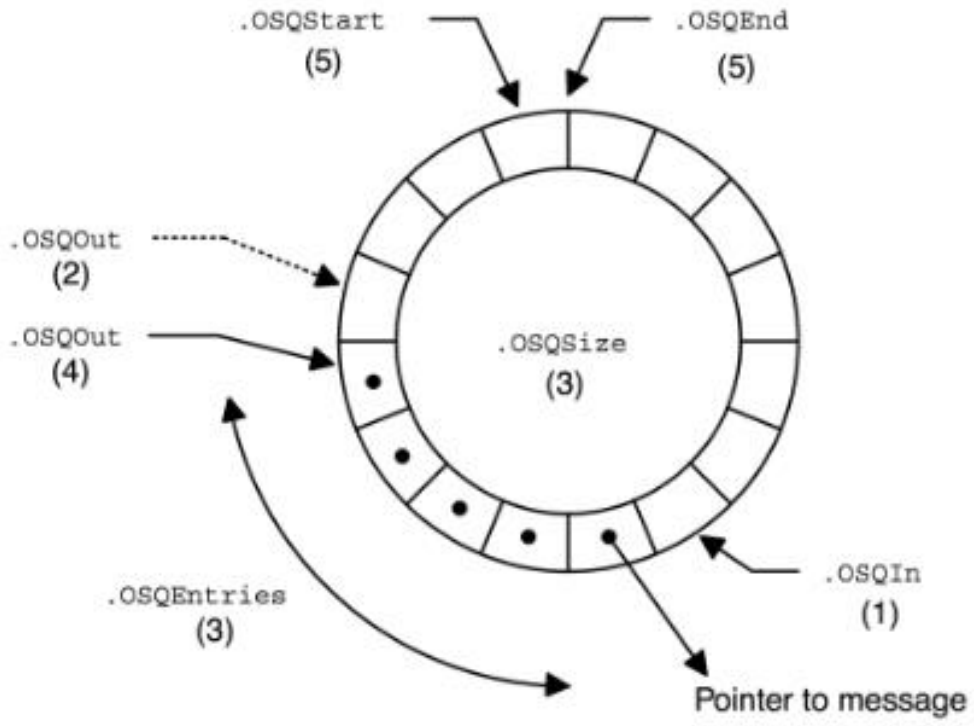
Eventos - Queues

- Entidad que permite transferir una cola de mensajes.
- Es posible almacenar mensajes siguiendo una lógica FIFO o LIFO.



Eventos - Queues

- Se almacenan los mensajes en una estructura de buffer circular.



Eventos - Queues

- Ejemplo de USO

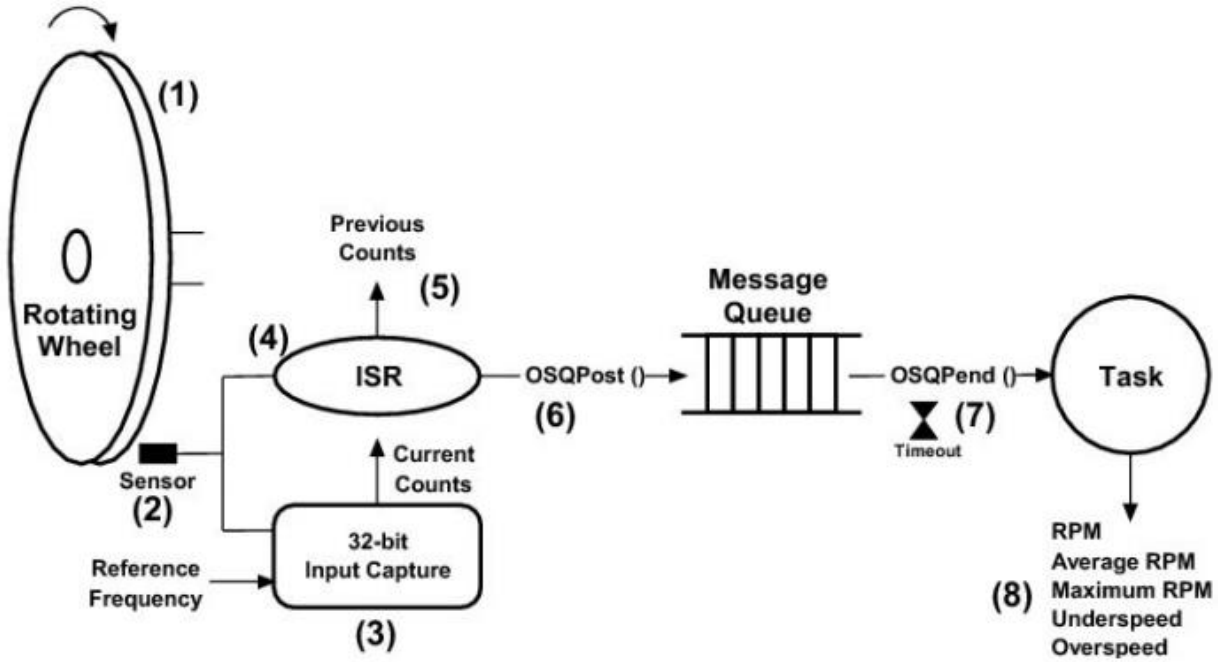


Figure - Measuring RPM

Sincronización desde ISR

- Es posible usando EVENTOS para sincronizar una Tarea con la ocurrencia de una Interrupción.