Software en Tiempo Real

Msc. Ing. Carlos Centeno
Ingeniería Electrónica
UTN FRC

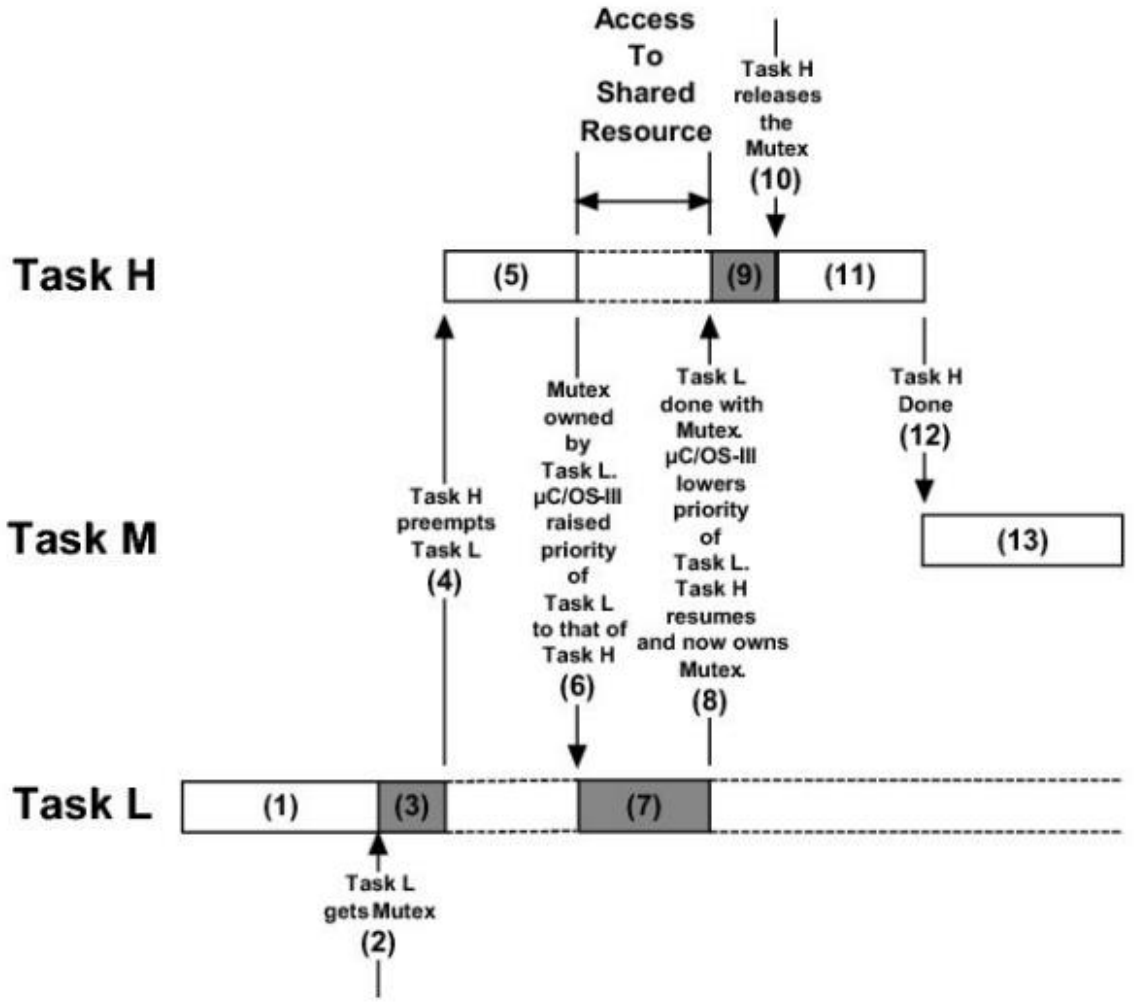
Año 2023

Eventos - Mutex

- Se pueden usar MUTEX cuando se requiere usar un recurso/periférico por más de una tarea.

- En el ejemplo se presenta el porque es útil un mutex.

Eventos - Mutex

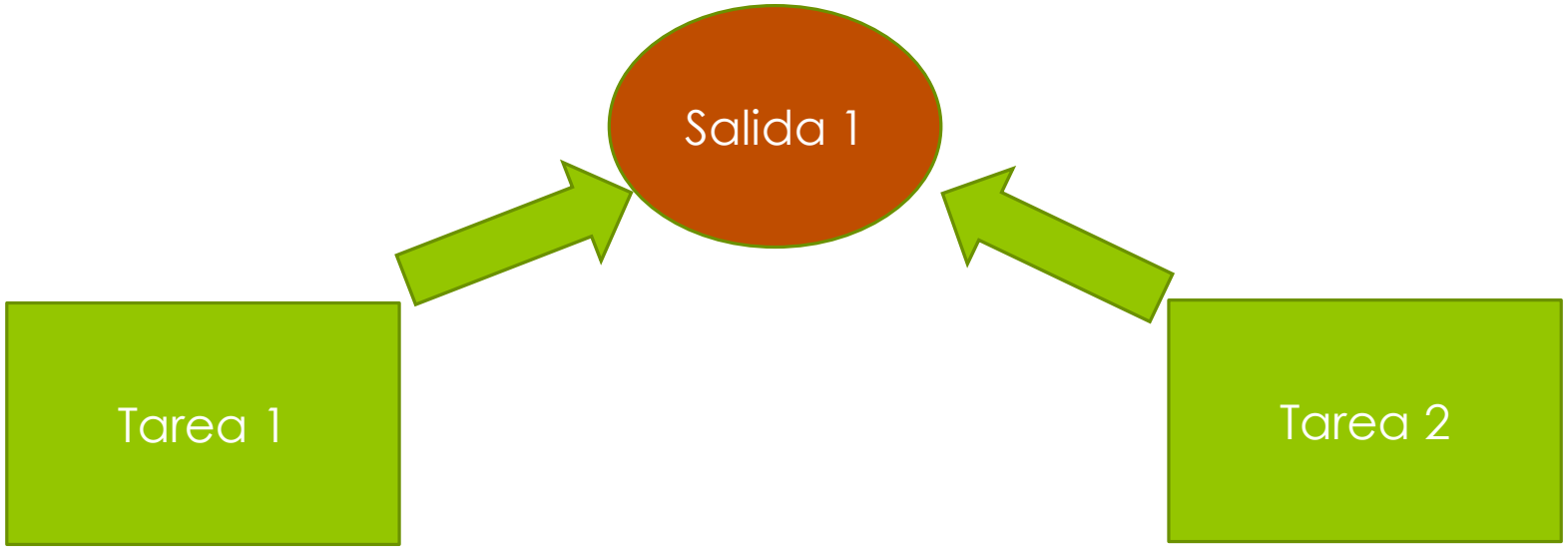


Eventos - Mutex

- PRECAUCIONES de USO
  - Es la alternativa de uso más sencilla.
  - Se modifican las reglas de ejecución.
    - La tarea de mas alta prioridad es la que se ejecuta.
  - Se debe tener presente :
    - La inversión de prioridades
    - El deadlock.

Ejemplo

- Dos tareas desean acceder a un puerto I/O.



Ejemplo

Ejemplo

- Se pueden usar diversos mecanismos
  - Usar Activación/ Desactivación de Interrupciones.
  - Usar Activación/Desactivación del Scheduler.
  - Usar Semáforos binarios
  - Usar Mutex

Solución

```
for(;;) {
    OS_ENTER_CRITICAL();

    salidaLED_1 = 1;
    salidaLED_2 = 1;

    OS_EXIT_CRITICAL();

    OSTimeDly(2);

    salidaLED_1 = 0;
    salidaLED_2 = 0;
    OSTimeDly(2);
}
```

```
for(;;){
    OSSchedLock();

    salidaLED_1 = 1;
    salidaLED_3 = 1;
    OSTimeDly(5);

    salidaLED_1 = 0;
    salidaLED_3 = 0;
    OSTimeDly(5);

    OSSchedUnlock();
```

Solución

- Uso MUTEX

```
for(;;) {
    OSMutexPend(ADC_Mutex, 0, &err);

    salidaLED_1 = 1;
    salidaLED_2 = 1;
    OSTimeDly(5);

    salidaLED_1 = 0;
    salidaLED_2 = 0;
    OSTimeDly(2);

    OSMutexPost(ADC_Mutex);
}
```

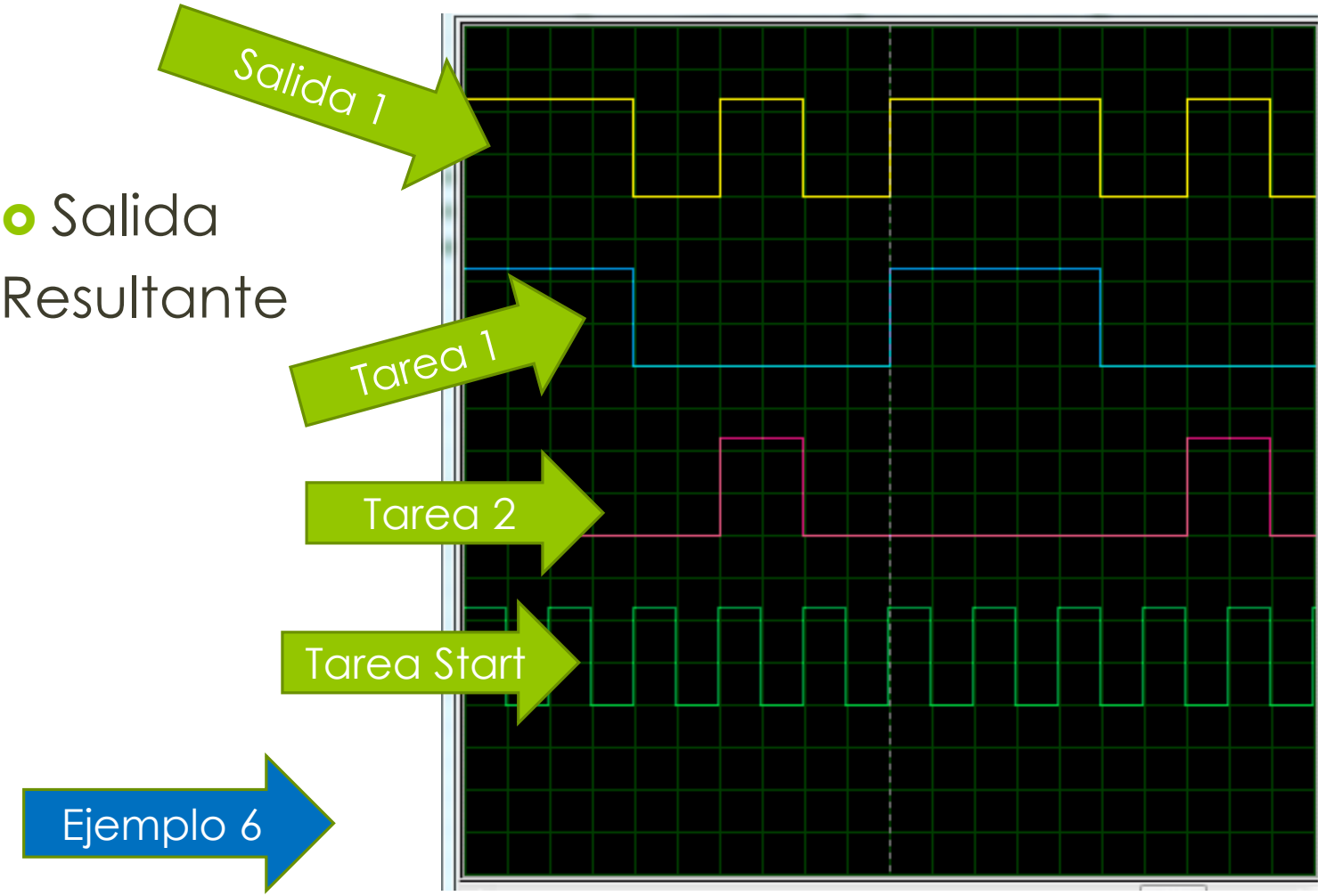
```
for(;;){
    OSMutexPend(ADC_Mutex, 0, &err);

    salidaLED_1 = 1;
    salidaLED_3 = 1;
    OSTimeDly(2);

    salidaLED_1 = 0;
    salidaLED_3 = 0;
    OSTimeDly(2);

    OSMutexPost(ADC_Mutex);
}
```

- Salida Resultante



RTOS - Deadlock

- Existe DEADLOCK cuando dos tareas están esperando los recursos que poseen la otra.

Ocurrencia de Deadlock

```
INT16U timeOut = 0;
for(;;)
{
    OSSemPend(Semaforo2,timeOut,error);
    switch(*error){
        case OS_NO_ERR:
            Nop();
            break;
        case OS_TIMEOUT:
            Nop();
            break;
    }

    salidaLed_1 = 1;
    OSSemPost(Semaforo1);
    OSTimeDly(20);
}
```

```
INT16U timeOut = 20;
for(;;)
{
    OSSemPend(Semaforo2,timeOut,error);
    switch(*error){
        case OS_NO_ERR:
            Nop();
            break;
        case OS_TIMEOUT:
            Nop();
            break;
    }

    salidaLed_1 = 1;
    OSSemPost(Semaforo1);
    OSTimeDly(20);
}
```

```
for(;;)
{
    OSSemPend(Semaforo1,0,err);
    salidaLed_2 = 1;
    OSSemPost(Semaforo2);
    OSTimeDly(20);
}
```

RTOS - Deadlock

- SOLUCION
  - La manera más sencilla de evitar deadlock es utilizar timeout.
  - Es necesario cuando se libera el semáforo binario o el semáforo mutex, verificar el resultado de la operación.
  - OSMutexPend() devuelve el resultado.
    - OS_ERR_NONE
    - OS_ERR_MUTEX_NESTING
    - ETC

Cambios de Contexto

- Se producen cambios de contexto cuando expira el tiempo establecido para funcionar.
- Se producen cuando una Tarea solicita un Servicio del RTOS.
- Los cambios de contexto pueden provocar instancias de Deadlock si no son tenidos en cuenta.

Cambios de Contexto

- Se puso para el ejemplo un led que se activa y desactiva cuando se produce el Context Switch.
- Se puso un led que se activa cuando se ejecuta la ISR del timer.

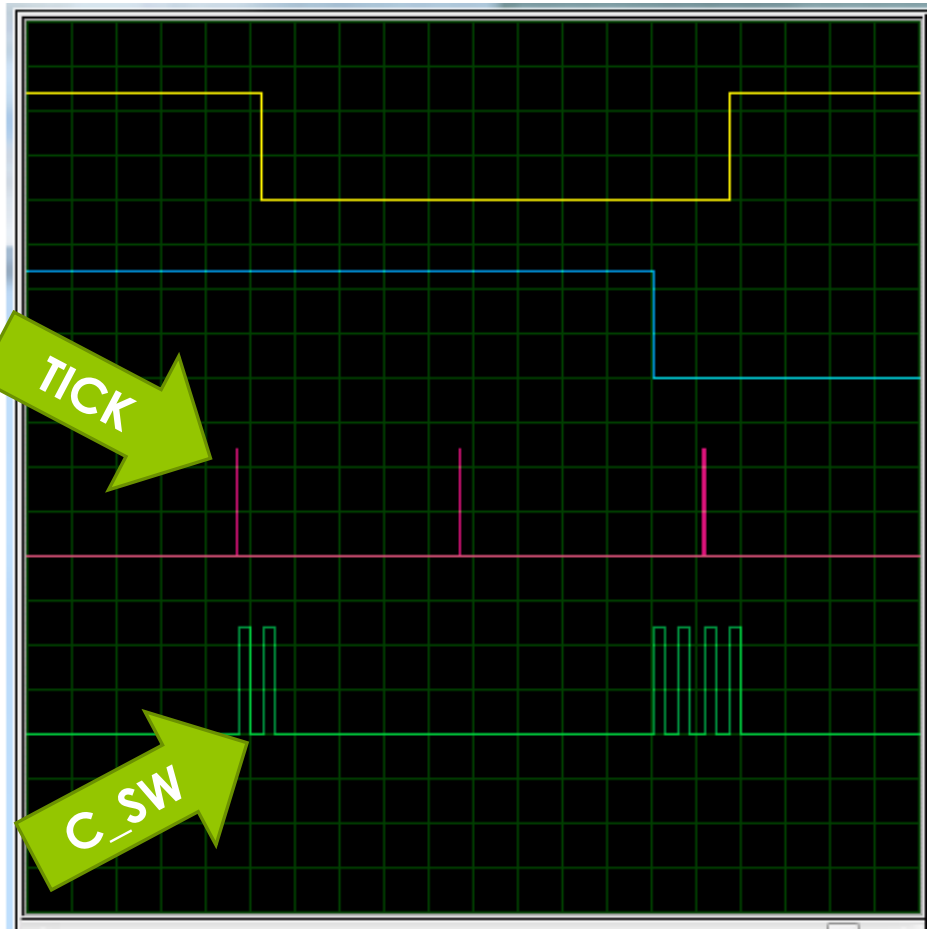
```
void CPUInterruptHook(void)
{
    if(INTCONbits.TMR0IF) {

        salidaLed3 = 1;

        INTCONbits.TMR0IF = 0;

        TMR0H = 60;
        TMR0L = 251;

        OSTimeTick();
        salidaLed3 = 0;
    }
}
```




```
void OSTaskSwHook (void)
{
    salidaLed4 = 1;
    Delay10TCYx(10L);
    salidaLed4 = 0;
    Delay10TCYx(10L);
}
```