Carrera de Especialización en Sistemas Embebidos

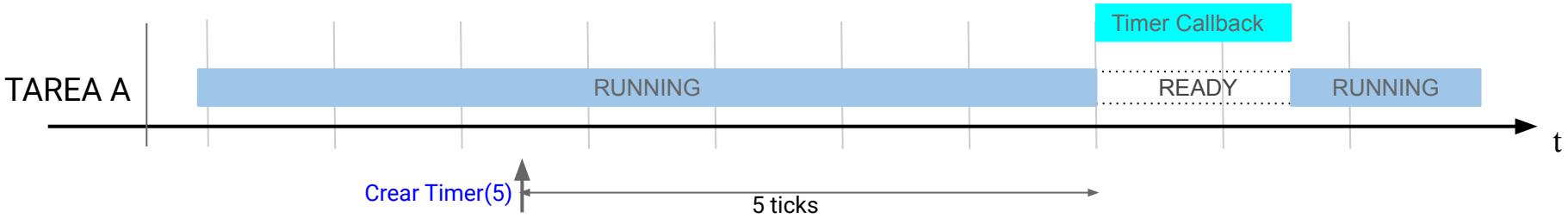
Sistemas Operativos en Tiempo Real II

Clase 3: Timers en FreeRTOS

ACSE
Asociación Civil para la Investigación, Promoción y Desarrollo de los Sistemas Electrónicos Embebidos
FACULTAD DE INGENIERIA
Universidad de Buenos Aires
CIM
CADIEEL

Timers

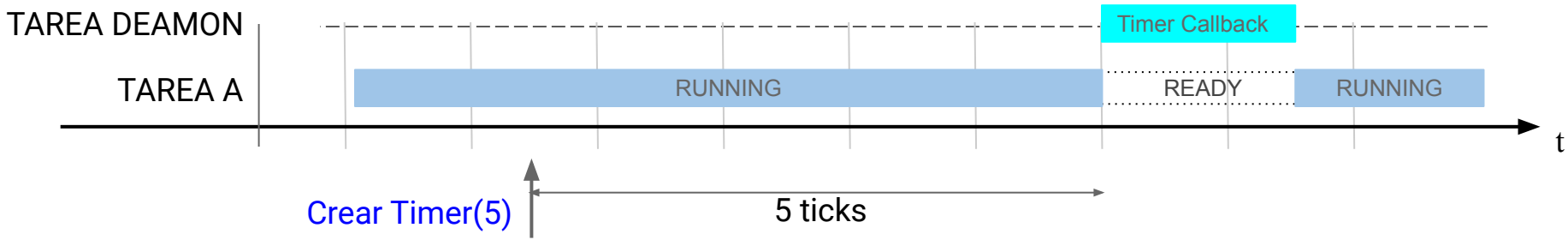
- Los timers en FreeRTOS sirven para generar eventos de timeout asincrónicos con la ejecución de las tareas.
  - Otra manera de expresarlo es que planifican la ejecución de una cierta función de manera temporizada a futuro.
  - Ésta función es denominada "callback del timer".
- Son objetos del kernel del OS, y está implementado por software (no dependen de ningún timer por hardware, solamente del tick del OS)



- Para usarlos:
  - Se deberá compilar timers.c
  - Se deberá configurar en FreeRTOSConfig.h configUSE_TIMERS == 1

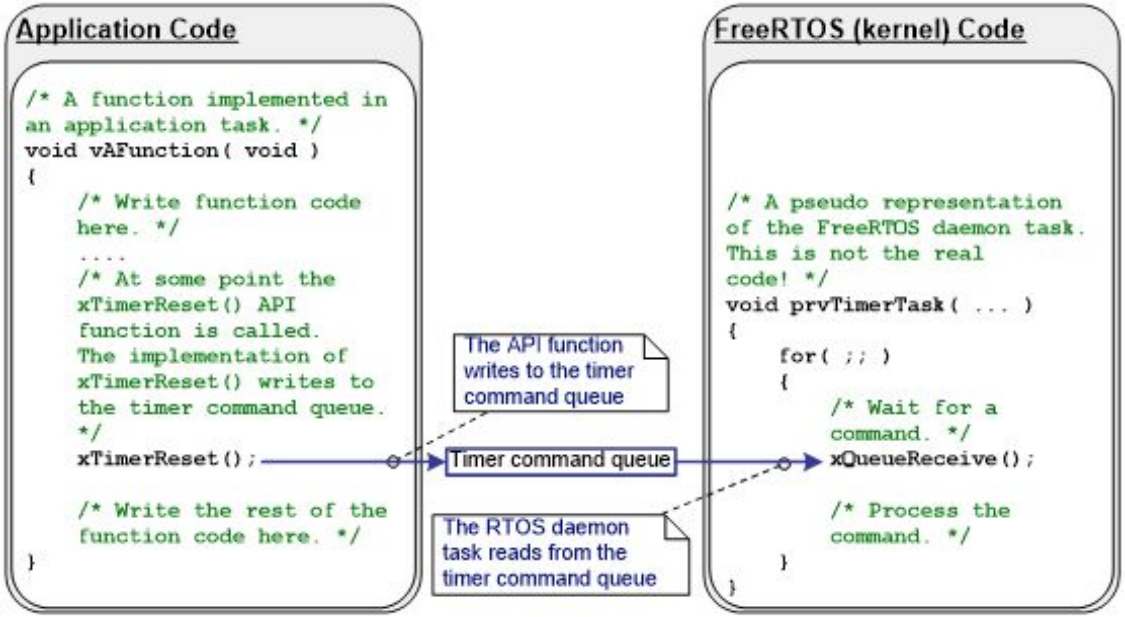
Timers : RTOS Deamon Task

- ¿ En qué contexto corren los callbacks ? ¿ isr ? ¿ tareas ?
- Tarea DEAMON: Es una tarea que crea el OS al arrancan al Scheduler, sin intervención del usuario.
- La prioridad de la tarea DEAMON la define el usuario en la macro configTIMER_TASK_PRIORITY en FreeRTOSConfig.h
- El tamaño de stack de esta tarea, la define el usuario en la macro configTIMER_TASK_STACK_DEPTH en FreeRTOSConfig.h



Timers : RTOS Deamon Task

- ¿ Cómo se comunica con las otras tareas ?
  - Implementa una cola de comandos de entrada (cuya cantidad de elementos se configura con configTIMER_QUEUE_LENGTH)



Timers: Implementacion de Callbacks

- El prototipo no es como una tarea.
  - `void timer_callback( TimerHandle_t handle );`
- Deberán tener un corto tiempo de ejecución.
- Por ejecutarse en contexto de tarea, pueden utilizarse todas las funciones de la API de FreeRTOS, pero... OJO!
- No deben utilizar funciones de la API que bloquee la tarea DEAMON.
  - Puede utilizar cualquier función de la API para enviar señales a otras tareas.
  - Si la funcion posee TicksToWait, el valor debe ser 0.

API: Creación

```
TimerHandle_t timer = xTimerCreate( const char * const pcTimerName,
                                    TickType_t xTimerPeriodInTicks,
                                    UBaseType_t uxAutoReload,
                                    void * pvTimerID,
                                    TimerCallbackFunction_t pxCallbackFunction  );
```

pcTimerName = Nombre de fantasia (usado para debuguear)

xTimerPeriodInTicks = Tiempo para la ejecución del callback.

uxAutoReload = Si es pdTrue, el callback se llama periódicamente.

pvTimerID = Puntero a estructura de usuario para compartir entre entre la tarea que arranca el timer y el callback

pxCallbackFunction = Referencia a la función que ejecuta el evento de timeout.

Retorna el handler al timer.

API: Borrado

```
BaseType_t result = xTimerDelete( TimerHandle_t xTimer,
                                  TickType_t xBlockTime );
```

xTimer = handler al timer

xBlockTime = tiempo a esperar a que la operación se envía al DEAMON y se efectúe.

Retorna pdTrue si se destruyó
pdFalse, si dio Timeout.

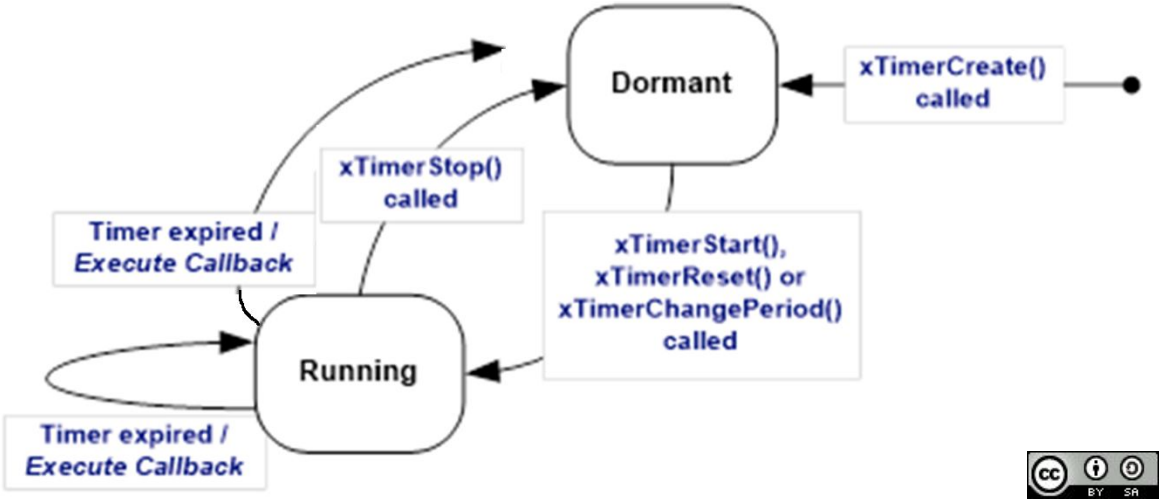
API: Start / Stop

```
BaseType_t result = xTimerStart( TimerHandle_t xTimer,
                                 TickType_t xBlockTime );
```

- Arranca el timer si estaba frenado
- Reinicia el timer si estaba corriendo (equivalente a xTimerReset(...))

```
BaseType_t result = xTimerStop( TimerHandle_t xTimer,
                                TickType_t xBlockTime );
```

- Frena el timer.



Timers: USO

```
TimerHandle_t Timer1;
TimerHandle_t Timer2;
char array1[10];
char array2[30];

void callback( TimerHandle_t xTimer )
{
   char *array =( char* ) pvTimerGetTimerID( xTimer );

   if( xTimer == Timer1 )
   {
      ...
   }
   if( xTimer == Timer2 )
   {
      ...
   }
}
```

```
void main( void )
{
   long x;

   Timer1 = xTimerCreate ( "", 300 , pdTRUE, array1 , callback);
   Timer2 = xTimerCreate ( "", 200 , pdTRUE, array2 , callback);

   xTimerStart( Timer1 , 0 );
   xTimerStart( Timer2 , 0 );

   vTaskStartScheduler();
}
```

Bibliografia

- Amazon FreeRTOS - Software Timers

Licencia

"Timers en FreeRTOS"