Trabajo Práctico 3

Seminario de Sistemas Embebidos

June 14, 2011

Contents

1 Temporización

Se practicarán las funciones de temporización de FreeRTOS:
vTaskDelay (portTickType ticks);
vTaskDelayUntil (portTickType tiempoUltimaActivacion, portTickType ticks);
y el uso del vApplicationTickHook (void) para incrementar un contador y medir así tiempo trasncurrido.

1.1 Demoras fijas

Caso de uso: Activar un actuador o válvula o esperar un tiempo conocido.
Ejemplo: Activar un led durante 500ms cada 1s.

1.2 Períodos fijos

Caso de uso: Hacer que un proceso repetitivo se ejecute siempre con la misma periodicidad, como ser activar una señal modulada por ancho de pulso.
Ejemplo: Hacer una onda cuadrada de período 1s, con ciclos de actividad incrementándose 100ms, 200ms, 300ms...

1.3 Medir tiempo transcurrido: ENTREGA OBLIGATORIA

Caso de uso: Medir cuánto tiempo duró un proceso determinado.
Se puede hacer de dos maneras:

1. Se puede consultar el contador de ticks del RTOS, mediante llamadas a xTaskGetTickCount() para obtener el tiempo del sistema (en ticks) al inicio y al fin del mismo. En este caso hay que preveer que esta variable puede desbordar, dando entonces la resta resultados erroneos. Si bien es un suceso poco probable (el timer del sistema tendría que desbordar en el medio de la medición, y ya que es una variable de 32 bits cuenta MUCHO tiempo, unos 50 días con un tick de 1 ms), en un sistema en producción puede darse.

2. También se puede usar la funcionalidad de vApplicationTickHook() para que incremente una variable contador del usuario. Esto no tiene el anterior problema, el único riesgo potencial es que haya una sección crítica del sistema demasiado larga y en ese interim se pierdan interrupciones de timer, pero esto es una condición de mal diseño que no debiera darse. Se recomienda esta última manera que además aporta el uso del TickHook.

Ejemplo: Medir usando el vApplicationTickHook() el tiempo de pulsación de un botón, aplicándole al mismo anti-rebote. Luego destellar un led durante el tiempo medido.

1.4 Ejemplo integrador

Escribir un programa con dos tareas:

1. Una tarea medirá el tiempo de pulsación de un botón, aplicándole al mismo anti-rebote.

2. La otra destellará un led con un período fijo de 1s, y tomando como tiempo de activación el último tiempo medido.

Dado que todavía no se ha practicado el manejo de colas, el tiempo medido se puede comunicar entre tareas mediante una variable global, protegiendo ambas operaciones dentro de una sección crítica.

2 Sincronización de tareas mediante semáforos

Se practicará el uso de semáforos binarios para sincronizar tareas entre sí (la sincronizacion con interrupciones se deja para más adelante). También se practirá la instanciación múltiple de una tarea. Es importante afianzar la noción del paradigma productor-consumidor de datos.

2.1 Sincronizar dos tareas mediante un semáforo binario

Caso de uso: Una tarea consumidora está bloqueada esperando recibir un semáforo, mientras que una tarea generadora lo liberará cuando genere o descubra un dato.

Ejemplo: Una tarea sensa un pulsador y libera el semáforo cuando confirma el fin de la pulsación. La otra tarea destella un led cuando recibe el semáforo. La tarea esperará al semáforo indefinidamente.

2.2 Sincronizar varias tareas mediante un semáforo binario

Caso de uso: Varias tareas esperando un evento. Por ejemplo, un cambio de estado del sistema que debe ser atendido por varias tareas con prioridades diferentes.

Ejemplo: Instanciar 3 veces la tarea que destella el led, cada una aplicada a un color del led RGB del base board y hacer que las tres tareas generen el destello cuando se reciba el semáforo. Las tareas esperarán al semáforo indefinidamente.

El semáforo no es más que un mensaje al scheduler, recibido por cuantas tareas estuvieran esperando por él. Prestar atención al detalle del parámetro de tarea, este debe indicar a cada instancia creada qué color le corresponde.

2.3 Tiempo de bloqueo: ENTREGA OBLIGATORIA

Caso de uso: Una tarea espera un evento durante un tiempo acotado, ya que la ausencia del mismo podría indicar un problema en el generador del evento o la caducidad del dato esperado.

El tiempo de bloqueo es el tiempo máximo por el que se espera al semáforo. Cuando se usa uno, debe chequearse el valor de retorno de la llamada bloqueante para verificar si se recibió el semáforo o si expiró el tiempo de espera.

Ejemplo: Una tarea muestrea un pulsador y libera un semáforo cuando confirma su liberación. Otra tarea espera el semáforo por 1s, encendiendo el led verde al recibir el semáforo y encendiendo en cambio el rojo si expira el tiempo de bloqueo.

3 Comunicación de datos entre tareas

Se practicara el uso de colas para comunicar datos y para bloquear a las tareas aperiódicas que no deben recibir tiempo de CPU mientras no tienen eventos que procesar.

3.1 Pasaje de datos por copia: ENTREGA OBLIGATORIA

Este es el uso más simple de las colas del RTOS: Los mensajes se pasan por copia dentro de la cola, dándole persistencia a los mismos una vez que la tarea productora sale de contexto.

Caso de uso: Una tarea manejadora de evento bloquea hasta recibir el mismo.

Ejemplo: Una tarea mide el tiempo de opresión de un pulsador. Cuando lo obtiene lo envía por cola a otra tarea que destella una led durante el tiempo recibido.

3.2 Lectura no bloqueante de una cola: ENTREGA OBLIGATORIA

Caso de uso: El mensaje recibido por cola modifica de alguna manera el comportamiento de la tarea receptora, pero la misma debe seguir trabajando (no bloquear) aunque no reciba mensajes

Ejemplo: Una tarea destella contínuamente un led, con una frecuencia constante y un ciclo de actividad que recibe de otras tareas mediante una cola. La tarea no debe bloquearse, ya que mientras no reciba mensajes debe mantener el led titilando.

Recuerde el uso de vTaskDelayUntil para mantener la periodicidad.

3.3 Pasaje de datos por referencia

Caso de uso: Los mensajes a enviar por cola son demasiado grandes para pasar por copia (por ejemplo, un buffer a ser transmitido por puerto serie). Se pasa entonces por cola la dirección de los datos a procesar, pero en este caso debe asegurar el programador que los mismos se mantienen válidos hasta que el destinatario los recibe. Para esto se usa memoria global, y según el caso debe señalizarse la validez del mensaje (p. ej, mediante flags o variables de estado).

Ejemplo: Repetir la tarea del ejemplo anterior, pero en vez de pasar el dato por cola, pasar la direccion del mismo. Recuerde usar variables globales para mantener válido el dato cuando la tarea sale de contexto.

4 Manejo de recursos

Se practicará el uso de mutex para resguardar recursos compartidos entre tareas. Se practicará también el uso de semáforos contadores de eventos (ascendentes).

4.1 Cuenta de eventos con sincronizacion

Caso de uso: Sincronizar la generación de un evento con su consumidor, cuando la generación del evento es más rápida que su manejo (se acumulan eventos).

Ejemplo: Repetir el ejercicio 2.1 (semáforo binario), pero agregue la posibilidad de que se contabilicen hasta 3 pulsaciones del botón, que a su vez generarán 3 destellos del led.

4.2 Exclusión mutua de un recurso compartido: ENTREGA OBLIGATORIA

Caso de uso: Dos tareas comparten un recurso, debe turnarse el acceso al mismo para que no lo corrompan.

Ejemplo: Escribir un programa con tres tareas:

1. Una tarea que destella un led a 0,5Hz (500ms ON, 500ms OFF).
2. Otra tarea que mide el tiempo de pulsación de un botón. En cuanto se suelta, lo enviará por cola a la tarea 3 que hará destellar al led durante este tiempo.

Use un mutex para turnar el acceso al LED y que no se perturbe ninguna de las formas de onda. Lo importante es no interrumpir el tiempo de alto en exhibición, y no "pegarse" al tiempo de alto de la otra tarea (deje un tiempo de off como guardabanda).

5 Sincronización con interrupciones

Se practicará la sincronizacion de tareas con eventos generados o descubiertos por interrupciones.

5.1 Generar/Descburir un evento en una interrupción.

Caso de uso: Una interrupción de entrada de algun periférico o una IO descubre un evento. Se debe manejar el mismo en una tarea sincronizada y no en la ISR.

Ejemplo: Generar un pulso en un led sincronizado a una interrupción. Use un semáforo binario dado en la interrupción.

5.2 Procesar datos en una interrupción: ENTREGA OBLIGATORIA

Caso de uso: Situacion inversa al anterior el manejo del evento se produce en la interrupción de algún periferico de salida. Por ejempo un "transmitter empty" de una UART.

Ejemplo: Idem anterior, pero usando una cola, de modo de poder hacer el destello de tiempo variable.