Laboratorio 12:

Aplicaciones multihebra bajo el RTOS uC/OS-II

Programación de sistemas y dispositivos

José Manuel Mendías Cuadros
Dpto. Arquitectura de Computadores y Automática
Universidad Complutense de Madrid

Presentación

- Crear una aplicación multihebra bajo uC/OS-II.
  - El punto de partida es una aplicación funcional que consta de 6 tareas concurrentes:
    1. Cada 500 ms, alterna el led que se enciende.
    2. Cada 100 ms, muestrea el keypad y, si hay una tecla pulsada, envía su scancode a otras tareas.
    3. Cada segundo, envía por la UART0 la hora del RTC.
    4. Cada 10 segundos, envia por la UART0 el número de ticks del sistema transcurridos desde su inicio.
    5. Cada vez que reciba un scancode lo envía por la UART0.
    6. Cada vez que reciba un scancode lo muestra por el display 7-segmentos.
  - Adicionalmente, existirá una hebra adicional: una RTI por pulsación de pulsador
    - Cada vez detecte una pulsación de cualquier pulsador, enviará por la UART0 un mensaje.
  - La UART0, como recurso compartido, esta protegido por un semáforo.
  - Las 3 tareas que producen/consumen scancodes utilizarán una cola tipo FIFO.
  - Esta aplicación se ampliará con 2 tareas concurrentes adicionales:
    7. Mostrará por el LCD cada una de las teclas pulsadas.
       - Deberá ser un consumidor adicional de la cola tipo FIFO de scancodes
    8. Mostrará por el LCD los segundos transcurridos desde que se inició.
       - Incrementará cada segundo un contador interno y lo visualizará por el LCD

Aplicación multihebra

declaraciones

```
...

#define TASK_STK_SIZE    10*1024

OS_STK Task1Stk[TASK_STK_SIZE];
OS_STK Task2Stk[TASK_STK_SIZE];
...
OS_STK TaskStartStk[TASK_STK_SIZE];

#define KEYPAD_QUEUE_SIZE 64

OS_EVENT *uart0Sem;
OS_EVENT *keypadQueue;
void     *keypadQueueTable[KEYPAD_QUEUE_SIZE];

void Task1( void *id );
void Task2( void *id );
...
void TaskStart( void *pdata );

extern void OSTickISR( void );
extern void OS_CPU_isr_pb( void );
void isr_pb( void );
```

Declara las pilas de cada tarea

Declara semáforode protección de la UART0

Declara cola de scancodes

Reserva espacio para mensajes

Declara tareas

RTI por presión de pulsador

RTI (wrapper) por presión de pulsador

Función invocada en la RTI para atención del dispositivo
*(**sin atributo interrupt**)*

Aplicación multihebra

programa principal

```
void main( void )
{
  sys_init();
  timers_init();
  ...
  keypad_init();

  uart0_puts( "\n\n Ejecutando uCOS-II (version " );
  uart0_putint( OSVersion() );
  uart0_puts( ")\n" ) ;
  uart0_puts( "--------------------------------\n\n" ) ;

  OSInit();

  uart0Sem = OSSemCreate( 1 );
  keypadQueue = OSQCreate( &keypadQueueTable[0], KEYPAD_QUEUE_SIZE );

  OSTaskCreate( TaskStart, NULL, &TaskStartStk[TASK_STK_SIZE - 1], 0 );

  OSStart();
}
```

Inicializa dispositivos

Muestra versión

Inicializa el Kernel

Crea recursos (semáforo y cola)

Crea la tarea inicial de arranque

Inicia multitarea

Aplicación multihebra

tarea inicial

```
void TaskStart( void *pdata )
{
  const char id1 = '1';
  const char id2 = '2';
  ...
  const char id6 = '6';

  OS_ENTER_CRITICAL();
  timer0_open_tick( OSTickISR, OS_TICKS_PER_SEC );
  pbs_open( OS_CPU_isr_pb );
  OS_EXIT_CRITICAL();

  OSTaskCreate( Task1, (void *)&id1, &Task1Stk[TASK_STK_SIZE - 1], 1 );
  OSTaskCreate( Task2, (void *)&id2, &Task2Stk[TASK_STK_SIZE - 1], 2 );
  ...
  OSTaskCreate( Task6, (void *)&id6, &Task6Stk[TASK_STK_SIZE - 1], 6 );

  OSTaskDel(OS_PRIO_SELF);
}
```

Identificadores de tareas

Instala RTI

Crea tareas

La tarea inicial de arranque se auto-elimina

Aplicación multihebra

tareas (i)

```
void Task1( void *id )
{
  INT8U err;

  OSSemPend( uart0Sem, 0, &err );
    uart0_puts( "  Task" );
    uart0_putchar( *(char *)id );
    uart0_puts( " iniciada.\n" );
  OSSemPost( uart0Sem );

  led_on( LEFT_LED );
  led_off( RIGHT_LED );

  while( 1 )
  {
    OSTimeDly( 50 );
    led_toggle( LEFT_LED );
    led_toggle( RIGHT_LED );
  }
}
```

Protege el acceso a la UART con un semáforo

Muestra un mensaje de presentación por la UART0

La tarea realiza su función indefinidamente

Suspende la tarea durante 0,5 segundos (50 ticks)

Conmuta el estado de los leds

Aplicación multihebra

tareas (ii)

```
void Task2( void *id)
{
  INT8U err;
  uint8 scancode;

  OSSemPend( uart0Sem, 0, &err );
    uart0_puts( "  Task" );
    ...
  OSSemPost( uart0Sem );

  while( 1 )
  {
    OSTimeDly( 10 );
    scancode = keypad_scan();
    if( scancode != KEYPAD_FAILURE )
    {
      OSTimeDly( 3 );
      OSQPostOpt( keypadQueue, (void *) scancode, OS_POST_OPT_BROADCAST );
      while( scancode == keypad_scan() )
        OSTimeDly( 10 );
      OSTimeDly( 10 );
    }
  }
}
```

Protege el acceso a la UART con un semáforo

Muestra un mensaje de presentación por la UART0

Muestrea el teclado esperando presión cada 0,1 segundos (10 ticks)

*Encola el scancode para que sea leído por **todos** los consumidores*

Espera rebote de presión

Muestrea el teclado esperando depresión cada 0,1 segundos (10 ticks)

Espera rebote de depresión

Aplicación multihebra

tareas (iii)

```
void Task3( void *id )
{
  INT8U err;
  rtc_time_t rtc_time;

  OSSemPend( uart0Sem, 0, &err );
    uart0_puts( "  Task" );
    ...
  OSSemPost( uart0Sem );

  while( 1 )
  {
    rtc_gettime( &rtc_time );
    OSSemPend( uart0Sem, 0, &err );
      uart0_puts( "  (Task" );
      uart0_putchar( *(char *)id );
      uart0_puts( ") Hora: " );
      uart0_putint( rtc_time.hour );
      ...
    OSSemPost( uart0Sem );
    OSTimeDly( 100 );
  }
}
```

Protege el acceso a la UART con un semáforo

Muestra un mensaje de presentación por la UART0

Lee la hora del RTC

Protege el acceso a la UART con un semáforo

Muestra la hora del RTC por la UART0

Suspende la tarea durante 1 segundo (100 ticks)

Aplicación multihebra

tareas (iv)

```
void Task4( void *id )
{
  INT8U err;
  INT32U ticks;

  OSSemPend( uart0Sem, 0, &err );
    uart0_puts( "  Task" );
    ...
  OSSemPost( uart0Sem );

  while( 1 )
  {
    ticks = OSTimeGet();
    OSSemPend( uart0Sem, 0, &err );
      uart0_puts( "  (Task" );
      uart0_putchar( *(char *)id );
      uart0_puts( ") Ticks: " );
      uart0_putint( ticks );
      uart0_puts( "\n" );
    OSSemPost( uart0Sem );
    OSTimeDly( 1000 );
  }
}
```

Protege el acceso a la UART con un semáforo

Muestra un mensaje de presentación por la UART0

Obtiene del RTOS el número de ticks trasncurridos

Protege el acceso a la UART con un semáforo

Muestra la hora del RTC por la UART0

Suspende la tarea durante 10 segundos (1000 ticks)

Aplicación multihebra

tareas (v)

```
void Task5( void *id )
{
  INT8U err;
  uint8 scancode;

  OSSemPend( uart0Sem, 0, &err );
    uart0_puts( "  Task" );
    ...
  OSSemPost( uart0Sem );

  while( 1 )
  {
    scancode = (uint8) OSQPend( keypadQueue, 0, &err );
    OSSemPend( uart0Sem, 0, &err );
      uart0_puts( "  (Task" );
      uart0_putchar( *(char *)id );
      uart0_puts( ") Tecla pulsada: " );
      uart0_puthex( scancode );
      uart0_puts( "\n" );
    OSSemPost( uart0Sem );
  }
}
```

Protege el acceso a la UART con un semáforo

Muestra un mensaje de presentación por la UART0

Desencola el scancode

Protege el acceso a la UART con un semáforo

Muestra el scancode por la UART0

Aplicación multihebra

tareas (vi)

```
void Task6( void *id )
{
  INT8U err;
  uint8 scancode;

  OSSemPend( uart0Sem, 0, &err );
    uart0_puts( "  Task" );
    ...
  OSSemPost( uart0Sem );

  while( 1 )
  {
   scancode = (uint8) OSQPend( keypadQueue, 0, &err );
   segs_putchar( scancode );
  }
}
```

Protege el acceso a la UART con un semáforo

Muestra un mensaje de presentación por la UART0

Desencola el scancode

Muestra el scancode por el display 7-segmentos

Aplicación multihebra

RTI

```
void isr_pb( void )
{
  if( OSSemAccept( uart0Sem ) )
  {
    uart0_puts( "  (INT) Se ha pulsado algún pushbutton...\n" );
    OSSemPost( uart0Sem );
  }
  EXTINTPND = BIT_RIGHTPB;
  EXTINTPND = BIT_LEFTPB;
  I_ISPC = BIT_PB;
}
```

Muestra un mensaje por la UART0

Protege el acceso a la UART con un semáforo (no bloqueante)

```
.include "../os_port/os_cpu_isr_wrapper.asm"

  .extern isr_pb
  .global OS_CPU_isr_pb

.section .text

OS_CPU_isr_pb:
  OS_CPU_ISR_WRAPPER isr_pb
```

Tareas

1. Crear el proyecto **`lab12`** a partir de una copia de uno anterior.
2. Descargar de la Web en el directorio **`lab12`** los ficheros:
   - **`lab12.c`** y **`lab12-isr-wrappers.asm`**
3. Descargar del CV y descomprimir en el directorio **`lab12`** el fichero:
   - **`uCosII276.rar`**
4. Refrescar el proyecto **`lab12`**.
5. Compilar el proyecto original **`lab12`**.
6. Crear una configuración de depuración **`lab12`** a partir de una anterior.
7. Arrancar Termite.
8. Conectar la placa y encenderla.
9. Arrancar OpenOCD.
10. Arrancar la configuración de depuración **`lab12`**.
11. Repetir el proceso añadiendo las tareas adicionales propuestas.