Laboratorio 14:

Multitarea bajo RTOS uC/OS-II

Programación de sistemas y dispositivos

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

Presentación

- Objetivo: Desarrollar una aplicación multitarea bajo uC/OS-II.

- El punto de partida es una aplicación que realiza de 7 tareas :
  1. Cada 500 ms, alterna el led que se enciende.
  2. Cada 50 ms, lee el keypad y, si hay una tecla pulsada, envía su scancode a otras tareas.
     - Usa un mailbox (facilitado por uC/OS-II) como mecanismo de comunicación.
  3. Cada segundo, muestra por la UART0 la hora leída del RTC.
  4. Cada 10 s, muestra por la UART0 el número de ticks transcurridos desde el inicio.
     - Usa una llamada al sistema uC/OS-II para conocer el número de ticks.
  5. Muestra por la UART0 cada una de las teclas pulsadas.
  6. Muestra en el display 7-segmentos cada una de las teclas pulsadas.
  7. Indica por la UART0 la detección (por interrupción) de la presión de un pulsador.
     - La RTI activa un flag (implementado mediante un semáforo facilitado por uc/OS-II) cada vez que detecta presión.
- Adicionalmente, todas las tareas envían un mensaje a la UART0 a su inicio.

Presentación

- Las tareas se ejecutan en un kernel de planificación expropiativo.
  - Las tareas son hebras en foreground diferentes que compiten por la UART0
    - Al ser un recurso compartido, está protegido por un semáforo (facilitado por uC/OS-II)

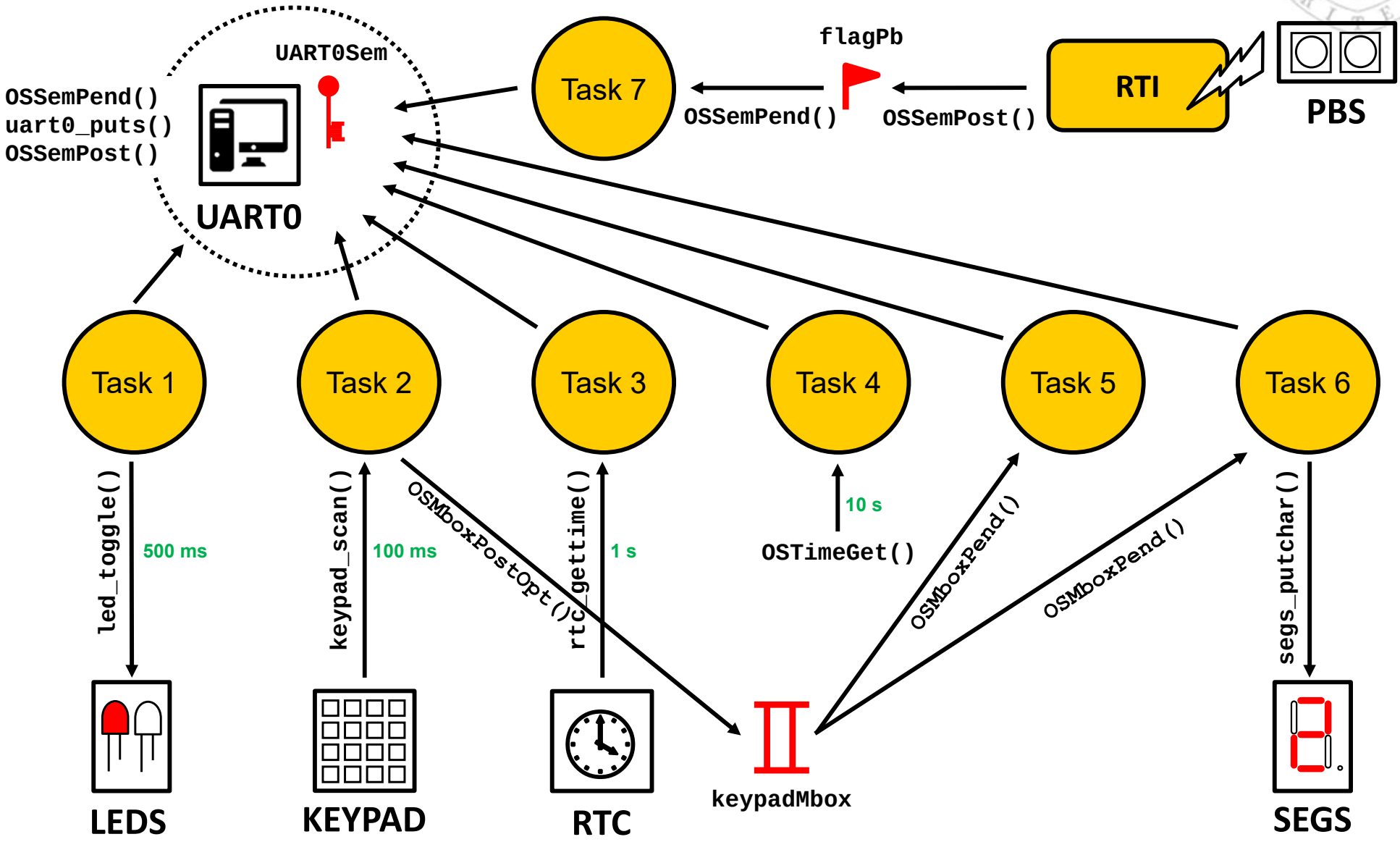
- En los directorios os_core y os_port se facilita el código fuente del sistema operativo completo (véase tema 7)

- Esta aplicación deberá ampliarse con 2 tareas adicionales:
  8. Muestra en el LCD cada una de las teclas pulsadas.
  9. Cada segundo, muestra en el LCD los segundos transcurridos desde el inicio.

J.M. Mendias
2020

Aplicación multitarea

arquitectura software



Aplicación multitarea

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];

OS_EVENT *uart0Sem;
OS_EVENT *keypadMbox;
OS_EVENT *flagPb;

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 semáforo de protección de la UART0

Declara buzón de scancodes

Declara flag (implementado como semáforo)

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 multitarea

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 );
  keypadMbox = OSMboxCreate( NULL );
  flagPb      = OSSemCreate( 0 );

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

  OSStart();
}
```

Inicializa dispositivos

Muestra versión

Inicializa el Kernel

Crea recursos (semáforo, buzón y flag)

Semáforo UART0 abierto

Buzón vacío

Flag desactivado

Crea la tarea inicial de arranque

Inicia multitarea

Aplicación multitarea

tarea inicial

```
void TaskStart( void *pdata )
{
  const char id1 = '1';
  const char id2 = '2';
  ...
  const char id7 = '7';

  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], 6 );
  OSTaskCreate( Task2, (void *)&id2, &Task2Stk[TASK_STK_SIZE - 1], 1 );
  ...
  OSTaskCreate( Task7, (void *)&id7, &Task7Stk[TASK_STK_SIZE - 1], 2 );

  OSTaskDel(OS_PRIO_SELF);
}
```

Identificadores de tareas

Instala RTI

Crea tareas para planificación Rate Monotonic

La tarea inicial de arranque se auto-elimina

Aplicación multitarea

tarea 1: alternancia de leds

```
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 );
  }
}
```

Muestra un mensaje de presentación por la UART0

Protege el acceso a la UART con un semáforo

Cuando tiene el acceso exclusivo envía el mensaje

La tarea realiza su función indefinidamente

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

Conmuta el estado de los leds

Aplicación multitarea

tarea 2: lectura del keypad y difusión del scancode

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

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

  while( 1 )
  {
    while( !keypad_pressed() )
      OSTimeDly( 5 );
    scancode = keypad_scan();
    if( scancode != KEYPAD_FAILURE )
      OSMboxPostOpt( keypadMbox, (void *) &scancode, OS_POST_OPT_BROADCAST );
    while( keypad_pressed() )
      OSTimeDly( 5 );
  }
}
```

Muestra un mensaje de presentación por la UART0

Protege el acceso a la UART con un semáforo

Cuando tiene el acceso exclusivo envía el mensaje

Muestrea el teclado esperando presión cada 50 ms (5 ticks)

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

Muestrea el teclado esperando depresión cada 50 ms (5 ticks)

Aplicación multitarea

tarea 3: informe de la hora

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

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

  while( 1 )
  {
    OSTimeDly( 100 );
    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 );
  }
}
```

Muestra un mensaje de presentación por la UART0

Protege el acceso a la UART con un semáforo

Cuando tiene el acceso exclusivo envía el mensaje

Suspende la tarea durante 1 segundo (100 ticks)

Lee la hora del RTC

Protege el acceso a la UART con un semáforo

Muestra la hora del RTC por la UART0

Aplicación multitarea

tarea 4: informe de los ticks transcurridos

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

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

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

Muestra un mensaje de presentación por la UART0

Protege el acceso a la UART con un semáforo

Cuando tiene el acceso exclusivo envía el mensaje

Suspende la tarea durante 10 segundos (1000 ticks)

Obtiene del RTOS el número de ticks transcurridos

Protege el acceso a la UART con un semáforo

Muestra los ticks transcurridos por la UART0

Aplicación multitarea

tarea 5: informe de la tecla pulsada

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

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

  while( 1 )
  {
    scancode = *((uint8 *) OSMboxPend( keypadMbox, 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 );
  }
}
```

Muestra un mensaje de presentación por la UART0

Protege el acceso a la UART con un semáforo

Cuando tiene el acceso exclusivo envía el mensaje

Desencola el scancode

Protege el acceso a la UART con un semáforo

Muestra el scancode por la UART0

Aplicación multitarea

tarea 6: visualización en display de la tecla pulsada

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

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

  while( 1 )
  {
    scancode = *((uint8 *) OSMboxPend( keypadMbox, 0, &err ));
    segs_putchar( scancode );
  }
}
```

Muestra un mensaje de presentación por la UART0

Protege el acceso a la UART con un semáforo

Cuando tiene el acceso exclusivo envía el mensaje

Desencola el scancode

Muestra el scancode por el display 7-segmentos

Aplicación multitarea

tarea 7: aviso de pulsador presionado

```
void Task7( void *id )
{
  INT8U err;

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

  while( 1 )
  {
    OSSemPend( flagPb, 0, &err );
    OSSemPend( uart0Sem, 0, &err );
      uart0_puts( "  (Task" );
      uart0_putchar( *(char *)id );
      uart0_puts( ") Se ha pulsado algún pushbutton...\n" );
    OSSemPost( uart0Sem );
  }
}
```

Muestra un mensaje de presentación por la UART0

Protege el acceso a la UART con un semáforo

Cuando tiene el acceso exclusivo envía el mensaje

Espera por la activación del flag y lo desactiva

Protege el acceso a la UART con un semáforo

Muestra aviso

Aplicación multitarea

RTI

```
void isr_pb( void )
{
  OSSemPost( flagPb );  ..... Señaliza la presión de cualquier pulsado
  EXTINTPND = BIT_RIGHTPB | EXTINTPND = BIT_LEFTPB;
  I_ISPC = BIT_PB;
}
```

```
.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
```

Acerca de *Creative Commons*

- Licencia CC (Creative Commons)

  - Ofrece algunos derechos a terceras personas bajo ciertas condiciones. Este documento tiene establecidas las siguientes:

Reconocimiento (*Attribution*):
En cualquier explotación de la obra autorizada por la licencia hará falta reconocer la autoría.

No comercial (*Non commercial*):
La explotación de la obra queda limitada a usos no comerciales.

Compartir igual (*Share alike*):
La explotación autorizada incluye la creación de obras derivadas siempre que mantengan la misma licencia al ser divulgadas.

Más información: https://creativecommons.org/licenses/by-nc-sa/4.0/