

SISTEMAS EMPOTRADOS

3º Grado en Ingeniería Informática

PRÁCTICA 9: Aplicación integrando un sistema operativo *Instant-Up*

9.1. Introducción

Vamos a realizar un pequeño ejercicio para utilizar el sistema operativo *Instant-Up* en un programa que active una señal periódica por el puerto P4.24. La señal tendrá una duración de 0'1 segundos tanto a nivel alto como a nivel bajo.

9.2. Material necesario:

- Ordenador personal con *Windows*.
- Placa de desarrollo MCB2300 de *Keil*.
- Adaptador USB-JTAG de la familia ULINK para depurar programas.
- Dos cables USB A-B.

9.3. Desarrollo de la práctica:

Crearemos una carpeta nueva para la realización de la práctica donde copiaremos los ficheros `serial.c`, `retarget.c`, `misTipos.h`, y `LPC2300.s` de cualquiera de las prácticas anteriores. Los ficheros en que realizaremos modificaciones son los siguientes:

- `Practica_SO_GPIO.c`: En este fichero se realizará el programa principal de la aplicación, desde donde llamaremos al resto de las funciones.
- `HAL.c`: Fichero para definir el *hardware* que utilizaremos en la aplicación, en esta práctica sólo el GPIO P4.24 y el *timer* 1.

El fichero fuente del sistema operativo estará disponible en la plataforma *Moodle*.

Al crear un nuevo proyecto deberemos de incluir estos archivos de manera que el manejador de proyectos quede, por ejemplo, como se indica en la figura 9-1.

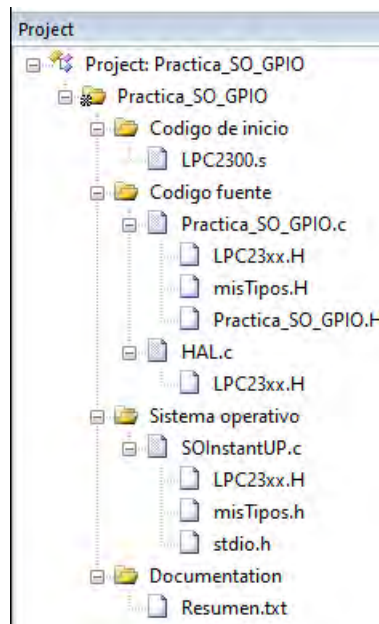


Figura 9-1: Vista previa del manejador de proyectos (*Project Manager*).

El sistema operativo se integra como parte de una nuestra aplicación. Por tanto, además de disponer del código fuente, deben hacerse una serie de inicializaciones y llamadas a funciones de inicialización del sistema operativo.

Los pasos a seguir en la función *main* de nuestra aplicación son:

- Se deben inicializar las variables globales del sistema operativo llamando a la función:

```
SO_Inicio_TCB_Var( )
```

- Se inicializa el temporizador *hardware* asociado al sistema operativo (*timer 1* en el código original) habilitando la interrupción y arrancando la cuenta:

```
ActVIC_T1( )           // coloca los valores en el VIC para el T1
```

```
startT1( )             // timer asociado al SO
```

- Una vez inicializados el resto de los periféricos, se arranca el sistema operativo habilitando la interrupción de tiempo real (en el código original la ISR asociada al *timer 1*):

```
SOStart( )
```

- Se definen las tareas al sistema operativo haciendo una llamada por tarea a la función:

```
SOAgrego_Tarea( pointerTask, delay, period );
```

- En la última línea de la función *main* se llama continuamente al planificador/despachador:

```
while(1)                                /* Llamada continua al planificador/despachador */
{
    SODespachador_Tareas();
}
```

```
1  /* ***** */
2  /* Practica 9: Aplicación utilizando el sistema operativo Instan-Up */
3  /* ***** */
4  #include <LPC23xx.H>
5  #include "misTipos.H"
6  #include "Practica_SO_GPIO.H"
7  UINT32 image_pin24;
8
9  extern void hardwareInit(void);
10 extern void SO_Inicio_TCB_Var(void);
11 extern void startTl(void);
12 extern void ActVIC_Tl(void);
13 extern void SOStart(void);
14 extern void SODespachador_Tareas(void);
15 extern unsigned int SOAgrego_Tarea(int (*)(void),const unsigned int ,const unsigned int );
16
17 int MyTask(void)
18 {
19     if (image_pin24==0)
20     {
21         PIN_P4_24_HIGH;
22         image_pin24=1;
23     }
24     else
25     {
26         PIN_P4_24_LOW;
27         image_pin24=0;
28     }
29     return 0;
30 }
31 int main(void)
32 {
33     PIN_P4_24_LOW;
34     image_pin24 =0; /* Inicializo la variable */
35     hardwareInit(); /* Inicializo recursos Hardware */
36     SO_Inicio_TCB_Var(); /* Llamo a la función que inicializa las variables del SO */
37     startTl(); /* Inicializo el timer asociado al sistema operativo, Tl */
38     ActVIC_Tl(); /* Coloca los valores en el VIC para el Tl */
39     SOStart(); /* Arranca el sistema operativo */
40     SOAgrego_Tarea(MyTask,100,100); /* Se definen las tareas al sistema operativo */
41     while(1) /* Llamada continua al planificador/despachador */
42     {
43         SODespachador_Tareas();
44     }
45 }
```

Figura 9-2: Inicialización del sistema operativo en el programa principal (*main*).

```
1  /* ***** */
2  /* Practica_SO_GPIO.h: definiciones para la práctica de S.O. */
3  /* Sistemas Empotrados. Universidad de Córdoba */
4  /* ***** */
5
6  #define PIN_P4_24_HIGH FIO4SET3 = 0x01; /* Pin P4.24 a alto */
7  #define PIN_P4_24_LOW FIO4CLR3 = 0x01; /* Pin P4.24 a bajo */
```

Figura 9-3: Fichero cabecera (*header*) con las definiciones de este ejercicio.

```

ca_SO_GPIO.c HAL.c Practica_SO_GPIO.H SOInstantUP.c misTipos.H Resumen.txt
1  /*****
2  /* HAL.C: funciones generales que acceden al hardware. */
3  /* Capa de abstracción del hardware (Hardware Abstract Layer) */
4  /* Sistemas Empotrados. Universidad de Córdoba */
5  *****/
6  #include <LPC23xx.H> /* LPC23xx definitions */
7  *****/
8  /* pinesSignalInit: Esta función configura los pines */
9  /* P4.24 y P4.25 como salida */
10 *****/
11 void pinesSignalInit(void)
12 {
13     PINSEL9 = 0x00000000;
14     PINMODE9 = 0x00000000;
15     FIO4DIR3 = 0x03;
16 }
17 *****/
18 /* hardwareInit: Esta función se llama al comienzo del programa */
19 /* para inicializar el Hardware */
20 *****/
21 void hardwareInit(void)
22 {
23     pinesSignalInit(); /* Configura los pines del circuito */
24 }
25

```

Figura 9-4: Fichero de la capa de abstracción del hardware (HAL).

9.4. Resultado de la práctica:

Los periféricos que interesa visualizar son el GPIO 4 y el *timer* 1. En el analizador lógico mostraremos la señal P4.24. Los valores de la señal, 0'1 segundos a nivel alto y a nivel bajo se deben observar tanto en el analizador lógico como en el osciloscopio.

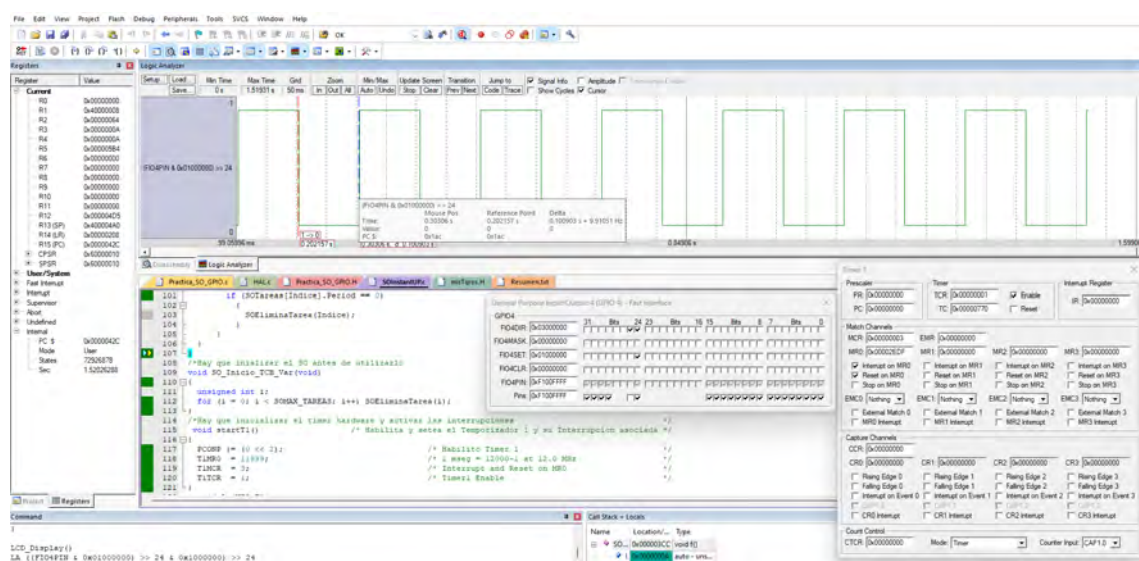


Figura 9-5: Resultado esperado.