SISTEMAS EMPOTRADOS
3º Grado en Ingeniería Informática

PRÁCTICA 7: Programación con el puerto serie

7.1.– Introducción

En esta práctica vamos a aplicar las dos prácticas anteriores. Partimos de la práctica 5, en las que se generaban dos señales por un puerto de salida utilizando un solo *timer* y trabajando por interrupciones, se va a realizar un interfaz de usuario que cambie la frecuencia de las señales. Se van a generar dos señales cuadradas cuyo semiperiodo será programable por el usuario a través del Terminal y por tanto podremos variar la frecuencia de estas señales digitales. Para descartar cualquier valor el usuario podrá pulsar la tecla ESC. Por otro lado, seguimos utilizando el puerto serie de la práctica anterior (práctica 6).

7.2.– Objetivos

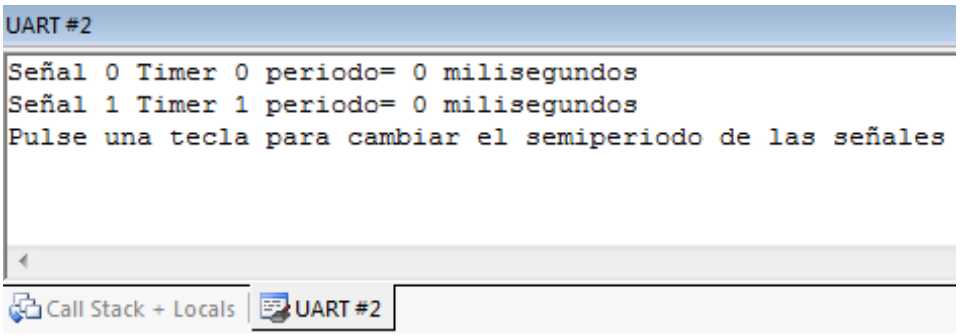
El objetivo de esta práctica será realizar un pequeño menú que se comunique con el usuario utilizando uno de los puertos serie de la placa MCB2300 y conectando al otro lado de la línea un ordenador personal que simule un terminal, por ejemplo, con el *Hyperterminal* de Windows XP.

7.3.– Material necesario:

- Ordenador personal con Windows XP.
- Placa de desarrollo MCB2300 de Keil.
- Adaptador USB–JTAG de la familia ULINK para depurar programas.
- Dos cables USB A–B.
- Cable serie RS232.

7.4.– Desarrollo de la práctica:

El interfaz de usuario que se mostrará en la pantalla del terminal tendrá un aspecto similar a este:



Una vez pulsada una tecla se pasará al menú de usuario que hará las siguientes preguntas:

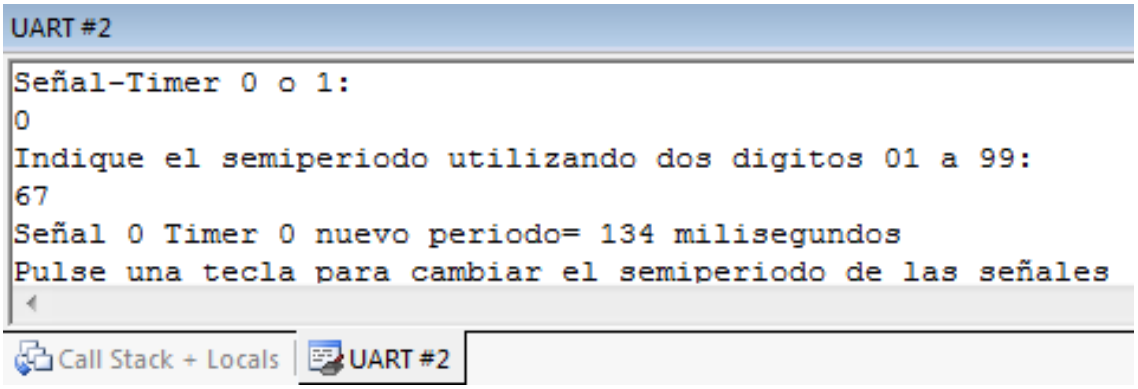
Señal–Timer 0 ó 1:

La aplicación espera a que el usuario pulse 0 ó 1 y mostrará el eco en pantalla. Si se pulsa una tecla distinta de 0 ó 1 la aplicación no mostrará eco y descartará la tecla.

A continuación. preguntará por el semiperiodo. Se aceptarán valores de dos dígitos y se descartarán las teclas distintas de un valor BCD:

Indique el semiperiodo utilizando dos digitos 01 a 99:

A continuación, indicará el *timer* que se ha cambiado y la nueva frecuencia mostrando algo similar a esto:



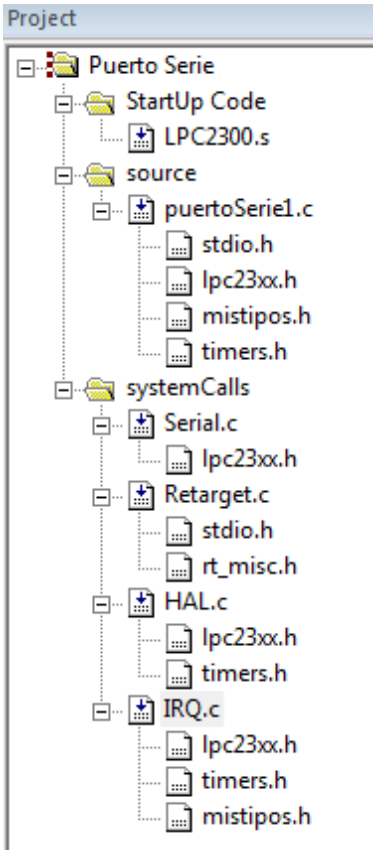
- **Indicaciones:**

Separar el menú del resto de la aplicación. Para ello se creará una función con el menú. El prototipo será:

UINT32 menuTimer(UINT32* timer, UINT32* semiperiodo)

La función devolverá un valor que indique a la función que llame, si el usuario ha abortado el menú pulsando la tecla ESC. Como argumentos tendrá el puntero a las variables que quedarán modificadas de la función que llama. El primer argumento estará relacionado con el *timer* (señal) y el segundo con el nuevo periodo.

Necesitaremos los ficheros de retarget.c y serial.c al igual que en la práctica anterior. También necesitaremos los ficheros IRQ.c y HAL.c de la práctica de los *timers* software con interrupciones. El manejador de proyectos tendrá una apariencia como la siguiente:



La parte de código que tiene el fichero serial.c para configurar el puerto serie del microcontrolador LPC2378 de nuestra placa es la siguiente:

```
36 void init_serial (void)  {                /* Initialize Serial Interface       */
37   #ifdef UART0
38     PINSEL0 |= 0x00000050;                 /* Enable TxD0 and RxD0              */
39   #elif defined (UART1)
40     PINSEL0 |= 0x40000000;                 /* Enable TxD1                       */
41     PINSEL1 |= 0x00000001;                 /* Enable RxD1                       */
42   #endif
43   UxFDR    = 0;                          /* Fractional divider not used       */
44   UxLCR    = 0x83;                       /* 8 bits, no Parity, 1 Stop bit     */
45   UxDLL    = 78;                         /* 9600 Baud Rate @ 12.0 MHZ PCLK    */
46   UxDLM    = 0;                          /* High divisor latch = 0            */
47   UxLCR    = 0x03;                       /* DLAB = 0                          */
48 }
```

Se observa que en este fichero se ha definido el reloj de 12 MHz del LPC2378 y una velocidad de transmisión de 9600 baudios. Por tanto, nuestra conexión del *hyperterminal* tendremos que definirla para 9600 baudios, 8 bits, sin paridad, un bit de parada y sin control de flujo.

El generador de velocidad de transmisión es un pre–escalador de dieciséis bits que divide el PCLK para generar una señal de reloj en la UART igual a 16 veces la velocidad de transmisión. Por tanto, la fórmula utilizada para calcular la velocidad de transmisión de la UART es:

```
Divisor = Pclk/(16 x BAUD)
```

En el caso de 12 MHz

```
Divisor = 12.000.000/(16 x 9600) = (approx) 78 o 0x4E
```

Con el valor aproximado de 78 se obtiene una velocidad de transmisión real de 9615 baudios. Normalmente no es posible obtener una velocidad de transmisión exacta para la UART, sin embargo, se trabaja con un error máximo del 5% en la temporización.

Tendremos que conectar con un cable serie RS232 nuestra placa por su COM1 a un puerto serie del ordenador, y observar el menú a través del *hyperterminal* de Windows.

Los valores obtenidos podremos medirlos en el osciloscopio, al igual que en las prácticas anteriores. Recordar que las señales son cuadradas y lo que hemos programado es el valor del semiperiodo.

7.5.– Ficheros auxiliares:

- **Fichero HAL.c:**

puertoSerie1.c | HAL.c | IRQ.c | misTipos.h | timers.h

```
/* HAL.C: funciones generales que acceden al hardware                         */
/* Sistemas Empotrados Universidad de Cordoba                                 */
/****************************************************************************/
#include <LPC23xx.H>                    /* LPC23xx definitions                */
#include "timers.h"
/* Import external IRQ handlers from IRQ.c file                               */
extern __irq void T0_IRQHandler  (void);
extern void init_serial (void);
/****************************************************************************/
/* pinesSignalInit                                                          */
/****************************************************************************/
/* Esta funcion configura los pines   P4.24 y P4.25 como salida             */
/****************************************************************************/
void pinesSignalInit(void)
{
  PINSEL9  = 0x00000000;
  PINMODE9 = 0x000000000;
  FIO4DIR3 = 0x03;
}
/****************************************************************************/
/* timer0Init                                                               */
/****************************************************************************/
/* Esta funcion configura el timer 0 con los parámetros que no cambian durante*/
/* la aplicacion                                                            */
/****************************************************************************/
void timer0Init(void)
{
  T0PR  = 0x00;   /* activa el preescalador a cero */
  //controlador de interrupciones
  VICVectAddr4  = (unsigned long)T0_IRQHandler;/* Set Interrupt Vector       */
  VICVectCntl4  = 15;                  /* Priority use it for Timer0 Interrupt */
  VICIntEnable  = (1  << 4);           /* Enable Timer0 Interrupt              */
}
/****************************************************************************/
/* hardwareInit                                                             */
/****************************************************************************/
/* Esta funcion se llama al comienzo del programa para inicializar el Hardware*/
/****************************************************************************/
void hardwareInit(void)
{
  pinesSignalInit();   // Configura los pines del circuito
  timer0Init();        // Inicializa el timer 0
  init_serial();
 }
```

- **Fichero timers.h:**

puertoSerie1.c | HAL.c | IRQ.c | misTipos.h | timers.h

```
/****************************************************************************/
/* timers.h: definiciones para la practica de timers  y algunas funciones */
/* Sistemas Empotrados Universidad de Cordoba                                 */
/****************************************************************************/
#define SIGNAL0_PIN_HIGH  FIO4SET3 = 0x01;      // Pin señal 0 a alto P4.24
#define SIGNAL0_PIN_LOW   FIO4CLR3 = 0x01;      // Pin señal 0 a bajo P4.24
#define SIGNAL1_PIN_HIGH  FIO4SET3 = 0x02;      // Pin señal 1 a alto P4.25
#define SIGNAL1_PIN_LOW   FIO4CLR3 = 0x02;      // Pin señal 1 a bajo P4.25

```

- **Fichero IRQ.c:**

puertoSerie1.c | HAL.c | IRQ.c | misTipos.h | timers.h

```
1  /*****************************************************************/
2  /* IRQ.C: manejadores de interrupcion                            */
3  /* Sistemas Empotrados Universidad de Cordoba                    */
4  /*****************************************************************/
5  #include <LPC23xx.H>                       /* LPC23xx definitions */
6  #include "timers.h"
7  #include "mistipos.h"
8  extern UINT32 timersw0, timersw0InitialCount;;
9  extern UINT32 timersw1, timersw1InitialCount;;
10 extern BOOL signal0High;
11 extern BOOL signal1High;
12 /* Timer0 IRQ                                                    */
13 __irq void T0_IRQHandler (void) {
14   timersw0--;
15   timersw1--;
16   if (timersw0==0)  // comprueba si el timer sw ha finalizado
17     {
18       timersw0=timersw0InitialCount;
19       if (signal0High==TRUE)
20       {
21         SIGNAL0_PIN_LOW;
22         signal0High=FALSE;
23       }
24       else
25       {
26         SIGNAL0_PIN_HIGH;
27         signal0High=TRUE;
28       }
29     }
30   if (timersw1==0) // comprueba si el timer sw ha finalizado
31   {
32     timersw1=timersw1InitialCount;
33     if (signal1High==TRUE)
34     {
35       SIGNAL1_PIN_LOW;
36       signal1High=FALSE;
37       }
38     else
39     {
40       SIGNAL1_PIN_HIGH;
41       signal1High=TRUE;
42     }
43   }
44   T0IR        = 1;                  /* Clear interrupt flag  */
45   VICVectAddr = 0;                  /* Acknowledge Interrupt */
46 }
47
```

- **Fichero misTipos.h:**

puertoSerie1.c | HAL.c | IRQ.c | misTipos.h | timers.h

```
1  /*****************************************************************************/
2  /* misTipos.h: definiciones de tipos de variables para practicas con LPC2378  */
3  /* Sistemas Empotrados Universidad de Cordoba                                 */
4  /*****************************************************************************/
5  /*                                                                           */
6  /*****************************************************************************/
7  #ifndef __misTipos_H
8  #define __misTipos_H
9
10 /* Byte */
11 #define UINT8 unsigned char
12 #define INT8 char
13
14 /*16 bits */
15 #define UINT16 unsigned short int
16 #define INT16  short int
17
18 /*32 bits WORD para el LPC2378 */
19 #define UINT32 unsigned int
20 #define INT32  int
21
22 /* Tipos para control */
23 #define STATUS UINT32
24
25 /* Boolenas */
26 #define BOOL   UINT32
27 #define FALSE (unsigned int)0x000000000
28 #define TRUE  (unsigned int)0x000000001
29
30 /* flags */
31 #define FLAG  BOOL
32 #define ESC 0x1B
33 #define OK 0x01
34 #endif
35
```