

SISTEMAS EMPOTRADOS

3º Grado en Ingeniería Informática

PRÁCTICA 6: Introducción al puerto serie

6.1.– Introducción

Independientemente de que la aplicación final en un sistema empotrado no tenga interfaz de usuario, en otras fases del desarrollo es necesario tener una vía de comunicación con el usuario para depurar la aplicación, para depurar el *hardware*, para mantenimiento o diagnóstico del sistema, o para informar de sucesos.

Si el sistema no dispone de un interfaz visual, hay que dotar al *hardware* de un interfaz económico y que no consuma demasiado tiempo de CPU. La comunicación serie a través de una UART (*Universal Asynchronous Receiver/Transmitter*) integrada en el sistema o una UART virtual utilizando un puerto de entrada/salida digital, es la más económica y utilizada. Mediante esta comunicación serie tenemos una manera sencilla para mostrar o aceptar información del usuario si al otro lado de la línea serie conectamos un terminal.

6.2.– Objetivos

El objetivo de esta práctica será realizar una pequeña aplicación 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.

En la plataforma *Moodle* está disponible un ejemplo de terminal gratuito **YAT** (*Yet Another Terminal*) aunque hay otros disponibles en la red. También está disponible en dicha plataforma un *software* para el adaptador USB–RS232 por si nuestro ordenador no dispone de puerto serie.

6.3.– 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.
- Cable serie RS232 o bien un adaptador USB puerto serie.

6.4.- Desarrollo de la práctica:

Se creará un programa que realice el producto de dos números BCD de un dígito. El resultado será un número de dos dígitos BCD. El interfaz de usuario que se mostrará en la simulación del *Keil µVision 5* tendrá un aspecto similar a este:

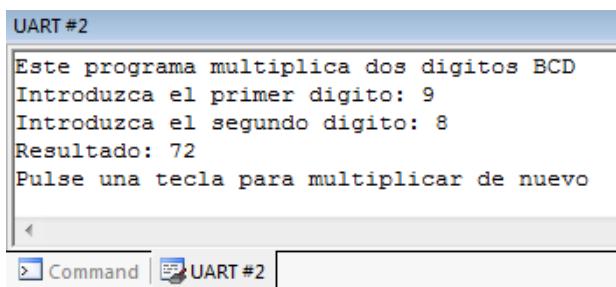


Figura 6-1: Interfaz de usuario a obtener.

Se aceptarán valores de un dígito y se descartarán las teclas distintas de un valor BCD. Hay que tener en cuenta que los códigos ASCII de los números del 0 al 9 están comprendidos entre 0x30 y 0x3A, por lo que al valor introducido por el teclado habrá que restarle 0x30.

6.5.- Indicaciones:

Necesitaremos los ficheros de retarget.c y serial.c de alguna aplicación conocida (por ejemplo, *Blinky* de MCB2300).

En HAL.c solamente es necesario introducir la función `init_serial` que a su vez llama al serial.c, donde se definen las características del puerto serie del microcontrolador LPC2378 de nuestra placa.

```
Practica_6_Puerto_Serie.c | HAL.c | Serial.c | multibench | Segmentos
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 /* hardwareInit: Esta función se llama al comienzo del
7 /*           programa para inicializar el Hardware */
8 //*****
9 extern int init_serial (void);
10 void hardwareInit(void)
11 {
12     init_serial();
13 }
14
```

Figura 6-2: Fichero de la capa de abstracción del hardware (HAL).

El manejador de proyectos tendrá un entorno como éste:

```

3  ****
4  /* Sistemas Emplorados. 3º de Graduado en Ingeniería Informática */
5  /* Universidad de Córdoba
6  ****
7  /* Apellidos y nombre del alumno/s:
8  ****
9
10 #include <stdio.h>
11 #include <LPC23xx.H>                                /* LPC23xx definitions */
12 #include "misTipos.h"
13
14 extern void hardwareInit (void);
15 extern int getkey (void);
16
17 int main (void)
18 {
19     UINT32 charTmp;
20     UINT32 primerDigito, segundoDigito, resultado;
21     // inicialización hardware
22     hardwareInit();
23     while (1)
24     {
25         printf("Este programa multiplica dos digitos BCD \n");
26         // primer dígito. Sólo son válidos los caracteres de 0 a 9
27         printf("Introduzca el primer digito: ");
28         do
29         {
30             charTmp=getkey();
31             if ((charTmp<0x3A) & (charTmp>=0x30)) printf("%c", (char) charTmp); //Eco
32         } while (!((charTmp<0x3A) & (charTmp>=0x30)));
33         primerDigito=(charTmp-0x30);
34         // segundo dígito. Sólo son válidos los caracteres de 0 a 9
35         printf("\nIntroduzca el segundo digito: ");
36         do
37         {
38             charTmp=getkey();
39             if ((charTmp<0x3A) & (charTmp>=0x30)) printf("%c", (char) charTmp); //Eco
40         } while (!((charTmp<0x3A) & (charTmp>=0x30)));
41         segundoDigito=(charTmp-0x30);
42         //resultado
43         resultado=primerDigito*segundoDigito;
44         printf("\nResultado: %d ", resultado);
45         printf("\nPulse una tecla para multiplicar de nuevo");
46         (void)getkey();
47         printf("\n \n");
48     }
49 }

```

Figura 6-3: Programa principal de la aplicación.

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

```

36 void init_serial (void) {                                /* Initialize Serial Interface */
37     #ifdef UART0
38         PINSEL0 |= 0x00000050;                            /* Enable TxDO and RxDO */
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 }

```

Figura 6-4: Configuración del puerto serie.

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.

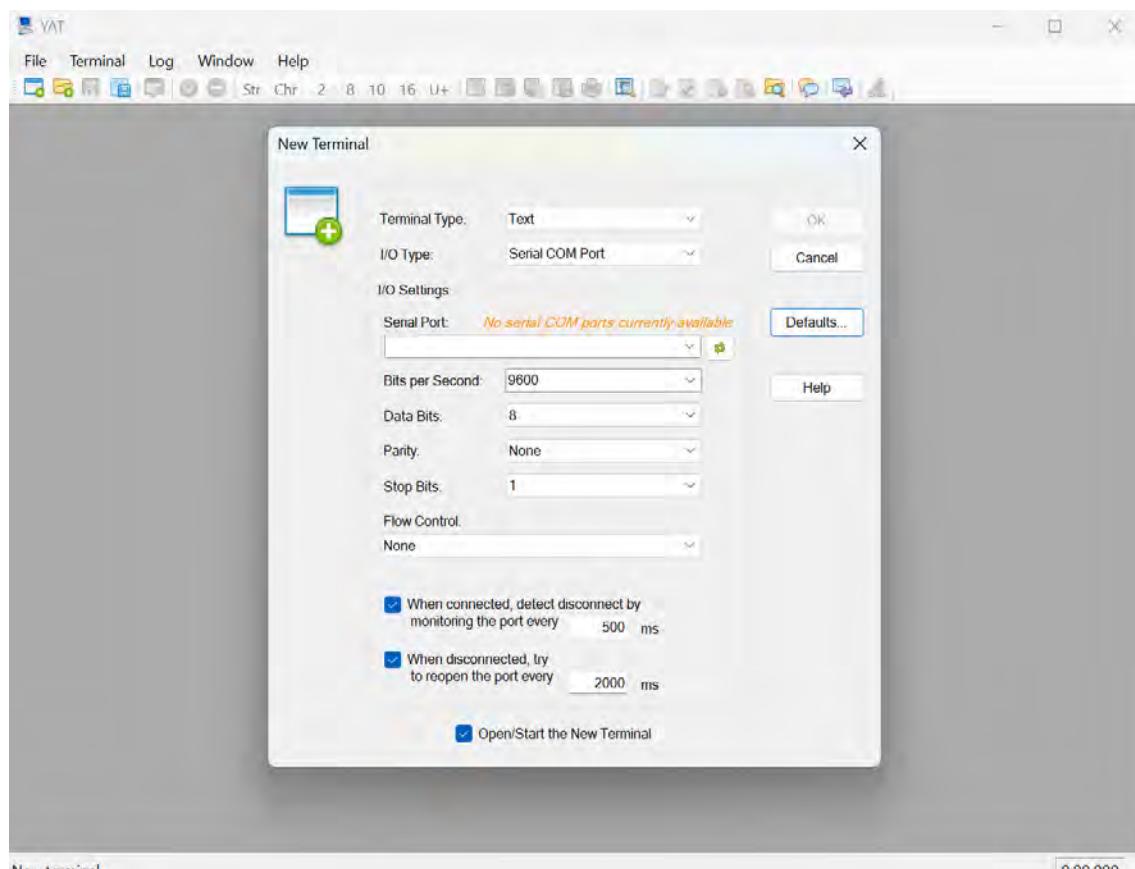


Figura 6-5: Configuración del hyperterminal YAT.

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:

$$\text{Divisor} = \text{Pclk} / (16 \times \text{BAUD})$$

En el caso de 12 MHz

$$\text{Divisor} = 12.000.000 / (16 \times 9600) = (\text{approx}) \ 78 \text{ o } 0x4E$$

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.

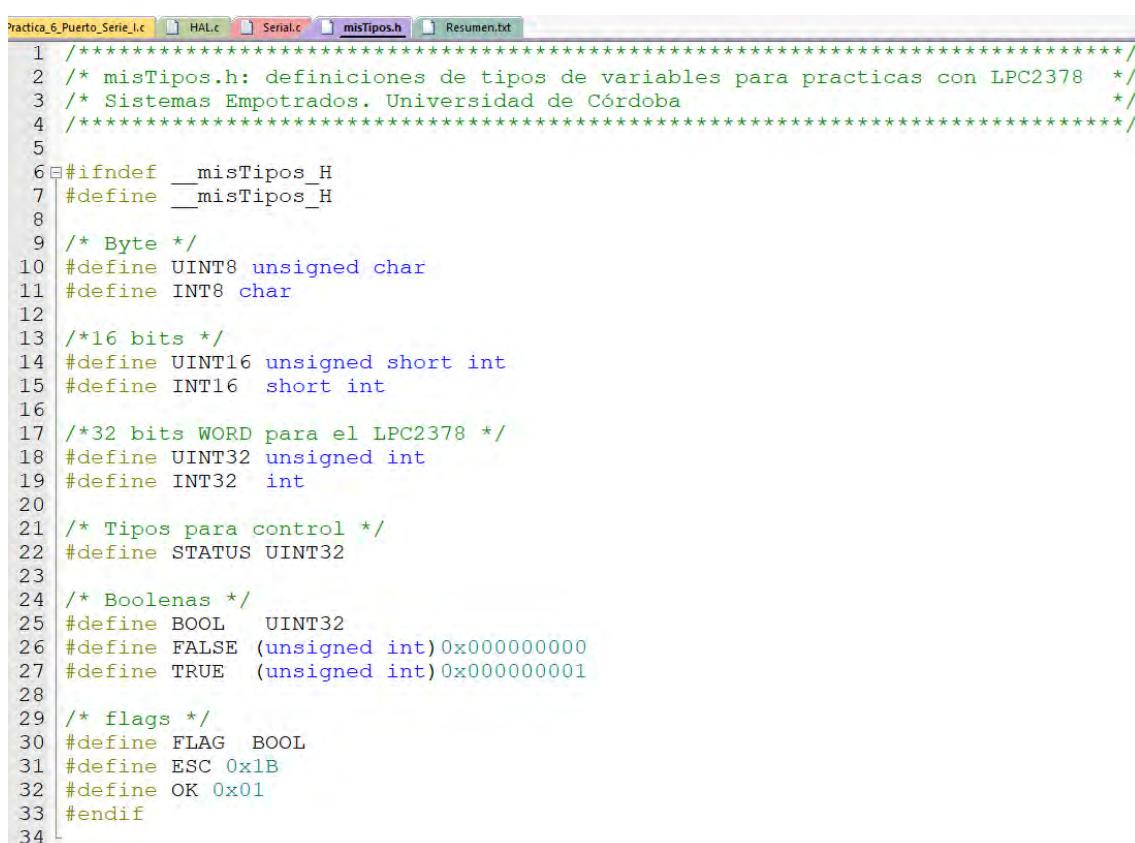
La parte del fichero serial.c de *Keil* para implementar los comandos printf y getkey queda definida en la figura siguiente:

```

51 /* Implementation of putchar (also used by printf function to output data) */
52 int sendchar (int ch) { /* Write character to Serial Port */
53
54     while (!(UxLSR & 0x20));
55
56     return (UxTHR = ch);
57 }
58
59
60 int getkey (void) { /* Read character from Serial Port */
61
62     while (!(UxLSR & 0x01));
63
64     return (UxRBR);
65 }
```

Figura 6-6: Fichero de definición de tipos de variables.

El fichero de definición de tipos de variables quedará igual que en las prácticas anteriores.



```

1 /*****
2  * misTipos.h: definiciones de tipos de variables para practicas con LPC2378 */
3 /* Sistemas Empotrados. Universidad de Córdoba */
4 *****/
5
6 #ifndef __misTipos_H
7 #define __misTipos_H
8
9 /* Byte */
10#define UINT8 unsigned char
11#define INT8 char
12
13/*16 bits */
14#define UINT16 unsigned short int
15#define INT16 short int
16
17/*32 bits WORD para el LPC2378 */
18#define UINT32 unsigned int
19#define INT32 int
20
21/* Tipos para control */
22#define STATUS UINT32
23
24/* Boolenas */
25#define BOOL UINT32
26#define FALSE (unsigned int)0x00000000
27#define TRUE (unsigned int)0x00000001
28
29/* flags */
30#define FLAG BOOL
31#define ESC 0x1B
32#define OK 0x01
33#endif
34
```

Figura 6-7: Fichero de definición de tipos de variables.

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 terminal YAT. Otra opción será utilizar un adaptador USB–UART.

El único periférico que interesa visualizar en esta práctica es la UART utilizada que, según la configuración de serial.c de *Keil*, es la UART1.

En cambio, en el depurador, para ver la salida e interactuar con el puerto serie, debemos seleccionar en el menú *View → Serial Windows → UART #2*.

El resultado en el depurador (*Debugger*) del software *Keil* sería algo como el de la siguiente imagen:

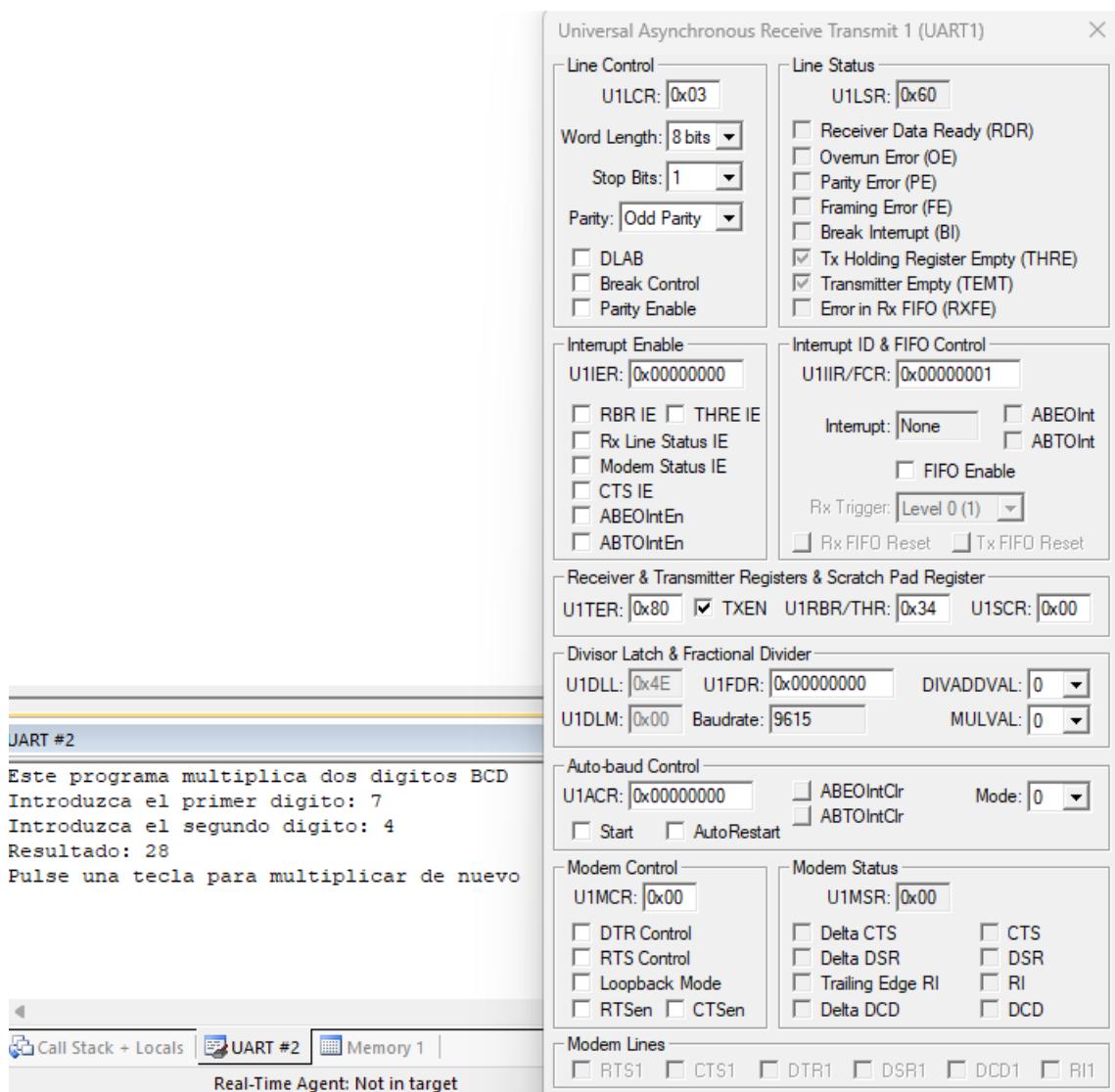


Figura 6-8: Visualización del puerto serie y de la simulación.

Finalmente hay que visualizar lo mismo en el *hyperterminal YAT*.