Diseño Basado en Microprocesadores
Práctica 11
UARTs del LPC4088

Índice

1. Objetivos

En esta práctica usaremos la UART0 del microcontrolador LPC4088 aplicándola en programas que permitan el intercambio de datos con un PC.

2. Introducción

El microcontrolador LPC40xx cuenta con cinco UARTs que le permiten intercambiar datos en serie con otros dispositivos. Las UARTs del LPC40xx se nombran como UART0, UART1, UART2, UART3 y UART4.

Tras el reset, las UARTs 0 y 1 están operativas mientras que las UARTs 2, 3 y 4 están en modo de bajo consumo. Por tanto, si queremos usar alguna de las UARTs 2, 3 o 4 será necesario habilitarlas activando el correspondiente bit del registro PCONP del *System Control*.

También será necesario configurar adecuadamente las funciones de los pines que deseemos que realicen funciones asociadas a la UART o UARTs que queramos usar. Además, debemos asegurarnos que el pin o pines que realicen funciones de recepción serie tengan desactivados las resistencias de pull-down internas.

Las UARTs usan como señal primaria de reloj el reloj de periféricos PCLK. A partir de éste obtienen por división de frecuencia la tasa de comunicación en baudios y las demás señales de sincronización internas.

Las UARTs 0, 2, 3 y 4 del LPC40xx no cuentan con líneas de control de modem, mientras que la UART1 sí posee la capacidad de manejar las señales de control de modem habituales (DTR, DSR, RTS, CTS, DCD y RI). Salvo por la carencia de líneas de control de modem en las UARTs 0, 2, 3 y 4, todas ellas son compatibles con el estándar RS-232C que también usan los puertos serie de los ordenadores PC y muchos otros dispositivos.

El uso de una UART del microcontrolador resulta muy útil en sistemas que carecen de teclado y pantalla ya que, mediante la conexión a un puerto serie de un PC y ejecutando un programa de terminal en el mismo, podemos usar la pantalla y el teclado del PC para interactuar con el microcontrolador.

En la presente práctica desarrollaremos un conjunto de funciones que nos permitirán un uso básico de las UARTs del microcontrolador. En la tarjeta microcontroladora LPC4088 Developer's Kit, las señales de la UART0 están conectadas a un circuito integrado convertidor RS232-USB modelo FT232. Este circuito usa el mismo conector USB que empleamos habitualmente para alimentar la tarjeta, así que la comunicación entre ésta y el PC se realizará a través del mismo cable USB que suministra la alimentación. El sistema operativo Windows 8 que está instalado en los ordenadores del laboratorio dispone de drivers para el FT232, de manera que reconoce automáticamente la tarjeta microcontroladora como un puerto serie. Para averiguar el nombre que Windows ha asignado al puerto debe abrirse el administrador de dispositivos y desplegar el apartado "Puertos (COM y LPT)". El nombre del puerto será COMx, siendo x un número.

3. Ejercicios

3.1. Ejercicio 1

Escribe un conjunto de funciones para manejar las UARTs del LPC40xx que incluya las siguientes funciones:

- ```
  void uart_inicializar(LPC_UART_TypeDef *uart_regs,
                        uint32_t  baudrate,
                        uint32_t  bits_datos,
                        paridad_t  paridad,
                        uint32_t  bits_stop,
                        uint32_t  *baudrate_obtenido);
  ```

La función `uart_inicializar` servirá para inicializar una UART usando la configuración dada por los argumentos:

uart_regs: puntero al bloque de registros de la UART.

baudrate: velocidad de comunicación en baudios.

bits_datos: número de bits de datos entre 5 y 8.

paridad: indicará si se usa paridad o no y, si se usa seleccionará par o impar. Puede tomar los valores NINGUNA, IMPAR o PAR.

bits_stop: número de bits de stop, que podrá ser 1 ó 2.

baudrate_obtenido: si no es NULL, la función usa este puntero para devolver el baudrate real que se ha podido obtener, el cual puede diferir del solicitado. Si es NULL, la función no usa este argumento.

La función debe activar los buffers FIFO de transmisión y recepción de la UART. La función usará la macro ASSERT para comprobar que los argumentos son correctos.

- ```
  void uart_transmitir_dato(LPC_UART_TypeDef *uart_regs, uint8_t dato)
  ```

La función uart_transmitir_dato enviará a través de la UART indicada por el primer argumento el dato que recibe como segundo argumento. La función debe esperar a que el registro *Transmit Holding Register* esté libre. Esto se hará consultando el bit *Transmitter Holding Register Empty* (THRE) del registro *Line Status Register* (LSR). Una vez que este bit esté a uno, escribiremos el argumento dato en el registro THR. La función usará la macro ASSERT para comprobar que los argumentos son correctos.

- **bool_t** uart_hay_dato_disponible(**LPC_UART_TypeDef** *uart_regs)

La función uart_hay_dato_disponible comprobará si hay datos en el buffer de recep- ción de la UART indicada. Retornará FALSE si no hay ningún dato en el buffer y TRUE si hay uno o más datos en el buffer. Puede conocerse el estado del buffer de recepción de la UART consultando el bit *Receiver Data Ready* (RDR) del registro *Line Status Register* (LSR). La función usará la macro ASSERT para comprobar que el argumento es correcto.

- **uint8_t** uart_leer_dato (**LPC_UART_TypeDef** *uart_regs)

La función uart_leer_dato obtendrá un dato del buffer de recepción de la UART indicada, leyendo para ello el registro *Receiver Buffer Register* (RBR), y retornará dicho dato. Esta función solo debe ser llamada si se sabe que hay datos disponibles en el buffer de recepción. La función usará la macro ASSERT para comprobar que el argumento es correcto.

- **uint8_t** uart_esperar_recibir_dato (**LPC_UART_TypeDef** *uart_regs)

La función uart_esperar_recibir_dato esperará la llegada de un dato a través de la UART indicada y retornará dicho dato. La función esperará primero a que haya datos disponibles en el buffer de recepción de la UART consultando el bit *Receiver Data Ready* (RDR) del registro *Line Status Register* (LSR). Cuando esto ocurra, se retornará el dato obtenido leyendo el registro *Receiver Buffer Register* (RBR). La función usará la macro ASSERT para comprobar que el argumento es correcto.

Usa las funciones anteriores en un programa para el microcontrolador que use la UART0 para intercambiar datos con un programa de terminal (tal como Coolterm) que se ejecutará en el PC. El programa debe actuar de la siguiente forma:

- Si el joystick se mueve hacia arriba se enviará al PC el carácter 'A'.
- Si el joystick se mueve hacia abajo se enviará al PC el carácter 'B'.
- Si el joystick se mueve hacia la izquierda se enviará al PC el carácter 'I'.
- Si el joystick se mueve hacia la derecha se enviará al PC el carácter 'D'.
- Si se recibe el carácter '1' desde el PC, el LED de P1[5] debe cambiar de estado.
- Si se recibe el carácter '2' desde el PC, el LED de P0[14] debe cambiar de estado.
- Si se recibe el carácter '3' desde el PC, el LED de P0[13] debe cambiar de estado.
- Si se recibe el carácter '4' desde el PC, el LED de P1[18] debe cambiar de estado.

Inicialmente, los cuatro LEDs deben estar apagados. Los LEDs se encienden cuando el correspondiente pin está a nivel bajo.

Deben usarse los pines P0[2] y P0[3] para realizar las funciones TXD y RXD de la UART0.

Una vez que hayas escrito el programa:

1. Instala en el PC un programa de terminal, por ejemplo Coolterm http://freeware.the-meiers.org.
2. Conecta el puerto USB conectado a la UART0 del microcontrolador a un puerto USB de PC.

3. Selecciona los siguientes parámetros de comunicación tanto en el programa en el microcontrolador como en el programa de terminal: 9600 baudios, 8 bits de datos, sin paridad, 1 bit de stop.
4. En el programa de terminal en el PC debes elegir el puerto serie (COM) al en el que ha quedado asignado el adaptador USB-RS232.
5. En el programa de terminal no debe usarse protocolo hardware para que ignore las líneas de control de modem DTR, DSR, RTS, CTS, DCD y RI.
6. También debes activar el eco local de los caracteres tecleados en el pro- grama de terminal.

3.2. Ejercicio 2

Escribe funciones que faciliten la transmisión y recepción de cadenas de caracteres a través de las UARTs.

- **void** `uart_transmitir_cadena(LPC_UART_TypeDef *uart_regs, const char *cadena)`

La función uart_transmitir_cadena debe enviar a través de la UART indicada por el primer argumento cada uno de los caracteres de la cadena a la que apunta el segundo argumento hasta que se encuentre el terminador nulo de la cadena. El terminador nulo no se enviará. Para enviar cada uno de los caracteres se usará la función uart_transmitir_dato.

- **void** `uart_recibir_cadena(LPC_UART_TypeDef *uart_regs, char *buffer)`

La función `uart_recibir_cadena` debe quedar a la espera de recibir una cadena de caracteres a través de la UART indicada por el primer argumento. Para esperar la llegada de cada uno de los caracteres se usará la función `uart_esperar_recibir_dato`. A medida que se reciban caracteres se irán almacenando a partir de `buffer`.

La recepción de la cadena finalizará cuando en el terminal se pulse la tecla ENTER. En Windows, cuando se pulsa la tecla ENTER, se generan dos caracteres de control: un carácter CR (*carriage return* o retorno de carro), con código ASCII 0x0D, y un carácter LF (*line feed* o alimentación de línea), con código ASCII 0x0A. En cambio, en los sistemas tipo Unix (como Linux), cuando se pulsa ENTER, sólo se genera un carácter LF.

La función `uart_recibir_cadena` interpretará la llegada de un carácter LF como señal de que se ha pulsado la tecla ENTER y por tanto de que debe terminal el proceso de recepción de la cadena. Cuando se reciba el carácter LF, se almacenará un terminador nulo al final de la cadena que ha quedado en buffer y se retornará de la función. El carácter de control LF no se almacenará en el buffer.

Usar LF para detectar la pulsación de ENTER asegurará que la función trabaje bien tanto en Windows como en Linux. Pero como en Windows antes de cada LF llegará un CR, la función deberá ignorar cualquier carácter CR que le llegue, es decir, deberá desecharlo sin interpretarlo ni almacenarlo en la cadena. De hecho, lo más conveniente sería ignorar todos los caracteres de control (códigos ASCII menores de 0x20), excepto el LF que, como se ha dicho, se interpretará como señal de fin del proceso de recepción.

Usa las funciones anteriores en un programa que envíe al terminal la temperatura captada por el termistor o la tensión en el cursor del potenciómetro de la tarjeta según desde el terminal se reciba la palabra temperatura o la palabra tension, respectivamente. Para determinar cuál es la orden recibida puede ser útil la función strcmp de la biblioteca estándar, cuyo prototipo está en string.h.