PROGRAMACIÓN DEL PUERTO SERIAL DEL PC

El puerto serial del PC es uno de los recursos más comunes para la conexión de periféricos, como pueden ser dispositivos de puntero (`mouse`) o de comunicación (`módem`, cables de conexión entre PCs, etc.). Está compuesto por un integrado UART del tipo 16550 en modelos actuales, mientras que en la "antigüedad" se hablaba del **8250**.

La siguiente tabla nos muestra el ejemplo de 4 puertos COM típicos. Lo mas normal es que tanto el **COM1** como el **COM2** estén ahora integrados en la placa base, o puestos en una tarjeta controladora. Los otros 2 se suelen configurar con el módem (interno)

Com	Dirección Base	IRQ
Com1	3F8	4
Com2	2F8	3
Com3	3E8	4
Com4	2E8	3

Tenga en cuenta que el **Com1** y el **Com3** comparten la misma **IRQ**. Lo mismo pasa con **Com2** y **Com4**.

Configuración del puerto serial

Cada uno de los puertos COM tiene 7 registros a comentar:

Registro base+0: tiene 3 funciones

- **Transmitter Holding Register** (THR): Su función es la de transmitir un dato (palabra) por el puerto.
- **Reciver Data Register** (RDR): Su función es la de recibir un dato (palabra) del puerto.
- **Baud Rate Divisor Low** (BRDL): Velocidad del puerto, parte Baja

Registro base+1: tiene 2 funciones

- **Baud Rate Divisor High** (BRDH): Velocidad del puerto, parte Alta
- **Interrupt Enable Register** (IER): Activa o desactiva las interrupciones para el puerto COM

Registro base+2:

- **Interrupt ID Register** (IIR): Controla la prioridad de las interrupciones

Registro base+3:

- **Line Control Register (LCR)**: Controla los parámetros de configuración del puerto serial (velocidad...)

Registro base+4:

- **Modem Control Register** (MCR): Activa las señales del módem

Registro base+5:

- **Line Status Register** (LSR): Muestra el estado del puerto serial (errores, etc)

Registro base+6:

- **Modem Status Register** (MSR): Muestra el estado del módem

Algunos de los registros antes mencionados no los usaremos. Tan soló son importantes LCR, BRDH, BRDL, LSR, THR y RDR.

Debemos configurar correctamente el puerto serie **ANTES** de trabajar con el.

Primero configuramos el LCR, pero poniendo el bit DLAB a 1. Configuramos luego el BRDL, seguidamente el BRDH y finalmente volvemos a configurar el LCR con los mismos bits que antes, pero esta vez el bit DLAB a 0.

Explicación de los Registros

LCR : Con este registro configuramos los parámetros de la comunicación serial. **DLAB** es el bit de más peso y **Num(1)** el de menos peso.

DLAB	Break	Stick	Tipo	Paridad	Stop	Num(2)	Num(1)

Num (1) y **Num(2)**: Indica el numero de bits de datos en cada palabra que vamos a utilizar. Se configura con la siguiente tabla:

Num2	Num1	Nº Bits
0	0	5
0	1	6
1	0	7
1	1	8

Stop: Indica el número de bits de parada que enviara (o esperara) el puerto. Lo normal es poner un 0 en este bit para conseguir un bit de parada. Pero si ponemos un 1, el puerto usara 1,5 bits de parada si el número de bits de la palabra (configurada en Num) es de 5. En caso de que sean mas de 5, se usaran 2 bits de parada. Puede que en determinados momentos nos sea de utilidad, pero es posible que envié 10 bits en lugar de 9, cosa que implica más tiempo en el envío, por lo que se recomienda un 0 (un bit de parada).

Paridad: Indica si hay paridad en la comunicación serial. Con un 0 le diremos que no queremos paridad, y con un 1 que si. La paridad es una manera de detectar errores, pero debemos programar nuestro programa para que lo detecte.

Tipo: Este bit nos indicara el tipo de Paridad que vamos a usar. Con un 0 le indicaremos que vamos a mirar la paridad de modo impar, mientras que con un 1 miraremos la paridad de modo par.

Stick: Indica el nivel que usaremos para la paridad: si ponemos un 0, contaremos el numero de '1' para la paridad. Y si ponemos un '1' contaremos el numero de '0' para la paridad.

Break: Fuerza un corte de la comunicación. Si lo dejamos a 0 no pasara nada, pero si lo ponemos a 1 cortamos la comunicación y forzamos la salida a '0'

DLAB: Bit interno. Sin uso, pero no es decisivo. Lo que hacemos es configurar una vez el puerto poniendo a 1 este bit, configurar la velocidad (los 2 registros) y luego volver a configurar el puerto poniendo un 0.

Ejemplo:

Si queremos una configuración de 8 bits, un bit de parada, sin paridad para un COM2 lo que tendremos que hacer es enviarle a la dirección base más 3 (este caso como la dirección base es 2F8, nos dirigimos a 2F8+3=2FB) le enviaremos un 10000011 (0x83, con el DLAB a 1) configuramos la velocidad y volvemos a configurar el puerto pero con DLAB a 0 (00000011, 0x03).

BRDH y BRDL (Baud Rate Divisor, BRD)

Es un registro de 16 bits compuesto por BRDH y BRDL que define la velocidad de la comunicación del puerto de comunicación. El valor a cargarle lo obtendremos al aplicar la siguiente formula:

$$RD = \frac{Clock}{16 * Númerodebaudios}$$

Debemos tener en cuenta lo siguiente: que para una computadora XT el clock (velocidad de reloj) era de 1,78 MHz, y en una computadora AT es de 1,84 MHz.

Para un AT, si queremos 1200 baudios debemos hace la siguiente operación:

$$\frac{1{,}84M}{16*1200} = 95,83$$

Ese resultado lo redondeamos, 96 (0x60 en hexadecimal). Ese es el resultado, 00 96, lo que hay que enviarle. En BRDL el 96 y el 00 en BRDH. Juntando con el ejemplo de antes, una configuración de COM2 de 1200 baudios, sin paridad, 8 bits de datos y 1 bit de parada:

```
outportb (0x2FB,0x83);
outportb (0x2F8,0x60);
outportb (0x2F9,0x00);
outportb (0x2FB,0x03);
```

Enviar datos por el puerto serie

Para enviar una palabra por el puerto COM debemos enviar el dato a THR (a la dirección de hardware base del puerto). Si hablamos del COM2, deberemos hacer:

```
outportb (0x2F8,dato);
```

El dato sera enviado por el puerto serial, **SIEMPRE QUE ESTE LIBRE Y NO OCUPADO**. Para sabe si está ocupado debemos mirarlo en el LSR (*Line Status Register*, 2FD) de la siguiente manera

Si ((LSR&0x20) es igual a 0x20) entonces podremos enviar el dato. Si hablamos del COM2, la pregunta en lenguaje C seria:

```
if ((outportb (0x2FD)&0x20)== 0x20)
```

Poner un getch, un scanf o otro tipo de función de consola típica (espera que el usuario introduzca un dato) puede ser muy perjudicial para el programa, porque si ese momento alguien envía un dato, no se va a leer! va a llegar, pero si durante el tiempo que el usuario esta dudando que poner llega otro dato por el puerto serial la información anterior seria reemplazada por el nuevo dato, y perderíamos parte de la información!!!

Leer datos del puerto serie

Para leer una palabra del puerto COM debemos leer del registro RDR (de la dirección base del puerto). Si hablamos del COM2 deberemos hacer:

```
inport (0x2F8);
```

La lectura deberemos guardarla en algún sitio (una variable tipo char seria lo mas indicado).

Pero como en el caso de la escritura en el puerto, también nos interesaría que el puerto nos avisara cuando el puerto tiene un dato nuevo. Deberemos saberlo de la siguiente manera:

Si ((LSR&0x01) es igual a 0x01) entonces podremos recibir el dato. Si hablamos del COM2, la pregunta en lenguaje C seria:

```
if ((outportb (0x2FD)&0x01)== 0x01)
```

Un ejemplo de lo visto hasta ahora: Un programa que configura el puerto serial COM2, envía todo el rato el dato 'A' por el puerto, y en caso de que reciba algo lo muestra por pantalla.

```
// By Stealth
// Programa de muestra del funcionamiento de Envío/Recibo del puerto serial
// Code start

#include <stdio.h>
#include <dos.h>
```

```
void main (void)
{
    outportb (0x2FB,0x83);
    outportb (0x2F8,0x60);
    outportb (0x2F9,0x00);
    outportb (0x2FB,0x03);

    //Puerto configurado para 1200 baudios, un bit de parada, 8 bits de datos, sin paridad
    for(;;)
    {
        if ((outportb (0x2FD)&0x01)== 0x01) //en caso de recibir
        {
            printf ("%c",inportb(0x2F8)); // Lo mostramos por pantalla
        }
        if ((outportb (0x2FD)&0x20)== 0x20) //en caso de poder enviar
        {
            outportb (0x2F8,'A'); // Lo mandamos por el puerto
        }
    }
}

//Code ends
```