PUERTO PARALELO

Desde el punto de vista del software, el puerto paralelo son tres registros de 8 bits cada uno, ocupando tres direcciones de I/O consecutivas de la arquitectura x86.
Desde el punto de vista hardware, el puerto es un conector hembra DB25 con doce salidas latcheadas (que tienen memoria/buffer intermedio) y cinco entradas, con 8 líneas de masa.
La función normal es transferir datos a una impresora a través de las 8 líneas de datos, usando las señales restantes como control de flujo.
El documento lo separé en tres secciones principales:

Descripción general

Tabla de puertos paralelo

El puerto paralelo se identifica por su dirección de I/O base y se identifica ante sistemas DOS por el número LPT. Cuando arranca la máquina, la BIOS chequea direcciones específicas de I/O en busca de puertos paralelos y construye una tabla de las direcciones halladas en la posición de memoria 40h:8h (o 0h:0408h).
Esta tabla contiene hasta tres palabras de 16 bits. Cada palabra es la dirección de I/O base del puerto paralelo. La primera palabra corresponde a LPT1, la segunda a LPT2 y la tercera a LPT3. Hay que agregar que en DOS tenemos el dispositivo PRN que es un alias a uno de los dispositivos LPT (generalmente es LPT1, pero se puede cambiar con la orden MODE)
Las direcciones estándar para los puertos paralelos son 03BCh,0378h y 0278h (chequeadas en este orden). Para saber como detectar léase la sección Detectando SPP

Acceso directo al puerto

El puerto, como se mencionó antes, consiste de tres registros de 8 bits ubicados en direcciones adyacentes del espacio de I/O de la PC. Los registros se definen relativos a la dirección de I/O base (variable IOBase) y son:
IOBase+0 : registro de datos
IOBase+1 : registro de estado
IOBase+2 : registro de control

Registro de datos (D)

Se puede leer y escribir. La forma de leer y escribir puertos con lenguajes de programación estándares se puede ver en la sección Acceso a los puertos
Escribiendo un dato al registro, causa que el mismo aparezca en los pines 2 a 9 del conector del puerto. Leyendo el registro, se lee el último dato escrito (NO lee el estado de los pines; para ello hay que usar un puerto bidireccional).

```
Nro.Bit 7 6 5 4 3 2 1 0  Descripción
        x . . . . . . .  D7 (pin 9), 1=Alto, 0=Bajo
        . x . . . . . .  D6 (pin 8), 1=Alto, 0=Bajo
        . . x . . . . .  D5 (pin 7), 1=Alto, 0=Bajo
        . . . x . . . .  D4 (pin 6), 1=Alto, 0=Bajo
        . . . . x . . .  D3 (pin 5), 1=Alto, 0=Bajo
        . . . . . x . .  D2 (pin 4), 1=Alto, 0=Bajo
        . . . . . . x .  D1 (pin 3), 1=Alto, 0=Bajo
        . . . . . . . x  D0 (pin 2), 1=Alto, 0=Bajo
```

Cuando se indica Alto o Bajo se refiere a la tensión de salida (~5V para el 1 físico y ~0V para el 0 físico, respectivamente). Esto es porque la lógica puede ser positiva (un 1 lógico equivale a Alto o 5V) o negada (un 0 lógico equivale a Bajo o 0V). Con respecto a esto debemos decir que para negar algo le anteponemos el carácter / (representando la barra que se coloca encima).
El estándar es que las salidas sean LS TTL (low schottky TTL), aunque las hay que son de tipo OC (colector abierto).
La corriente que pueden entregar (source) es de 2,6 mA máximo y pueden absorber (sink) un máximo de 24 mA.
En el puerto original de IBM hay condensadores de 2,2 nF a masa.
Las tensiones para el nivel bajo son entre 0 y 0,8V y el nivel alto entre 2,4V y 5V.

Registro de estado (S)

El registro de estado está en IOBase+1. Es de sólo lectura (las escrituras serán ignoradas).
La lectura da el estado de los cinco pines de entrada al momento de la lectura.
En la tabla siguiente los nombres de los pines se dejaron en inglés porque es como generalmente se identifican.

```
Nro.Bit 7 6 5 4 3 2 1 0  Descripción
        x . . . . . . .  S7 : Busy     (pin 11), 0=Alto, 1=Bajo
        . x . . . . . .  S6 : Ack      (pin 10), 1=Alto, 0=Bajo
        . . x . . . . .  S5 : No paper (pin 12), 1=Alto, 0=Bajo
        . . . x . . . .  S4 : Selected (pin 13), 1=Alto, 0=Bajo
        . . . . x . . .  S3 : Error    (pin 15), 1=Alto, 0=Bajo
        . . . . . x x x  Sin definir
```

La línea `Busy` tiene, generalmente, una resistencia de pull-up interna. El estándar es que sean entradas tipo LS TTL.

Registro de control (C)

El registro de control se encuentra en IOBase+2. Es de lectura/escritura.

```
Nro.Bit 7 6 5 4 3 2 1 0
        x x . . . . . .  Sin usar
        . . x . . . . .  C5 : Control puerto bidireccional
        . . . x . . . .  C4 : Interrupt control, 1=enable, 0=disable
        . . . . x . . .  C3 : Select     (pin 17), 1=Bajo, 0=Alto
        . . . . . x . .  C2 : Initialize (pin 16), 1=Alto, 0=Bajo
        . . . . . . x .  C1 : Auto Feed  (pin 14), 1=Bajo, 0=Alto
        . . . . . . . x  C0 : Strobe     (pin 01), 1=Bajo, 0=Alto
```

Los cuatro bits inferiores son salidas. La lectura devuelve lo último que se escribió a dichos bits. Son TTL a colector abierto con resistencias de pull-up de 4700 ohms, por lo que un dispositivo externo puede forzar el estado de los pines sin dañar el driver.
Esto permite utilizar estas cuatro líneas como entradas. Para ello, ponemos en alto las cuatro salidas (escribiendo 0000100b en IOBase+2) lo que hace que las salidas "floten". Ahora, un dispositivo externo puede forzar a bajo alguna de las salidas con lo que, leyendo el puerto, sabemos si esto sucedió o no.
Es posible realizar esta técnica en salidas totem-pole (como D0-D7) pero no recomendamos su uso porque habría que tener un conocimiento preciso de la corriente ya que se puede sobrecargar los transistores de salida, dañando el driver (especialmente en puertos integrados LSI).

Bit de puerto bidireccional (compatible PS/2)

El bit C5, está disponible sólo si se trata de un puerto bidireccional; en los puertos comunes actúa como los bits C6 y C7. Si C5=1, el buffer de los datos de salida se pone en alta impedancia, "desconectando" dicho buffer de los pines 2 a 9 del conector del puerto (D0 a D7). Si se escribe al registro de datos, se escribe al buffer pero no a la salida. Esto permite que al leer el puerto, se lea el estado de las entradas y no lo que hay en buffer.
En las computadoras IBM PS/2, para habilitar el puerto paralelo bidireccional, además de lo antes descrito, se debe poner a 1 el bit 7 del registro del puerto 102h (opciones de configuración).
En computadoras que no tengan puerto paralelo bidireccional compatible PS/2 hay que modificar uno o más bits de algún puerto específico correspondiente al chipset de la placa. A veces se habilita por setup o por jumper en la placa del puerto.

Bit de interrupción

En trabajos normales de impresión ni el BIOS ni el DOS hacen uso de la interrupción.
El hecho de poseer una línea de interrupción que está conectada directamente al PIC (Programmable Interrupt Controller), lo hace muy útil para experimentación en data-loggers por ejemplo. El bit de interrupción está conectado al control de un buffer de tres estados. Cuando C4=1, se activa el buffer y su entrada, S6, se conecta a la línea IRQ (en general es IRQ7 o IRQ5). La lectura del bit, nos devuelve el estado del mismo (es decir si el buffer está en alta impedancia o no).
Se producirá una interrupción, cuando haya un flanco descendente en el pin correspondiente a S6 (ver Descripción del conector físico). A continuación, se describen los pasos para poder utilizar interrupciones.

Interrupciones con el puerto paralelo

Primero que nada debemos habilitar el buffer que conecta la línea ACK con la línea IRQ. Esto lo hacemos poniendo a 1 el bit 4 del registro de control (IOBase+2). Luego debemos preparar una ISR (Interrupt Service Routine) que atienda la interrupción recordando enviar la señal EOI (20h) al registro de control del PIC (puerto 20h) al salir de la rutina. La interrupción software corresponde a la número 0Dh para IRQ5 y 0Fh para IRQ7.

Finalmente se habilita con 0 la interrupción IRQ5 (o IRQ7) escribiendo al bit 5 (o 7) del registro de interrupciones del PIC (puerto 21h).
Para desinstalar la ISR, deshabilitamos la IRQ5 (o IRQ7) escribiendo un 1 al bit 5 (o 7) del registro de interrupciones del PIC (puerto 21h). Luego hacemos que C4=0.

Descripción del conector físico

Como se mencionó anteriormente, la conexión del puerto paralelo al mundo externo se realiza por un conector hembra DB25. Viendo el conector al frente y con la parte que tiene más pines hacia arriba, se numera de menor a mayor de derecha a izquierda en ambas filas de pines (1 a 13 arriba y 14 a 25 abajo).
En las tablas vistas en las secciones correspondientes a cada registro se vio que cada bit tiene un nombre característico. Estos nombres son las funciones que cumplen los bits en una conexión con una impresora. Además de la lógica que posee cada señal (que es un aspecto físico del conector), tenemos que la impresora tiene su propia lógica para con los nombres de los pines. Por eso no hay que confundirse si el nombre de la señal establece una lógica contraria a la lógica real del puerto; son cosas distintas. Por ejemplo, la expresión /Ack, donde Ack viene de Reconocido o Aceptado, nos indica que una aceptación (Ack=Alto=Verdadero) la tenemos que reconocer porque la impresora nos envía un Bajo, que por la característica de S6 equivale a un 0 lógico.
En la columna de entrada/salida (I/O) se refiere al lado de la PC. En la columna Cent pin, indicamos el pin correspondiente al conector Centronics usado en las impresoras.
Algunos nombres no se corresponden con los de las tablas anteriores. Es a propósito para indicar los posibles nombres que puede tener una señal.

I/O	**DB25 pin**	**Cent pin**	**Bit reg.**	**Señal**	**Descripción**
O	1	1	/C0	/Strobe	A bajo por más de 0.5uS para indicar a la impresora que se enviarán datos
O	2	2	D0	Data0	Bit menos significativo de Data
O	3	3	D1	Data1	
O	4	4	D2	Data2	
O	5	5	D3	Data3	
O	6	6	D4	Data4	
O	7	7	D5	Data5	
O	8	8	D6	Data6	
O	9	9	D7	Data7	Bit más significativo de Data
I	10	10	S6	/Ack	Pulso bajo de ~5uS, indica que se recibieron datos en impresora.
I	11	11	/S7	Busy	En alto nos indica que la impresora está ocupada
I	12	12	S5	PaperEnd	En alto nos indica que no hay papel
I	13	13	S4	SelectIn	En alto para impresora seleccionada
O	14	14	/C1	/AutoFd	Si ponemos en bajo, el papel se mueve una línea luego de la impresión
I	15	32	S3	/Error	En bajo nos indica error (no hay papel, está fuera de línea, error no det.)
O	16	31	C2	/Init	Si enviamos un pulso en bajo > 50uS la impresora se resetea
O	17	36	C3	/Select	En bajo seleccionamos impresora (en gral. no se usa, ya que SelectIn se fija a alto)
-	18-25	19-30 y 33		Masa	Masa retorno de par trenzado
	18-25	16		Masa	Masa lógica
	18-25	17		Masa	Masa chasis

Velocidad

Un puerto paralelo ISA normal toma un ciclo-ISA para leer o escribir. Como en muchos sistemas, la velocidad del bus ronda los 1,3 Mhz, podemos decir que la lectura la podemos hacer cada 1 uS (idealmente, ya que siempre tenemos otras instrucciones software, etc, etc.). Algunos puertos soportan un modo “turbo” que elimina los 3 estados de espera de la CPU, con lo que la velocidad de lectura/escritura al puerto se duplica (2,7 MHz).

Programación

Obtención del puerto

Como ya se mencionó anteriormente las direcciones de I/O de los puertos paralelo se almacenan en una tabla ubicada en 40h:8h (0h:408h). Entonces, éste sería un método de obtener las direcciones. A continuación se muestra como obtener dichas direcciones en distintos lenguajes.

Ensamblador

```
;en SI tengo la dirección de memoria:
;LPT1 = 0408h
;LPT2 = 0408h + 2h = 040Ah
;LPT3 = 040Ah + 2h = 040Ch
mov        si,0408h              ;SI = 0408h
xor        ax,ax                 ;AX = 0
push  ds                    ;Mete DS en la pila
mov        ds,ax                 ;DS = AX = 0
mov        ax,word ptr [SI]      ;AX = [0h:SI]
pop        ds                    ;recupero DS de la pila
;ahora en AX tengo la dirección base
```

Pascal

```
{en la variable lptadr guardo la dirección del puerto.
en lpt debo guardar el número del puerto paralelo, antes de
ejecutar la siguiente instrucción.}

lptadr := memW($0040,$8+(lpt-1)*2);

{lo que hace esta línea es utilizar el array memW que
directamente accede a la memoria en el segmento y offset
especificados}
```

C

```
/* En portnum debo guardar el número de puerto (1,2,3).
En la variable lptadr, tipo unsigned, obtengo la dirección.*/

portnum--;
lptadr = peek(0x0040,0x0008+portnum*2);
```

QuickBasic

```
'en la variable lptnum debo guardar el número de puerto
'en la variable lptadr, obtengo dirección
DEF SEG 0       'cambio a segmento 0 para próxima inst. Peek
lptadr = PEEK(8+(lptnum-1)*2)
DEF SEG              'restauro a segmento de Basic
```

Windows

En entorno Windows, se complica un poco ya que tenemos varios métodos.

1- Verificar en las direcciones estándar que el puerto tenga retención de datos escritos en él. Es armar la tabla que realiza la BIOS por nosotros mismos. Este método falla si el puerto es bidireccional (o ECP o EPP), si algún controlador prohíbe el acceso, si estamos bajo WinNT o si el puerto está en una dirección no estándar. Ver la sección Acceso a los puertos para más detalles y véase la sección Detección de tipo de puerto, para buscar por uno mismo las direcciones (y de paso detectar que tipo de puerto es). Ahora lo único que podría hacer fallar la prueba es si algún controlador de dispositivo prohíbe el acceso (o WinNT, claro).

2- Tener la posibilidad de leer la tabla que la BIOS genera cuando arranca la máquina. Debemos contar con alguna función que nos permita leer una dirección de memoria como si estuviéramos en modo real de la CPU. En entornos de 16 bit esto se facilita. En otros lenguajes desconozco, pero para Visual Basic hay una DLL (Vbasm.dll), que tiene la función Peek como en QuickBasic.

3- Bajo WinNT, podemos obtener (mediante las funciones de la API, RegOpenKey y RegQueryValue) la información del registro de:

[HKEY_LOCAL_MACHINE\Enum\Root\
[HKEY_LOCAL_MACHINE\Enum\BIOS\
[HKEY_DYN_DATA\Config Manager\Enum\

4- En Win9x,Me, podemos hacer el enfoque de la opción 2, pero con la API de Windows. Utilizamos la función de la API Toolhelp32ReadProcessMemory (que reside en Kernel32 y que no se encuentra en NT). A continuación se muestra un fragmento de código escrito por Bill McCarthy:

```
Declare Function Toolhelp32ReadProcessMemory Lib "KERNEL32" (ByVal
      th32ProcessID As Long, ByVal lpBaseAddress As Long, lpBuffer As Any, ByVal
      cbRead As Long, ByRef lpNumberOfBytesRead As Long) As Long

Sub Main()
      Dim i As Long, rtn As Long
    Dim portAddresses(3) As Integer
    Dim cbLen As Long
    cbLen = 8
    rtn = Toolhelp32ReadProcessMemory(0&,&H408&,portAddresses(0),8,cbLen)
    'Debug.Print rtn, cbLen
    For i = 0 To 3
      If portAddresses(i) = 0 Then Exit For
           Debug.Print "port address " & i & " = &H" & Hex$(portAddresses(i))
    Next i
End Sub
```

Acceso a los puertos

A continuación se darán las funciones a utilizar para leer y escribir puertos en distintos lenguajes. En Ms-Dos no tenemos ningún tipo de restricción de acceder a los puertos. En Windows 3.x, 9x y Me tampoco hay restricciones (a no ser que el puerto esté bajo el control de un controlador de dispositivo virtual). En Windows NT, el sistema operativo tiene control total sobre la máquina por lo que hay que pedir un permiso que se hace mediante un driver.

Ensamblador

Los opcodes IN y OUT permiten leer y escribir, respectivamente, datos a un puerto cualquiera. La secuencia a seguir para leer podría ser:

```
mov         dx,Puerto       'DX = Puerto (puede ser cte. o ref. de
                            'memoria si es variable.
in          al,dx           'Leo del puerto DX y lo guardo en AL
```

Y para escribir:

```
mov         dx,Puerto       'DX = Puerto (puede ser cte. o ref. de
                            'memoria si es variable.
out         dx,al           'Manda AL al puerto DX
```

Pascal

En Pascal utilizamos el array Port[]. Este array es de tipo byte y el índice indica el número de puerto. Si asignamos un valor a un elemento del array, se refleja en el puerto correspondiente y si utilizamos un elemento del array en una expresión se leerá del puerto correspondiente. Sólo se permite referirse a un sólo elemento y no se puede utilizar como parámetros variables para funciones o procedimientos.

C

Se utilizan las funciones outportb() e inportb().

QuickBasic

Con la función INP leemos y con la función OUT escribimos:

```
OUT Puerto,Dato
dato = INP(Puerto)
```

Visual Basic

VisualBasic no tiene una instrucción para lectura/escritura de puertos. Una segunda limitación es el tipo de sistema operativo y su configuración. En sistemas Windows 3.x, 9x, Me, se puede construir una librería de enlace dinámico (DLL) que provea a los programas realizados en VisualBasic de las funciones INP y OUT.
En el archivo INOUT.ZIP se encuentran las librerías de VB para el uso de las instrucciones INP y OUT de la misma forma que en QuickBasic (en los archivos hay información detallada acerca de su uso, así como también ejemplos de aplicación).

Delphi
Para la versión de 16 bits usar las funciones inport y outport. Para la versión de 32 bits (2.0 y superiores) usar ensamblador en línea (insertado).

Detección del tipo de puerto
En esta sección indicaremos los pasos a seguir para detectar si el puerto es SPP (común), bidireccional compatible PS/2, ECP o EPP. Para una referencia acerca de los puertos EPP y ECP vea Puertos extendidos.
Describiremos los pasos generales; luego se tendrá que adaptar al lenguaje en uso. Se testea en orden descedente de complejidad, es decir primero ECP, luego EPP, bidireccional y finalmente SPP (esto se debe realizar así para no fallar en la detección).

Detectando ECP
Este es el método que recomienda Microsoft:
Leer el control de registro extendido (ECR) en Base+402h. Base se refiere a la dirección inicial donde se comienza a mapear el puerto (03BCh,0378h y 0278h). Verificar que el bit 0 sea 1 (FIFO vacía) y que el bit 1 sea 0 (FIFO no está llena). Estos bits podrían ser distintos que los bits correspondientes en el puerto de control (Base+2h). Para verificar esto, cambiamos el estado de algún bit del puerto Base+2h y verificamos que no haya cambiado en Base+402h. Una prueba adicional es escribir 34h al ECR y leerlo. Los bits 0 1 y son de sólo lectura, por lo tanto, si leemos 35h, es probable que tengamos un puerto ECP.

Detectando EPP
Además de los tres registros de un puerto SPP, un puerto EPP tiene cinco registros más mapeados desde Base+3h a Base+7h. Estos registros adicionales proveen una manera de testear la presencia de un EPP, escribiendo ciertos valores y leyendo el resultado (de la misma manera que se testea un puerto SPP). Al igual que al detectar puertos SPP, se recomienda escribir un par de valores y leerlos. Dichos valores podrían ser 55h y AAh.
Hay que asegurarse de poner S0 a 0 (EPP timeout), antes de leer o escribir a estas direcciones extendidas.
Otro dato importante es que no existen puertos EPP en la dirección base 03BCh.

Detectando SPP
Para detectar si es SPP, hacemos el procedimiento que realiza la BIOS al arranque. Verificamos la retención del puerto, que será posible en un puerto SPP. Para ello escribimos un par de valores (55h y AAh) a las direcciones base (03BCh,0378h y 0278h) y leemos el registro verificando que sean los mismo valores escritos.

Detectando puerto bidireccional (PS/2)
Aquí debemos habilitar el puerto bidireccional (con C5=1) y hacer la misma prueba que para un SPP para verificar que no haya retención.

Puertos extendidos

Puerto bidireccional (compatible PS/2)

Véase la sección Bit de puerto bidireccional.

Puerto EPP (Enhanced Parallel Port)

Pueden leer y escribir datos a la velocidad del bus ISA. Este tipo de puerto se define por el estándar EPP 1.7. Son tan rápidos como el bus del sistema y pueden alcanzar transferencias por encima de 1 Mbyte/seg.
El EPP fue desarrollado por Intel, Xircom y Zenith. Otros fabricantes comenzaron a introducir EPP que no eran del todo compatibles con el introducido por Intel. De ahí que se formó un comité para estandarizar el puerto, formando el estándar EPP 1.7. Luego se mezcló con el estándar IEEE 1284 que describe puertos bidireccionales de alta velocidad para impresoras. Pero no se aceptó el EPP 1.7 original por lo que se modificó y ahora se llama IEEE 1284 EPP, teniendo ahora dos estándar.
Un puerto paralelo estilo IEEE 1284 soporta múltiples modos. Es decir que tenemos en un único puerto, los modos SPP, bidireccional PS/2, EPP (versión 1.7 y/o 1284) y ECP.
EPP se mapea por encima de las direcciones estándar, en cinco registros (de Base+3h a Base+7h). No hay EPP en la dirección estándar 03BCh (ya que se traslapa con las direcciones dedicadas a video):

Base +	Descripción
0	puerto de datos (como SPP)
1	puerto de estado (como SPP)
2	puerto de control (como SPP)
3	address strobe
4	data strobe 0
4	data strobe 1
4	data strobe 2
4	data strobe 3

Puerto ECP (Enhanced Capability Port)

Pueden, como EPP, leer y escribir a la velocidad del bus. Fue desarrollado por Microsoft y Hewlett Packard. Se distinguen por poseer capacidad de DMA, FIFO y compresión de datos. La velocidad puede superar fácilmente el Mbyte/seg. y en el futuro se ampliará. La ventaja de estos puertos es que tienen la emulación de otros modos como SPP y bidireccional PS/2. En la especificación original no está contemplado EPP, pero los fabricantes utilizan algún bit no utilizado por ECP para poder configurar como EPP. El uso externo de ECP está definido en IEEE 1284 como el modo ECP de 1284.
El puerto se mapea en Base+400h. En Base+402h tenemos el registro ECR (Extended Control Register). Con este registro podemos configurar los distintos modos. La siguiente tabla muestra el significado de cada bit:

Bit(s)	Descripción

```
 7-5        ECP mode
            000 ISA-compatible (SPP)
            001 PS/2-compatible (bidirectional port)
            010 ISA-compatible FIFO (fast centronics)
            011 ECP
            100 reserved (EPP)
            101 reserved
            110 test
            111 configuration
 4          disable ERROR interrupts
 3          enable DMA
            when bit 3 set and bit 2 clear, an interrupt is generated on
            the DMA terminal-count condition; this bit must be cleared to
            reset the TC interrupt
 2          disable FIFO/TerminalCount service interrupts
 1          (read-only) FIFO is full
 0          (read-only) FIFO is empty
```

Notas:
Si estamos en el modo 000 o 001, podemos cambiar a otro modo. Si estamos en otro modo distinto de 000 o 001, debemos cambiar al modo 000 o 001 y luego cambiar al modo deseado.
Si actualmente estamos en el modo 010 a 111 y el bit de dirección está a cero, debemos esperar a que la FIFO se borre antes de cambiar a los modos 000 o 001.

En el modo 111, la dirección Base+400h será el registro de configuración A y la dirección Base+401h será el registro de configuración B. A continuación se describen:

Configuración A de ECP:

```
Bit(s)      Descripción
 7-4        (read-only) implementation identification
            bit 7: ISA-style interrupt
            bit 4: eight-bit implementation
 3-0        reserved
```

Configuración B de ECP:

```
Bit(s)      Descripción
 7          reserved (0)
 6          IRQ status (refleja el valor actual ya sea en IRQ5 o IRQ7;
            se usa para chequear conflictos de interrupciones)
5-0         reserved (0)
```

Nota:
C4 debe valer 0 antes de que el bit 6 muestre el estado de la interrupción.

Autor: Javier Perez / Diciembre 2000

Fuentes:

Listas de Ralf Brown (ralf@pobox.com)
Mini-faq de Kris Heidenstrom (kheidens@actrix.gen.nz)
Interfacing the IBM PC Parallel Printer Port de Zhahai Stewart (zstewart@hisys.com) Version 0.96 9/1/94
Jan's Parallel Port FAQ de Jan Axelson (jan@lvr.com)