Proyecto BOTTLER

Control del puerto paralelo mediante el PC.

(Programación en Visual Basic 6.0)

En la actualidad, muchos equipos electrónicos se pueden controlar mediante el ordenador como medio sencillo y flexible para programarlos de manera rápida y segura o como control y procesamiento del sistema, evitando circuitería complementaria. En este caso, nos vamos a centrar en conocer y utilizar el puerto paralelo del PC para nuestro proyecto, que aparte de estar exclusivamente diseñado para imprimir documentos de diferentes impresoras también se puede aprovechar para controlar exteriormente otros equipos electrónicos mediante un programa, unos drivers y una tarjeta de interfaz diseñada según la utilidad que deseamos darle.

José Miguel Castillo Castillo

Índice del contenido

1. Introducción

El puerto paralelo es una herramienta simple y económica para crear proyectos y dispositivos controlados por el PC. La simplicidad y la facilidad de programación hacen que el puerto paralelo sea popular en el mundo de los aficionados a la electrónica. El puerto paralelo se usa a menudo en robots controlados por ordenadores, programadores Atmel / PIC, automatización del hogar, etc.

Mediante el puerto paralelo podemos controlar también periféricos como focos, motores entre otros dispositivos, adecuados para automatización.

Conoceremos los conceptos fundamentales sobre el puerto paralelo de impresora Centronics y nos adentraremos en un proyecto basado en el control de una máquina conectada a través de una tarjeta interfaz al puerto paralelo del ordenador y mediante una programación realizada en ***Visual Basic 6.0.***

Durante todo el documento se irán repitiendo los conceptos fundamentales sobre el puerto paralelo para obtener una mejor compresión de todos los elementos que la integran.

Es un tema interesante y práctico el trabajar con el puerto paralelo en la programación y control de determinados dispositivos y pequeños proyectos, como hobby o estudios como Ciclo Formativos Grado Medio, Ciclo Formativos Grado Superior e incluso para la universidad, puesto que la amplia utilidades que podemos darle evitaríamos dejarlo obsoleto.

2. Puerto paralelo Centronics

Fundamentalmente el uso principal que se le da al puerto paralelo es para conectar impresoras y por lo tanto está específicamente ideado para este propósito. Se le conoce con el nombre de puerto **Centronics**, este nombre proviene de una popular empresa de fabricación de impresoras **'Centronics'** que ideó algunos estándares para puertos paralelos. Puede verse el conector del puerto paralelo en la parte trasera del PC. Es un conector hembra de 25 pines DB25 al que está conectada la impresora. En casi todas los PC solo hay un puerto paralelo, pero puede agregar más comprando e insertando tarjetas de puerto paralelo ISA / PCI.

El puerto paralelo cumple más o menos con la norma IEEE 1284, que destaca por su sencillez y que transmite 8 bits. Se ha utilizado principalmente para conectar impresoras, pero también ha sido usado para programadores EPROM, escáner, interfaces de red Ethernet a 10 Mb, unidades ZIP, SuperDisk y para comunicación entre dos PC, MS-DOS trajo en las versiones 5.0 ROM a 6.22 un programa para soportar esas transferencias.

El puerto paralelo del PC, de acuerdo a la norma **Centronics**, está compuesto por un bus de comunicación bidireccional de 8 bits de datos, además de un conjunto de líneas de protocolo. Las líneas de comunicación cuentan con un retenedor que mantiene el último valor que les fue escrito hasta que se escribe un nuevo dato, las características eléctricas son:

- Tensión de nivel alto: 3,3 o 5 V.
- Tensión de nivel bajo: 0 V.
- Intensidad de salida máxima: 2,6 mA.
- Intensidad de entrada máxima: 24 mA.

Los sistemas operativos basados en MS-DOS y compatibles gestionan las interfaces de puerto paralelo con los nombres LPT1, LPT2 y así sucesivamente, Unix en cambio los nombra como /dev/lp0, /dev/lp1, y demás. Las direcciones base de los dos primeros puertos son:

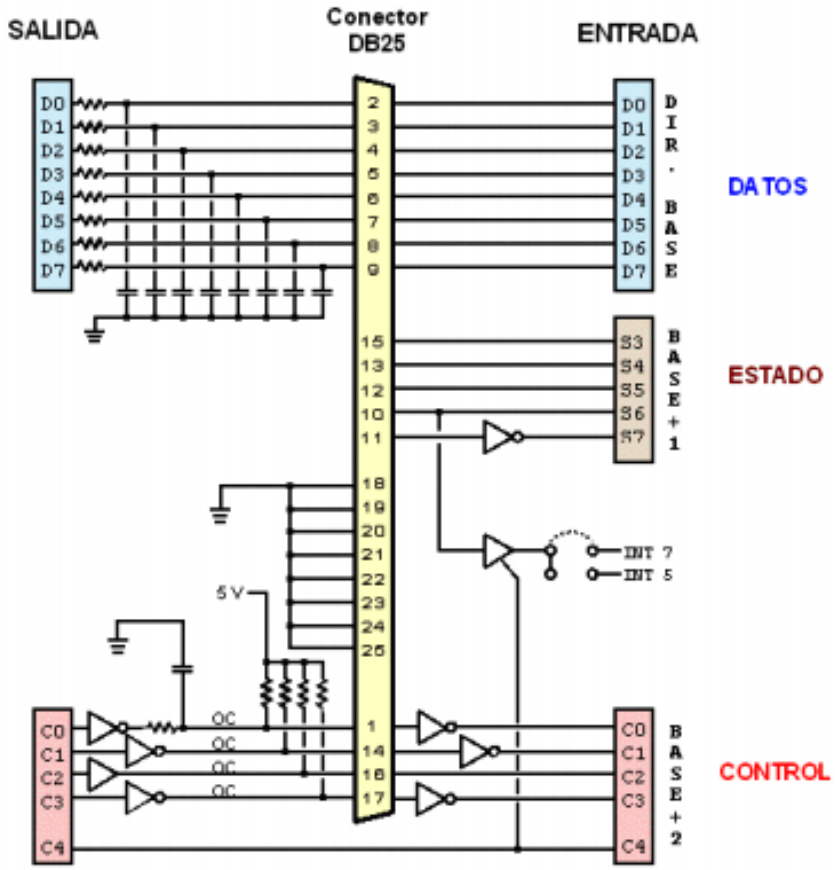
- **LPT1 = 0x378.**
- **LPT2 = 0x278**

No obstante, cuando lo que se desea es utilizar este puerto en programación para el control de determinados procesos, recibiendo señales que al ser procesadas por el programa desencadenarán una serie de señales de salida, dicho puerto se convierte en tres puertos independientes:

- **Registro de datos**: puerto bidireccional de 8 bits. Su dirección en el LPT1 es 0x378. Para utilizarlo como entrada se debe sacar previamente todos a unos por el registro de control.
- **Registro de estado**, se trata de un registro de entrada de información de 5 bits, su dirección en el LPT1 es 0x379.
- **Registro de control** es un bidireccional de 4 bits, con un bit de configuración que no tiene conexión al exterior, su dirección en el LPT1 es 0x37A.

La función normal del puerto paralelo consiste en transferir datos a una impresora mediante 8 líneas de salida de datos, usando las señales restantes como control de flujo. Cada línea o bit entrega una corriente máxima de 24mA. Sin embrago, puede ser usado como un puerto E/S de propósito general por cualquier dispositivo o aplicación que se ajuste a sus posibilidades de entrada/salida.

Existen dos métodos básicos para transmisión de datos en los PC actuales. En un esquema de transmisión de datos en serie un dispositivo envía datos a otro a razón de un bit a la vez a través de un cable. Por otro lado, en un esquema de transmisión de datos en paralelo un dispositivo envía datos a otro a una tasa de n número de bits a través de n número de hilos a un tiempo. La mayoría de los sistemas paralelos utilizan ocho líneas de datos para transmitir un byte a la vez, como en el caso del puerto paralelo de un PC.



3. Estructura e identificación del puerto paralelo del PC

El puerto paralelo se identifica por su dirección base de **E/S** (**E**ntrada**/S**alida de datos) y se reconoce en sistemas **MS-DOS** por el número **LPT** (lp en Unix/Linux). Cuando arranca la máquina, la BIOS (*Basic Input Output System*) chequea direcciones específicas de E/S 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 con el byte bajo primero seguido por el byte alto. Cada palabra es la dirección de **E/S base** del puerto paralelo (que denominaremos LPT_BASE en lo sucesivo). La primera corresponde a LPT1, la segunda a LPT2 y la tercera a LPT3. Hay que agregar que, en MS-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).

3.1. DIRECCIONES

Las direcciones base estándar para los puertos paralelos son:

- **03BCh**
- **0378h**
- **0278h**

La siguiente tabla muestra, como ejemplo, la memoria en un PC con dos puertos paralelos instalados en las direcciones hexadecimales **378** y **278**.

Identificador DOS	Dirección	Byte bajo	Byte alto	Hexadecimal	Decimal
LPT1	0000:0408/9	78	03	378	888
LPT2	0000:040A/B	78	02	278	632
LPT3 (no instalado)	0000:040C/D	00	00	0	0

3.2. REGISTROS

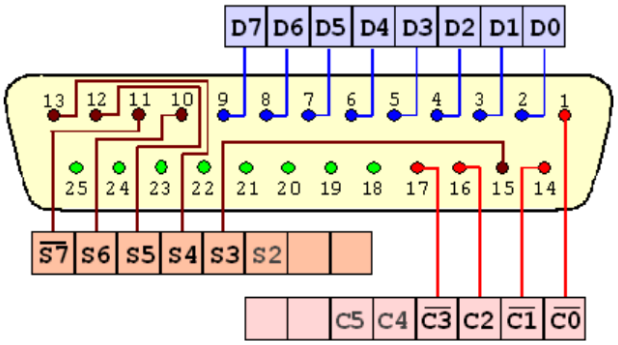
El puerto paralelo estándar (**SPP**) consta, como se mencionó antes, de tres registros de 8 bits localizados en direcciones adyacentes del espacio de E/S del PC. Los registros se definen relativos a la dirección de E/S base (LPT_BASE) y son:

- LPT_BASE + 0: registro de **DATOS**
- LPT_BASE + 1: registro de **ESTADO**
- LPT_BASE + 2: registro de **CONTROL**

		REGISTROS			Nombre habitual
		DATOS	ESTADO	CONTROL	
DIRECCIÓN E/S	Puerto	378h	379h	37Ah	LPT1
	Puerto	278h	279h	27Ah	LPT2
	Puerto	3BCh	3BDh	3BEh	MDA con p. paralelo

Se hará referencia a cada bit de los registros como una inicial que identifica el registro seguido de un número que identifica el número de bit, siendo 0 el LSB (bit menos significativo) y 7 el MSB (bit más significativo). Por ejemplo, D0 es el bit 0 del registro de datos, S7 es el bit 7 del registro de estado y C2 es el bit 2 del registro de control.

Se indican con una barra de negación los bits que utilizan lógica negativa. En lógica positiva un 1 lógico equivale a alto (~5 V TTL) y un 0 lógico a bajo (~0 V TTL). En lógica negativa 1 equivale a bajo (~0 V) y 0 equivale a alto (~5 V).



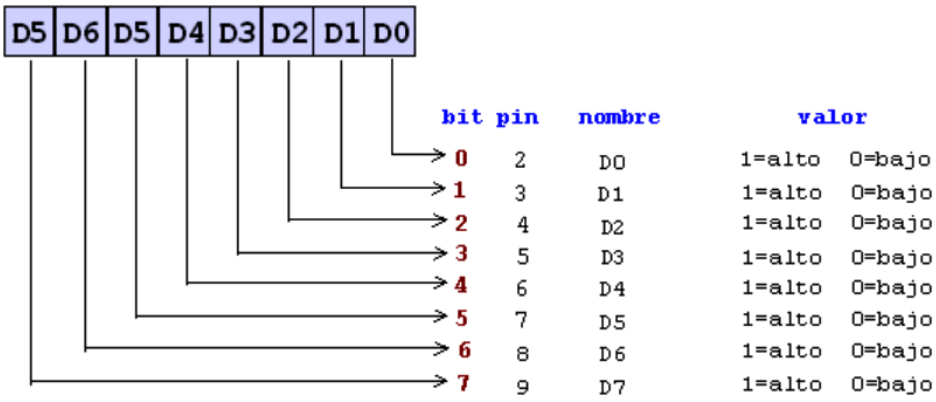
En la figura se muestra un conector DB25 donde se identifica los pines que corresponde al **registro de Datos** de D0 al D7, el **registro de Estado** del S2 a S7 y el **registro de Control** del C0 al C5.

Es preciso no confundir la lógica que sigue el puerto con la lógica que mantiene la impresora. Por ejemplo, la impresora pone a alto **Busy** (pin 11) para indicar que está ocupada. Pero en realidad, al leer el registro de estado, **Busy** la interpretamos como 0 (puesto que el pin 11 se corresponde con S7). Es decir, es como si fuera activa a nivel bajo (Busy).

3.2.1. Registro de datos (D)

El **registro de datos** se halla en LPT_BASE. Se puede leer y escribir. Escribir un dato en el registro causa que dicho dato aparezca en los pines 2 a 9 del conector del puerto.

Al leer el registro, se lee el último dato escrito (NO lee el estado de los pines; para ello hay que usar un puerto bidireccional).

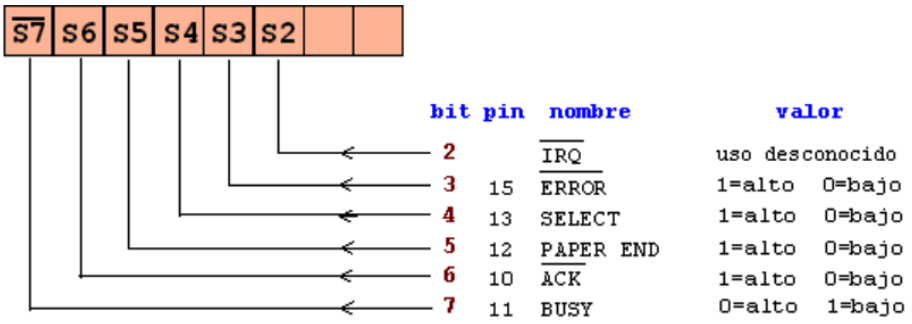


El estándar es que las salidas sean LS TTL (**low schottky** TTL), aunque las hay que son de tipo OC (colector abierto). La corriente que pueden entregar (modo **source**) es de 2,6 mA máximo y pueden absorber (modo **sink**) un máximo de 24 mA. n el puerto original de IBM hay condensadores de 2,2 nF a masa. Las tensiones para el nivel bajo son entre 0 y 0,8V y el nivel alto entre 2,4V y 5V.

3.2.2. Registro de estado (S)

El **registro de estado** está en LPT_BASE+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.

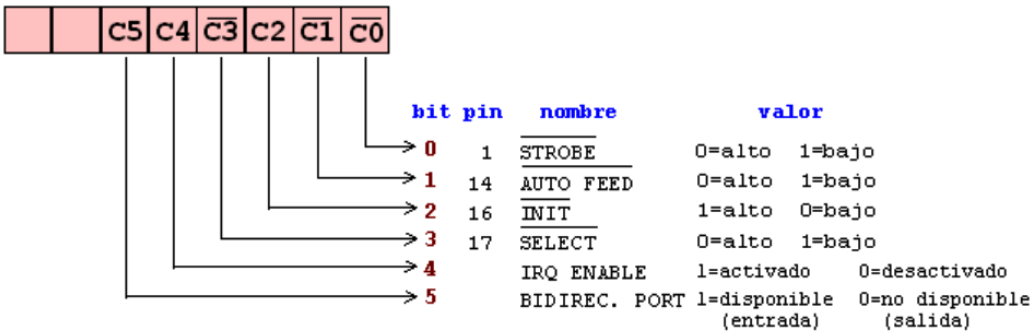
En la siguiente tabla se representan los nombres de los pines sen inglés porque es como generalmente se identifican.



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

3.2.3. Registro de control (C)

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



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 4.7KΩ, 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, se ponen en alto las cuatro salidas (escribiendo 0100b, es decir, 4h, en LPT_BASE+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 se recomienda su uso porque habría que tener un conocimiento preciso de la corriente, ya que se puede sobrecargar los transistores de salida y dañar el driver (especialmente en puertos integrados LSI).

3.3. BIT DE INTERRUPCIÓN

En trabajos normales de impresión ni el BIOS ni el S.O. hacen uso de la interrupción. El hecho de poseer una línea de interrupción que está conectada directamente al **PIC** (***P**rogrammable **I**nterrupt **C**ontroller*), 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. A continuación, se describen los pasos para poder utilizar interrupciones.

Finalmente, se muestra en la siguiente página una tabla que reúne los datos y configuración tanto del conector **DB25** como del conector **Centronics** del puerto paralelo.

DB25 pin	Centronics pin	Registro bit		Tipo (E/S)	Activo	Señal	Descripción
1	1	C0	Control 0	S	bajo	Strobe	Si está bajo más de 0.5 µs, habilita a la impresora para que reciba los datos enviados.
2	2	DO	Dato 0	S	alto	D0	Bit 0 de datos, bit menos significativo (LSB)
3	3	D1	Dato 1	S	alto	D1	Bit 1 de datos
4	4	D2	Dato 2	S	alto	D2	Bit 2 de datos
5	5	D3	Dato 3	S	alto	D3	Bit 3 de datos
6	6	D4	Dato 4	S	alto	D4	Bit 4 de datos
7	7	D5	Dato 5	S	alto	D5	Bit 5 de datos
8	8	D6	Dato 6	S	alto	D6	Bit 6 de datos
9	9	D7	Dato 7	S	alto	D7	Bit 7 de datos, bit más significativo (MSB)
10	10	S6 IRQ	Estado 6	E	alto	Ack	Un pulso bajo de ~11 µs indica que se han recibido datos en la impresora y que la misma está preparada para recibir más datos.
11	11	S7	Estado 7	E	bajo	Busy	En alto indica que la impresora está ocupada.
12	12	S5	Estado 5	E	alto	PaperEnd	En alto indica que no hay papel.
13	13	S4	Estado 4	E	alto	SelectIn	En alto para impresora seleccionada.
14	14	C1	Control 1	S	bajo	AutoFeed	Si está bajo, el papel se mueve una línea tras la impresión.
15	32	S3	Estado 3	E	alto	Error	En bajo indica error (no hay papel, está fuera de línea, error no det.).
16	31	C2	Control 2	S	alto	Init	Si se envía un pulso en bajo > 50 µs la impresora se reinicia.
17	36	C3	Control 3	S	bajo	Select	En bajo selecciona impresora (en gral. no se usa, ya que SelectIn se fija a alto).
18-25	19-30,33					GND	Masa retorno del par trenzado.
18-25	16						Masa lógica
18-25	17						Masa chasis

3.4. PROTOCOLO DEL PUERTO CON LA IMPRESORA

El *handshaking* ("apretón de manos" o protocolo) es un conjunto de reglas que ambos extremos de un sistema de comunicación tienen que seguir para que la comunicación sea correcta. El puerto paralelo, usado con una impresora, transmite datos y transmite/recibe las señales de protocolo. Las principales son **Strobe**, **Ack** y **Busy**.

La secuencia a seguir para enviar datos sería:

1. Colocar el byte a enviar en el registro de datos.
2. Verificar que la impresora no esté ocupada (Busy = bajo, S7 = 1).
3. Indicarle a la impresora que acepte los datos (Strobe = bajo , C0 = 1, pulso >5us).
4. En ese instante la impresora indica que está ocupada recibiendo los datos (Busy = alto, S7 = 0).
5. Finalmente, la impresora envía un pulso de aceptación indicando que se recibieron los datos y que se puede volver al paso 1 (Ack = bajo, S6 = 0, pulso de entre 5s y 15 s según impresora).

Las otras señales sirven para verificar el estado de la impresora (**Error**, **PaperEnd**), para reiniciarla (**Init**) y para configurarla (**AutoFeed**, **Select**). En los nuevos puertos, estas señales adquieren otra función, a veces parecida y otras totalmente distintas.

3.5. INTERRUPCIONES CON EL PUERTO PARALELO

En primer lugar, se debe 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 (LPT_BASE+2). Luego se debe 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 al 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, se deshabilita la IRQ5 (o IRQ7) escribiendo un 1 al bit 5 (o 7) del registro de interrupciones del PIC (puerto 21h). Luego se hace que C4=0.

3.6. VELOCIDAD

Un puerto paralelo ISA normal toma un ciclo-ISA para leer o escribir. En un sistema cuya velocidad de bus sea 1,3 Mhz, se puede decir que la lectura se puede hacer cada 1 s (idealmente, ya que siempre existen otras instrucciones software, etc. En la práctica pueden ser desde 1.2 s a 2 s). Algunos puertos soportan un modo "turbo" que elimina los 3 estados de espera de la CPU, con lo que la velocidad de lectura/escritura del puerto se duplica (2,7 MHz).

3.7. MODOS DE PUERTO PARALELO

El estándar IEEE 1284 que se publicó en 1994 define cinco modos de transferencia de datos para un puerto paralelo. Son:

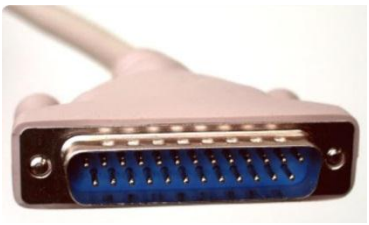
1. Modo de compatibilidad
2. Modo Nibble
3. Modo byte
4. EPP
5. ECP

Los programas, circuitos y otra información que se encuentra en este documento son compatibles con casi todos los tipos de puertos paralelos y se pueden usar sin ningún problema.

4. Cable y conectores para la conexión del puerto paralelo

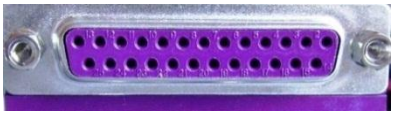
La parte de conexión hardware entre el PC y la impresora o el equipo interfaz se realiza mediante un cable y conectores específicos diseñados y fabricados para tal cometido.

El **cable** es el conductor y la unión física entre el puerto paralelo del PC y el dispositivo periférico, impresora o tarjeta de interfaz. Se compone de una manguera de **25 hilos** conectados entre sus extremos a **dos conectores macho** de **25 pines DB25**.



Cable 25 hilos y conector macho DB25 aéreo.

El **conector estándar hembra** de **25 pines DB25** aparece instalado en la parte trasera del PC con el propósito de servir de conexión entre la tarjeta base del PC y la impresora o periférico. El sistema operativo Windows cargado en dicho PC soporta hasta tres puertos paralelos asignados a los identificadores LPT1, LPT2 y LPT3, y cada puerto requiere tres direcciones consecutivas del espacio de E/S (Entrada-Salida) del procesador para seleccionar todas sus posibilidades. En un puerto paralelo habrá una serie de bits de control que irán en ambos sentidos por caminos distintos.



Conector hembra DB25 instalado en el PC

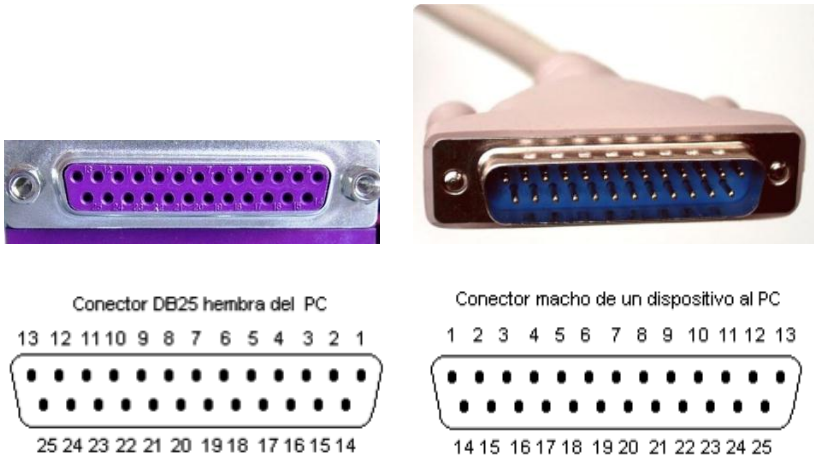
Desde el punto de vista del **hardware**, el puerto consta de un conector hembra **DB25** con doce salidas **latch** (poseen memoria/buffer intermedio) y cinco entradas, con ocho líneas de tierra.

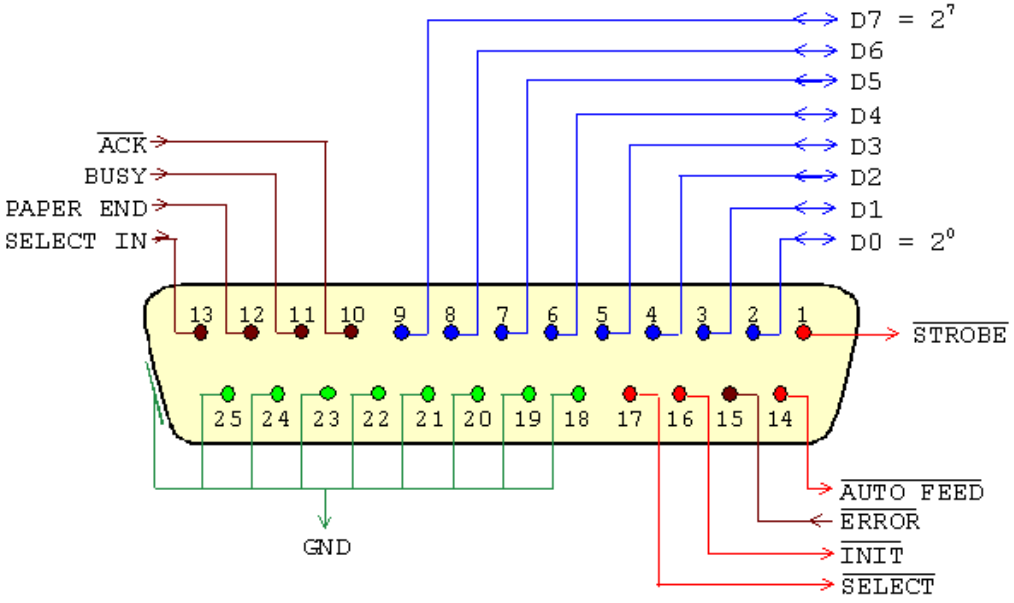
Desde el punto de vista del **software**, el puerto paralelo consta de tres registros (**datos**, **estado** y **control**) de 8 bits cada uno, que ocupan tres direcciones de E/S (I/O) consecutivas de la arquitectura x86 y que veremos más detalladamente.

4.1. DESCRIPCIÓN DEL CONECTOR DB25 MACHO Y HEMBRA

La conexión del puerto paralelo del PC al mundo exterior se realiza mediante un conector **hembra DB25** situado en la parte trasera del PC y se conecta mediante el cable aéreo (manguera de 25 hilos) y su conector **macho DB25**, a la impresora, tarjeta interfaz o equipo periférico.

Observando los conectores **hembra** y **macho DB25** de **frente** y con la parte que tiene mayor número de pines hacia arriba, se enumeran de derecha a izquierda para el conector hembra y de izquierda a derecha para el conector macho y de arriba a abajo, del 1 al 13 (arriba) y del 14 al 25 (abajo).





El conector **DB25** contiene las siguientes líneas:

- **8 líneas (pines)** son para salida de datos (bits de **DATOS**). Sus valores son únicamente modificables a través de software, y van del pin 2 (dato 0, D0) al pin 9 (dato 7, D7).
- **5 líneas** son de entrada de datos (bits de **ESTADO**), únicamente modificables a través del hardware externo. Estos pines son: 11, 10, 12, 13 y 15, del más al menos significativo.
- **4 líneas** son de control (bits de **CONTROL**), numerados del más significativo al menos: 17, 16, 14 y 1. Habitualmente son salidas, aunque se pueden utilizar también como entradas y, por tanto, se pueden modificar tanto por software como por hardware.
- las **líneas de la 18 a la 25** corresponde a la masa o tierra del puerto.

En la siguiente tabla se detallan la nomenclatura y descripción de cada línea. La columna "**Centronics pin**" se refiere a las líneas del conector tipo Centronics usado en las impresoras. La columna E/S se refiere al dato visto desde el lado del PC.

DB25 pin	Centronics pin	Tipo (E/S)	Señal	Descripción
1	1	S	Strobe	Si está bajo más de 0.5 µs, habilita a la impresora para que reciba los datos enviados.
2	2	S	D0	Bit 0 de datos, bit menos significativo (LSB)
3	3	S	D1	Bit 1 de datos
4	4	S	D2	Bit 2 de datos
5	5	S	D3	Bit 3 de datos
6	6	S	D4	Bit 4 de datos
7	7	S	D5	Bit 5 de datos
8	8	S	D6	Bit 6 de datos
9	9	S	D7	Bit 7 de datos, bit más significativo (MSB)
10	10	E	Ack	Un pulso bajo de ~11µs indica que se han recibido datos en la impresora y que la misma está preparada para recibir más datos.
11	11	E	Busy	En alto indica que la impresora está ocupada.
12	12	E	PaperEnd	En alto indica que no hay papel.
13	13	E	SelectIn	En alto para impresora seleccionada.
14	14	S	AutoFeed	Si está bajo, el papel se mueve una línea tras la impresión.
15	32	E	Error	En bajo indica error (no hay papel, está fuera de línea, error no det.).
16	31	S	Init	Si se envía un pulso en bajo > 50 µs la impresora se reinicia.
17	36	S	Select	En bajo selecciona impresora (en gral. no se usa, ya que SelectIn se fija a alto).
18-25	19-30,33		GND	Masa retorno del par trenzado.
18-25	16			Masa lógica
18-25	17			Masa chasis

El nombre de cada señal corresponde a la misión que cumple cada línea con relación a la impresora, el periférico para el que fue diseñado el puerto paralelo. Las señales activas a nivel bajo aparecen con la barra de negación (por ejemplo, **Strobe**). Cuando se indica alto o bajo se refiere a la tensión en el pin del conector. Alto equivale a ~5V en TTL y bajo a ~0V en TTL.

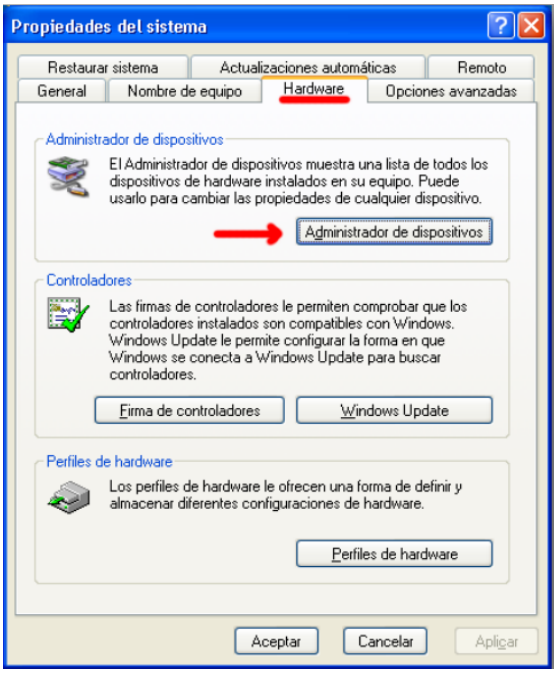
5. Acceso y configuración del puerto paralelo del PC

Desde la **BIOS** de tu ordenador o PC puedes comprobar si tienes habilitado o deshabilitado los puertos paralelos LPT. Si no lo están tienes que habilitarlo (ENABLE) para que el sistema operativo del PC lo pueda reconocer y proceder a su acceso y configuración.

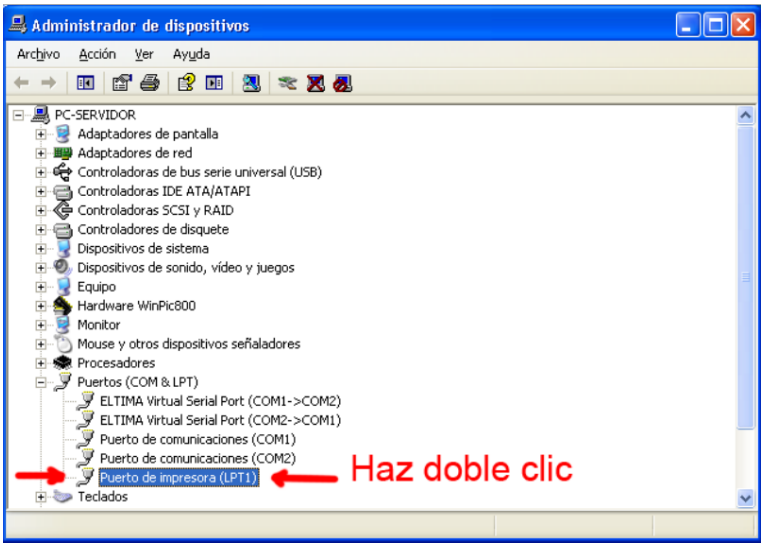
Los PC más modernos pueden configurar salidas o como entradas. Una vez habilitados puedes acceder para ver su configuración desde el sistema operativo, por ejemplo, Windows XP.

En este caso y desde **Windows XP**. Haz clic en el escritorio sin tocar ningún icono de tu monitor. Después pulsas la tecla de **Windows + Pausa**. Te aparecerá la ventana de "**Propiedades del sistema**".

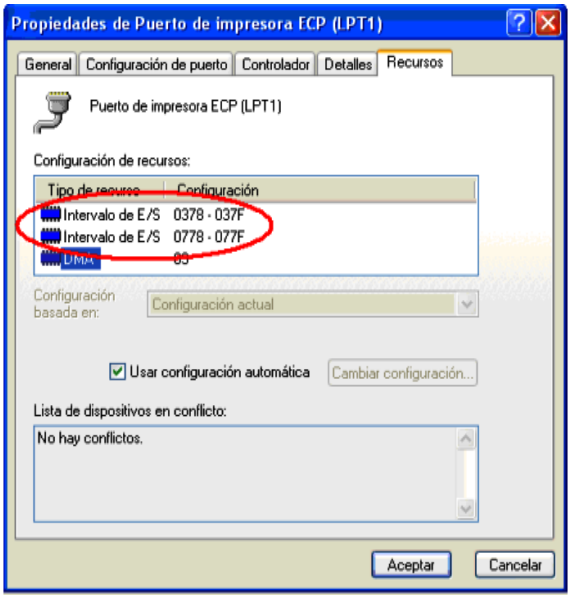
Selecciona en la ventana de **Propiedades del sistema** la pestaña **Hardware** y haz clic en **Administrador de dispositivos.**



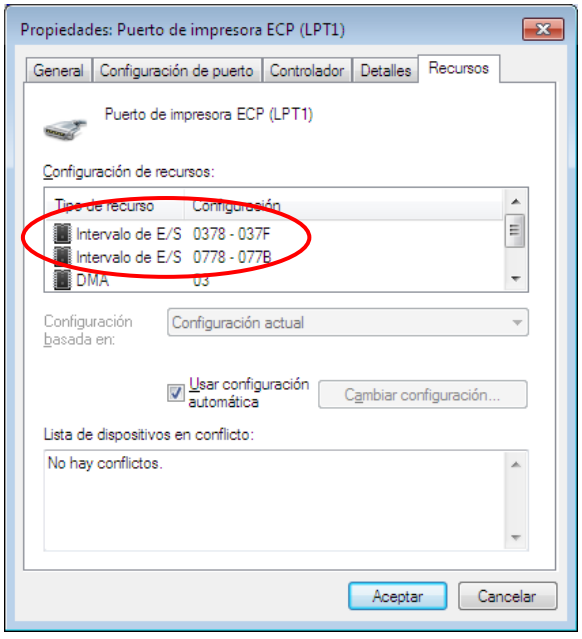
Nos aparece una lista de los controladores instalados en el sistema. Abre el **Puertos (COM & LPT)** y haz doble clic en **Puerto de impresora (LPT1).**



Como puedes ver en la pestaña **Recursos**, en la configuración está incluida la dirección **0378.**



Las direcciones de memoria que utiliza el puerto paralelo desde el sistema operativo **Windows 7** para funcionar se puede consultar haciendo: Clic derecho en Equipo → Propiedades → Administrador de dispositivos → Doble clic Puerto de impresora ECP (LPT1) → Recursos.



Observemos que los recursos son los mismos en ambos sistemas operativos. Es recomendable siempre comprobar las direcciones para no presentar errores ni confusiones.

6. El controlador (Driver) inpout32.dll

InpOut32 es una **DLL** y controlador de Windows de código abierto para dar acceso directo a los puertos de hardware (por ejemplo, el puerto paralelo y el puerto serie de los programas de nivel de usuario. Originalmente fue desarrollado por la gente de Logix4U para Windows 9x y 32bit variaciones de NT (NT / 2000 / XP / 2003 etc.).

La nueva Web donde las **.dll** están actualizadas para el sistema operativo **Windows** lo puedes encontrar en las siguientes Webs, siempre que no estén roto el enlace, puedes acceder a su descarga desde:

http://myelectronic.eu5.org/Descargas/InpoutDLL.zip

http://www.highrez.co.uk/Downloads/InpOut32/default.htm

https://github.com/ellysh/InpOut32/tree/master/bin

http://www.github.com/ellysh/InpOut32

Inpout32.dll está programado por varios lenguajes, en este ejemplo está hecho para Visual Basic, claro que también en la Web podrás encontrar ésta **.dll** hecho con otros lenguajes como pascal, Visual C++, etc.

6.1. INSTALACIÓN DE INPOUT32.DLL

Si se ha escrito una aplicación para hacer uso de **InpOut32.dll**, todo lo que se necesita hacer es colocar el archivo InpOut32.dll en la misma carpeta que la "original". Por lo general, esta sería la misma carpeta que el archivo EXE de las aplicaciones. La mayoría de las aplicaciones utilizan la versión de 32 bits de la DLL. Esta versión aún es capaz de instalarse y ejecutarse en sistemas operativos basados en x64, pero la primera vez que se ejecuta, debe elevarse en Vista y más tarde (ejecutarse como Administrador).

En la descarga binaria se incluye un pequeño programa "**InstallDriver.exe**" para hacer esto por el usuario.

6.2. PARA LOS DESARROLLADORES

El controlador (archivo .sys) está incluido, como un recurso en la DLL. Todo lo que se necesita hacer es vincular a la DLL apropiada en su programa y debería funcionar.

NOTA: Cuando la DLL se carga por primera vez, se instala y utiliza el controlador apropiado. Se requieren permisos elevados de Administrador en Windows Vista y posteriores para instalar el controlador.

Tenga en cuenta que en los sistemas operativos de 64 bits, muchas aplicaciones siguen siendo de 32 bits. Si su aplicación es de 32 bits, DEBE usar la versión de InpOut32.dll que contiene los controladores de 32 bits.

Si su aplicación es de 64 bits (x64), use InpOutx64.dll. Ambos archivos DLL pueden instalarse y comunicarse con el controlador de 64 bits (x64) en las ediciones x64 de Windows.

Con *VB6* está provista con la distribución original de Logix4U que debería funcionar exactamente de la misma manera que con otros sistemas. ***VB6* es SOLO de 32 bits** y siempre se debe usar InpOut32.dll. Para el programa de Visual Basic 6.0 se usa la librería inpout32.dll o io.dll. La librería io.dll. IO.DLL proporciona un conjunto de comandos útiles para leer y escribir en los puertos de E/S.

7. Comandos principales

El puerto paralelo nos permite manejar una cantidad de elementos y dispositivos que los hace muy atractivo a programadores y aficionados. En un programa específico diseñado para el PC podemos además de conectar y desconectar una máquina, controlar tiempos, recibir señales, etc. por medio de la ejecución de una secuencia de órdenes. En la salida del puerto paralelo tenemos 8 bornes que entrega una tensión que puede estar entre 0 y 3.5 - 4.8 voltios según el ordenador.

Con la líneas de Entrada y Salida de ocho bits podemos escribir en el puerto un total de 256 valores diferentes, cada uno de éstos representa un **byte** de información y cada byte puede representar una acción concreta que nosotros podemos definir de acuerdo a nuestras necesidades. Para ello, el objetivo principal es entender cómo se trabaja con el puerto paralelo, por lo tanto hay que hacer un programa que nos permita escribir un número cualquiera entre 0 y 255 de tal manera que sea posible visualizar el valor en formato binario.

Los puertos paralelos se denominan **&h378** o **&h278** o **&h3bc** según el ordenador. Los terminales del puerto que nos interesa son los terminales del 2 a la 9 (ambos inclusive).

Si observas esta tabla comprenderás como trabaja el puerto paralelo. Si mandamos el número 255 al puerto paralelo, el ordenador transformara dicho número a binario para después enviarlo a través del puerto en forma de señales eléctricas. Como solo disponemos de 8 entradas, el número máximo que podemos enviar será el 11111111, es decir el número 255 en decimal.

Se envían datos binarios y dependiendo del valor son los pines utilizados:

PIN	2	3	4	5	6	7	8	9
VALOR BINARIO	1	10	100	1000	10000	100000	1000000	10000000
VALOR DECIMAL	1	2	4	8	16	32	64	128

Por ejemplo:

Si queremos mandar señales por el pin 2 y el 5 pues se hace la suma.

1 + 8 = 9 → Y ese valor es el que se envía en binario 1001.

Digamos que necesitamos que se active el pin 5 y 9 la suma es:

8 + 128 = 136 → Y en binario es 10001000.

Si queremos que se encienda todos los Leds conectados al puerto paralelo pues se pone:

Valor.- 1111111 = 255

Si queremos apagar todos los Leds conectados al puerto paralelo pues se pone:

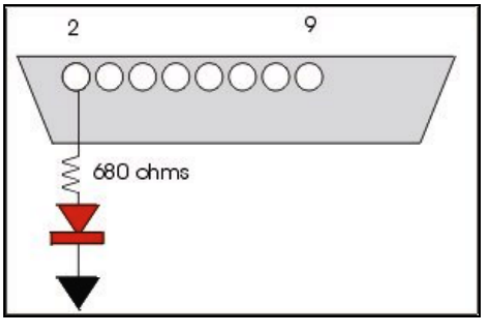
Valor.- 0 = 0

Si ponemos **Out &H378,19** se encenderán los led de los terminales 2-3 y 6, ya que el número 19 en Decimal, se escribe 10011 en binario.

La corriente eléctrica enviada al puerto corresponde a un voltaje de +5V, aprox. con lo que podemos poner a su salida algún circuito electrónico que recoja dicha señal eléctrica y la aplique a un **relé** o un **optoacoplador**. (El Optoacoplador es un componente electrónico que aísla y separa dos puntos de un circuito eléctrico con potenciales eléctricos diferentes. En los casos más usuales consta de un diodo emisor de luz LED enfrentado a un fototransistor que al recibir la luz del diodo LED el transistor empieza a conducir polarizando otros circuitos con diferentes potenciales).

En la programación se necesita comprender la forma en que serán enviados los datos, señales o pulsos al puerto. De la manera más sencilla tenemos:

1. Hay que tener en cuenta que cada vez que se envía un dato de salida en el grupo de 8 bits, se mantiene hasta que nuevamente volvamos a introducir otro dato diferente, es decir, cada vez que introduzcamos un valor de entrada o de salida diferente se va cambiando los 8 bits de golpe.
2. Para hacer visible las órdenes que demos desde el PC se puede montar un circuito sencillamente con un diodo led y una resistencia de 680Ω, soldando un cable en cada terminal cuidando que queden bien aislados entre sí y con respecto a chasis. Para circuitos más complejos se necesitan más protección para la placa base, para ello, se usan **optoacopladores**.



En el extremo libre de los cables soldamos una resistencia de 680 Ω y conectamos al ánodo del diodo LED y el otro extremo del led (cátodo) lo conectamos a chasis, masa o polo negativo del conector.

A modo global lo que hacemos primeramente es abrir un lenguaje de programación y escribimos los códigos con el nombre del puerto Ej.: **Out &H378,1** y lo compilamos para que se ejecute el programa. Esto manda al puerto de la impresora un bit. Inmediatamente se encenderá el led conectado a la salida 2 o BIT 1, y todos los demás estarán apagados. **OUT &H378, 1** Genera un bit 1 (Uno), en la patilla 2 y un bit 0 (Cero) en el resto de las patillas, es decir los conectores quedan de las siguiente manera **00000001**. Para ello, debemos tener instalados el driver en el directorio c:/windows/system32/inpout32.dll para reconocer las ordenes de lo contrario no funcionaría.

7.1. COMANDOS INP & OUT

Como hemos vistos anteriormente para establecer comunicación entre el ordenador y el puerto paralelo tenemos que utilizar dos comandos principales que nos permiten intercambiar ordenes entre el ordenador y la tarjeta de interfaz, estos son: **INP** genera un byte leído del puerto (hardware) de Entrada y Salida (E/S) y **OUT** envía un byte a un puerto (hardware) de E/S.

INP(puerto%)	puerto% Un número entre 0 y 65,535 que identifica el puerto.
OUT puerto%, datos%	datos% Una expresión numérica entre 0 y 255 que será enviada al puerto

Veamos un ejemplo:
1º. Creamos un proyecto en ***VB6*** en el formulario agregamos 2 botones.

Nombre	Texto
Command1	Prendido
Command2	Apagado

2º. Abrimos el código y agregamos.

```
Private Sub Command1_Click()
Out &H378, 1 ‘ enciende el led
End Sub
Private Sub Command2_Click()
Out &H378, 0 ‘apaga el led
End Sub
```

3º. Abrimos un módulo para establecer la ubicación del fichero inpout32.dll y ponemos:

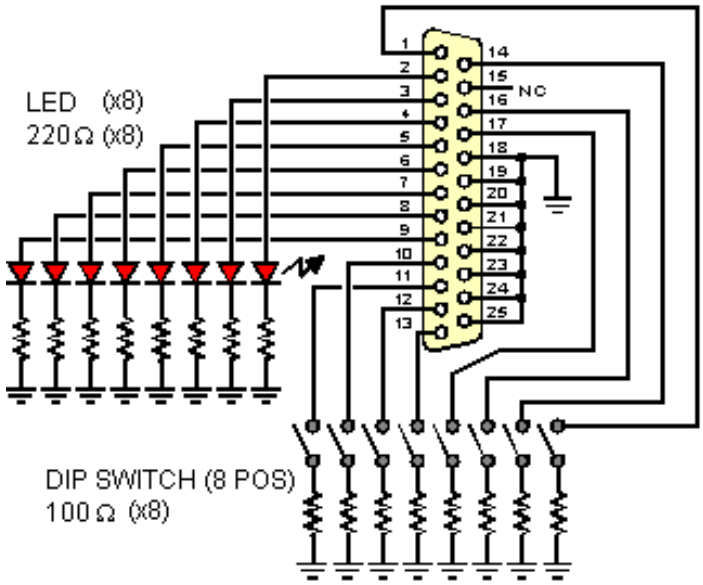
```
Public Declare Function Inp Lib "inpout32.dll" _
Alias "Inp32" (ByVal PortAddress As Integer) As Integer
Public Declare Sub Out Lib "inpout32.dll" _
Alias "Out32" (ByVal PortAddress As Integer, ByVal Value As Integer)
```

En el caso de que nuestro fichero inpout32.dll se encuentre ubicado en otra carpeta, supongamos que la tenemos en un pendriver o usb, con la letra D:\ asignada y una carpeta con nombre serial, tenemos que indicárselo en el módulo de la siguiente forma:

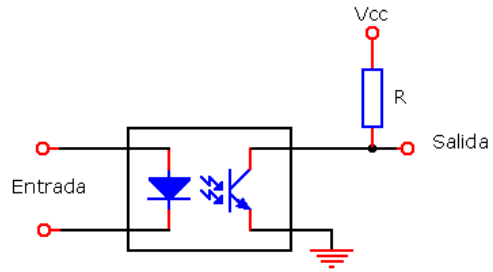
```
Public Declare Function Inp Lib "D:\ serial\inpout32.dll" _
Alias "Inp32" (ByVal PortAddress As Integer) As Integer
Public Declare Sub Out Lib "inpout32.dll" _
Alias "Out32" (ByVal PortAddress As Integer, ByVal Value As Integer)
```

4º. Una vez introducidos los códigos y el módulo proseguimos con la ejecución del programa pulsando la tecla **F5**. Veremos como pulsando el botón “**Prendido**” se enciende el Led y cuando se pulsa el botón “**Apagado**” se apaga el Led.

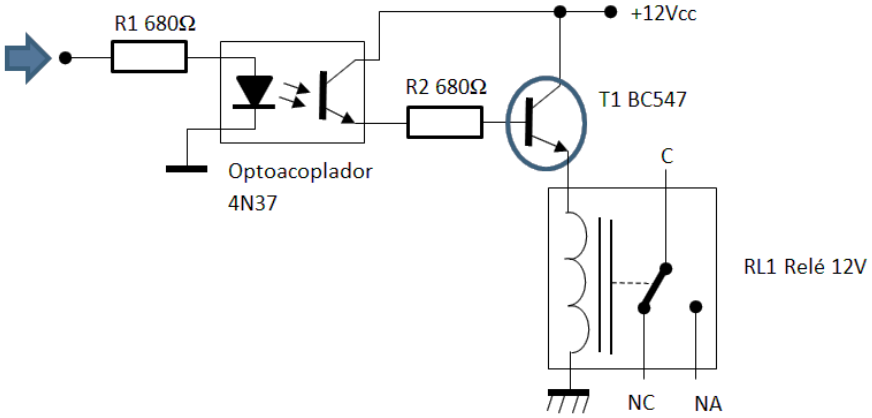
Este es un sencillo ejemplo de comienzo, la utilidad que le puedas dar al puerto paralelo puede estar más allá de donde pueda llegar tu imaginación. Observa la siguiente imagen el número de **salidas** del conector **DB25** (pines 2, 3, 4, 5, 6, 7, 8 y 9) y las **entradas** (pines 1, 10, 11,12, 13, 14, 16 y 17).

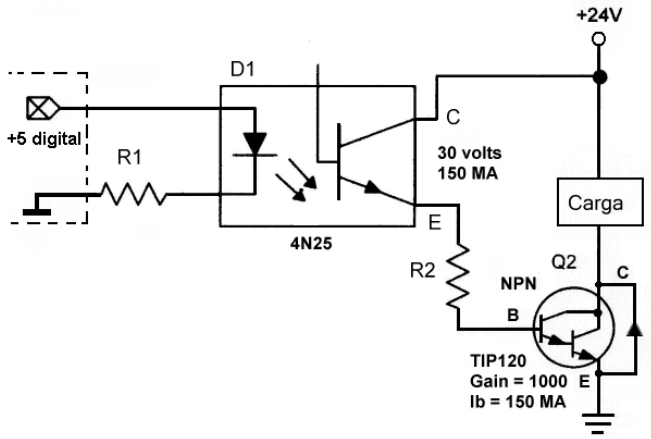


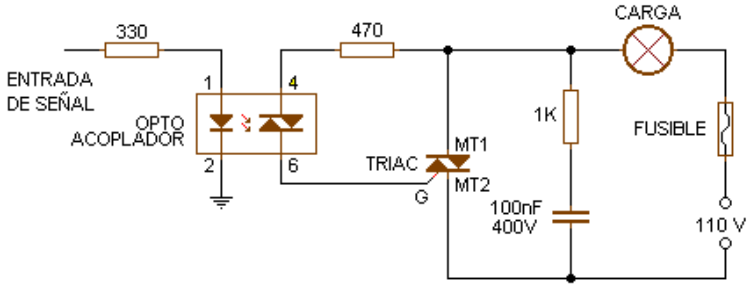
Para evitar posibles corrientes no deseadas en los terminales del puerto paralelo y poder dañar la placa base del ordenador, se suelen utilizar componentes electrónicos como pueden ser los **optoacopladores** de tal forma que con ellos separamos eléctricamente los circuitos eléctricos de la placa base del PC y los circuitos eléctricos de activación y control de salida (relés o luces) de la placa de interfaz. Con ello, se podrá controlar grandes cargas sin dañar el ordenador utilizando relés, tiristores, triacs, contactores…



Circuito típico con optoacoplador







8. Proyecto BOTTLER

El puerto paralelo del PC, aparte de que está diseñado exclusivamente para utilizarlo como puerto de impresora, actualmente está en desuso por la sustitución del **puerto USB**, desapareciendo en la mayoría de los ordenadores modernos.

Cabe destacar que su utilización puede resultar bastante útil para utilizarlo en una infinidad de proyectos. Para ello, hay que conocer perfectamente todos los elementos que lo integran, puesto que al ser un dispositivo hardware hay que conocer cómo interpretar las líneas de datos, controles y estados, así como saber programarlo en un lenguaje de alto nivel y compatible con el sistema operativo del PC, requiriendo sus drivers para controlar y comunicarse con la tarjeta de interfaz.

Por lo tanto, el puerto paralelo se puede utilizar perfectamente para controlar por medio del PC una tarjeta interfaz y ésta una máquina o equipo periférico para manejar y ejecutar determinadas funciones programadas: visualización de eventos de entradas, activación de aparatos, control de dispositivos sensores, activación de motores, etc.

Su principal característica, como se ha comentado anteriormente, es que posee un bus de 8 bits de datos de E/S que viajan juntos, enviando un paquete de un byte a la vez. Es decir, se implementa un cable o una vía física de 8 hilos para cada bit de datos formando un bus de datos de 8 bits que se envían todos a la vez.

En este proyecto se va a crear una **interfaz** (circuito electrónico) que se conectará al ordenador, por medio de un cable **DB25** conectado al puerto paralelo y, por otra parte, a la máquina de la que queremos controlar mediante una programación diseñada y creada para tal propósito.

El objetivo es saber cómo programar utilizando **Visual Basic 6.0** para controlar el funcionamiento del puerto paralelo de E/S. Saber cómo aplicar el archivo DLL como controlador, para que la programación en Visual Basic 6.0 pueda acceder a las entradas y salidas del puerto paralelo. Saber cómo obtener el valor de los datos de entrada a través del puerto de estado y el valor de los datos de salida a través del puerto de datos.

Para entender el funcionamiento de la interfaz, es necesario describir brevemente el funcionamiento del puerto paralelo. En la siguiente figura se muestra un esquema de la ubicación de los 25 pines del puerto paralelo de un PC a través de un conector hembra DB25. Los pines se agrupan en tres puertos y se muestran en un color diferente. Los pines 2 a 9 forman parte del puerto de datos (D), los pines 1, 14, 16 y 17 forman parte del puerto de control (C) y los pines 10 a 13 integran el puerto de estado (S). Los pines 18 a 25 van a masa o tierra.

En el presente proyecto se utilizarán **6 bits de salida** correspondiente al puerto de datos **D0, D1, D2, D3, D4** y **D5** con dirección **378h** y pines 2, 3, 4, 5, 6 y 7 y los **5 bits de entradas** correspondiente al puerto de estado de entrada **E3(S3), E4(S4), E5(S5), E6(S6)** y **E7(S7)** con dirección **379h** y pines 10, 11, 12, 13 y 15.

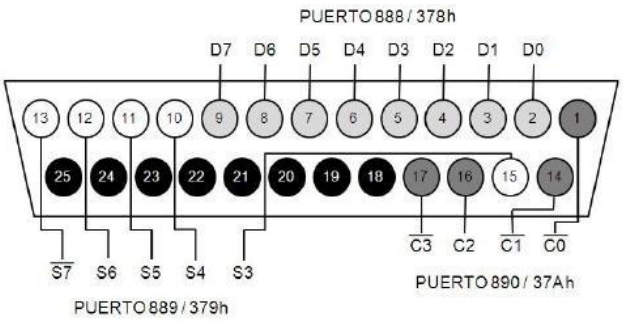
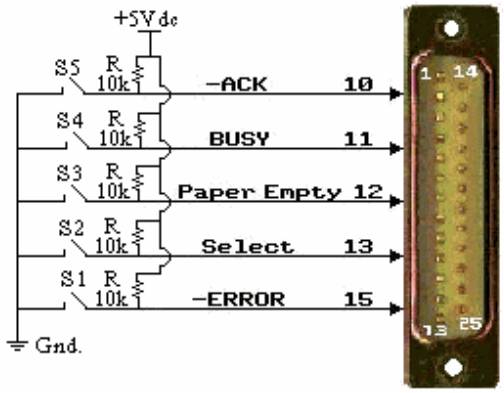


Figura 1. Diagrama de la ubicación de los pines en el puerto paralelo y las direcciones de los puertos.

Cada vez que un pin **E(S)** hace contacto con un pin a tierra, su valor cambia y podemos detectar el cambio usando *Visual Basic.* Más adelante se describe el uso de *Visual Basic* para leer las entradas y generar salidas. Los pines D0 a D7 generan 5 VDC cada vez que se cambia su estado desde *Visual Basic*. Es importante señalar que el puerto paralelo es delicado y una conexión incorrecta puede dañarlo permanentemente.

En realidad, obteniendo datos llamados "**Entrada**" **E(S)** desde el puerto de la impresora; la impresora enviará su señal a la interfaz con la computadora para mostrar el estado de la impresora, verificando el envío de datos a la impresora, como Error de impresora, sin papel, listo para trabajar, etc.



Todos estos pines 10, 11, 12, 13 y 15 se conectarán a través del **puerto de estado**, E3(S1), E4(S2), E5(S3), E6(S5) y E7(S4) que es un puerto para obtener una señal de entrada que es de solo lectura, que difiere de enviar la señal al puerto paralelo que enviaremos a través del **puerto de datos**, de 8 bits.

La utilidad de controlar el dispositivo que utilizaremos el **puerto de estado** de la impresora puede estar recibiendo la señal de muchos dispositivos, como conmutadores, relés, sensores, etc.

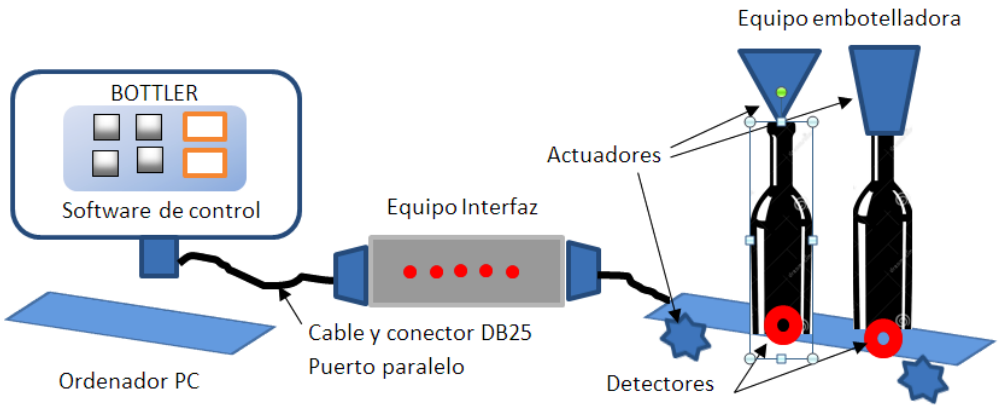
8.1. ¿EN QUÉ CONSISTE EL PROYECTO BOTTLER?

Si nos detenemos a pensar el significado de "*proyecto*" se podría decir que consiste en unas series de tareas, estudios y operaciones planificadas, todas ellas relacionadas con el tema específico del proyecto y con el objetivo de ponerla en la práctica y obtener los resultados esperados. Por ejemplo, en nuestro proyecto **BOTTLER** tenemos unas series de tareas que debemos preparar previamente:

1. *Tener muy clara la idea de lo que queremos hacer*
2. *Anotar y recopilar toda la información y datos importantes*
3. *Hacer un croquis de todos los elementos que intervienen*
4. *Dibujar el esquema eléctrico y componentes utilizados en la parte Hardware*
5. *Montar los componentes en placa mutitaladros o Project Boards*
6. *Verificar y comprobar con tensión el circuito eléctrico*
7. *Planificar los eventos, mensajes, instrucciones, bucles, etc. en la parte Software*
8. *En la programación en Visual Basic crear el proyecto*
9. *Introducir el nombre principal de nuestra aplicación*
10. *Introducir eventos y elementos*
11. *Establecer las propiedades de los eventos introducidos*
12. *Colocación y orden de tabulación para los botones y cajas de textos*
13. *Introducir las instrucciones relacionadas con los comandos y subrutinas*
14. *Ir ejecutando nuestro programa, de vez en cuando, para depurar errores*
15. *Conectar la parte hardware al PC con la programación en VB desarrollada*
16. *Comprobar y verificar nuestra aplicación y su correcto funcionamiento tanto software como hardware.*

Comentado esto, la idea general de nuestro proyecto consiste en la preparación y realización de un circuito de interfaz conectada al puerto paralelo del PC y realizar la programación en *Visual Basic 6.0* de dicho interfaz para controlar las operaciones de E/S de una máquina embotelladora (**BOTTLER**).

Según se aprecia en la siguiente imagen vemos las tres partes fundamentales de que consta el proyecto: **SOFTWARE DE CONTROL+INTERFAZ+MÁQUINA.** El propósito es controlar los actuadores y detectores de una máquina embotelladora a través de un equipo interfaz y controlarlo mediante un software específico instalado en el PC, donde, interactivamente, podemos manejar botones y ver mensajes a tiempo real de los procesos que se van produciendo, así como poner en marcha la máquina o apagarla, activar o detener los procesos o salir del programa. Pare ello, debemos conocer ampliamente el funcionamiento de la máquina, donde van colocados los sensores, actuadores, etc.



8.2. PROCESOS A DESARROLLAR

En este apartado vamos a describir los eventos que necesitamos para controlar las salidas y entradas:

1º. La máquina dispone de un interruptor general **ON/OFF** que irá conectado a la tarjeta interfaz y será controlado desde el mismo programa software por medio de un botón de "**Conectar máquina**" y otro de "**Apagar máquina**". Al conectar la máquina nos muestra los siguientes mensajes en la pantalla del ordenador:

"Máquina conectada"
"Active procesos"

2º. Al **desconectar la máquina** se desactivarán todos los procesos y nos muestra por pantalla el mensaje:

¿Confirmar apagar la máquina?
"Máquina desconectada"

3º. El programa dispondrá de otros dos botones, en uno, donde se podrá "**Activar procesos**" y el otro "**Detener procesos**". Cuando se pulsa **Activar procesos** aparece el siguiente mensaje:

"Máquina conectada"
"Procesos activados"

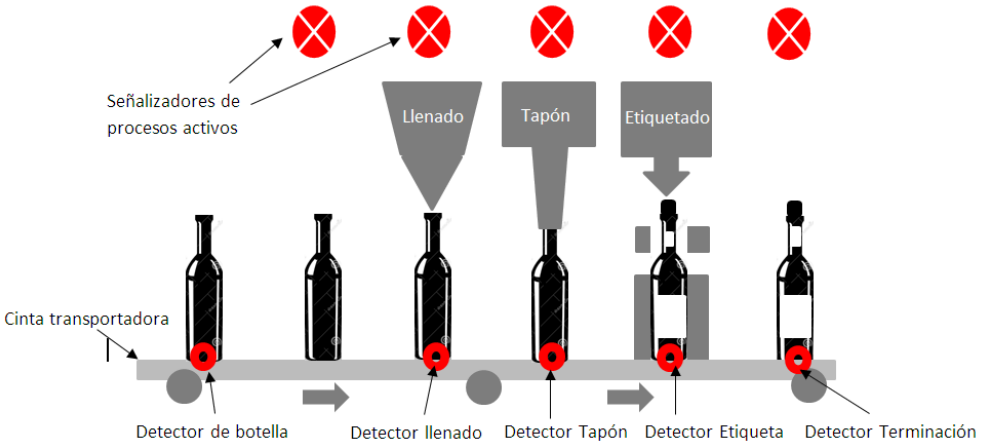
4º. Cuando se activa el evento "**Detener procesos**" nos muestra los siguientes mensajes:

"Máquina conectada"
"Procesos detenidos"

5º. Se dispondrá de un evento que permita salir del programa. El botón "**Salir del programa**" nos permite salir del programa apareciendo un mensaje de confirmación:

¿Confirma salir del programa?

6º. Cuando se active cualquier entrada de datos mediante los 5 pulsadores (microrruptores) llevando el pin a masa o tierra **E3, E4, E5, E6** y **E7,** cada uno de estos activara un proceso, que tendrá como salida activar un relé y este un motor, dispensador, contactor, etc., formando un ciclo operativo hasta ser detenido.



Según vemos en la imagen anterior, la máquina va a disponer de 5 sensores (microrruptores) instalados en diferentes posiciones de la máquina, que nos señaliza en cada momento la posición donde se encuentra la botella y su posterior activación del actuador/dispensador correspondiente. El cometido de cada pulsador-microrruptor es:

1. **Detectar la botella** (Arranque y desplazamiento de la botella mediante la cinta transportadora hacia el primer proceso)
2. **Llenado** (Se detiene, se llena la botella y continua)
3. **Colocación Tapón** (Se detiene, coloca el tapón y continua)
4. **Etiquetado** (Se detiene, coloca la etiqueta y continua)
5. **Detección terminación** (Finaliza el proceso y se prepara para su embalaje. La cinta se detiene y está preparada para activarse nuevamente en el momento que detecte otra botella, produciéndose un ciclo continuo hasta que no se paren los procesos o se apague la máquina).

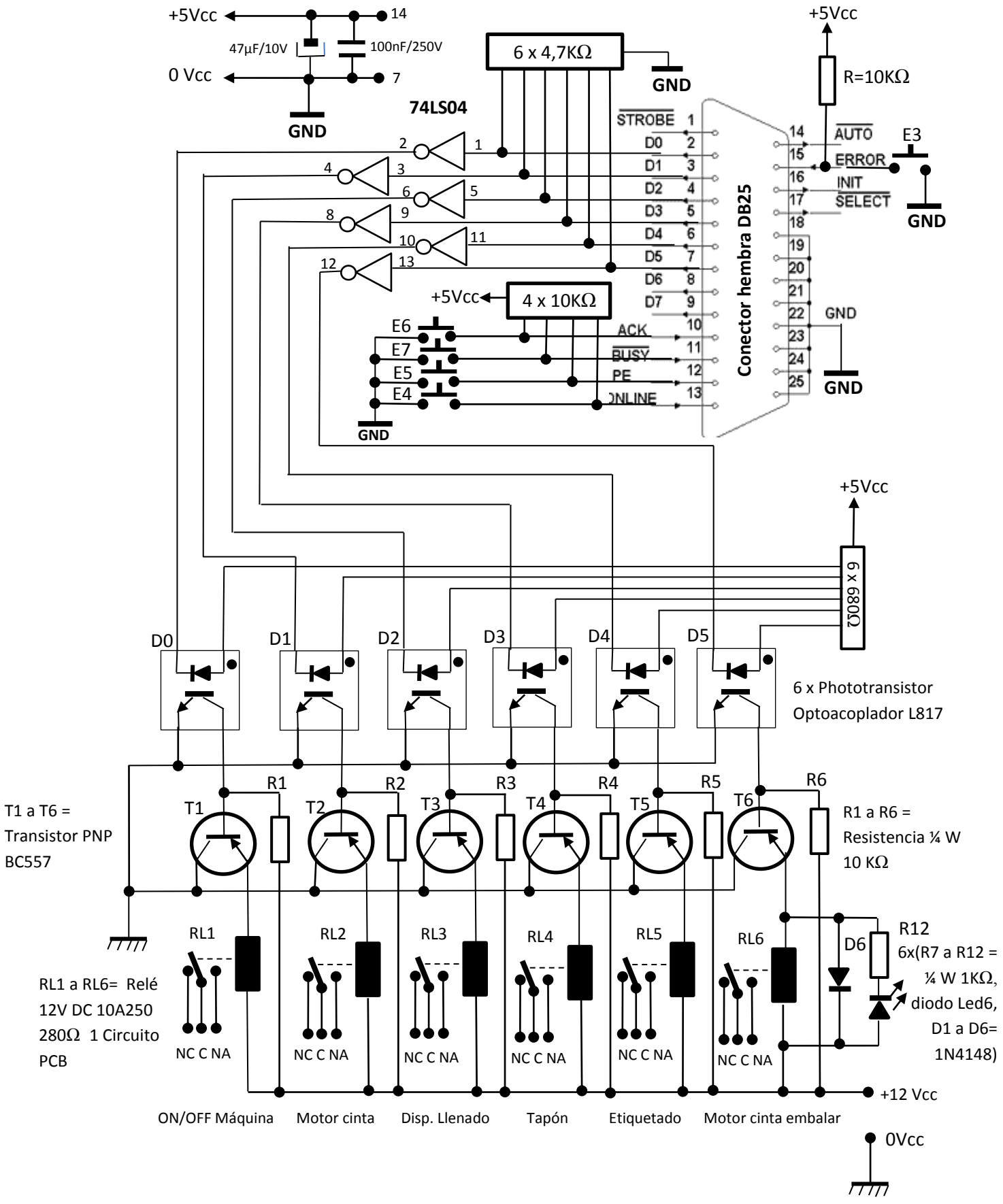
Mientras esté un pulsador accionado los otros cuatro pulsadores no actúan. Conforme se va posicionando la botella en las diferentes posiciones van apareciendo 5 mensajes de texto en la pantalla del ordenador, señalizando el proceso que en ese momento se encuentra la botella:

"Detección botella"
"Llenado"
"Colocación tapón"
"Etiquetado"
"Embalaje"

8.3. ESQUEMA ELÉCTRICO DEL EQUIPO INTERFAZ

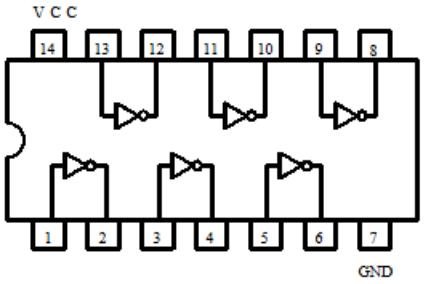
Ya hemos visto todo lo que queremos hacer en el proyecto. Pero ahora toca la parte hardware. Para que los detectores, señalizadores y actuadores se muevan controlados por el software necesitamos de una tarjeta de interfaz que al conectarla entre el ordenador (puerto paralelo) y la máquina le podamos enviar ordenes, señales eléctricas digitales, para que puedan actuar.

8.3.1. Esquema eléctrico



8.3.2. Descripción del circuito eléctrico

El circuito eléctrico consta de un bus de datos de 6 bits de salida **D0** a **D5** que nos actuará las salidas según la programación establecida. Esta línea de datos entra en el circuito integrado **74LS04 DIP14** formado de 6 inversores, correspondiendo un inversor para cada uno de los datos de salida. Este **74LS04** sirve de protección, separación y adaptación de las señales digitales que vienen del conector del puerto paralelo del ordenador. La señal de activación que es de nivel alto (1) el inversor lo invierte a nivel bajo (0) de esta forma lo que se hace posteriormente es polarizar el optoacoplador y el circuito de relé para que al final se activen con la señal a nivel alto. Se le añade una resistencia de Pull-Down de 4,7 KΩ. Se toman las salidas y entradas de datos del conector DB-25 del pin 2 al pin 7.

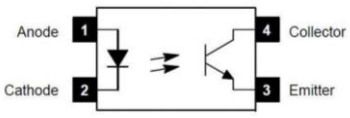


Si nos fijamos en el **esquema eléctrico**, necesitamos de 5 entradas E3, E4, E5, E6 y E7 que nos harán de detección de posición de la botella mediante la colocación de unos **microrruptores NA** (Normalmente Abierto) que cuando se vayan activando envía una señal a nivel bajo (0) al puerto paralelo y esto lo recibe el programa y controla 5 bits de salida D1, D2, D3, D4 y D5 actuando según la programación establecida: sobre 5 relés del RL2 a RL6 y 5 diodos leds del 2 al 6, que nos indicarán en cada momento el proceso que se va ejecutando. El relé RL1 y diodo Led 1 se activa directamente desde el control del programa con dos botones para apagar y conectar la máquina ON/OFF.

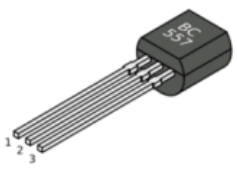
Las señales recibidas de los detectores ejecutarán el motor o dispositivo que le corresponda:

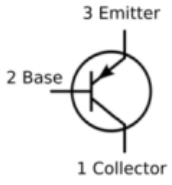
1. **Detector de botella** pone la cinta en movimiento (Motor de cinta)
2. **Detector de llenado** pone en funcionamiento el dispensador de líquido (Llenado)
3. **Detector de Tapón** pone en funcionamiento el dispensador de tapones (Tapón)
4. **Detector de etiqueta** pone en funcionamiento el dispensador de etiquetas (Etiqueta)
5. **Detector de embalaje** fin del proceso (Motor cinta de desplazamiento para su embalaje)

A la salida de estos 6 inversores se conectan a un optoacoplador **EL817 DIP4** para cada una de las líneas de datos y poder activar un relé para cargas superiores. La entrada del optoacoplador se polariza con una resistencia en serie de 680Ω, conectada permanentemente al positivo de la fuente de alimentación de 5 voltios y al terminal **ánodo** del diodo led, esta resistencia se usa para limitar la intensidad y no dañar el diodo led del optoacoplador cuando se le aplica una sobretensión a la que no está diseñado, evitando también que no se produzca una sobrecarga que pueda dañar la placa del puerto paralelo. Cuando le llega **tensión negativa,** del inversor de **IC1** al terminal **cátodo,** el led se encenderá.

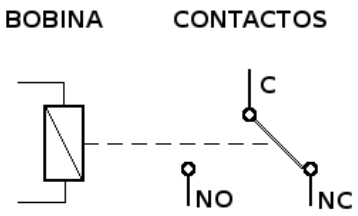


En la salida del opctoacoplador **EL817,** el fototransistor no tiene la suficiente capacidad de activar una carga grande, por lo tanto se le añade 6 transistores **BC557 PNP** al circuito para amplificar la corriente lo suficiente para que la bobina del relé pueda activar los contactos sin ningún problema.

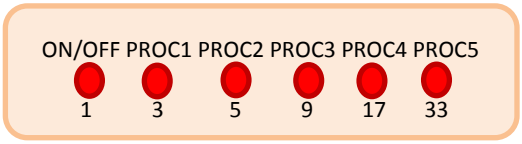




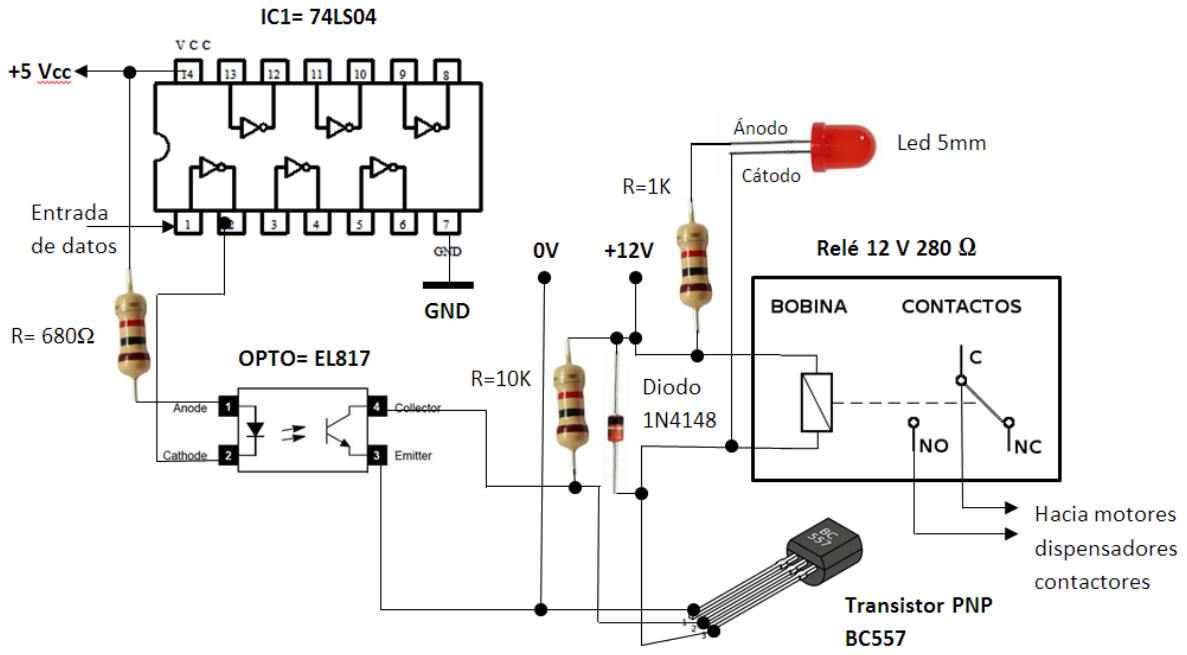
Se utilizarán **6 relés**. El **relé** es del tipo sin enclavamiento, para montaje en **PCB**, de **12V DC**, con resistencia de la bobina de **280Ω**. El material de los contactos debe ser plata y soportar un amperaje de **10A/250V** y de un circuito conmutado. En las bornas de alimentación de la bobina del relé se colocará un diodo de protección **1N4148** para evitar corrientes y fuerzas contra electromotrices que puedan dañar el circuito.



Para la señalización de cada evento se incorpora 6 diodos **Led rojo** montado en el panel del equipo interfaz que nos avisa del proceso que se encuentra activo. Los LEDs son diodos que tienen la capacidad de emitir luz cuando circula una corriente por ellos. Esta corriente debe ser del orden de los 10 mA (miliamperios). Para limitar la corriente que pasa por el diodo LED se usa una resistencia de valor de 1KΩ que protege al LED de su destrucción.



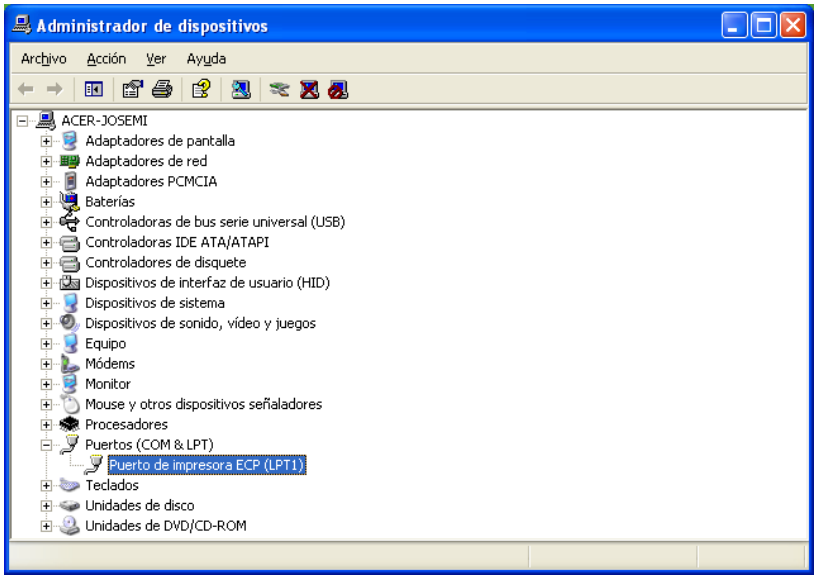
El circuito se alimenta de dos tensiones continuas de diferentes voltajes. Una tensión de 5 voltios para alimentar el circuito integrado **IC1** y otra de 12 voltios para el circuito de relé. También hay que tener en cuenta utilizar el pin 25, que es el pin de masa o tierra eléctrica para la adaptación en común de nuestro circuito de entrada y el PC. Tendremos únicamente en común las masas **GND** tanto del conector **DB25** con la masa de la fuente de alimentación de 5 voltios. La masa y la tensión de la fuente de 12 voltios deben estar totalmente aisladas con respecto a la tensión de 5 voltios.



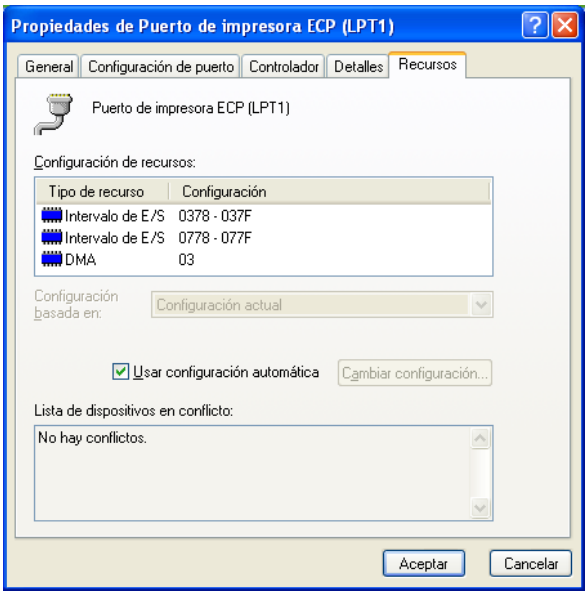
Estos elementos nos permiten señalizar y activar los dispositivos, dispensadores y motores, separándolo de las señales que nos vienen del puerto paralelo para no dañarlo. Es importante que las tensiones de alimentación que se utilicen estén lo suficientemente filtradas y estabilizadas.

8.4. ACTIVACIÓN DEL PUERTO PARALELO DEL PC

Para que nuestro proyecto **BOTTLER** funcione correctamente tenemos que activar el puerto paralelo LPT1 desde la BIOS (en el caso de que no lo esté) y seguidamente obtener la dirección base desde el sistema operativo del PC accediendo a: Inicio | Panel de control | Sistema | Hardware | Administrador de dispositivos | Puertos (COM y LPT) | doble clic en el puerto LPT de interés | Recursos.



En la información de la pestaña **Recursos**, tomamos como dirección base el límite inferior del primer rango de Entrada-Salida.



En la imagen anterior, el límite inferior del primer rango de E/S es **0x378**. Por lo tanto, al ejecutar el programa, en el campo **Dirección base** entramos este valor precedido por los caracteres **&H**, así: **&H378.**

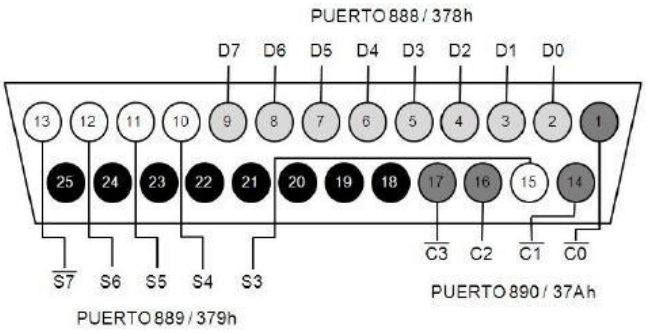


Figura 1. Diagrama de la ubicación de los pines en el puerto paralelo y las direcciones de los puertos.

La configuración más común del puerto paralelo lo ubica en la terminal LPT1 con las direcciones 888 (378h), 889 (379h), 890 (37Ah) como se muestra en la Figura 1. Para facilitar la descripción de la presente interface, en las siguientes secciones se asume que el puerto paralelo está configurado de esta forma. Sin embargo, dicha configuración puede variar y debe verificarse antes de proceder a controlar las entradas y salidas.

8.5. LA PROGRAMACIÓN

Para programar nuestro equipo interfaz es importante tener conocimientos, al menos, fundamentales de programación en ***Visual Basic***.

Si ya has programado anteriormente, con cualquier otro lenguaje, se te hará más fácil programar diversos proyectos para controlar máquinas, luces, motores, mecanismos eléctricos, etc., y desde el ordenador mediante el puerto paralelo. Una vez que tengamos el programa perfectamente funcionando solamente nos falta conectar el **cable DB25** del ordenador a la tarjeta de interfaz y ésta al dispositivo que va a ser controlado. En el monitor del ordenador veremos los eventos que se van produciendo en la máquina (máquina conectada, procesos activados, proceso etiquetado, etc.) y podremos actuar sobre los diferentes actuadores mediante los botones que aparecen en la aplicación del programa pulsando con el ratón para activar/desactivar, inicializar, detener o salir del programa.

Para realizar una secuencia de programación debemos tener presente que es lo que queremos hacer y cómo tiene que ser el resultado:

1) **¿Qué debe hacer el programa?** Reflexione sobre la tarea o tareas que debe realizar el nuevo programa.
2) **¿Qué aspecto tiene el formulario? ¿Qué controles se necesitan?** Determine la apariencia que van a tener los cuadros de diálogos, que en ***Visual Basic*** se denomina **formularios**. Realice un esquema del formulario en un papel tal como le gustaría que quedara. Tome nota de los controles que se necesitan: botones, etiquetas, cajas de texto, etc.
3) Crear los **formularios**.
4) Establecer las propiedades de los formularios y de todos los elementos que contienen, también llamados **controles**.
5) En este caso es donde realmente se lleva a cabo la verdadera programación.
6) Compruebe la funcionalidad del programa ejecutándolo de vez en cuando, durante su desarrollo, por si hubiera algún error y atajarlo antes de seguir engordando el programa.

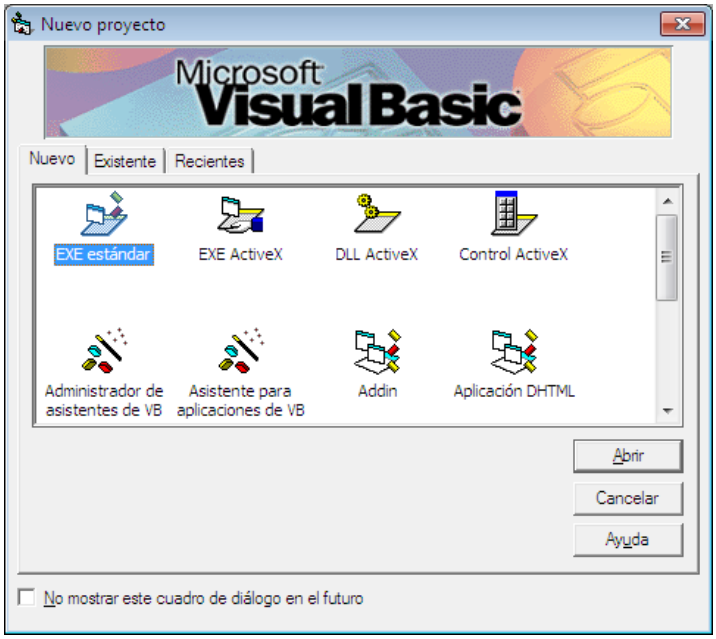
8.5.1. Introducción al Visual Basic 6.0

Visual Basic 6.0 es un entorno de desarrollo para aplicaciones de Windows, pensado para trabajar de una forma rápida y sencilla. Se podría definir como un lenguaje con herramientas gráficas de alto nivel pero con acceso a todas las funciones del sistema.

A pesar que se trabaja con unos controles que son similares a los objetos, ***Visual Basic*** no se considera un lenguaje de **P**rogramación **O**rientada a **O**bjetos (**OOP**), ya que no posee algunas de las principales características de estos lenguajes (herencias, polimorfismo, etc.).

La programación de *Visual Basic* no es la típica programación, en la que todo programa tiene un punto de entrada y un punto de salida, sino que es una programación **conducida por eventos**. Una aplicación de *Visual Basic* se compone de varias **funciones independientes** que se ejecutan en el momento en que ocurre algo, este algo es un evento, por ejemplo, aplicación típica de Windows que tiene un cuadro que puede realizar varios procesos en función del botón que se pulse, cada uno de estos botones tiene asociado un **código** (instrucciones del programa) que únicamente se ejecutará con ese botón y de ninguna otra forma.

Los **eventos** o **mensajes** los genera el usuario con sus acciones (mover el ratón, pulsar el teclado, etc.) y Windows es quién los recibe y los transmite a *Visual Basic* para ser procesados. A pesar de esto Visual Basic no procesa todos los mensajes de Windows, sólo los considerados más importantes. De la misma manera nuestras aplicaciones sólo procesarán los mensajes que nosotros deseemos controlar.



Toda aplicación de ***Visual Basic*** está compuesta por los siguientes elementos:

- **Formularios y controles**: Corresponde al diseño de aplicación, las ventanas, botones, listas, cuadros de diálogo, etc.
- **Propiedades**: Características particulares de cada control.
- **Eventos**: Mensajes que deseamos procesar.
- **Procedimientos**: Trabajos que realizará Visual Basic al recibir y procesar un mensaje.

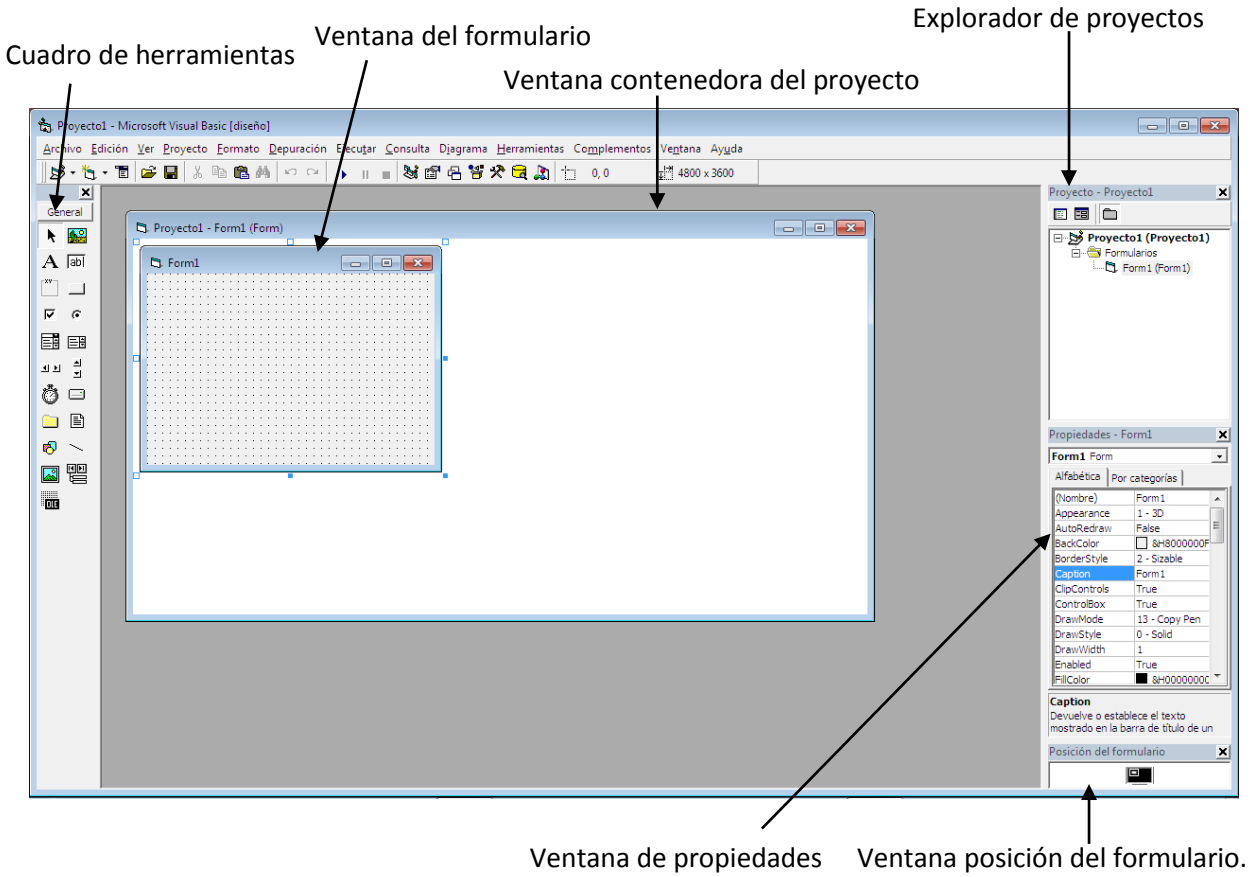
Los **controles** pueden ser botones, barras de desplazamiento, listas, casillas de selección, botones de opción, etc., y cada uno de ellos tiene sus eventos, que puede ser una pulsación del ratón sobre él, modificación de alguna de sus propiedades como el tamaño, la situación, etc.

Debemos tener en cuenta que un **formulario** puede tener tantos controles del mismo tipo como sean necesarios (objetos de una clase) y cada uno de ellos tendrá sus propiedades y procesará sus eventos de forma individual.

Básicamente, un **programa** es una colección de instrucciones. En *Visual Basic* las instrucciones se integran en **procedimientos**. Un procedimiento comienza con la línea *Sub Nombre()* y termina con *End Sub*.

Sub, End, Private, etc. son términos especiales de *Visual Basic*, las denominadas palabras clave o claves. Comienzan automáticamente con mayúsculas y se representa en la ventana de código en color azul.

Al ejecutar ***Visual Basic 6.0*** nos aparece en pantalla el entorno de programación con los siguientes elementos:



En el **Cuadro de herramientas** se encuentran los diferentes componentes que puede aplicar en los formularios. Si el cuadro de herramientas no aparecen en pantalla active su vista con la opción *Ver/Cuadro de herramientas*.

La **Ventana contenedora de proyecto** se muestra el esquema del formulario. Para activar su vista haga una doble pulsación en el botón principal del ratón sobre el formulario en el *Explorador de proyectos.*

La **Ventana de formulario** contiene todos los cuadros de diálogo utilizados en el programa se denomina en Visual Basic formularios.

En el **Explorador de proyectos** se encuentran todos los componentes del proyecto de programación actual. En el menú **Ver** dispone de una opción para activar o desactivar la vista del Explorador de proyectos.

La **Ventana de propiedades** muestra la configuración del elemento actualmente seleccionado en la ventana de proyecto. Dicha configuración se puede modificar directamente desde la ventana *Propiedades* o en el mismo código de programa. Si la ventana *Propiedades* está oculta puede activar su vista con la opción *Ver/Ventana Propiedades*.

La **Ventana Posición del formulario** se podrá definir la posición del mismo en la pantalla una vez que se ejecute el programa.

Cuando se está desarrollando un programa de un proyecto en *Visual Basic*, hay que guardarlo de vez en cuando. Esto sería lo más lógico, para así evitar perder demasiado trabajo en caso de un bloqueo del sistema o fallos en la red eléctrica.

Para **generar un programa ejecutable**, puesto que hasta ahora únicamente se podría ejecutar el programa desde el entorno de desarrollo de *Visual Basic*, hay que crear un programa que pueda funcionar sin iniciar antes la aplicación de *Visual Basic*, para ello, deberá generar un archivo ejecutable mediante la selección de la opción ***Archivo/Generar Proyecto1.exe***

Los **controles** en *Visual Basic* se encuentran en el **cuadro de herramientas**. Todos estos controles nos permiten añadirlo a nuestra ventana de objetos según nos vaya necesitando para la programación. Controles como *PictureBox, Label, TextBox, Frame, CommadButton, CheckBox, OptionButton, ComboBox, ListBox, HScrollBar, VScrollBar, Timer, DriveListBox, etc.* Para seleccionar un control sólo hay que pulsar el botón del ratón sobre él. Si desea seleccionar varios controles al mismo tiempo, mantenga presionada la tecla <**Mayús**> mientras selecciona uno tras otro los controles utilizando el ratón. Para definir y cambiar las propiedades de un control que se encuentra insertado en el formulario debemos utilizar la ventana de propiedades y en las opciones que se encuentra a la izquierda cambiamos el nombre (**Name**) por otro que nos interese, por ejemplo, si es un botón de inicio escribimos **cmdInicio** (cmd es abreviatura de command). Para cambiar el nombre que aparece en el botón nos vamos a la opción **Caption** y en la columna derecha escriba el nuevo texto: **Inicio.**

Los nombres que le demos a los elementos del programa deben de estar relacionados adecuadamente a nuestros controles y formularios. Esto lo hacemos más exclusivos y, por lo tanto, más fácil de entender su lectura en un futuro.

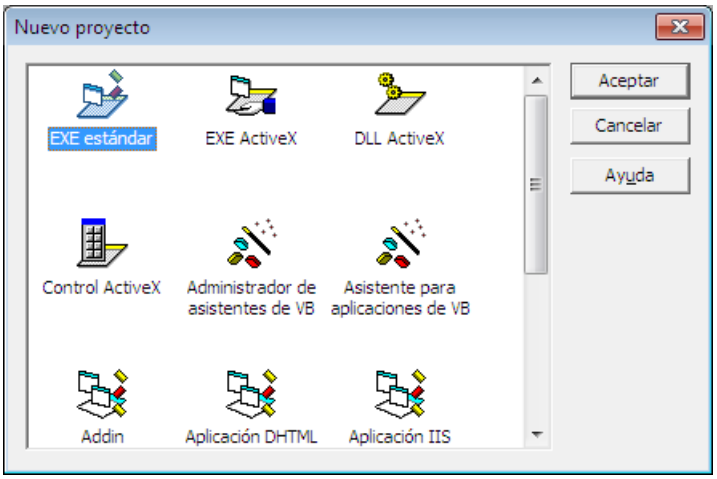
Hasta ahora en el formulario no pasa absolutamente nada. Todo lo que el programa debe hacer todavía lo tiene que definir el programador. Por tanto, deberá programar lo que va a suceder al pulsar el botón de **Inicio**. Pulsar un botón se denomina en *Visual Basic* **evento.** La parte del programa que se activa a través de un evento se conoce como procedimiento (de evento).

En un programa en *Visual Basic* puede utilizar tantos formularios como desee. Generalmente el primero de los formularios se establece como formulario de inicio, es decir, se muestra el primero después de iniciar el programa. También se puede crear subprogramas cuando en un programa principal algunas instrucciones se ejecutan varias veces o cuando necesita las mismas instrucciones en procedimientos diferentes.

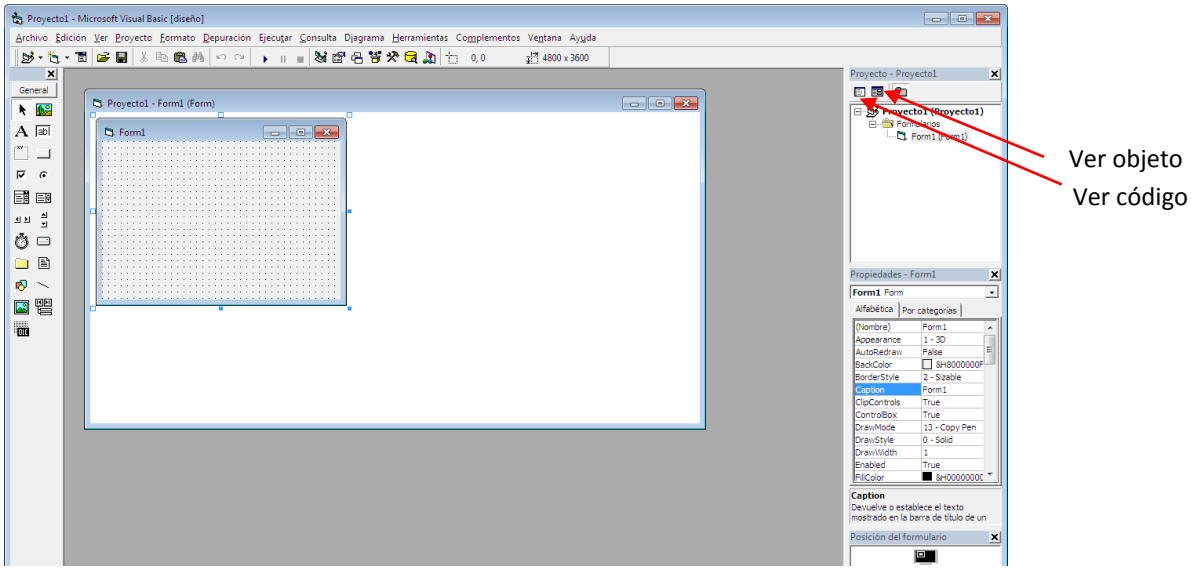
8.5.2. Programación proyecto BOTTLER

Estamos a mitad del proyecto y para ello ya sabemos de los elementos de entrada y salida E/S que tenemos que controlar.

Primeramente vamos abrir ***VB6* (*Visual Basic 6.0*)** y ejecutamos **Nuevo proyecto** y seleccionamos **EXE estándar** y pulsamos **Aceptar**.



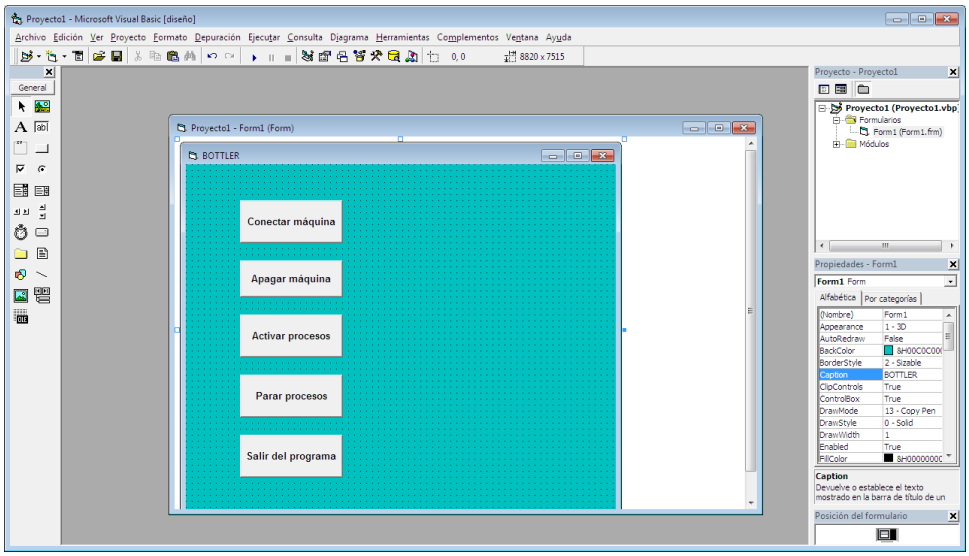
Nos aparece el **Proyecto1 (Form)** y **Form1** en este último introduciremos los elementos de la caja de herramientas que se irán añadiendo al Proyecto1 (Form).



En **Form1** podrás cambiar las dimensiones de la ventana en **Propiedades** que se encuentra en el margen derecho de la aplicación e igualmente también se podrá cambiar las propiedades con cualquier otro elemento de la caja de herramientas: *Button*, *TextBox*, *Label*, *PictureBox*, etc.

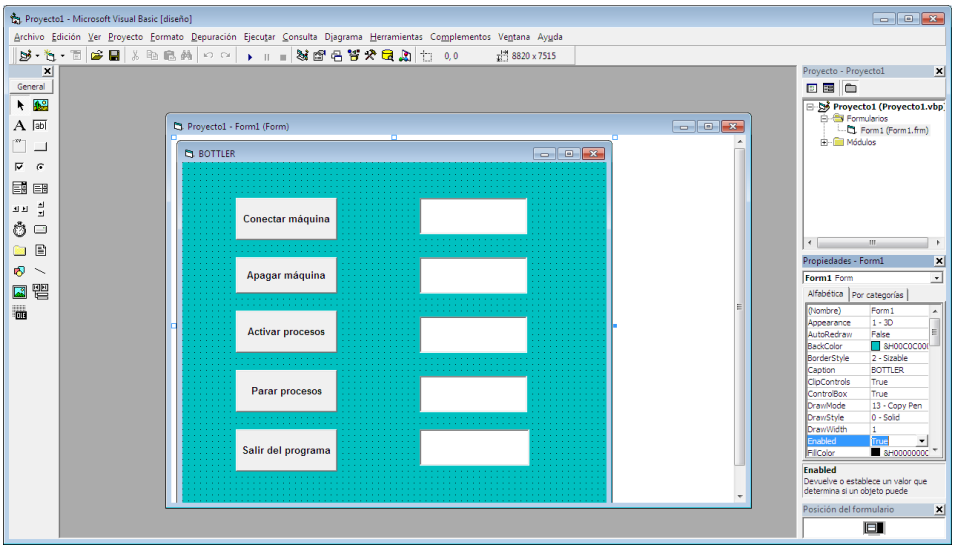
Recordemos que para nuestro proyecto necesitamos 5 controles que activemos desde el programa, para ello, crearemos **5 botones**:

- Conectar máquina
- Apagar máquina
- Activar procesos
- Detener procesos
- Salir del programa



Hemos añadido 5 botones de la caja de herramientas y se ha cambiado el tamaño, font y el nombre (**Caption**) en la ventana de propiedades. Se ha cambiado el tamaño, nombre y el color de fondo de la ventana de control a través de la ventana de **propiedades**. Lo que estamos haciendo es introduciendo los elementos de control, adornándolo un poco… todavía no hemos empezado a programar.

Seguidamente vamos a introducir los **TextBox** que son 5 cajitas de textos vacías. En ellas, irán apareciendo los mensajes de textos conforme los dispositivos detectores, pulsadores o microrruptores, se vayan activando. Realmente vamos a tener una correspondencia interactiva entre el operario y la máquina (**H**ombre/**M**áquina **H/M**).

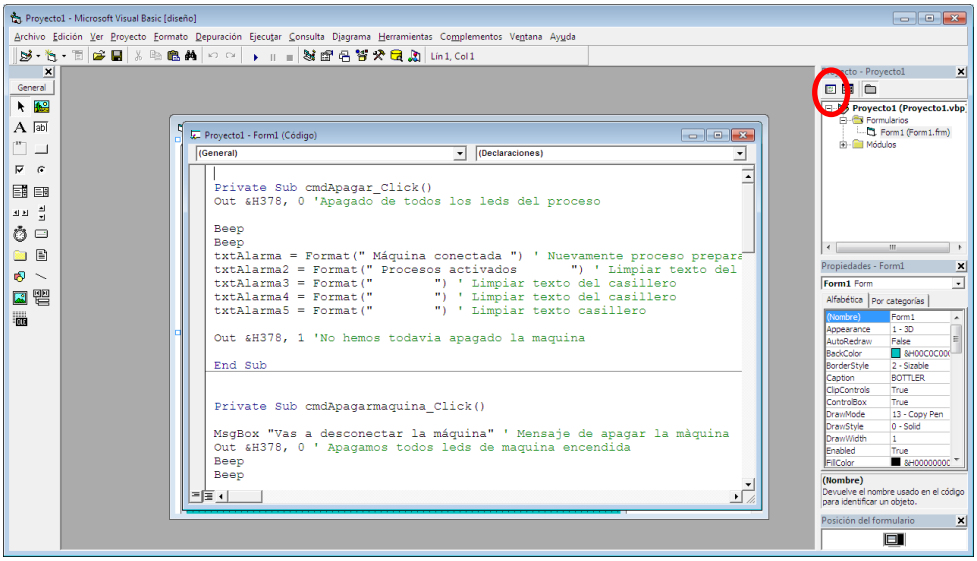


Cada vez que seleccionemos un evento, botón o caja de texto, aparece la configuración en la ventana de propiedades donde podemos modificar las propiedades de cualquier elemento del formulario: nombre, colores, tamaño, tipo de letra, etc. Es muy importante ir guardando nuestro proyecto de vez en cuando por si hacemos algo que *Visual Basic* lo desconozca y nos deje bloqueado el programa.

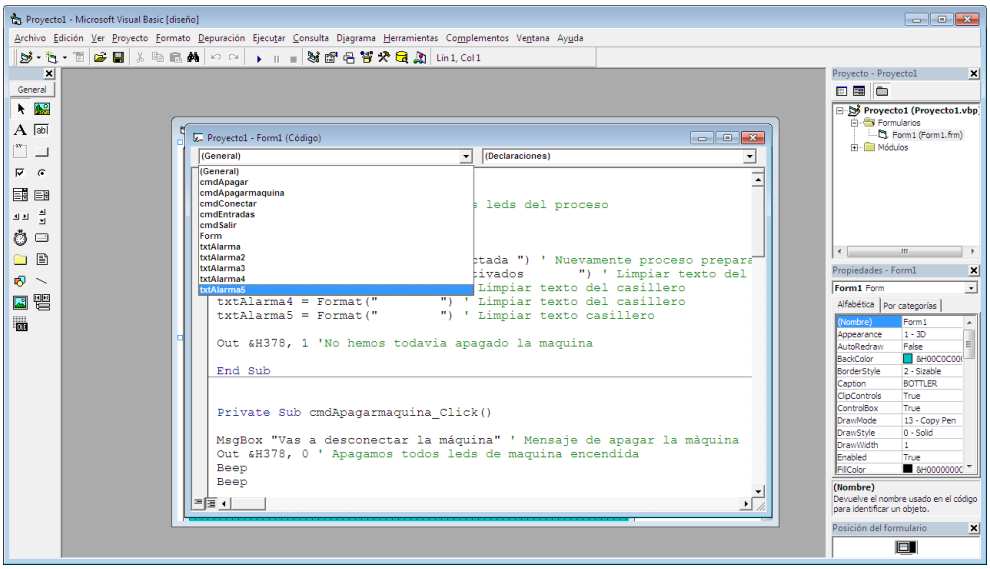
Cada uno de los controles posee sus propios eventos, de manera que cada evento tendrá su ventana de código. Una vez que tengamos todos los eventos necesarios añadidos a nuestra **Form1** pasaremos a abrir la ventana de **códigos** de programación.

Recuerda que hay dos formas de trabajar con *Visual Basic* una con la **ventana de Objetos** y otra con la **ventana de Códigos**, cuando le demos a ejecutar la programación o **F5** nos aparece la ventana de objetos para interactuar con todos los eventos programados.

Para ver y escribir nuestro código de instrucciones, debemos pulsar en la ventana del explorador de proyectos en el icono **Ver código**. De momento aparecen vacíos los objetos introducidos anteriormente. No hay más que ir añadiendo en cada objeto la instrucción que le corresponda.



En la ventana del código aparece dos opciones con alternativas: **General** y **Declaraciones**. Si pinchamos en **General** nos aparecen todos los nombres de los objetos introducidos, seleccionamos cualquiera de ellos y nos sitúa directamente en el código de ese evento.



Vamos a ir recordando algunas cosas que hemos comentado anteriormente. Recordemos que los puertos paralelos están configurados como **&h378** o **&h278** o **&h3bc** según está establecido en el PC y lo vemos en el administrador del sistema. Los terminales del puerto que nos interesa van de los terminales 2 al 9, ambos inclusive. Recuerda que el pin de menor peso es el pin 2 por lo que hay que empezar a contar por ahí.

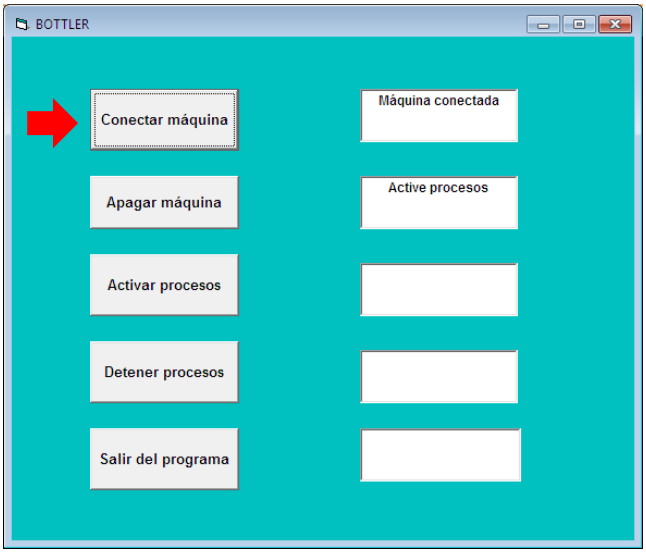
Para poder enviar y recibir datos al puerto paralelo desde ***Visual Basic*** tenemos que utilizar necesariamente un controlador, que es un archivo especialmente creado para interpretar las órdenes que le llega del sistema operativo **Windows** y por medio de la programación en ***Visual Basic***. Este archivo es fundamental y se instala en la librería DLL de Windows\System32 y se denomina **inpout32.dll** que nos permite establecer una línea de comunicación para poder usar el comando **Inp** y **Out**.

Como los pines de datos están enumerados del 2 al 9, ambos inclusive, D0, D1, D2, D3, D4, D5, D6 y D7, en nuestro caso solamente utilizaremos los **6 primeros bits** que corresponde a los datos de salida **D0, D1, D2, D3, D4** y **D5** y que activarán en cada salida un relé y un indicador luminoso.

Salidas	Pin DB25	Decimal	Proceso
D0	2	1	Conexión máquina
D1	3	3	Detección botella
D2	4	5	Llenado botella
D3	5	9	Tapón
D4	6	17	Etiquetado
D5	7	33	Embalaje

Vamos a programar el primer evento de nuestro **proyecto BOTTLER** utilizando el comando **Out** para "**Conectar máquina**", y le damos el nombre en propiedades **cmdConectar().** Su sintaxis es: **Out &H378, 1** (&H indica que el valor es hexadecimal y separado por una coma el número de pin que será usado "**pin con menor peso**", dato en decimal o binario). Por ejemplo si fuera **Out &H378, 255,** esto activaría los 8 bits de salida a nivel alto (1), pero en nuestro caso solo sería 6 bits.

```
Private Sub cmdConectar_Click()

Beep 'aviso acustico
Beep  'aviso acustivo
Out &H378, 1 'Activamos maquina

txtAviso = Format("Máquina conectada") ' se conecta la maquina
txtAviso2 = Format("Active procesos")   ' Texto Active proceso


End Sub
```

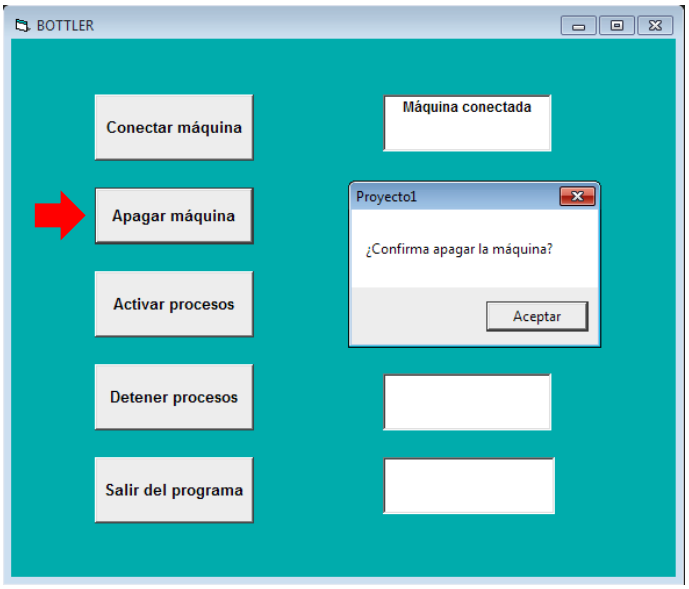


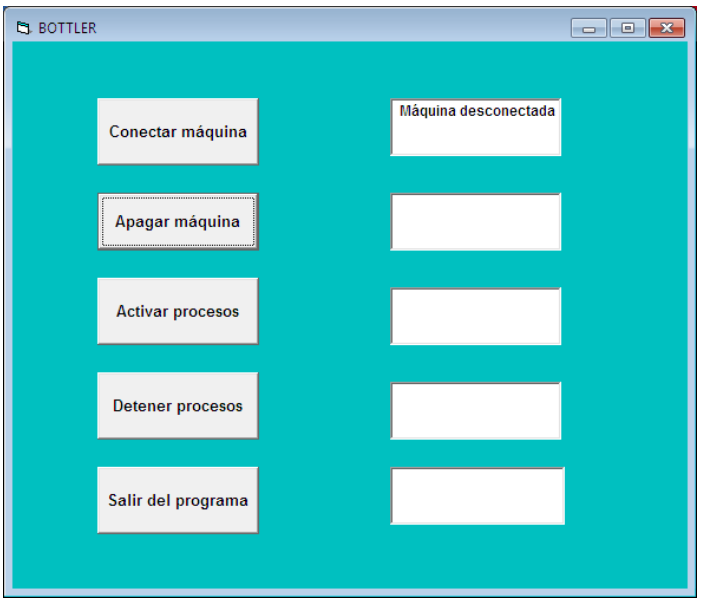
Para "**Apagar máquina**" utilizamos **Out &H378,0:**

```
Private Sub cmdApagarmaquina_Click()

MsgBox "¿Confirma apagar la máquina?" ' Mensaje de apagar la màquina
Out &H378, 0 ' Apagamos todos leds de maquina encendida
Beep ' aviso acustico
```

```
Beep ' aviso acustico
txtAviso = Format(" Máquina desconectada") ' aviso maquina apagada
txtAviso2 = Format("                ") ' Limpiar texto del casillero
txtAviso3 = Format("                ") ' Limpiar texto del casillero
txtAviso4 = Format("                ") ' Limpiar texto del casillero
txtAviso5 = Format("                ") ' Limpiar texto del casillero

End Sub
```

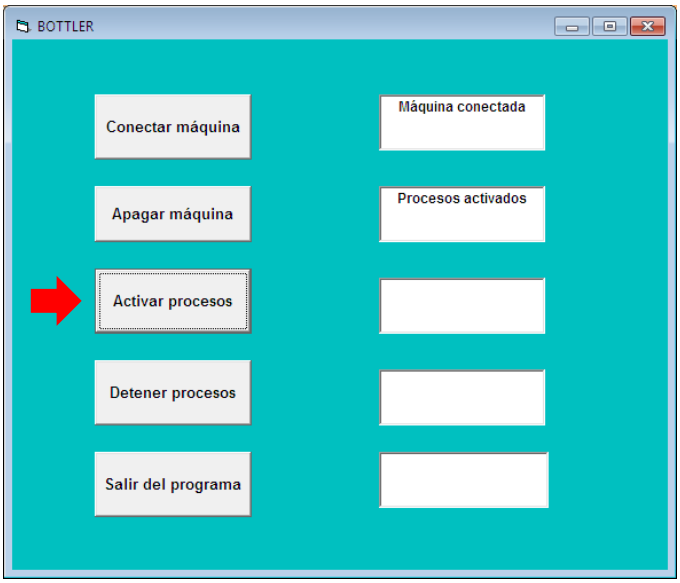
En este evento cuando pulsamos **Apagar máquina** nos aparece una ventana de confirmación pulsamos **Aceptar** y la aplicación muestra el mensaje de **Máquina desconectada**.





El siguiente evento es "**Activar procesos**". La máquina cuando se conecta a la red eléctrica nos aparece un mensaje de "Máquina conectada" y otro mensaje de que se puede "**Activar procesos**". El objeto del botón de "**Activar procesos**" es activar el grupo de pulsadores de entrada con la orden **Inp.** En ese momento el programa está a la espera de recibir esa orden para activar las señales de los pulsadores y poder procesar las operaciones programadas. Comprende el siguiente código de instrucciones:

```
Private Sub cmdEntradas_Click()

txtAviso = Format("Máquina conectada") ' Texto aviso proceso activado
txtAviso2 = Format("Procesos activados") ' aviso procesos activados
Beep ' aviso acustico
Beep ' aviso acustico
Dim entradas As Integer ' Dimensiona variable entradas como numero entero
Do While entradas < 100 ' Hace que entradas sea menor de 5
entradas = DoEvents   ' Activa Doevent para no bloquear el sistema
' entradas = entradas + 4 ' Variable contador
varRespuesta = (Inp(&H379)) ' La variable varRespuesta depende del valor
de Input
If varRespuesta = 119 Then Out &H378, 3 ' Activa proceso 1
If varRespuesta = 119 Then txtAviso4 = Format("Detección botella") ' aviso
de botella en proceso
If varRespuesta = 119 Then txtAviso3 = Format("Proceso 1 activo") ' aviso
de proceso en marcha
If varRespuesta = 119 Then txtAviso5 = Format("") ' limpiar casilla
If varRespuesta = 119 Then txtAviso2 = Format("Procesos activados") '
aviso procesos activados
If varRespuesta = 255 Then Out &H378, 5 ' Activa proceso 2
If varRespuesta = 255 Then txtAviso4 = Format("Llenado botella") ' aviso
de llenado
If varRespuesta = 255 Then txtAviso3 = Format("Proceso 2 activo") '  aviso
de proceso en marcha
If varRespuesta = 255 Then txtAviso5 = Format("") ' limpiar casilla
If varRespuesta = 255 Then txtAviso2 = Format("Procesos activados") '
aviso procesos activados
If varRespuesta = 95 Then Out &H378, 9 ' Activa proceso 3
If varRespuesta = 95 Then txtAviso4 = Format("Colocación tapon") ' aviso
colocacion tapon
If varRespuesta = 95 Then txtAviso3 = Format("Proceso 3 activo") ' aviso
de proceso en marcha
If varRespuesta = 95 Then txtAviso5 = Format("") ' limpiar casilla
If varRespuesta = 95 Then txtAviso2 = Format("Procesos activados") ' aviso
procesos activados
If varRespuesta = 111 Then Out &H378, 17 ' Activa proceso 4
If varRespuesta = 111 Then txtAviso4 = Format("Etiquetado") ' aviso de
etiquetado
If varRespuesta = 111 Then txtAviso3 = Format("Proceso 4 activo") ' aviso
de proceso en marcha
If varRespuesta = 111 Then txtAviso5 = Format("") ' limpiar casilla
If varRespuesta = 111 Then txtAviso2 = Format("Procesos activados") '
aviso procesos activados
If varRespuesta = 63 Then Out &H378, 33 'Activa proceso 5
If varRespuesta = 63 Then txtAviso4 = Format("Embalaje") ' aviso de
embalaje
If varRespuesta = 63 Then txtAviso3 = Format("Proceso 5 activo") ' aviso
proceso en marcha
If varRespuesta = 63 Then txtAviso5 = Format("Fin del Proceso") ' aviso de
botella terminada
If varRespuesta = 63 Then txtAviso2 = Format("Procesos activados") ' aviso
procesos activados
Loop ' Bucle

End Sub
```



Al **activar procesos** se conecta el grupo de entrada de datos. Para la entrada de datos usaremos los pines numerados 15, 13, 12, 10 y 11 para los cinco pulsadores de entradas: **E3, E4, E5, E6** y **E7**.

Para registrar las respuestas en los pines de las entradas **E3** a **E7** es necesario leer el valor del puerto con dirección **379h**. En este caso sería necesario ejecutar la función **inp()** repetidamente para leer el estado de las respuestas en "tiempo real" y leer las entradas directamente determinando el valor de las variables que en este caso de este proyecto, son los valores **63**, **95**, **111**, **119** y **255**. Cuando no se ha registrado ninguna respuesta el resultado de leer el valor del puerto es 127 (120 o 126 en algunos sistemas). Por ejemplo, en nuestro caso y con la siguiente instrucción, si no se ha introducido una respuesta, es decir, no se ha pulsado ningún microrruptor el valor de la variable **varRespuesta** es igual a 127.

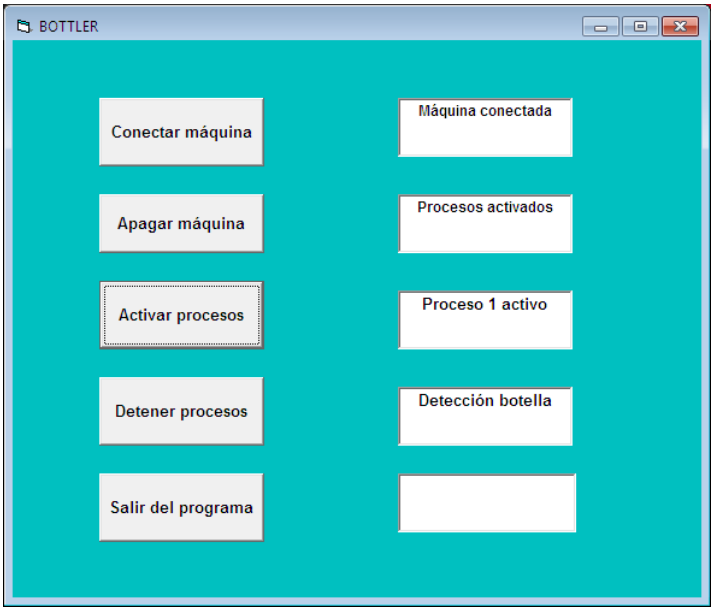
```
varRespuesta = (Inp (&H379))

print varRespuesta
```

Si se pulsa o se lleva a masa o tierra la entrada del pin **E3** se produce una respuesta apareciendo en pantalla un valor decimal que corresponde al valor de la variable **varRespuesta** de esa entrada, que es igual a **119**. Al pulsar las demás entradas **E4, E5, E6 y E7** van apareciendo también el valor decimal de cada una. Como se ha dicho anteriormente si no se pulsa ninguna entrada el valor de la variable es igual a 127.

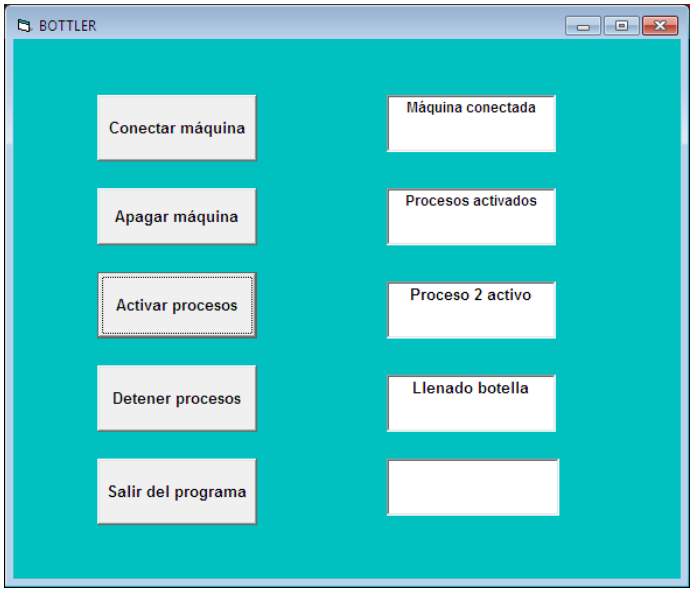
NOTA: Para leer un dato desde el puerto se usa la instrucción **Inp**. Para visualizar el valor decimal de una entrada utilizamos la variable **varRespuesta = (Inp(&H379))** y añadimos el comando **print varRespuesta** para visualizar el valor de la entrada. Cuando pulsamos una entrada, por ejemplo, E3 nos aparece en pantalla el valor decimal de esa entrada y hay que anotarla para aplicarla a nuestro programa, ver tabla.

Entrada	Salida	Pin DB25 de entrada	Pin DB25 de salida	Valor decimal entrada	Valor decimal salida	Proceso entrada	Proceso salida
E3	**D1**	15	3	119	3	CINTA	RL2
E4	**D2**	13	4	225	5	LLENADO	RL3
E5	**D3**	12	5	95	9	TAPON	RL4
E6	**D4**	10	6	111	17	ETIQUETA	RL5
E7	**D5**	11	7	63	33	EMBALAR	RL6

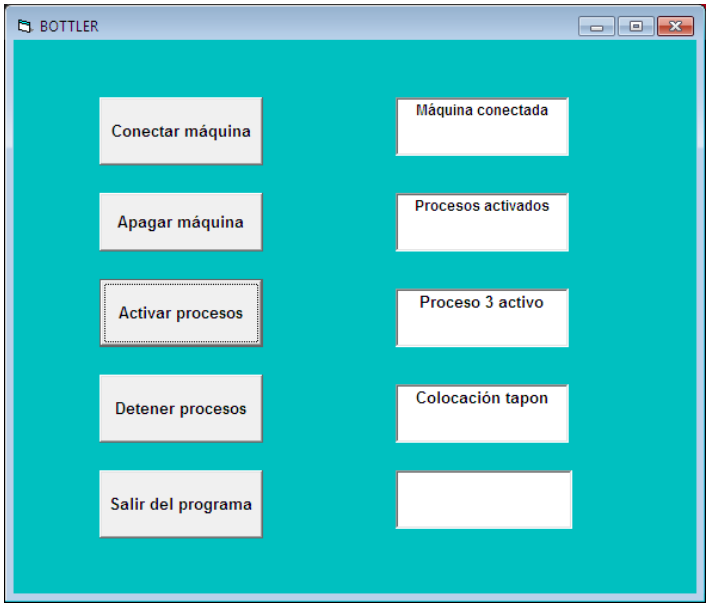
Veamos los mensajes que van apareciendo en cada una de las entradas de datos cuando se activa.



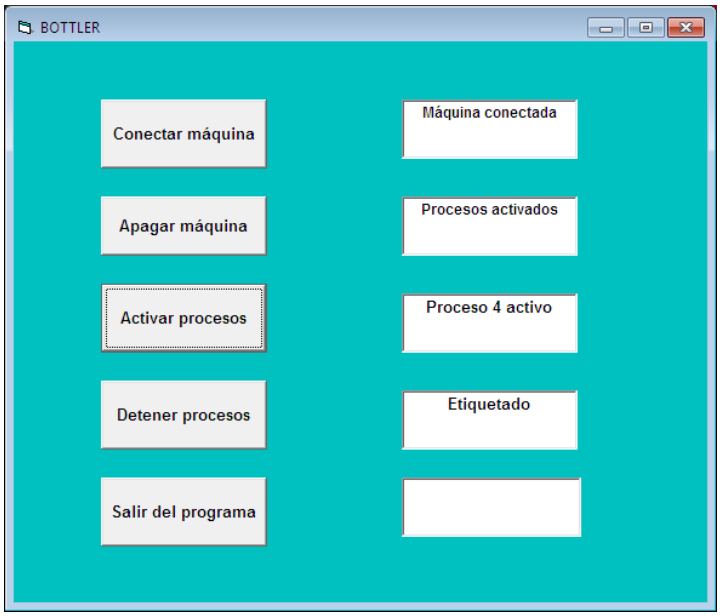
Proceso 1. E3 a masa o tierra.



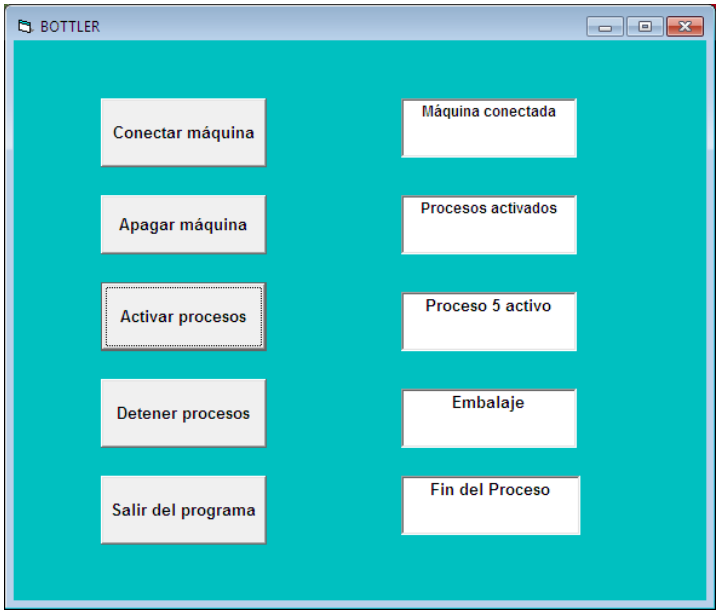
Proceso 2. E4 a masa o tierra.



Proceso 3. E5 a masa o tierra.



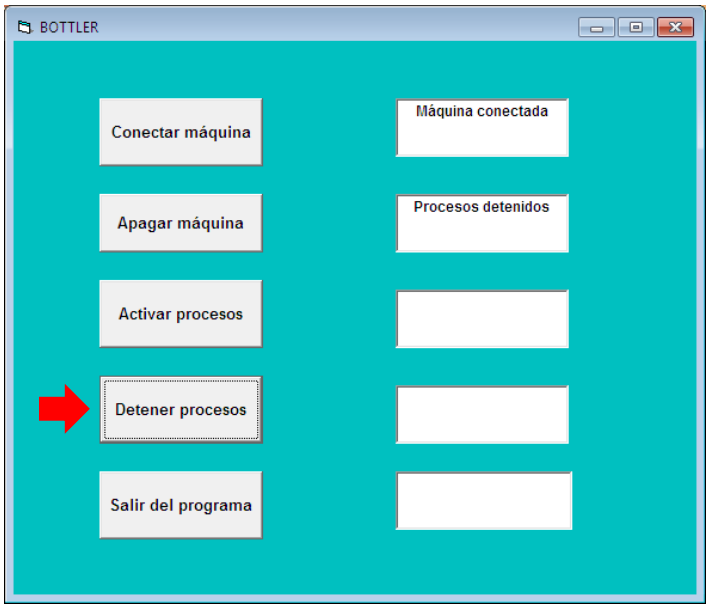
Proceso 4. E6 a masa o tierra.



Proceso 5. E7 a masa o tierra.

Para "**Detener los procesos"**:

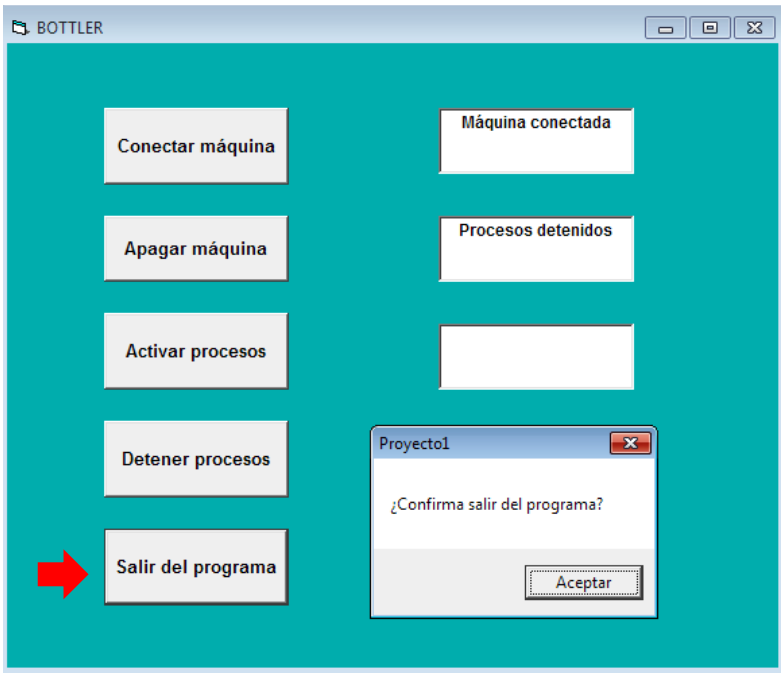
```
Private Sub cmdDetener_Click()

Out &H378, 0 'Desconectadas todas las salidas del proceso
Beep ' aviso acustico
Beep ' aviso acustico
txtAviso = Format(" Máquina conectada ") ' aviso de máquina conectada
txtAviso2 = Format(" Procesos detenidos ") ' aviso procesos detenidos
txtAviso3 = Format("        ") ' Limpiar texto del casillero
txtAviso4 = Format("        ") ' Limpiar texto del casillero
txtAviso5 = Format("        ") ' Limpiar texto casillero
Out &H378, 1 'Maquina conectada

End Sub
```



Este control nos permite desconectar los procesos que se encuentren en esos momentos activados, volviendo a ponerlos a todos desde el inicio.

El evento **salir del programa** nos permite desconectar el programa y quitarlo del sistema operativo pero **la máquina sigue conectada**, para ello, hay que volver a entrar en el programa y desconectar la máquina.

```
Private Sub cmdSalir_Click()
MsgBox "¿Confirma salir del programa?" 'Mensaje salir programa
End ' Nos salimos del programa pero los procesos siguen funcionando
End Sub
```



Al pulsar el botón salir del programa nos aparece una ventana de confirmación que al hacer clic en **Aceptar** nos salimos del programa completamente.

8.5.3. Introducir el módulo del controlador inpout32.dll

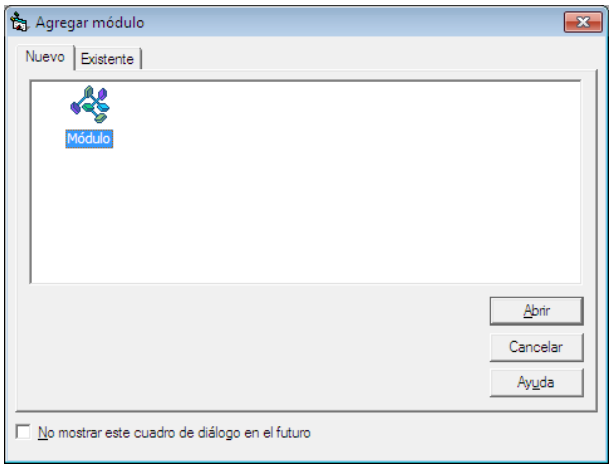
Continuamos con el desarrollo de la programación y como todos sabemos ***Visual BASIC*** es un lenguaje de alto nivel y como tal, no le es posible comunicarse directamente con el puerto paralelo. Para ello, se necesita de una librería llamada **inpout32.dll**, la cual se encarga de interpretar las órdenes que recibe del programa en *Visual Basic* y la transmite al puerto paralelo para ejecutarla. Con ello, se controla los relevadores y registrar las respuestas mediante el puerto paralelo, necesario para acceder a las funciones que están incluidas en el archivo inpout32.dll. Este archivo puede descargarse de manera gratuita en diversos sitios de internet y es de distribución libre:

http://myelectronic.eu5.org/Descargas/InpoutDLL.zip

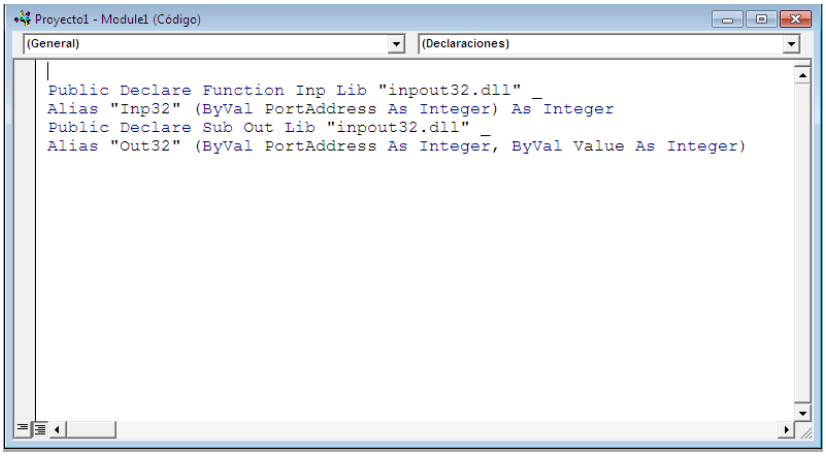
El archivo debe copiarse en la carpeta **C:\Windows\System32.**

*Copiamos y pegamos la librería **inpout32.dll** en el directorio **C:\WINDOWS\SYSTEM32**.*

Una vez copiado el archivo, debe crearse un módulo dentro de nuestro proyecto en *Visual Basic 6.0* que servirá para controlar las salidas y registrar las respuestas. En este módulo deben añadirse las siguientes líneas para habilitar las funciones del archivo inpout32.dll, para ello, seleccionamos **Proyecto** y pulsamos **Agregar módulo.**

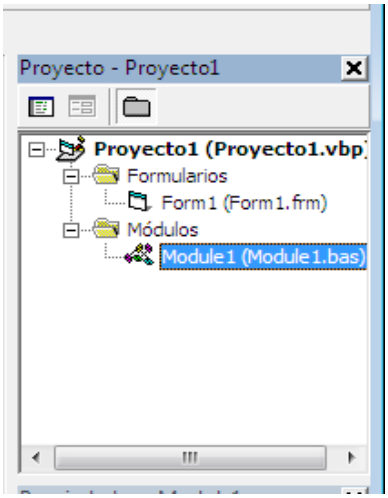


Añadimos el siguiente código:



```
Public Declare Function Inp Lib "inpout32.dll" _
Alias "Inp32" (ByVal PortAddress As Integer) As Integer
Public Declare Sub Out Lib "inpout32.dll" _
Alias "Out32" (ByVal PortAddress As Integer, ByVal Value As Integer)
```

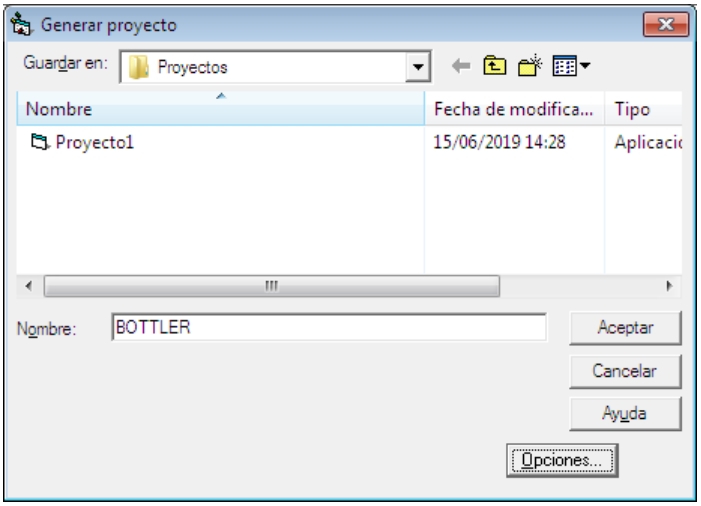
Cabe destacar que la dll debe de estar en el directorio del sistema **c:/windows/system32** o en el directorio del proyecto que hayamos creado en *Visual Basic*. Esto se quedará reflejado en la ventana del Explorador de proyectos:

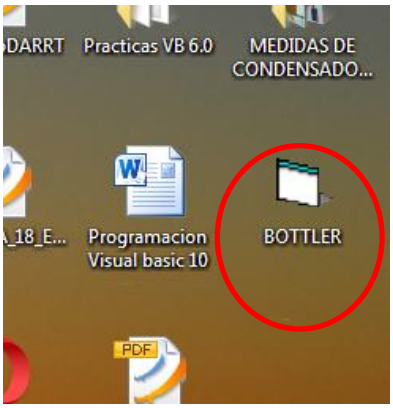


8.5.4. Convertir nuestro programa en ejecutable

A continuación crearemos nuestro programa **BOTTLER** en un programa ejecutable, sin tener que depender del entorno de trabajo de *Visual Basic*.

Cuando se crea el programa ejecutable se puede abrir desde cualquier parte sin la necesidad de tener instalado *Visual Basic*. Lo que se hace es generar un archivo ejecutable desde el mismo *Visual Basic 6.0*, seleccionando la opción **Archivo/Generar BOTTLER.exe…**





Si lo guardamos en el directorio **Proyectos**, desde aquí hacemos una copia y pega y lo llevamos al escritorio, o a una memoria USB, CD, etc. Desde aquí se podrá abrir el programa tantas veces que quieras sin tener que estar dentro del entorno de *Visual Basic*.

Es importante no olvidar que el archivo **inpout32.dll** nos lo tenemos que llevar donde vaya el programa y bajo el sistema operativo Windows, de lo contrario no funcionaría. Si el programa nos lo llevamos a otro PC con el sistema operativo Windows, tenemos que copiarlo en el directorio Windows\System32\inpout32.dll de ese ordenador.

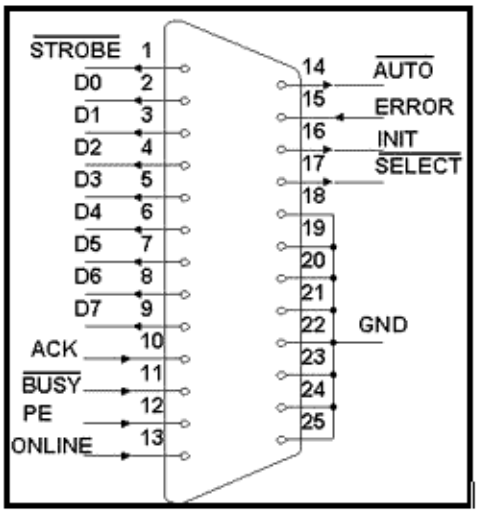
Clic aquí para ver el vídeo del proyecto BOTTLER.

9. Resumiendo.

El PC puede controlar la potencia electrónica mediante una programación que nos interesa que ejecute, simplemente lo que nos interesa es manipular y detectar elementos hardware desde el PC y con una buena programación se ponga en funcionamiento. Para ello, sería interesante tener por ejemplo, un software que te permita controlar las luces de la casa entera, o un software que te avise que puerta de la casa ha sido abierta, o un software que te ayude a regar las platas automáticamente desde el PC. Aunque la verdad, la aplicación que le des a la programación, está en los límites de la imaginación de cada uno, proponer la idea y seguidamente ponerse en marcha.

Antes que nada hay que conocer cuál es el puerto paralelo del PC. Físicamente, es un conector del tipo hembra, que consta de 25 terminales que se encuentra instalado en la parte trasera del PC y se conoce técnicamente como **DB25**. Estos 25 pines están divididos en tres "bytes" llamados **dataport**, **statusport** y **controlport**, todos orientados al manejo de una impresora que se conecta en este puerto, por ello, se le denomina **puerto impresora Centronics**, puesto que está dedicada a la gestión de la impresora, desde el control del papel hasta si la impresora está ocupada, etc...

Aquí tenemos en detalle la numeración de los pines del **conector hembra DB25** del puerto paralelo visto de frente, que se encuentra situado en la parte trasera del PC.



Dataport	Statusport	ControlPort	GND
pin 2 - D0 pin 3 - D1 pin 4 - D2 pin 5 - D3 pin 6 - D4 pin 7 - D5 pin 8 - D6 pin 9 - D7	pin 10 - ACK pin 11 - BUSY pin 12 - PAPER END pin 13 - SELECT IN pin 15 - ERROR	pin 1 - STROBE pin 14 - AUTO FEED pin 16 - INIT pin 17 - SELECT	Del18 al 25

Para saber cuál pin es cual, acércate al conector **hembra DB25** del puerto paralelo de tu PC y si te fijas cuidadosamente te darás cuenta que al lado de cada pin o terminal tiene su número, esta numeración corresponde a la parte exterior donde se conecta el **conector macho DB25**, para soldar un cable te tienes que poner por el lado inverso y la numeración del pin 1 que estaba a la derecha del conector, en la parte de la soldadura está a la izquierda, cuidado con ello.

Recordando la figura del **conector DB25** y de todos los pines que lo constituye es importante tener el concepto claro de cuales son de entradas y cuáles son de salidas. Teniendo en cuenta principalmente que:

El **Dataport** se usa como salida
El **Statusport** se usa como entrada
El **Controlport** se usa de las dos formas anteriores

Es decir que para encender leds, mandar voltaje al puerto usaremos el **Dataport**, para recibir niveles de voltaje usaremos el **Statusport** (un ejemplo del uso del statusport es para cuando en un sistema de riego que al terminar de regar, este haga un cambio de voltaje en algún dispositivo (como un relé) se detecta el cambio de cero a 5 voltios y se puede hacer "la magia" de la detección de acciones físicas desde el PC, en este caso desde *Visual Basic* para empezar es más que suficiente trabajar con estos dos bytes.

Ahora que ya se ha comentado un poco sobre el puerto, hablemos de la programación necesaria para trabajar con este puerto. Hay que saber que para trabajar con este puerto debes tener en tu pc, una librería **.dll** que gestiona toda esta programación, una puede ser la **io.dll** y otra puede ser la **inpout32.dll** en este caso usaremos la **inpout32.dll**. En realidad no cambian mucho la programación entre las dos. Puede descargar esta librería en:

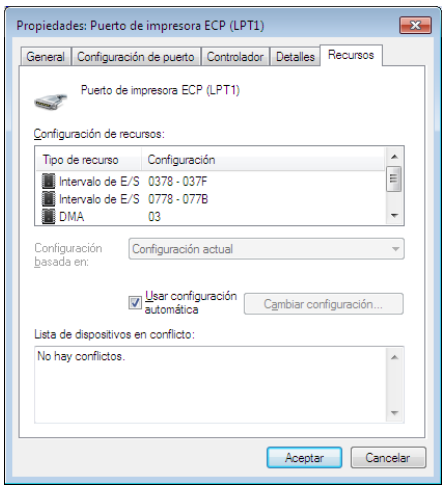
http://myelectronic.eu5.org/Descargas/InpoutDLL.zip

https://es.dll-download-system.com/dlls-i-/inpout32.dll.html

Antes de continuar se nos hará necesario saber la posición de memoria del puerto paralelo en nuestro PC o la dirección en memoria, que viene a ser lo mismo, generalmente la dirección de memoria del puerto paralelo se ve así:

378------dataport
379------statusport
37A------controlport

Si no estás seguro puedes hacer varias cosas para saber la dirección, una es durante el **power on self test** de la **BIOS** irse con pause o **F2** hasta que veas algo como el intervalo de salida/entrada del puerto paralelo. En **Windows XP** puedes irte a propiedades de Mi PC→ ficha hardware → administrador de dispositivos → puertos COM y LPT → da click en el + y luego posiciónate en el LPT → click derecho→ propiedades → dale a la ficha recursos > y ahí tendrás el intervalo entrada/salida del puerto ocuparemos primero la dirección **378** para poder enviar voltaje al puerto paralelo.



Agregamos a tu formulario lo siguiente:

- nuevo textbox
- nuevo botón

La instrucción (con la inpout32.dll) para poder enviar datos al puerto paralelo es la más lógica que pueda existir, es **Out**, su sintaxis es así: out (puerto, dato) Por ejemplo:

Out &H378, 255

La instrucción (con inpout32.dll) para poder detectar o recibir datos desde el puerto paralelo, entonces usamos la instrucción **inp**

INP(puerto%)	puerto% Un número entre 0 y 65,535 que identifica el puerto.
OUT puerto%, datos%	datos% Una expresión numérica entre 0 y 255 que será enviada al puerto

Hay que tener en cuenta también el tipo de dato que le enviaremos al puerto:

PIN	2	3	4	5	6	7	8	9
VALOR BINARIO	1	10	100	1000	10000	100000	1000000	10000000
VALOR DECIMAL	1	2	4	8	16	32	64	128

Eso sería para enviar voltaje uno por uno a cada pin del **dataport** o pines de datos que es lo mismo para poner en cero voltios a todos los pines de datos se usa el **0** (**out &H378, 0**) para poner un 1 lógico (5 voltios) en el pin D1 se usa **2** (**out &H378, 2**) y así sucesivamente hasta que para poner todos los pines de datos a nivel alto se utiliza el **255** (**out &H378, 255**).

Si se quisiera enviar un 1 lógico a varios pines de datos, únicamente debes sumar los datos: por ejemplo, para mandar datos a la vez a D1 y a D2 se usaría 2 + 4, es decir quedaría así la orden: **out &H378, 6** y así sucesivamente.

Si queremos detectar o captar desde el puerto paralelo, entonces usamos la instrucción **inp.** El siguiente ejemplo es un programa en *Visual Basic* con estos códigos.

```
Private Sub Timer2_Timer()
If Image4(1).Visible = True Then
Image4(2).Visible = True
Image4(1).Visible = False
Text2.Text = Str(Inp(Val(&H379))) '   aqui puedes notar como el textbox
cogia el valor
Text2.Refresh
If Text2.Text = 207 Then
''MsgBox "entrada 15 detectada"
Shape1.FillColor = &HFF&
End If
If Text2.Text = 71 Then
MsgBox "entrada 11 detectada"
End If
If Text2.Text = 79 Then
MsgBox "entradas 11 y  15 detectada"
End If
Exit Sub
```

```
End If
If Image4(2).Visible = True Then
Image4(3).Visible = True
Image4(2).Visible = False
Text2.Text = Str(Inp(Val(&H379)))
Text2.Refresh
If Text2.Text = 207 Then
''MsgBox "entrada 15 detectada"
Shape1.FillColor = &HFF&
End If
If Text2.Text = 71 Then
MsgBox "entrada 11 detectada"
End If
If Text2.Text = 79 Then
MsgBox "entradas 11 y  15 detectada"
End If
Exit Sub
End If
If Image4(3).Visible = True Then
Image4(1).Visible = True
Image4(3).Visible = False
Text2.Text = Str(Inp(Val(&H379)))
Text2.Refresh
If Text2.Text = 207 Then
''MsgBox "entrada 15 detectada"
Shape1.FillColor = &HFF&
End If
If Text2.Text = 71 Then
MsgBox "entrada 11 detectada"
End If
If Text2.Text = 79 Then
MsgBox "entradas 11 y  15 detectada"
End If
Exit Sub
End If
End Sub
```

El problema que se me daba a la hora de practicar la detección de "eventos" en un circuito electrónico, era que no encontraba el evento en *Visual Basic* adecuado para colocar el código que se encargaría de gestionarme todo lo que sucedería según el estado del **statusport**. Entonces se me ocurrió hacer un "**simulador de activación**" es decir como un monitoreo a la activación, por ejemplo se "encienden" foquitos en el formulario algo así:

0=apagado
1= encendido

```
100
010
001
100
010
001
100
010
001
```

Aprovechando la secuencia de cambio automático por medio del timer pude encontrar el evento que me diera la detección automática de la presencia de personas en una zona por medio de sensores conectados por medio de una interface al puerto paralelo (al **statusport**)

```
If Image4(1).Visible = True Then
Image4(2).Visible = True
Image4(1).Visible = False
```

Cuando la imagen4(1) fuera visible, la imagen 4(2) se encendía y al instante se apagaba la imagen4(1) y así sucesivamente para la detección de códigos del status me apoye de un textbox

```
Text2.Text = Str(Inp(Val(&H379)))
```

Este me cogía el valor en el **statusport** que varía según el estado de sus terminales te genera una combinación, la cual tienes que manipular según te convenga y según la conexión que hayas hecho...

```
If Text2.Text = 207 Then
''MsgBox "entrada 15 detectada"
End If
```

Si el código era 207 el **msgbox** lo dice todo se puede dar el mismo caso que al enviar datos al puerto, que mandes a dos pines a la vez en ese caso:

```
If Text2.Text = 79 Then
MsgBox "entradas 11 y  15 detectada"
End If
```

Todo es cuestión de probar uno a uno las combinaciones posibles que te va a generar lo de usar un textbox es nada más para la hora de practicar, hacer pruebas y copiar los códigos que te genera el **statusport** y si a la hora de hacer tu propia aplicación te da flojera modificarlo como es mi caso generalmente entonces nada más lo pones invisible y todo igual.

```
text2.visible = false '=P
```

Con respecto al aspecto de la seguridad y del cuido del puerto paralelo hay interfaces especiales que se encargan de echarte una mano y de cuidar a la vez el puerto paralelo que puedes dañar si no tienes cuidado.

Pues, en lo que generalmente se basa una interface para el puerto paralelo en la amplificación de salida que lastimosamente la salida directa de corriente del puerto paralelo es muy pobre, y se hace necesario agregar una interface estabilizadora con buffers o con transistores, y con otro montón de cosas que puedes usar hay muchas páginas que manejan como algo totalmente serio el uso de interfaces de protección, porque puedes quemar el puerto paralelo, pero si no vas a manejar cosas como motores, u otras cosas que exigen más, y solo quieres practicar con leds por ejemplo, no es necesario armar o comprar una interface ya que funciona perfectamente aunque siempre hay que tener el debido cuidado de no conectar mal algo. Todo esto está bien para comenzar a trabajar con esto tan interesante que es la manipulación y detección de variables físicas desde la PC.