Polibits

ISSN: 1870-9044

polibits@nlp.cic.ipn.mx

Instituto Politécnico Nacional

México

Herrera Lozada, Juan Carlos; González Robles, Juan Carlos; Cruz Contreras, Agustín

Interfaces para el Puerto Paralelo de la PC, en Modo Bidireccional

Polibits, núm. 31, 2005, pp. 9-16

Instituto Politécnico Nacional

Distrito Federal, México

Interfaces para el Puerto Paralelo de la PC, en Modo Bidireccional

M. en C. Juan Carlos Herrera Lozada,
M. en C. Juan Carlos González Robles,
Ing. Agustín Cruz Contreras;
Profesores del CIDETEC – IPN

Este es el primero de una serie de artículos enfocados al uso del puerto paralelo de la PC, para propósitos especiales y con tendencias actuales. En esta primera entrega se explica de manera concreta, cómo manejar el puerto de impresión de la PC en modo bidireccional, con la intención de interactuar con un hardware externo. Se aportan soluciones mínimas en software bajo Windows en todas sus versiones, incluyendo Windows XP. La metodología expuesta para configurar el puerto se hace extensiva para aplicaciones diversas con requerimientos similares.

Introducción

El puerto paralelo se apega al estándar IEEE 1284, liberado en 1994 y que define 4 modos de operación soportados aún en la actualidad:

1. Puerto Paralelo Estándar (SPP)
2. Puerto Paralelo PS/2 (Bidireccional)
3. Puerto Paralelo Mejorado (EPP)
4. Puerto Paralelo con Capacidades Extendidas (ECP).

La mayoría de las computadoras personales recientes, tanto de escritorio como portátiles[1], presentan por omisión una configuración del puerto paralelo en dos direcciones de datos (bidireccional) para cualquier sistema operativo. Los sistemas operativos menos recientes, Windows 98 y anteriores, también son capaces de soportar este tipo de esquema para recibir y enviar datos por el puerto de impresión, siempre y cuando se configure manualmente dicha característica, preferentemente desde el *SETUP*.

El motivo central de este artículo es discutir el modo de operación bidireccional. Para el análisis mostrado se consideran dos vertientes: la programación del puerto bajo el modo MS –DOS (*Micrososft Disk Operating System – Sistema Operativo en Disco*), y como segunda derivación, la programación en Modo Windows. En ambos casos se revisan interfaces unidireccionales y bidireccionales.

Programación del Puerto Paralelo en Modo MS - DOS

El modo MS- DOS es válido en Windows 98 y versiones anteriores (95, 3.1, etc.). En esta condición es posible escribir directamente a los registros del puerto.

Puerto Paralelo Unidireccional

Considerando el modo de una sola dirección, comúnmente llamado Puerto Paralelo Estándar (*SPP*), existen tres direcciones consecutivas asociadas con un puerto paralelo; estas direcciones pertenecen al registro de datos (*Data Register*), el registro de estado (*Status Register*) y el registro de control (*Control Register*). Se le denomina dirección base a la que indica la propia del registro de datos, por lo general 0x378; así se tendría para el registro de estado la dirección inmediata siguiente 0x379 y para el registro de control la dirección 0x37A

Existen alternativas diferentes para encontrar la dirección de los puertos, dado que ésta puede cambiar dependiendo de la arquitectura y organización interna de la PC. Es posible acceder directamente al *panel de control* de Windows y verificar el *sistema;* dentro de los recursos *hardware* se encuentra el *administrador de dipositivos*. El puerto paralelo se utiliza para la conexión de impresoras, por lo que aparecen etiquetados como LPT1, LPT2, LPT3 ó LPT4, según las características de la PC.

[1] En las portátiles, se tiende a encaminar todos los periféricos hacia USB, eliminando en algunos casos el puerto paralelo y el puerto de juegos; así como la unidad de disco flexible, obligando a que ésta sea externa, o bien, utilizar la alternativa del disco compacto.

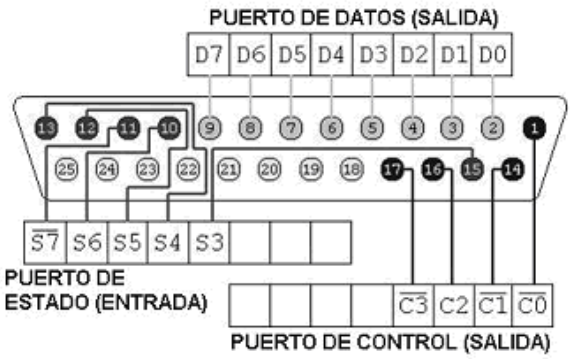

Figura 1.a. Puerto Paralelo Estándar, disposición de terminales del conector DB25.

Es importante recordar que en el modo estándar, el puerto de datos sólo es de salida, de ahí que se le conozca como *unidireccional*, y es de 8 bits. El puerto de estado es de sólo entrada con 5 bits referidos en el conector y el propio de control tiene 4 bits de sólo salida. En resumen, bajo este modo se tienen 12 líneas de salida (de las cuales, 3 son de tipo activo bajo) y sólo 5 de entrada (con una sola línea de tipo activo bajo), tal y como se aprecia en la **Figura 1.a**.

El puerto paralelo utiliza un conector hembra clase *D* de 25 terminales (DB-25), definido como *TIPO A* por el estándar IEEE 1284 (obsérvese la **Figura 1.b**). Este conector es el de interes en este artículo.

En las **Figuras 1.a** y **1.b**, se aprecia la distribución física de los pines en el conector DB-25. Para fines de análisis, se considera que los tres registros del puerto son de 8 bits, por lo que se tiene un orden significativo que es necesario respetar cuando se forma una palabra de configuración; por ejemplo, en el caso del registro de estado, se tiene disponible a partir del bit 4 y hasta el bit 8 (S7, S6, S5, S4, S3), los demás están comprometidos o reservados para otros propósitos. De acuerdo al diagrama interno aproximado mostrado en la **Figura 1.b**, el bit más significativo del registro de estado (S7), trabaja con lógica negativa y está físicamente ubicado en la terminal 11 del conector. Si se requiere leer una palabra de entrada a través de este registro, es importante considerar con qué lógica funciona cada línea.

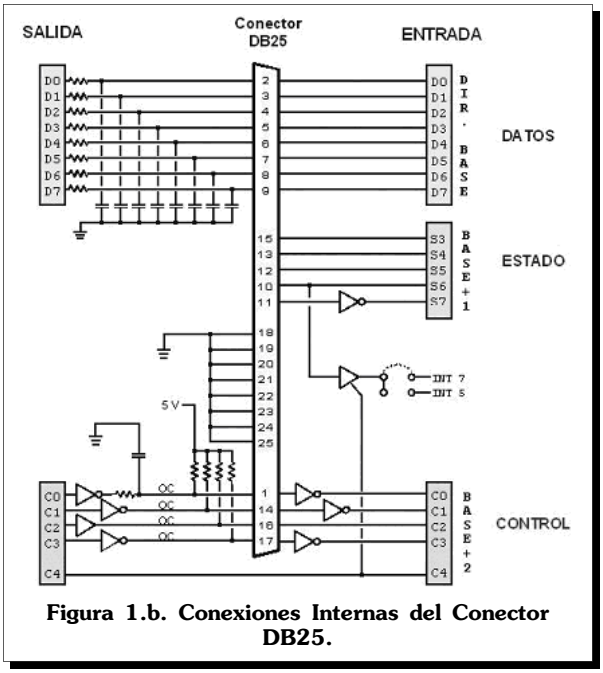

Figura 1.b. Conexiones Internas del Conector DB25.

El registro de control, para fines prácticos en el diseño de interfaces en modo estándar, es sólo de salida y utiliza los primeros cuatro bits (C3, C2, C1, C0) del registro, los restantes cuatro están reservados. En este registro, los bits C3, C1 y C0, trabajan con lógica invertida y están localizados físicamente en las terminales 17, 14 y 1 del conector.

Como ya se comentó, los ocho bits del registro de datos se utilizan sólo como salidas y todos trabajan con lógica positiva, ubicándose en orden significativo de la línea 2 a la 9 del conector, tal y como se aprecia en la **Figura 1.a**.

El siguiente programa escrito en Lenguaje C, muestra de manera sencilla cómo se envían y reciben datos. El circuito secuenciador de la **Figura 2** se activa con el botón externo conectado al bit 4 del registro de estado (*S3*), ubicado físicamente en la terminal 15 del conector.

```
/* {jcrls, acruz, jgrobles}@ipn.mx */

/*Se incluyen las bibliotecas generales, No se utilizan todas en este
ejemplo*/

#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include <dos.h>

void main(void)
{
int entrada;
int leds[9] = {0,1,2,4,8,16,32,64,128};   /*Datos de la secuencia, en
                                            decimal*/
int i;

while(1)
 {
 entrada = inportb(0x379);
 if  (((entrada)&0x08)==0)   /*Si el bit S3 está en 0, el botón está
                             presionado*/

 {
```

```
for(i = 0; i < 9; i++)
{
  outport(0x378,leds[i]);
     /*Recorre uno a uno los bits de izquierda
     a derecha*/
  sleep(1);
}
}
else
 outportb(0x378,0x00);
     /*Si el botón no es presionado, los LED
     se apagan*/
 }
}
```

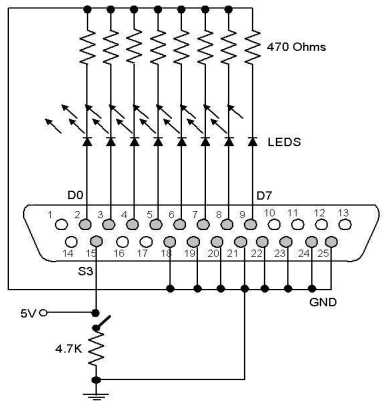

Figura 2. Circuito secuenciador en modo estándar.

Las conexiones básicas requieren de las terminales 2, 3, 4, 5, 6, 7, 8 y 9 del conector, D0, D1, D2, D3, D4, D5, D6 Y D7 del registro de datos, por lo que se conectarán leds tal y como lo indica el diagrama de la **Figura 2**. Para la conexión del botón de activación se utilizará la línea 15 del conector. Se recomienda utilizar una fuente externa de 5 volts y acoplar las tierras.

Como ya se indicó, este programa sólo funciona en modo MS- DOS. Para que funcione también sobre sistemas operativos superiores a Windows 98, es necesaria una biblioteca de enlace dinámico (dll) que declare nuevas funciones para acceder al puerto paralelo, esta cuestión se detallará más adelante en este mismo documento.

Puerto Paralelo Bidireccional

En algunas aplicaciones prácticas se requieren más líneas de entrada que las disponibles en el puerto de estado; por ejemplo, leer los 8 bits de un convertidor analógico – digital paralelo o interactuar con una pantalla de LCD. Dado que las entradas del puerto de estado están restringidas a sólo 5 bits, es necesario adecuar el programa escrito hacia una lógica de multiplexaje que lea un dato de 4 bits, se almacene en una localidad de memoria y después de un tiempo se lea la otra parte del dato, pensando en una entrada de 8 bits ó más.

Es admisible configurar el puerto de datos para que sus ocho terminales puedan ser también entradas. Esto se logra accediendo al puerto de control y cambiando el bit número 6 del registro de un estado natural bajo a un estado alto. Cuando C5 está a *0* lógico, las 8 líneas del puerto de datos son salidas y cuando C5 está a *1* lógico, se comportan como entradas. La **Figura 3**, muestra la disposición física de los pines del registro del puerto de control.

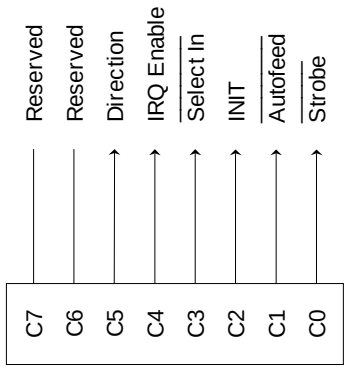

Figura 3. Puerto de Control

Considerando una dirección base 378H para el puerto de datos, se lista el siguiente fragmento en lenguaje C para explicar de manera más concreta la idea anterior.

```
{
  unsigned int Valor, temp;
  outportb(0x37A,  0x20);
  Valor=inport(0x378);
  printf («Valor Leído: %u \n», Valor);
  getch();
}
```

Obsérvese que en la primera instrucción outportb (0x37A, 0x20) se escribe al puerto de control con la dirección 0x37A un valor hexadecimal 0x20, traducido a binario de 8 bits como 00100000, especificando que el bit número 6 se establece a un nivel lógico alto, por lo que el puerto de datos (0x378) está configurado como entrada. En la siguiente instrucción Valor=inport(0x378) se leen las 8 líneas de datos y se asignan a una variable sin signo previamente definida.

Como ejemplo práctico, se considera un ADC0804 supervisado a través del puerto paralelo en modo bidireccional, como lo expone el diagrama de la **Figura 4**. En este artículo no se expone a detalle el funcionamiento del convertidor por lo que se recomienda consultar la hoja de especificaciones del dispositivo. Para este diseño en particular, la entrada analógica a convertir proviene directamente de una resistencia variable; sin embargo, ésta se puede sustituir (con las adecuaciones necesarias) por algún sensor, por ejemplo, un dispositivo *LM35* para concebir un termómetro digital.

Para las señales de control *WR* y *RD*, se consideran los bits 2 y 0, respectivamente del puerto de control (0x37A).

Como ya se mencionó, C5 es el bit que permite configurar el puerto de datos como entrada o salida, por lo que no tiene una línea exterior.

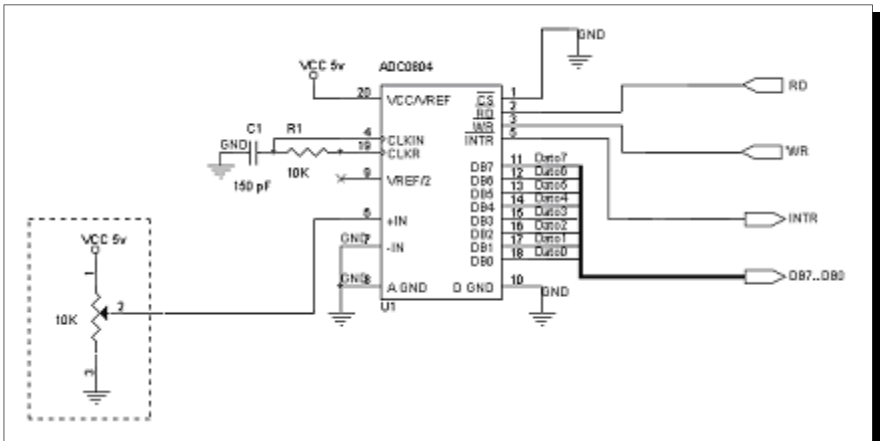

Figura 4. Diagrama del convertidor conectado al puerto paralelo.

En el programa en lenguaje C listado a continuación, se decidió utilizar C2 para *Write* y C0 para *Read* en el ADC. Físicamente, y con referencia a la **Figura 3**, C2 está asignado a la terminal 16 del conector y C0 a la línea 1 del mismo. Nótese que el hardware interno del puerto de control especifica que la línea 1 del conector tiene lógica negativa, por lo que se debe considerar este aspecto en el momento de generar las señales de escritura y lectura.

Para leer el bit correspondiente a la señal INTR que indica el fin de la conversión del ADC, se utiliza el bit S3 del puerto de estado (0x379), ubicado en la terminal 15 del conector.

Para capturar un dato convertido, se utilizan los 8 bits del puerto de datos (0x378) conectados de manera tradicional, del menos significativo al más significativo; en otras palabras, DB0 corresponderá al D0 del puerto (terminal 2), DB1 a D1 (terminal 3) y así, sucesivamente, hasta DB7 que corresponderá a D7 (terminal 9).

Al principio del programa listado se solicita el número de muestras a convertir (cuántas veces se repetirá el programa). Es una manera sencilla de terminar el programa después de "n" lecturas. También es posible asignar una tecla para salir. Se recomiendan pocas muestras cuando el retardo es de un valor alto.

```
/* ADC0804 por el puerto paralelo, {jcrls, acruz, jgrobles}@ipn.mx
8-bits utilizando modo bidireccional (ECP).*/

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

main()

{
unsigned int pdatos, pestado, pcontrol, temp, dato_leido;
int contador, muestra;
pdatos=0x378;
pestado=pdatos+1;  pcontrol=pdatos+2;
temp=inportb(pcontrol); /*permite restaurar el valor original del puerto*/

clrscr();
 /* «muestra» indica el número de lecturas que deseas realizar*/
 for (muestra=0; muestra<50; muestra++)
 {
 /* Comienza la conversión poniendo en bajo «write» y manteniendo C5 en alto para la bidireccionalidad.
Posteriormente pondremos en alto «write» para deshabilitarlo; así se genera el pulso negativo para accionar «write»*/
  outportb(pcontrol, 0x20);
  delay(10000);
  outportb(pcontrol, 0x24);
/* Espera hasta que la conversión haya concluido recibiendo el bit INTR proveniente del ADC*/
/* «contador» permite terminar el programa después de 256 ciclos si no se ha conectado el ADC al puerto*/
 contador=0;
 do { contador++; }
 while (((inportb(pestado) & 0x08)==0) && (contador!=256));
 if (contador==256) printf («No hay convertidor!!!\n»);
 else
 {
 delay(10000); /*Es posible omitir el retardo*/
  outportb(pcontrol, 0x25);
  delay(10000);
  dato_leido=inportb(pdatos);
  outportb(pcontrol, 0x24);
   clrscr();
  printf («ADC value:%i»,dato_leido);
  getch(); /* Únicamente espera por una tecla, una vez que concluyó el programa*/
 }
 }
outportb(pcontrol, temp); /*restaura puerto de control a su valor original*/
return 0;
}
```

De igual forma que en el modo unidireccional, este código funciona correctamente sólo en modo MS-DOS.

Programación del Puerto Paralelo en Modo Windows

Windows NT, 2000 y *XP* no permiten manejar el puerto paralelo en modo MS-DOS, como sucedía con las versiones anteriores de este mismo sistema operativo. Para solucionar este inconveniente, y dar acceso al puerto en cualquier versión de Windows (en modo Windows) se requieren *bibliotecas de enlace dinámico* (dll) que gestionan ante el sistema operativo el manejo del puerto.

Una dll se diseña, por lo general, en C++ y se utiliza en lenguajes de alto nivel como el mismo C++, Delphi o Java, entre otros. En este artículo se utiliza una dll *freeware* (uso libre, no comercial) para crear interfaces en *Visual Basic 6.0*.

Ninguno de los compiladores para Windows ha incluido un componente para el manejo de puertos en forma general sólo para comunicaciones e impresión. La ventaja que tiene Visual.

Basic sobre otros lenguajes es la facilidad del entorno visual. La biblioteca *io.dll* se descarga gratuitamente de *http://www.geekhideout.com* y se copia al directorio ubicado en C:/Windows/System32/. En Internet existen disponibles otras bibliotecas similares; la que aquí se indica es la que se ha probado con éxito en todos los diseños presentados.

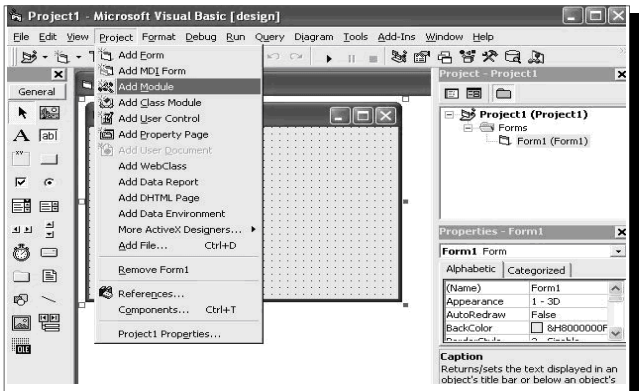

Figura 5. Pantalla de diseño en Visual Basic 6.0, adicionando un módulo.

El siguiente paso consiste en crear un proyecto estándar en Visual Basic y adicionar un módulo que incluya los prototipos de la dll para Visual Basic, listados en la misma página web de donde se descargó dicha dll (referirse a la **Tabla 1**).

Lo anterior se logra activando la opción del menú *Proyecto* y seleccionando *Adicionar Módulo*. En el espacio de edición del mismo módulo se escriben los prototipos de la **Tabla 1**. Para salvar el proyecto completo se requiere nombrar primeramente la *Forma*, después el *Proyecto* y finalmente el *Módulo*. Obsérvese la pantalla de la **Figura 5**.

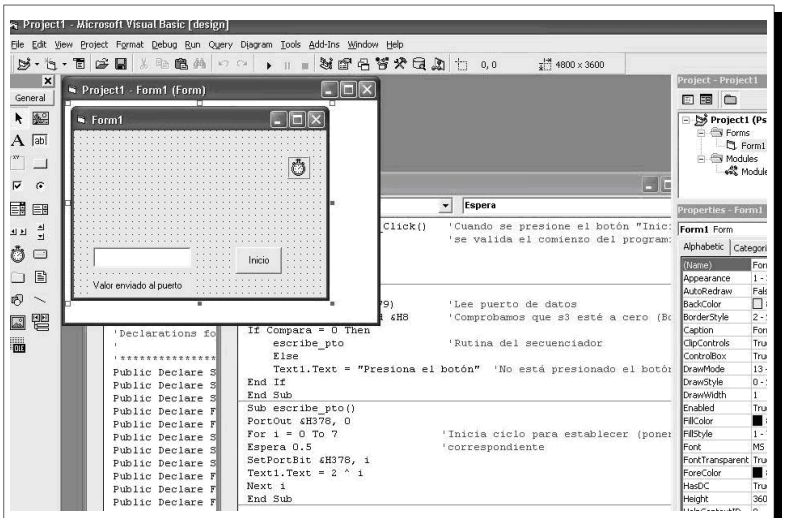

Figura 6. Pantalla en tiempo de diseño para el secuenciador.

Básicamente y en correspondencia con los prototipos de la dll, se tienen 13 funciones para manejo del puerto paralelo; nos enfocaremos sólo a cuatro: *PortOut*, *PortIn*, *SetPortBit* y *ClrPortBit*. Las dos primeras permiten enviar y recibir respectivamente un byte por el puerto indicado. *SetPortBit* y *ClrPortBit*, manipulan sólo un bit del registro, ya sea para establecerlo o para limpiarlo.

Programación Unidireccional en Visual Basic

Considerando el mismo ejemplo en modo estándar propuesto por el diagrama de la **Figura 2**; en Visual Basic no se tienen las directivas *Sleep* ni *Delay*, propias de C, por lo que para temporizar se requiere un *Timer* sincronizado en milisegundos. La pantalla mostrada en la **Figura 6**, es la que se utilizó para concretar el *secuenciador* de ejemplo.

```
Public Declare Sub PortOut Lib "IO.DLL" (ByVal Port As Integer, ByVal Data As Byte)
Public Declare Sub PortWordOut Lib "IO.DLL" (ByVal Port As Integer, ByVal Data As Integer)
Public Declare Sub PortDWordOut Lib "IO.DLL" (ByVal Port As Integer, ByVal Data As Long)
Public Declare Function PortIn Lib "IO.DLL" (ByVal Port As Integer) As Byte
Public Declare Function PortWordIn Lib "IO.DLL" (ByVal Port As Integer) As Integer
Public Declare Function PortDWordIn Lib "IO.DLL" (ByVal Port As Integer) As Long
Public Declare Sub SetPortBit Lib "IO.DLL" (ByVal Port As Integer, ByVal Bit As Byte)
Public Declare Sub ClrPortBit Lib "IO.DLL" (ByVal Port As Integer, ByVal Bit As Byte)
Public Declare Sub NotPortBit Lib "IO.DLL" (ByVal Port As Integer, ByVal Bit As Byte)
Public Declare Function GetPortBit Lib "IO.DLL" (ByVal Port As Integer, ByVal Bit As Byte) As
Boolean
Public Declare Function RightPortShift Lib "IO.DLL" (ByVal Port As Integer, ByVal Val As Boolean) As
Boolean
Public Declare Function LeftPortShift Lib "IO.DLL" (ByVal Port As Integer, ByVal Val As Boolean) As
Boolean
Public Declare Function IsDriverInstalled Lib "IO.DLL" () As Boolean
```

Tabla 1. Prototipos de io.dll a escribirse en un módulo de Visual Basic.

Una vez creado el proyecto, y después de haber adicionado el módulo de declaraciones de la dll, se procede a colocar una caja de texto (*TextBox*) que por omisión llevará el nombre *Text1*.

Después se colocará debajo de la caja una etiqueta (*Label*) que por default está referida como *Label1*. Dentro de las propiedades de este

objeto se debe buscar la que se refiere a *Caption* (Texto de etiqueta) y escribir "Valor enviado al puerto".

Se coloca un botón de acción (*CommandBotton*) que por omisión se llamará *Command1*. Dentro de sus propiedades, y en *Caption*, se escribe "Inicio".

Por último, se inserta un *Timer*, con el nombre *Timer1*. Este objeto sólo se visualiza en tiempo de diseño, es decir, en la aplicación ejecutable no se verá.

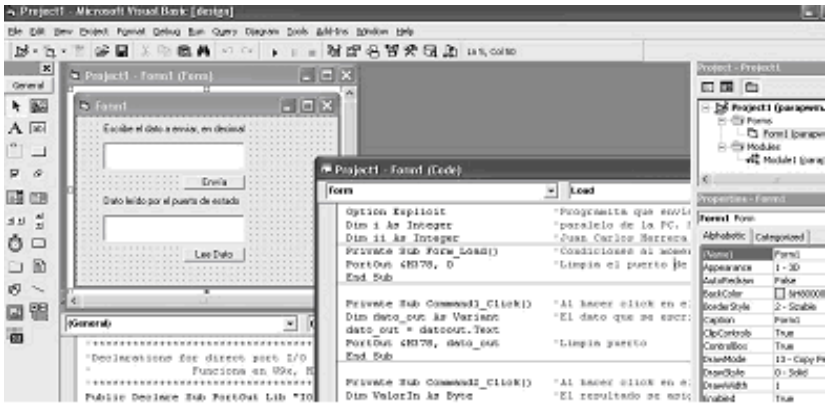

Figura 7. Interfaz unidireccional de propósito general.

En el editor de la forma (*View Code*) se copia el código siguiente y se presiona el botón *Run* para ejecutar la aplicación.

```
'{jcrls; acruz; jgrobles}@ipn.mx
Option Explicit   'Programa que envía y recibe datos del puerto
                  paralelo de la PC. Se utiliza io.dll.'
Dim i As Integer
Dim ValorIn As Byte
Dim Compara As Byte

Private Sub Form_Load()
'no hagas nada, cuando se abra la aplicación'
End Sub

Private Sub Command1_Click() 'Cuando se presione el
                                botón «Inicia»se vali
                                da el comienzo del
                                programa'
While (1)
Comienza
Wend
End Sub

Sub Comienza()
ValorIn = PortIn(&H379)   'Lee puerto de datos'
Compara = ValorIn And &H8 'Comprobamos que s3 esté
                            a cero (Botón presionado)'
If Compara = 0 Then
 escribe_pto 'Rutina del secuenciador'
 Else
 Text1.Text = «Presiona el botón» 'No está presionado el botón'
End If
End Sub

Sub  escribe_pto()
PortOut &H378, 0
For i = 0 To 7 'Inicia ciclo para establecer (poner a 1) el bit
                'correspondiente'
Espera 0.5
SetPortBit &H378, i
Text1.Text = 2 ^ i
Next i
End Sub

Sub Espera(t As Double) 'Rutina de tiempo'
Timer1.Interval = t * 1000 'Milisegundos'
Timer1.Enabled = True 'Habilitamos timer'
Do While Timer1.Enabled
 DoEvents
Loop
End Sub

Private Sub Timer1_Timer() 'Habilitación del timer1, relacionado con
                            la rutina Espera'
Timer1.Enabled = False
End Sub

Private Sub Form_OKClick() 'Salir de la aplicación'
 App.End
End Sub
```

El siguiente código, diseñado para una interfaz en modo unidireccional, permite enviar un dato escrito en la caja de texto correspondiente a "Enviar dato" directamente al puerto de datos (0x378). También es posible leer un dato del puerto de estado (0x379) en la caja de texto correspondiente a "Leer dato". La pantalla en tiempo de diseño se muestra en la **Figura 7**.

```
Option Explicit 'Programa que envía y recibe datos del puerto
                {jcrls; acruz; jgrobles}@ipn.mx'

Private Sub Form_Load() 'Condiciones al momento de abrir la
                        aplicación'
PortOut &H378, 0        'Limpia el puerto de datos'
End Sub

Private Sub Command1_Click() 'Al hacer click en el botón Command1
                        («Envía») El dato que se escriba en
                        la caja de texto, en decimal'
Dim dato_out As Variant
dato_out = datoout.Text
PortOut &H378, dato_out      'Limpia puerto'
End Sub

Private Sub Command2_Click() 'Al hacer click en el botón Command2
                        («Lee Puerto») El resultado se asig
                        nará a la variable ValorIn'
Dim ValorIn As Byte
ValorIn = PortIn(&H379)
dato_in.Text = ValorIn & « ,          'está en decimal»
End Sub
Private Sub Form_OKClick()            'Salir de la aplicación'
 App.End
End Sub
```

Programación Bidireccional en Visual Basic

Para la programación bidireccional del puerto de datos se sigue la misma lógica descrita en los programas en Lenguaje C anteriores, es decir, se establece el bit 6 (C5) del puerto de control para que el registro de datos acepte un byte proveniente del exterior y se limpia el mismo bit si se desea que el registro de datos sea sólo de salida.

Retomando la interfaz para monitorear el ADC0804, algunos autores desestiman utilizar todas las señales del ADC. Con una frecuencia sustentable, mucho menor que los 8KHz naturales del dispositivo, es posible obtener circuitos más simples como el denominado *Free Running* que propone utilizar un inversor hacia *RD* de la señal *WR*, obligando a que *RD* siempre sea el complemento de *WR* y viceversa (ver **Figura 8**). El inversor se puede implementar dentro de código, y dado que la frecuencia de trabajo para reportar lecturas se considera lenta, es posible omitir la espera de la respuesta de la señal INTR. Se recomienda reestablecer el puerto de control a su estado original, antes de salir de la aplicación.

La preparación del puerto desde el *SETUP* es fundamental para que funcione correctamente este programa.

El siguiente código en Visual Basic muestra la solución propuesta. Obsérvese que sólo se envía la señal de reloj por el puerto de control, a la vez que se reciben de forma paralela los 8 bits del ADC por el puerto de datos de la PC.

```
'{jcrls; acruz; jgrobles}@ipn.mx

Option Explicit 'Programa que recibe datos de un ADC0804
Dim DatoPuerto As Byte 'con conexiones mínimas Free Running
Dim Temporal As Byte 'Modo del puerto paralelo: ECP (Bidireccional)

Private Sub Form_Load() 'Limpia puerto de datos al cargar la aplicación
PortOut &H378, 0
End Sub

Private Sub Command1_Click() 'Con el botón «Inicio»
Temporal = PortIn(&H37A) 'Almacena el valor del puerto de control
While (1) 'se valida el comienzo de la adquisición
Comienza
Wend
End Sub

Private Sub Command2_Click() 'Botón «Restaurar Puerto»
PortOut &H37A, Temporal 'Regresa a la configuración original pto. de control
PortOut &H378, 0 'Limpia nuevamente el puerto de datos; detiene programa
End Sub

Sub Comienza()
PortOut &H37A, &H20 'Configura C5, bidireccional y genera flanco negativo
Espera 0.5 'Rutina de espera (1/2 segundo)
PortOut &H37A, &H24 'Mantiene bidireccionalidad y genera flanco positivo
Espera 0.5
DatoPuerto = PortIn(&H378) 'Lee puerto de datos y escribe equivalente en Volts
Text1.Text = DatoPuerto * 0.0196 & « Volts» '255 binario = 5 Volts analógicos
End Sub

Sub Espera(t As Double) 'Rutina de tiempo
Timer1.Interval = t * 1000 'Milisegundos
Timer1.Enabled = True 'Habilitamos timer
Do While Timer1.Enabled
 DoEvents
Loop
End Sub


Private Sub Timer1_Timer() 'Habilitación del timer1, relacionada
Timer1.Enabled = False 'con la rutina Espera
End Sub

Private Sub Form_OKClick() 'Salir de la aplicación
App.End
End Sub
```

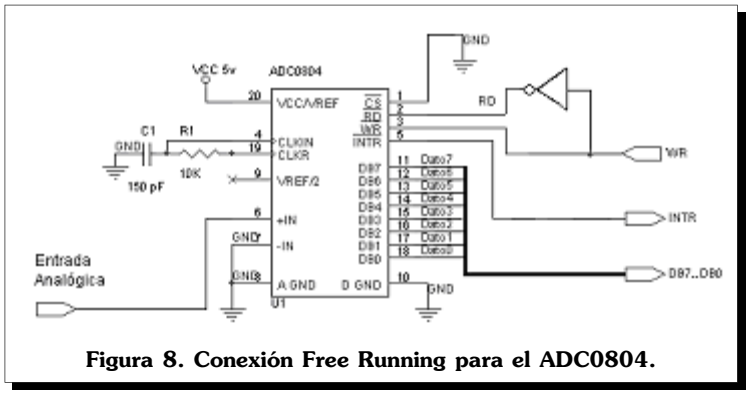

Figura 8. Conexión Free Running para el ADC0804.

Las **Figuras 9a** y **9b**, muestran de manera respectiva, las pantallas correspondientes al tiempo de diseño y al tiempo de ejecución de la interfaz en dos direcciones para el ADC0804. Particularmente en **9b** se aprecia el resultado arrojado por la aplicación para un dato de 255_{10} equivalente a 5 Volts, enviado por el ADC trabajando sobre Windows XP.

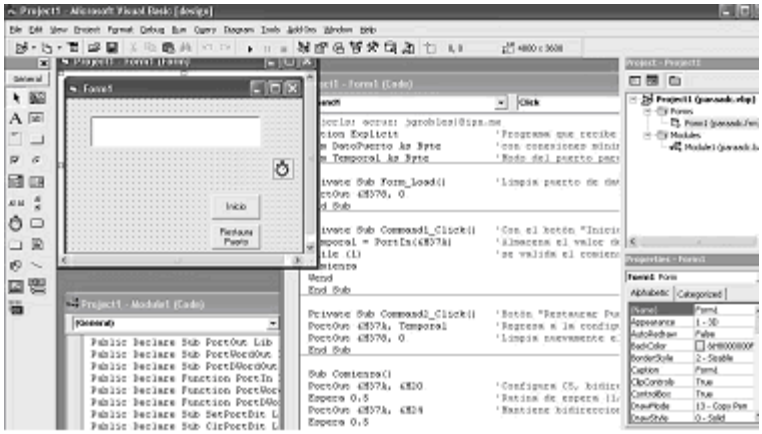

Figura 9a. Pantalla en modo de diseño para la conexión Free Running del ADC0804.

Figura 9b. Aplicación bajo Windows XP.

Conclusiones

Se presentó una metodología sencilla para crear interfaces para el puerto paralelo de la PC, utilizando Lenguaje C para modo MS-DOS y Visual Basic 6.0 para modo Windows. Las soluciones aportadas son adaptables sin cambios drásticos en el código a cualquier aplicación similar o más sofisticada, que requiera intercambio de datos en una o dos direcciones.

Todos los ejercicios mostrados se implementaron físicamente en dos PC con distinto sistema operativo. Las aplicaciones creadas en Lenguaje C se ejecutaron en modo MS-DOS bajo Windows 98 y las de Visual Basic bajo Windows XP. En ambos casos, para la bidireccionalidad de los datos, se configuró el modo *ECP* en el *SETUP.*

Cabe mencionar que es posible encontrar en Internet bibliotecas de enlace dinámico para diferentes lenguajes, como se puede advertir en el mismo sitio de donde se descargó io.dll (*www.geekhideout.com*) y en otros como *www.zealsoftstudio.com*, por mencionar algunos.

Se presentó el caso particular de un ADC0804 en un esquema bidireccional. La solución disminuye la cantidad de hardware y aumenta la velocidad de la respuesta, tomando en consideración que para plataformas superiores a Windows 98 existen dos modos de velocidad para acceso a puertos: *Normal* y *Fast*.

En un próximo artículo se detallará cómo utilizar el puerto paralelo como una alternativa al extinto bus ISA de la PC.

Referencias Bibliográficas e Internet

[1] Hans-Peter Messmer. *The indispensable PC hardware book*. Ed. Addison-Wesley, 1999.

[2] Dhananjay V. Gadre.*Programming the Parallel Port: Interfacing the PC for Data Acquisition & Process Control*. Ed. CMP Books, 1999.

[3] David I. Schneider. *An Introduction to Programming with Visual Basic 6.0*. Ed. Prentice – Hall, 2000.

[4] Francisco J. Ceballos. *Visual Basic 6, Curso de Programación*. Ed. AlfaOmega-Rama, 2002.

[5] *http://www.geekhideout.com*.

[6] Datasheet, National Semiconductor. *http:// www.national.com/pf/AD/ADC0804.html*.

[7] *http://www.doc.ic.ac.uk/~ih/doc/par/* : Interfacing to the IBM-PC Parallel Printer Port.

[8] *http://www.lvr.com/parport. htm* : Parallel Port Central

[9] *http://www.beyondlogic.org/ecp/ecp.htm*

[10] *http://www.epanorama.net/links/pc_inter face.html*

[11] *http://www.logix4u.net/para llelport1.htm*

[12] *http://www.zealsoftstudio. com*