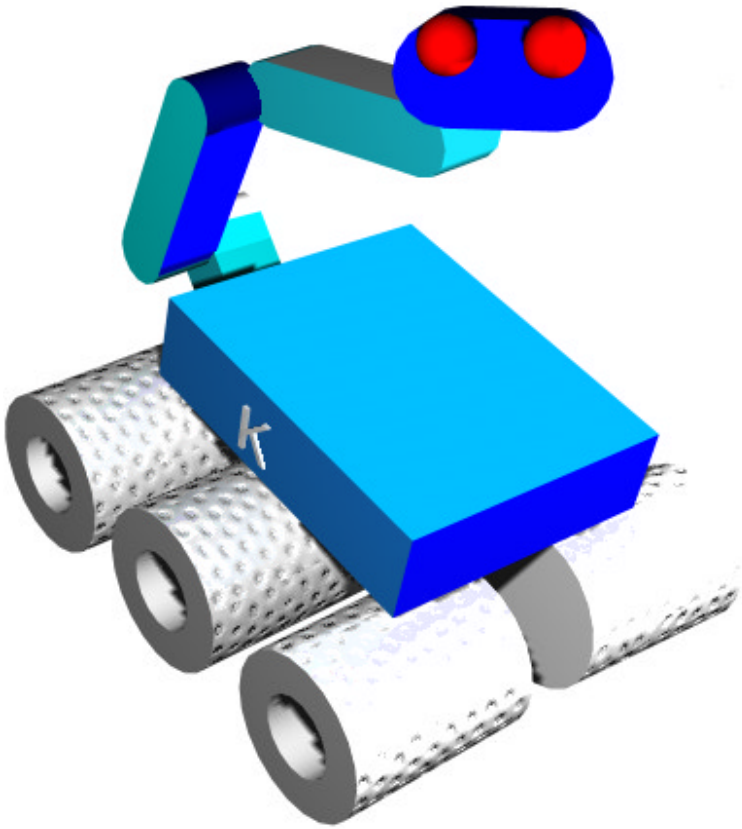
INTERFAZ DIDÁCTICA i-723

MANUAL DEL USUARIO



Simple Informática®

Publicación: 1998.
Impreso en Argentina.

Simple
Informática®

Carlos Tejedor 1597
CP.: 1602 Florida
Buenos Aires, Argentina
Telefax: (054-011)-4761-4165

Bienvenido,

yo soy el Asistente que te guiará paso a paso para que aprendas todo lo que debes saber acerca de la interfaz i-723 y del sistema didáctico de Simple Informática.

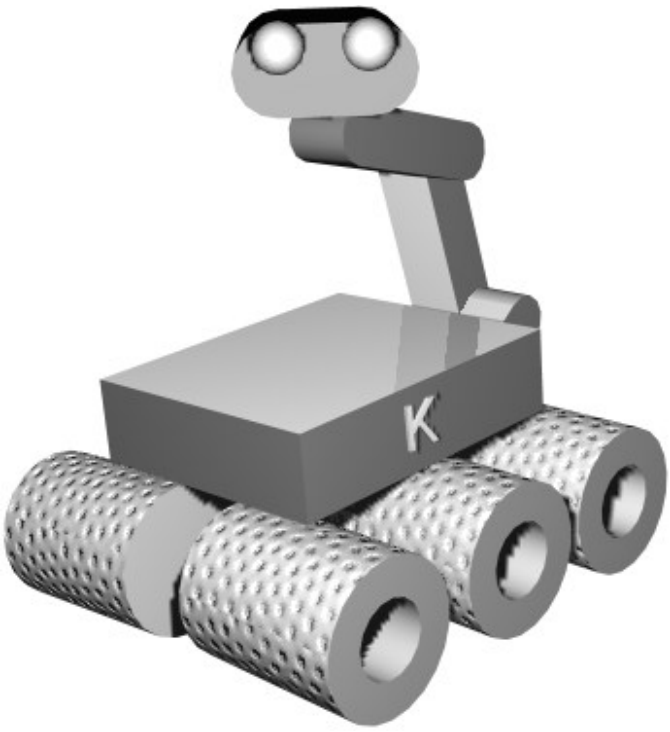
Estás a punto de darle nuevas capacidades a tu computadora. Ahora podrás hacer que ésta interactúe con el mundo real.

El objetivo principal de todo el sistema didáctico de Simple Informática es hacer que puedas manejar de la forma más sencilla e intuitiva posible, los proyectos más complejos.

Con la interfaz i-723 podrás trabajar tanto con los lenguajes de programación a los que probablemente ya estés acostumbrado (como LOGO, BASIC o PASCAL), como con aquellos de avanzada tecnología y aplicación profesional, como Delphi, Visual Basic o C++. Además, se incorpora un nuevo lenguaje icónico (en el que prácticamente no hay que escribir nada), con el que pueden trabajar incluso personas que aún no saben programar. Este es **Minibloques**, un lenguaje especialmente diseñado para ser aplicado en robótica educativa.

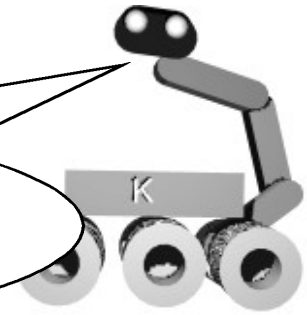
En cuanto al conexionado físico, la interfaz, los sensores, motores y demás componentes pueden ser interconectados sin dificultad con otros sistemas e incluso con piezas no estandarizadas (o hasta con material reciclado).

Con todo esto, tus proyectos de robótica ya no estarán limitados a un lenguaje específico o a un sistema cerrado. De esta forma, podrás concentrarte en el proyecto en sí y no en problemas y complicaciones secundarias.



CONTENIDO:

PRECAUCIONES:



Las siguientes son normas de uso de la interfaz i-723. No respetarlas puede ocasionar serios daños tanto al equipo como al personal que lo utiliza.

No sobrepases las tensiones máximas de trabajo en las entradas y salidas de la interfaz. (El apéndice B te informará con precisión sobre las mismas.).

La tensión de alimentación de la fuente que se incluye con el equipo está comprendida entre 220 y 230 VAC. Respétala.

Desconecta el equipo de la red de 220V una vez que finalizas las actividades. No dejes una habitación sola con equipos encendidos al terminar la jornada de trabajo.

Evita que los niños trabajen sin la presencia de adultos con el equipo. El mismo está pensado para niños de edades a partir de los 10 años.

Si la interfaz o la fuente de alimentación reciben un impacto, no prosigas el trabajo con ella. Consulta a tu distribuidor para que verifique los daños.

Utiliza siempre sensores y actuadores que se ajusten a las características de entrada/salida de la i-723. (El Capítulo II y el Apéndice B contienen mayor información sobre este tema).

Si por alguna razón se daña la fuente de alimentación de la interfaz, desenchufa la misma con suma precaución inmediatamente y consulta a tu distribuidor para realizar la reparación necesaria.

Si notas daño alguno en la interfaz, consulta inmediatamente a tu distribuidor.

La garantía de la i-723 es válida por 6 (seis) meses a partir del momento de la entrega. Para que la misma tenga efecto, se deberá presentar la interfaz ante el distribuidor. La misma contiene un número de serie. Si el número de serie se encuentra dañado y/o adulterado, la garantía pierde automáticamente su validez. Abrir la interfaz o intentar desarmarla anula la garantía. La garantía sólo cubre defectos de fabricación.

CAPÍTULO I:

COMENZANDO...

Instalación

Al abrir la caja, junto a la interfaz i-723 encontrarás:

- **D**os discos titulados "Sistema de Control de la interfaz i-723: **Utilitarios**" y "Sistema de Control de la interfaz i-723: **Librerías**". De aquí en adelante, en este manual serán llamados "Disco de Utilitarios" y "Disco de Librerías", respectivamente.
- **L**a fuente de alimentación para la interfaz.
- **E**ste manual.

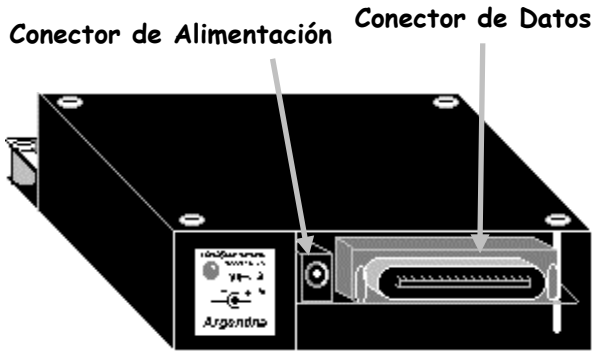


Figura 1.1: Conectores de la Interfaz.

Antes de que comiences a instalar tu interfaz, debes conocer los conectores de la misma. La figura 1.1 los muestra.
Para instalar la i-723, sólo sigue los siguiente pasos:

1) Apaga la PC en la que conectarás el equipo.

2) Conecta el cable de la fuente de alimentación provista con el **Conector de Alimentación** de la interfaz (figura 1.1)

3) Conecta un cable de impresora al **Conector de Datos** (figura 1.1) de la interfaz. (El otro extremo del cable debe ser conectado a un puerto paralelo - o puerto de impresora - de tu PC).

4) Enchufa la **fuente de alimentación** provista a la red de 220 Volts.

5) Enciende la computadora.

6) Los pasos que siguen son para instalar los programas que controlan a la interfaz. La instalación de aquí en más dependerá del sistema que tengas y de las tareas que pienses realizar con la interfaz. Si dispones en tu PC de Windows 3.1, Windows 3.11, Windows 95 o Windows 98, prosigue la instalación como se indica en el paso 7. Si el sistema operativo que posees es DOS (sin Windows), sólo podrás utilizar tu interfaz programándola con algún lenguaje que funcione bajo DOS (como por ejemplo LOGO, QBASIC o PASCAL). En este último caso, copia el Disco de Librerías a tu disco rígido (si tienes uno) y luego de leer el siguiente capítulo para familiarizarte con la i-723, lee el apartado del Capítulo "Programación avanzada de la interfaz: Las librerías para lenguajes de alto nivel", correspondiente al lenguaje que piensas utilizar.

7) Entra a Windows. Si tienes Windows 3.1 ó 3.11 debes tipear "Win" en tu PC y presionar ENTER. Si, en cambio tienes Windows 95, basta con encender tu PC para ingresar al mismo.

8) Coloca el **Disco de Utilitarios** (disco 1/2) en tu disquetera de $3\frac{1}{2}$ (si no tienes una, deberás copiar los discos de la interfaz a discos $5\frac{1}{4}$ para que tu PC los acepte).

9) Ejecuta el programa llamado **INSTALAR.EXE** del Disco de Utilitarios. Para hacer esto tienes varias posibilidades y depende de si dispones de Windows 3.1 (o 3.11) o Windows 95 ó 98:

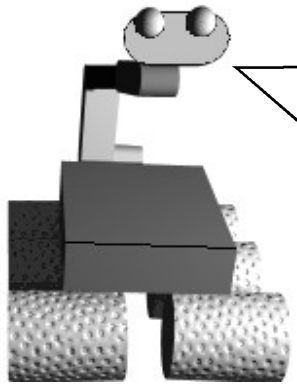
- **Windows 3.1 ó 3.11**: debes ir al menú Archivo (File, en inglés) y seleccionar Ejecutar (Run). Busca INSTALAR.EXE en tu disquetera 3 $\frac{1}{2}$ (A: o B:, dependiendo de cómo tengas configurada tu PC) y da click con el mouse en el botón ACEPTAR (botón "OK", en la versión en inglés).

- **Windows 95 ó 98**: en el botón Inicio (Start, en inglés), selecciona Ejecutar (Run). Busca INSTALAR.EXE en tu disquetera 3 $\frac{1}{2}$ (A: o B:, dependiendo de cómo tengas configurada tu PC) y da click con el mouse en el botón ACEPTAR (botón "OK", en la versión en inglés).

10) Deberá aparecer la pantalla del **Asistente de Instalación** (figura 1.2). El mismo te guiará paso por paso para que puedas instalar fácilmente todos los programas utilitarios provistos con la interfaz.

Figura 1.2: Asistente de Instalación.

11)



¡ **Listo** ! Ahora ya podrás utilizar la i-723. **Lo primero que deberás hacer es probarla y familiarizarte con sus entradas y salidas.** En este y en el próximo capítulo, se explicará esto con detalle. Antes de seguir, es conveniente que hagas una copia de cada uno de los discos de la interfaz, para evitar trabajar siempre con los originales.

Probando la interfaz

Una vez que haz instalado los programas utilitarios, ya puedes probar tu interfaz. Si tienes Windows 3.1 ó 3.11, verás que el Asistente de Instalación ha creado un grupo de programas en el Administrador de Programas (o Program Manager) llamado Simple Informática (a menos que tú le hayas dado otro nombre durante la instalación). En cambio, si tu sistema operativo es Windows 95 ó 98, lo que tendrás es una carpeta así llamada dentro de la carpeta Programas (o Programs en la versión en inglés) del botón Inicio (Start). La carpeta (o el grupo de programas) creada por el Asistente de Instalación contendrá cuatro íconos, los cuales corresponden a los cuatro programas utilitarios incluídos con la interfaz. El programa que figura como "Prueba de la Interfaz i-723" es el que debes utilizar para probar las funciones de la misma. Éste te permite controlar manualmente todas las salidas de la i-723 así como ver el estado de las entradas de sensor.

Antes de comenzar las pruebas, es necesario conocer los terminales del frente de la interfaz, para entender lo que ocurre cuando accionas sus salidas desde el programa de prueba. Los mismos están identificados con colores y números; además, están agrupados según su función en tres grupos: **Salidas de Relé**, **Salidas de Motor** y **Entradas de Sensor** (firura 1.3).

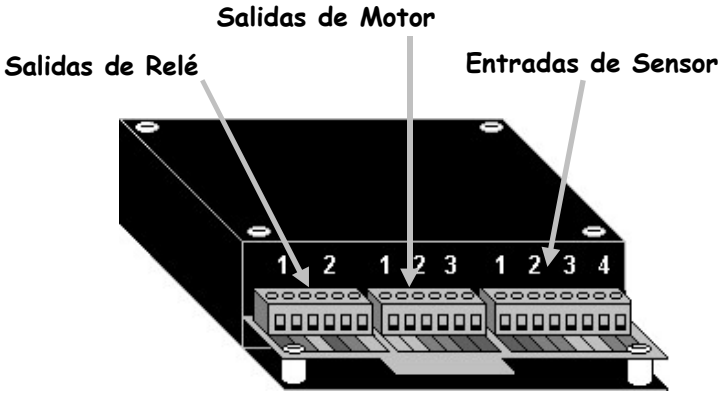


Figura 1.3: Frente de la interfaz i-723

Los terminales de las salidas de relé poseen tres conectores cada uno, en tanto que los de motores y los de sensores poseen dos. La figura 1.4 muestra el diagrama de colores. En ella se distinguen los tres grupos mencionados (Salidas de Relé, Salidas de Motor y Entradas de Sensor):

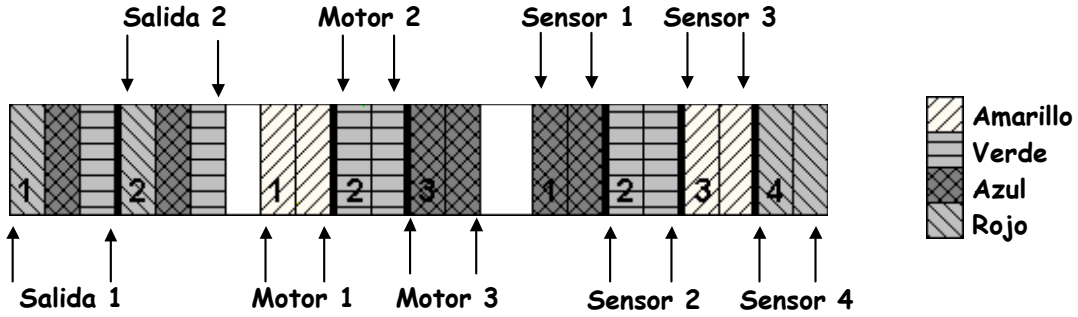


Figura 1.4: Terminales de conexión

Ahora ya puedes correr el programa de prueba; Si tienes Windows 3.1 ó 3.11, deberás darle un doble click con el mouse al ícono titulado "Prueba de la Interfaz i-723". En Windows 95 ó 98, tendrás que hacer un único click. Con él podrás probar la interfaz de un modo fácil e intuitivo. El mismo presenta varios botones en la pantalla. Hacia la parte inferior izquierda se encuentran los botones para controlar las salidas de relé de la interfaz (en el Capítulo II se explica el funcionamiento de estas salidas). Presionando uno de los botones que dicen "SI", deberá encenderse la luz amarilla de la salida accionada en la interfaz. Si no sucede esto, haz click con el mouse sobre las casillas de

selección que aparecen bajo el título "Port de Conexión" (pequeños botones redondos en la esquina superior derecha de la pantalla) que dicen Lpt1, Lpt2 y Lpt3. Con ellas puedes configurar el programa indicando en qué puerto de impresora está conectada la i-723 (puerto 1, 2 ó 3, respectivamente).
Para accionar los motores, haz click con el mouse en los botones que tienen las flechas. Para detener un motor, utiliza los botones marcados con una "X" azul. (El botón con la "X" negra apaga todas las salidas, tanto de motor como de relé).
Como verás, hay tres botones de control para cada salida de motor, mientras que hay sólo dos por cada salida de relé. Esto es así porque las salidas de relé tienen dos estados posibles: Activado y Desactivado (botones "Si" y "No" respectivamente), mientras que las salidas de motor tienen tres estados: motor girando hacia la derecha, motor girando hacia la izquierda (botones con las flechas) y motor detenido (botón con la X azul).

Si tu interfaz responde a todas las órdenes que le des desde el programa de prueba, es porque fue correctamente instalada. Cada vez que lleves la interfaz a otra computadora, podrás utilizar este programa para probar que todo esté bien.

Requerimientos mínimos

Para utilizar la i-723 desde DOS, mediante algún lenguaje de programación para ese sistema operativo, podrás utilizar una PC/XT con 640 Mb de ram y una disquetera. (Si la disquetera es para discos de 5 ¼, deberás copiar las librerías del disco 2/2 a un disco 5 ¼ para que lo acepte esa disquetera).

Si vas a utilizar los programas para Windows, será necesario un PC/AT 386 o superior, con 4 Mb de memoria ram, que posea mouse y al menos 1,4 Mb libres en el disco rígido.

Para poder correr Minibloques 1.02, es necesaria una PC basada en un procesador 486 DX o superior y que tenga 8 Mb de ram. Además, para Minibloques es recomendable un monitor VGA color o superior.

Cualquiera de las computadoras mencionadas debe tener un puerto paralelo (o puerto de impresora) para poder conectar la interfaz.

¿Si no funciona?

En caso de que la interfaz no encienda (luz roja de encendido apagada), verifica el cable de la fuente de alimentación y si ésta está correctamente enchufada tanto a la interfaz como a la red de 220 Volts.

Si la interfaz está encendida, pero no logras que responda a las órdenes de la PC, verifica el cable de conexión entre la interfaz y la computadora.

Si el cable está bien conectado, cambia el puerto paralelo de conexión (Lpt1, Lpt2 ó Lpt3) en la configuración del programa.

En caso de que el problema persista, consulta a tu distribuidor o directamente a Simple Informática.

CAPÍTULO II:

TRABAJANDO CON LA INTERFAZ

El sistema de conexiones

La interfaz i-723 está dotada de un sistema de conexiones abierto, que permite conectarle una gran variedad de accesorios, sean estos estándar o no. Incluso es posible conectar accesorios de tipo "casero" o de otros sistemas didácticos. La única limitación es que cada accesorio a conectar debe respetar las características técnicas de la i-723 (ver Apéndice B).

En el frente de la interfaz hay varios conectores. Antes de entrar en detalles describiéndolos, veamos como se conecta algo físicamente en ellos. Cada conector tiene orificios en la zona frontal y tornillos en su parte superior. Esta configuración permite enchufar en los conectores cables de diferentes tamaños. Mediante el tornillo (y contando con un destornillador), se puede ajustar y desajustar el cable introducido en el orificio. Es importante hacer un ajuste adecuado de cada tornillo. Así, las conexiones resultantes serán firmes y se evitarán fallas causadas por falsos contactos.

Es importante recordar que tanto los accesorios como la interfaz tienen conectores de tipo "hembra" exactamente iguales. Para conectarlos es preciso utilizar un cable. Puedes usar cable provisto por Simple Informática o cualquier cable que tengas. Esta manera de conexión es ideal si quieres conectar accesorios a distancias grandes, ya que puedes armarte tus propios cables con la longitud que necesites. Como verás, todo el sistema de conexionado es independiente de un tipo especial de ficha, o de un tipo específico de cable, lo que convierte a todo el sistema en altamente compatible con piezas de orígenes variados.

Salidas de motor

La i-723 posee tres salidas con inversión de polaridad, pensadas especialmente para ser utilizadas en el control de motores de corriente continua, como los de los juguetes a pilas o los de los juegos de construcción. La figura 2.1 muestra la conexión de un motor a una de estas salidas.

La corriente soportada en total es de 0,9 Amperes de consumo constante con la fuente provista. Esto permite conectar tres motores de 260 mA cada uno, o dos motores de consumos tales que sumándolos, en total no sobrepasen 0,9 Amperes de corriente constante (la i-723 puede soportar corrientes instantáneas mayores, como por ejemplo las corrientes de arranque de los

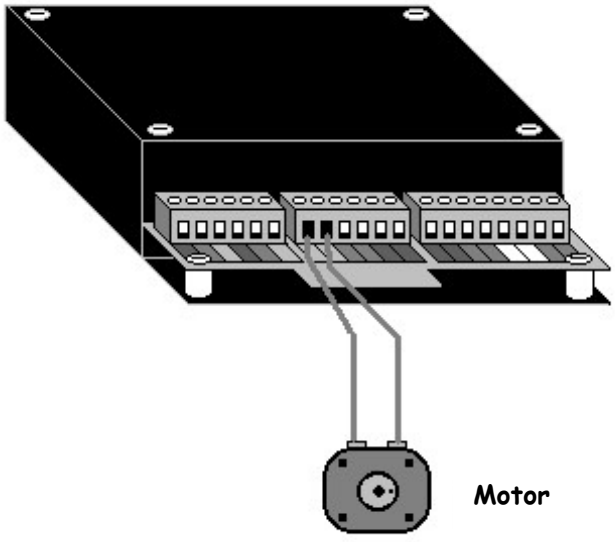


Figura 2.1: Conexión de un motor a la interfaz.

motores. Esto quiere decir que aunque los motores sobrepasen el límite de corriente en momentos críticos como suele ser el arranque, esto no daña a la interfaz). No es preocupante el hecho de que por equivocación se conecten elementos de mayor consumo que el especificado, ya que la interfaz tiene un sistema de **protección electrónica**. En caso de que sea sobrepasado el valor máximo de corriente, se apagarán todas las salidas automáticamente y comenzará a titilar la luz roja de encendido. Si ocurriese un cortocircuito en estas salidas, sucederá exactamente lo mismo. Cada salida posee un indicador luminoso que cambia de color (rojo o verde) según el sentido de giro del motor en cuestión. Utiliza el programa de prueba para familiarizarte con estos indicadores.

Es posible también, utilizar estas salidas para conectar luces comunes, pero se estará "desperdiciando" la capacidad de las salidas en cuanto a inversión de polaridad.

Otro uso de estas salidas es la conexión de luces especiales que cambian de color según la polaridad y de otros accesorios tales como los generadores de sonido (Consulta a tu proveedor por esta clase de accesorios). Estos últimos deben ser conectados con una polaridad definida (en general el cable oscuro es negativo y el rojo positivo) por lo cual es altamente recomendable conectarlos a las salidas de relé.

Salidas de relé y fuentes externas

Las dos salidas de relés de la i-723 agregan la ventaja de que soportan corrientes y tensiones mayores que las salidas de motor. Se las puede ver como si fueran llaves o interruptores controlados por el ordenador: ellos sólo abren o cierran el paso de la energía que alimenta al elemento conectado a las mismas. Claro que la alimentación no es provista por tales salidas, por lo que debe ser tomada de la fuente incluida con la i-723 o de fuentes externas que se ajusten al caso. La principal ventaja reside en poder controlar tensiones y corrientes elegidas por el usuario de acuerdo a sus necesidades específicas (**siempre y cuando se respeten los valores máximos absolutos**).

Los márgenes de tensión y corriente son:

Corriente Continua: 24 Volts; 2 Amperes
Corriente Alterna: 48 Volts; 2 Amperes.

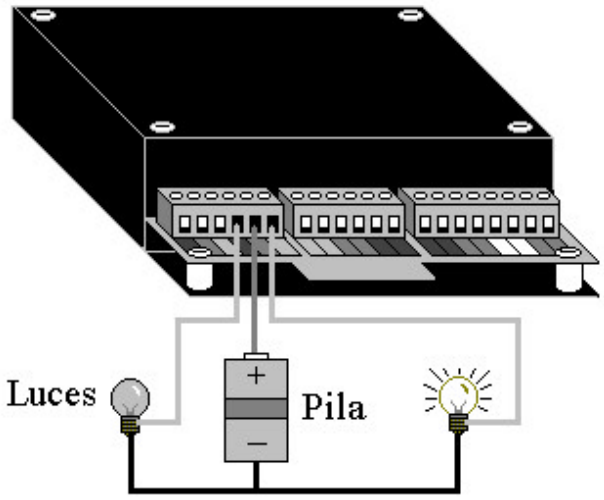


Figura 2.2: Salidas de relé.

Hay que tener siempre en cuenta que estos límites son **ABSOLUTOS**. La figura 2.2 muestra una salida de relé controlando dos lámparas de modo tal que al encenderse una, se apaga la otra. La fuente en el ejemplo es una pila común. Es posible claro, conectar cualquier otra fuente, como por ejemplo un transformador, una batería, etc. La fuente de la i-723 también es una posibilidad. Si observas la figura 1.4, podrás apreciar los colores de identificación de las salidas de relé. Cuando la salida se encuentra "apagada" (cada salida tiene una luz amarilla que indica su estado), no hay contacto entre la bornera identificada con el color azul y la identificada con rojo, pero sí entre la azul y la verde. Esto se debe a que estas salidas son

inversoras. Por lo tanto, cuando se encienda la salida (luz amarilla prendida), lo que se estará haciendo en realidad será establecer contacto entre las salidas azul y roja. La salida azul es lo que se llama el "punto medio". Ella siempre está conectada con la bornera roja o con la verde, pero nunca con ambas al mismo tiempo. De este modo, las salidas de relé son en verdad llaves inversoras o conmutadores. Si vuelves a ver la figura 2.2, podrás notar que cuando una de las luces se enciende, la otra se apaga. Nunca están las dos encendidas. Este mismo ejemplo es aplicable a electroimanes, generadores de sonido, motores, etcétera. Hay que tener en cuenta que algunos dispositivos tienen una polaridad que debe ser respetada.

Sensores

A la i-723 pueden ser conectados hasta 4 sensores digitales (sensores de dos estados posibles: activado o desactivado). La figura 2.3 muestra la conexión de dos tipos de sensores diferentes: de luz y de choque (o "fin de carrera"). Existen varios tipos de sensores para realizar distintas mediciones. Algunos de ellos son:

- **d**e temperatura
- de luz e infrarrojos
- pulsadores mecánicos
- magnéticos

Todos ellos se conectan a las entradas de sensor indistintamente. En el Apéndice C (Fichas Didáctidas), podrás aprender más sobre el uso y las aplicaciones de los diferentes tipos de sensor. Además, puedes armar tú mismo los sensores que desees, siempre y cuando respetes las especificaciones de la interfaz (mira el Apéndice B: Especificaciones técnicas). Es imprescindible respetar las especificaciones de la interfaz y utilizar los sensores indicados, para evitar dañar el equipo. Ten en cuenta, si piensas armarte tus sensores que los utilizados por la i-723 son todos de tipo resistivo.

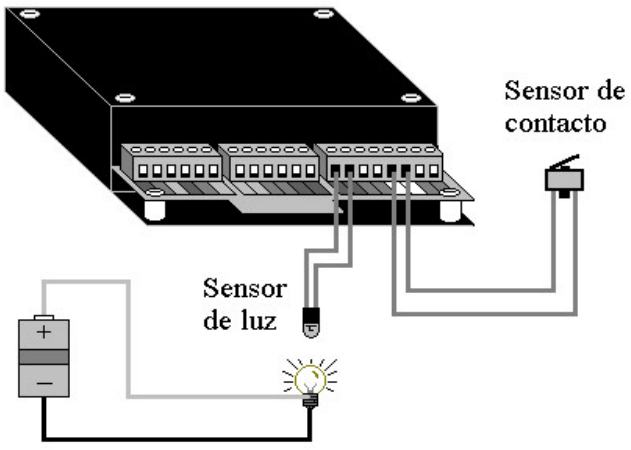


Figura 2.3: Conexión de sensores.

Por medio de los sensores se pueden realizar diversas mediciones, como ser conteos de vueltas, medición de velocidades, detección de obstáculos, control de temperaturas, etc… Además, una misma PC puede soportar más de una interfaz si cuenta con un puerto paralelo extra, y hasta es posible que una computadora sense a los elementos controlados por otra, o que haciendo una combinación entre salidas de relé y entradas de sensor se establezca comunicación entre interfaces conectadas en diferentes computadoras. Pronto verás que las posibilidades son muchísimas.

CAPÍTULO III:

USO DE LOS UTILITARIOS PROVISTOS CON LA INTERFAZ

Además del programa de Prueba mencionado en el Capítulo I, se suministran con la interfaz tres programas ejecutables (archivos con extensión EXE) más. En el grupo de programas creado por el Asistente de Instalación (si tienes Windows 3.1 ó 3.11 éste estará en el Administrador de Programas; sin posees Windows 95 ó 98 lo verás en la carpeta Programas del botón Inicio) encontrarás cuatro íconos, uno de los cuales, como ya se explicó, es el del programa de prueba. Los otros tres son:

- Minibloques 1.02
- Simulador de la Interfaz i-723
- Controlador de Puertos

Minibloques es un moderno sistema de programación basado en íconos. Con él es posible que personas con muy pocos conocimientos de programación puedan hacer sus propios programas. El próximo capítulo se dedica por completo a este lenguaje. Los dos programas restantes se describen a continuación:

El Simulador de la Interfaz

El Simulador de la Interfaz es un programa muy útil para el trabajo en grupos. Su función es la de simular o "hacer de cuenta" que hay una interfaz i-723 conectada a la PC que está siendo utilizada para programar. Esto permite, por ejemplo, que varias personas trabajen en distintas computadoras y que puedan hacer algunas pruebas de los programas que estén realizando, sin tener en realidad una interfaz en cada computadora. Como el mismo funciona bajo Windows, se puede utilizar un lenguaje de programación en otra ventana (incluso lenguajes en ventana de DOS) y ver en el Simulador lo que sucedería si realmente hubiera una i-723 conectada a la computadora. Es como tener una interfaz "virtual" en la PC. La figura 3.1 muestra la pantalla completa del programa.

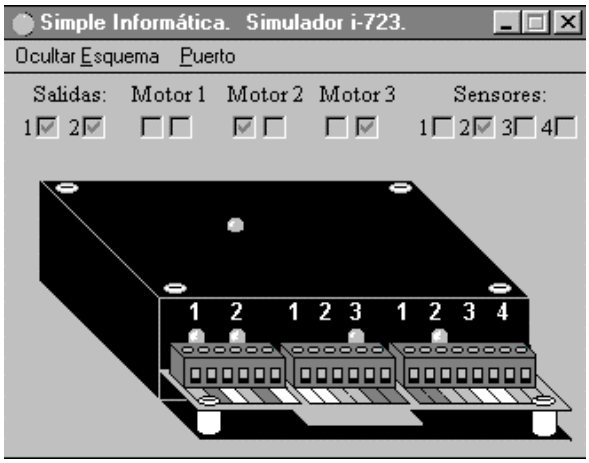


Figura 3.1: Pantalla completa del programa Simulador de la Interfaz i-723.

Como se podrá observar en la figura, el programa presenta un esquema de la interfaz en pantalla. La idea es que cualquier orden que se le dé a la interfaz desde un lenguaje de programación, sea reflejada en

el esquema del programa. Veamos un ejemplo. Supongamos que estás programando en LOGO y tienes en tu PC un puerto de impresora, pero no haz conectado la interfaz al mismo, ya que ésta está en uso en otra computadora. Si el puerto de impresora es el Lpt 1 y quieres simular una interfaz allí conectada, lo que debes hacer es lo siguiente:

Figura 3.2: Pantalla reducida del programa Simulador de la Interfaz i-723.

1) Abre una ventana de DOS para correr el LOGO (esto si el LOGO que posees es para DOS).

2) Ejecuta el Programa Simulador (da doble click sobre su ícono si estás trabajando con Windows 3.1 ó 3.11, o un único click si estás en Windows 95 ó 98).

3) En el menú PUERTO del programa, selecciona Lpt 1.

4) En el LOGO, cargua las librerías de la interfaz (para más información consulta el apartado dedicado a LOGO en el capítulo titulado "Programación avanzada de la interfaz: Las librerías para lenguajes de alto nivel").

5) Inicializa desde el LOGO la interfaz como si ésta estuviera realmente conectada al Lpt 1.

6) Ahora, cada orden de control de motores o salidas de relé que ejecutes desde la ventana de LOGO, se verá reflejada en la pantalla del simulador, como si éste fuese una interfaz virtual conectada al puerto paralelo seleccionado (en nuestro ejemplo, el Lpt1).

Si la pantalla del programa de simulación resultase molesta en tu monitor por ser demasiado grande, tienes una opción en el menú para que el esquema de la interfaz se oculte, quedando visible el estado de la i-723 en forma de panel, como se muestra en la figura 3.2:

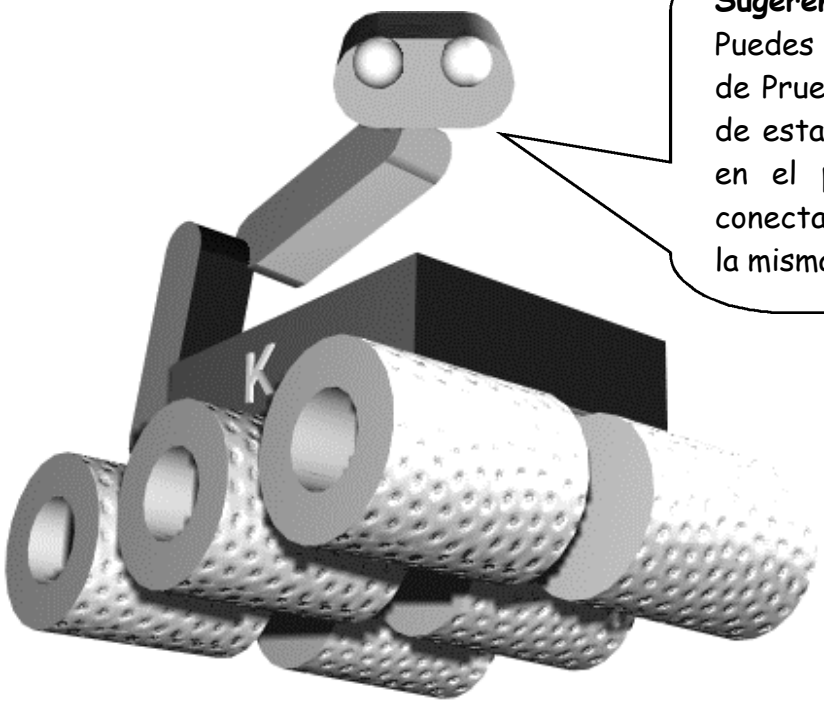
Funciones avanzadas: el Controlador de Puertos

Si eres es un programador con experiencia, es probable que encuentres el programa Controlador de Puertos (PRTCTRL.EXE) de gran utilidad. El mismo te permite monitorear y controlar cualquier puerto de Entrada/Salida (E/S) de tu PC mientras diseñas y depuras tus programas. Con él podrás ver qué ocurre exactamente, a nivel de bit, en el puerto seleccionado. Además, resulta un utilitario de gran ayuda a la hora de programar cualquier periférico o de hacer rutinas de control en un nuevo lenguaje. Si eres usuario de algún lenguaje no soportado directamente por la i-723, prueba el Controlador de Puertos, ya que éste te será de gran ayuda a la hora de hacer las nuevas rutinas de control.

La figura 3.3 muestra la pantalla de este utilitario. La misma resulta intuitiva para quienes están acostumbrados a programar en bajo nivel. Cada casilla de selección representa un bit en la dirección de E/S indicada abajo. Algunas veces, notarás que aunque hagas click en una casilla, ésta no cambiará de estado. Esto se debe a que algunas direcciones de E/S son de sólo lectura, y por ello no pueden ser modificados los valores de sus bits. Debes ser cuidadoso con las modificaciones en los distintos bits de las direcciones seleccionadas, ya que algunos (como por ejemplo el bit 3 –cuarta casilla, ya que comienza del bit 0- del puerto de control de un puerto paralelo) llaman interrupciones, las cuales podrían hacer que tu sistema "se cuelgue". Simple Informática no se responsabiliza de ningún daño o pérdida de información causado por el uso del software o del hardware incluido.

El programa tiene dos modos de trabajo: En el modo utilizado por defecto, transfiere inmediatamente cualquier cambio en las casillas de los bits al puerto físico, además de monitorearlo continuamente (tiempo real). Presionando el botón "Congelar Valores", ninguno de los cambios realizados en las casillas afecta al puerto, hasta que se vuelva a presionar el botón (cuyo título habrá cambiado por el de "Aplicar Valores"). Esto permite "inyectar" un byte completo "de una sola vez", en lugar de hacerlo a nivel bit, uno por uno. Una última salvedad acerca de este programa: los paneles de Status y Control sólo aparecen cuando el puerto seleccionado es un puerto paralelo (Lpt1, Lpt2 ó Lpt3), ya que esos son los puertos que poseen una zona de Control y otra de Status.

Figura 3.3: Pantalla del Controlador de Puertos.



Sugerencia:
Puedes usar conjuntamente el Programa de Prueba y el Controlador de Puertos y de esta forma podrás ver lo que sucede en el puerto donde la interfaz está conectada cada vez que das una orden a la misma.

CAPÍTULO IV:
PROGRAMACIÓN MÁS QUE FÁCIL: MINIBLOQUES 1.02

¿Qué es Minibloques?

Minibloques es un sistema de programación de uso sencillo. Está diseñado especialmente para ser utilizado en Robótica Educativa. Con él es posible programar la Interfaz Didáctica i-723 y controlar así cualquier mecanismo o sistema conectado a la misma. Programar con Minibloques es tan fácil como dibujar.

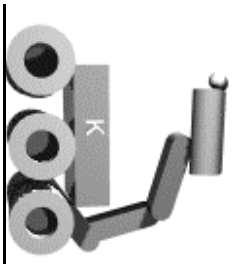
Cómo entrar a Minibloques

Si haz seguido los pasos de instalación del Capítulo I, Minibloques ya está instalado en tu PC. Si estás trabajando con Windows 3.1 ó 3.11, para ejecutar Minibloques debes dar doble click sobre el ícono llamado "Minibloques 1.02" del grupo de programas del Administrador de Programas (Program Manager en inglés) que ha sido creado por el Asistente de Instalación (Capítulo I). Si tu sistema operativo es Windows 95 ó 98, bastará con un solo click en el ícono de Minibloques 1.02 que está en la carpeta creada por el Asistente de Instalación dentro de la carpeta "Programas" del botón "Inicio".

Primeros pasos

Al correr Minibloques, aparecerá una pantalla de presentación. Al dar click en el botón "Aceptar" de dicha pantalla, entrarás en la Pantalla de Trabajo. En la figura 4.1 se la muestra, indicando sus partes.

Figura 4.1: Pantalla de Trabajo de Minibloques 1.02.



Antes de poder hacer algo con Minibloques, deberás indicarle al mismo el puerto donde se halla conectada la interfaz. La **zona de configuración del puerto de la interfaz**, tiene tres casillas de selección tituladas "Lpt 1", "Lpt 2" y "Lpt 3". Dando un click en la casilla llamada "Lpt 1" le estarás indicando a Minibloques que la interfaz está conectada en el puerto de impresora número 1 (llamado precisamente Lpt 1). Si tienes en tu PC más de un puerto de impresora, puede que quieras conectar tu interfaz en el puerto Lpt2 o Lpt3. En este caso deberás marcar la casilla correspondiente.

La barra de herramientas

La **barra de herramientas** se divide en dos grupos de botones. El grupo de la izquierda es el que contiene los "**bloques**" , mientras que el grupo de la derecha contiene las funciones para guardar o ejecutar el programa, obtener ayuda, salir de Minibloques, etc.. En la figura 4.2 se pueden apreciar los dos grupos mencionados, así como la función de cada botón.

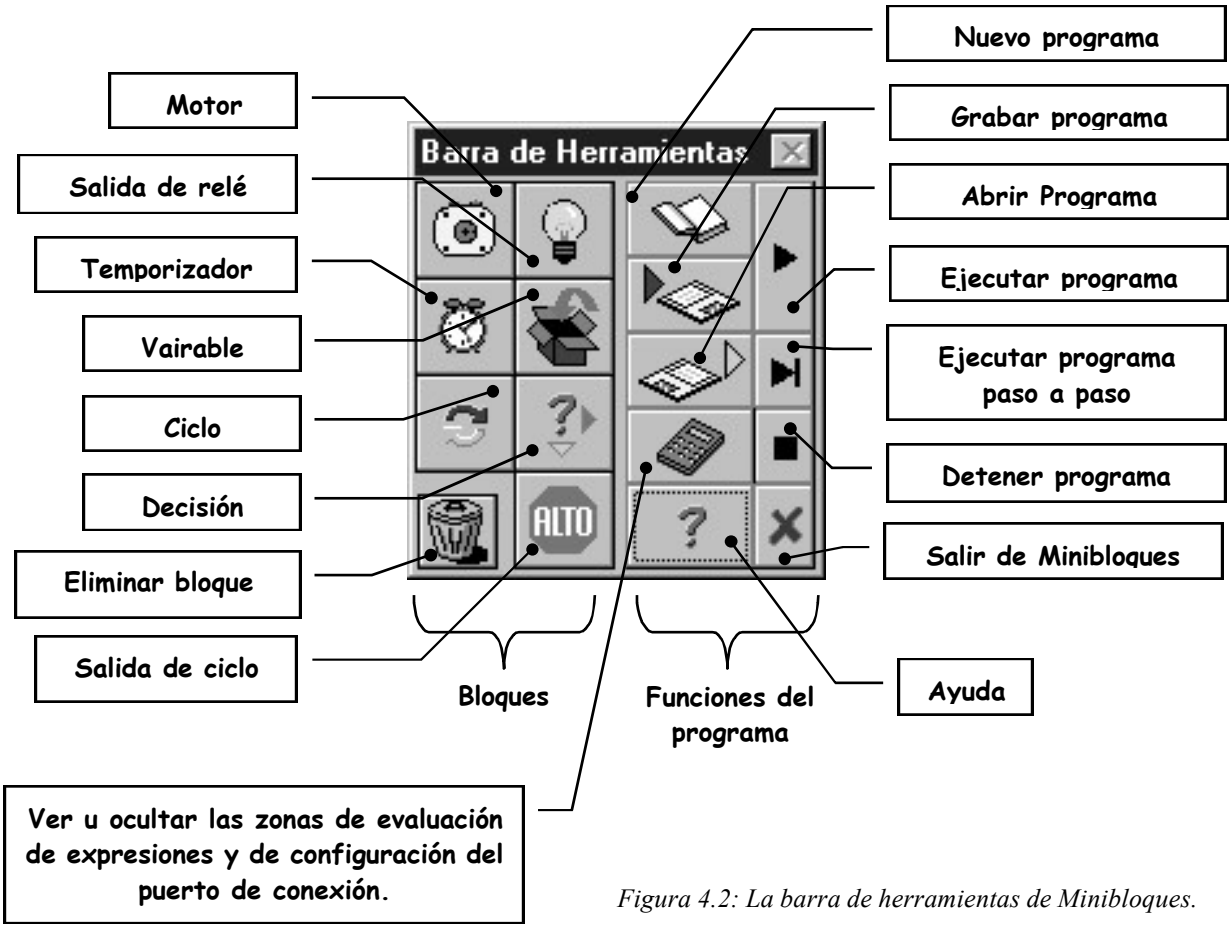


Figura 4.2: La barra de herramientas de Minibloques.

Los bloques son con los que construirás tus programas (podemos decir que son los verdaderos "elementos del lenguaje").. Hay en total siete bloques (el botón "Eliminar bloque" no es un bloque). Cada bloque representa una sentencia de programación. Por ejemplo, con el bloque de motor puedes ordenarle a la interfaz que haga girar un motor conectado a sus salidas hacia la derecha o hacia la izquierda. Al hacer click con el mouse en los botones correspondientes a los bloques, verás que se

agrega el bloque elegido a la zona de programación (la figura 4.3 muestra el resultado de presionar el botón de bloque de motor, luego el de temporizador y nuevamente el de motor).

Lo que ves en la figura 4.3 ya es un programa Minibloques. El único problema es que ese programa no hace nada (aún). Si te fijas en la pantalla de Minibloques, verás que hay un bloque pintado de color celeste. Ese es el "bloque actual". Si presionas el botón de eliminar bloque (el que tiene dibujado un cesto de basura), el bloque borrado será el bloque actual (pruébalo). Si, por otro lado agregas un nuevo bloque a tu programa (presionando uno de los siete botones de bloques), éste aparecerá **después** del bloque actual. Para seleccionar un bloque como actual, sólo debes hacer click con el mouse sobre él. Si deseas insertar un bloque al principio del programa, debes marcar el bloque "Comienzo" y luego agregar el nuevo bloque, el cual aparecerá entre comienzo y el bloque que le seguía. (Nota: "Comienzo" no puede ser borrado, es un bloque fijo y sólo está para indicar el lugar por donde comenzará la ejecución del programa).

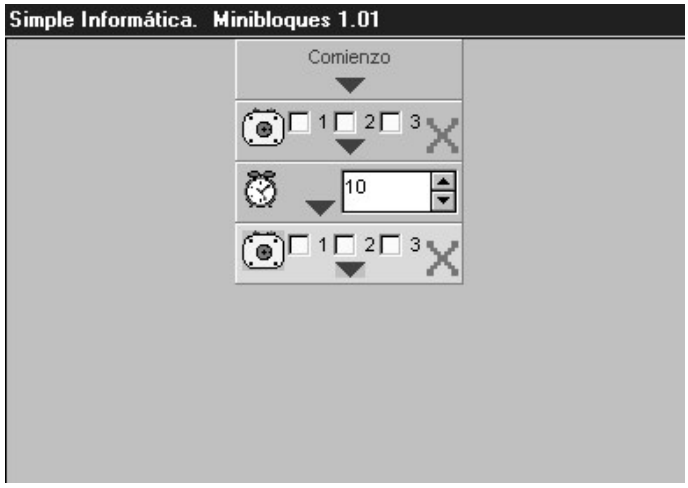


Figura 4.3: Vista de la zona de programación tras agregar algunos bloques.

Nuestro primer programa en Minibloques

Haz click en el botón de "Nuevo programa" (botón con un libro en blanco dibujado). Aparecerá un mensage prejuntándote si deseas grabar el programa actual. Selecciona "No" (si así lo quieres). Ahora comencemos nuestro primer programa en Minibloques. El mismo hará que la salida 1 de motor se active hacia la derecha (derecha e izquierda cuando se trata del sentido de giro de un motor son conceptos arbitrarios: para nosotros derecha será cuando la luz de la salida se torne roja e izquierda cuando esté en verde; la luz apagada significa que la salida está apagada). Luego nuestro programa esperará por un breve intervalo de tiempo y hará que la salida se active hacia la izquierda (luz en verde). Finalmente volverá a esperar y apagará la salida.

Para lograr este programa, agrega un bloque de motor, luego un bloque temporizador y después otro más de motor. Coloca otro temporizador y otro bloque de motor al final. El programa se verá como el de la figura 4.4.A. Lo siguiente es seleccionar qué salida de motor será activada por los bloques que hemos colocado. Para esto debes seleccionar una de las casillas de los bloques de motor. En nuestro programa queremos activar la salida 1, por lo que seleccionaremos la primer casilla desde la izquierda. Una vez seleccionada la salida, se debe indicar la

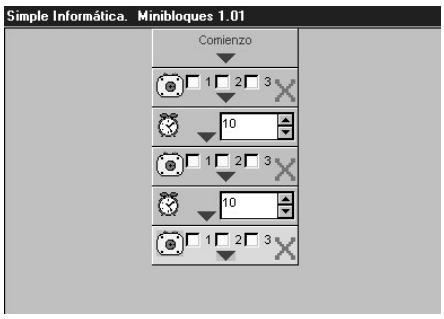


Figura 4.4.A: El programa en construcción.

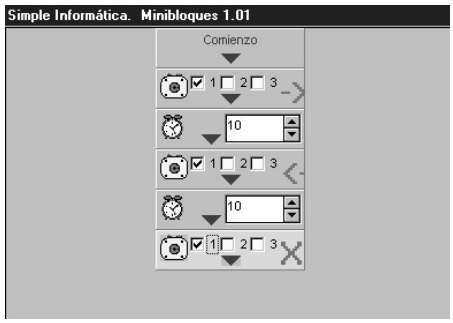


Figura 4.4.B: El programa terminado.

acción que ésta realizará. Esto se hace haciendo click sobre el ícono en rojo de la esquina derecha de los bloques de motor. Con cada click cambia el ícono indicando la acción que será realizada (**<-**: salida activada hacia la izquierda; **->**: salida activada hacia la derecha, **X**: salida apagada). La figura 4.4.B muestra el programa terminado. Para ejecutarlo presiona el botón de ejecutar programa (figura 4.2). Notarás que (si todo está bien), la salida 1 de motor se pone en rojo, luego pasa un intervalo de tiempo, la misma se pone en verde y finalmente se apaga tras otra breve espera. Así de fácil es como funciona Minibloques. Sólo hay que poner un bloque tras otro, seleccionar la salida que será afectada por el bloque y luego seleccionar la acción a realizar.

Si presionas el botón de **Abrir programa** y buscas en el cuadro de diálogo que se desplegará, encontrarás un directorio llamado "ejemplos". En él hay algunos programas que te muestran el funcionamiento de los distintos bloques.

Ciclos

Para hacer que una acción se repita, es necesario colocar un **ciclo**. Estos bloques están formados en realidad por dos bloques: uno que da comienzo al ciclo y otro que le da fin. Todo lo que se encuentre entre ambos se ejecutará repetidamente. Para detener un ciclo hay que agregar un bloque de **salida de ciclo**. Veamos. Comienza un nuevo programa y agrega un ciclo. Ahora agrega dentro del mismo todos los bloques que utilizaste en el primer programa y selecciona la salida 1 y las mismas acciones que en aquel programa. El resultado deberá verse como muestra la figura 4.5.A. Al ejecutarlo, el programa no se detendrá, sino que volverá a comenzar una y otra vez, hasta que lo detengamos manualmente (botón **Detener programa**; figura 4.1). Cada vez que el programa llega a un **Fin de ciclo** vuelve al **Principio de ciclo** (es como si hiciera un "salto"). Aquí es preciso notar un hecho: aunque inmediatamente antes del fin de ciclo hay un bloque para apagar la salida de motor, al ver la interfaz descubrirás que la salida no se detiene nunca, sino que siempre está en rojo o en verde. Esto ocurre porque entre un bloque y otro la ejecución es inmediata, a menos que halla un

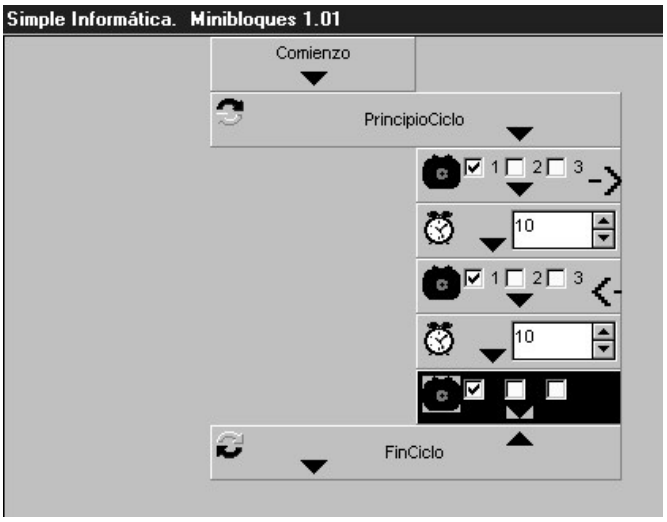


Figura 4.5.A: Programa con un ciclo.

temporizador entre ambos bloques. Lo mismo ocurre entre un bloque de final de ciclo y el de principio de ciclo: se ejecutan y el tiempo transcurrido entre ellos es prácticamente nulo. Para que la salida permanezca apagada un tiempo perceptible hay que agregar un temporizador entre el bloque que la apaga y el fin de ciclo (o entre el principio de ciclo y el bloque que le sigue). La figura 4.5.B muestra el agregado. Al ejecutar ahora el programa, la salida 1 permanecerá

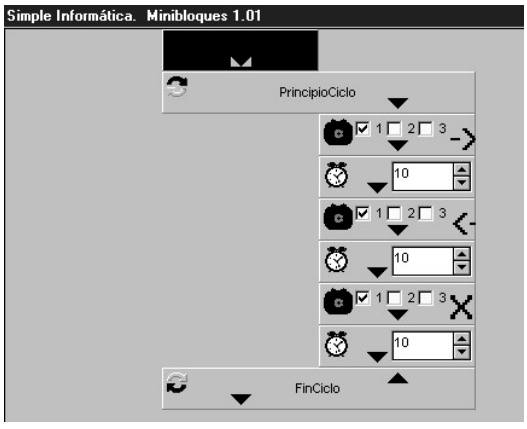


Figura 4.5.B: Agregado del temporizador.

detenida por un intervalo similar al que permanece en los otros dos estados (derecha e iquierda).

Variables, decisiones y salidas de ciclo

Generalmente, lo que queremos no es que un conjunto de acciones se repitan indefinidamente como en el programa anterior, sino que queremos que ocurran un número determinado de veces. Para lograr esto contamos con las variables, las salidas de ciclo y las decisiones. Una variable es como una "cajita' en la memoria de la computadora donde almacenamos números que luego nos servirán en el programa. Cada variable tiene un nombre que la identifica y que se lo ponemos nosotros. En Minibloques los nombres de las variables no pueden tener más de **seis caracteres** y sólo pueden estar compuestos por letras. Por ejemplo: CASA, NUMERO, HOLA o CONT son nombres válidos, mientras que CONTADOR, NUMEROGRANDE, N1 o 567 no son nombres correctos. Cada bloque de variable tiene dos casillas de texto. En la de la izquierda debes escribir el nombre de la misma y en la derecha el valor que tomará una vez que la ejecución del programa "pase por ese bloque". El bloque de la figura 4.6.A asigna el valor 4 a la variable NUM. En la figura 4.6.B muestra cómo se le puede asignar el valor de una expresión matemática a una variable. Para obtener más información sobre las expresiones matemáticas en Minibloques, lee en la ayuda del programa el apartado "Expresiones matemáticas". Además, si quieres probar una expresión matemática y conocer su valor, escríbela en la caja de texto de la izquierda en la zona de evaluación de expreiones (figura 4.1) y presiona el botón con el signo =. Verás el resultado en la caja de texto de la derecha. Si la expresión es incorrecta aparecerá la palabra ERROR. El evaluador de expresiones también sirve para conocer el valor de una variable. Sólo escribe en él el nombre de la misma y presiona con el mouse el botón = de la pantalla. Ten en cuenta, que una vez que agregas una variable, ésta pasa a ser un objeto utilizable en las expresiones matemáticas (por ejemplo, si la variable NUM vale 4, la expresión NUM * 2 en el evaluador dará por resultado 8). Al asignar una expresión a una variable también puedes utilizar dentro de la expresión otras variables y escribir cosas como "(NUM * 2) + 4" o "HOLA + 3 * NUM", donde NUM y HOLA son nombres de variables ya creadas en sus respectivos bloques.

Figura 4.6.A: Asignación de valores a variables.

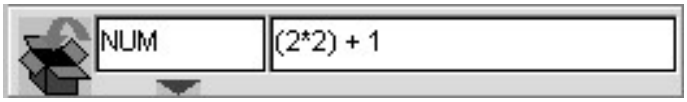


Figura 4.6.B: Asignación de expresiones a variables.

Con todo esto ya casi podemos escribir nuestro programa que ejecuta un ciclo un número determinado de veces. La figura 4.7 lo muestra. Allí la variable J es inicializada en 0 (o sea que su valor inicial será 0). Al entrar al ciclo se encenderá la salida 1 de relé, luego un temporizador hará que el programa espere unos segundos y luego la salida se apagará. Tras una nueva espera debida al siguiente temporizador, el programa entrará en un bloque de **decisión**. El bloque decisión consta de tres partes: un objeto a ser evaluado (caja de texto izquierda), un operador (caja central) que puede ser un = (igual que), un <> (distinto que), un <= (mayor o igual que), etc. (Nota: La lista de operadores disponibles cambia según el objeto seleccionado para ser evaluado, ya que algunos operadores no tienen sentido con algunos objetos.) y una expresión (caja de texto de la derecha). La expresión puede ser cualquier expresión matemática soportada por Minibloques, el operador sólo puede ser uno de los que figuran en la lista desplegable de la caja de texto central y el objeto sólo puede ser uno de los incluídos en la lista desplegable de la caja de la izquierda. Una vez que el programa llega al bloque

de decisión, pregunta si se cumple la condición establecida por los tres elementos mencionados. Por ejemplo, si el objeto fuese una variable (en nuestro programa será "J"), el operador fuese "=" y la expresión fuere "4", la condición sólo se cunplirá si J vale 4. Si la condición se cumple, todo lo que esté entre el bloque decisión y el bloque fin de decisión se ejecutará, pero si la condición es falsa, el programa "saltará" directamente al bloque de fin de decisión sin ejecutar lo que esté entre ambos bloques. Así, en nuestro ejemplo, antes del bloque decisión la variable J se incrementa (ya que el bloque variable dice: J = J + 1) y luego la decisión pregunta si J vale 10 o no. Si J vale 10, entonces se ejecuta lo que está adentro de la decisión y esto no es ni más ni menos que una salida de ciclo, por lo que el programa saldrá del ciclo una vez que J haya alcanzado el valor 10, y no antes. Es así como el ciclo se ejecutará 10 veces y concluirá. El siguiente cuadro muestra el equivalente en LOGO de este programa:

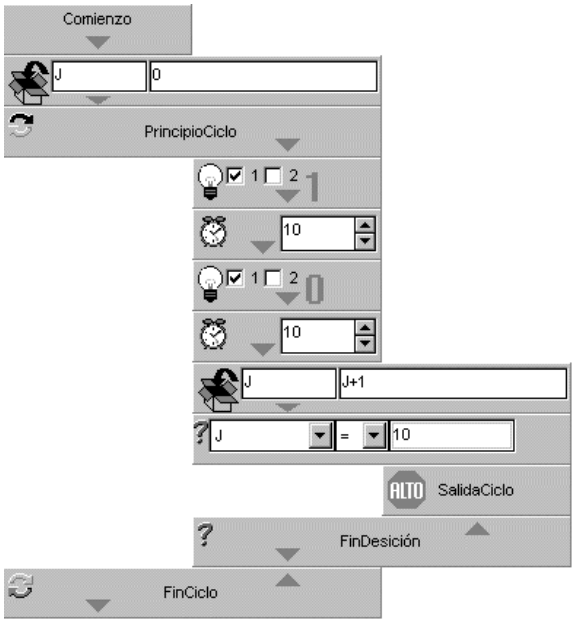


Figura 4.7: Variaable utilizada como contador.

```
REPITE 10 [
    EscribeSalida 1 1
    Espera 10
    EscribeSalida 1 0
    Espera 10
]
```

Otro uso de los ciclos, las decisiones y las variables es el de esperar hasta que ocurra algo. Si tenemos un vehículo cuyo motor está conectado a la salida 1 de la interfaz, y hemos colocado un sensor de choque en su frente, querremos que el vehículo avance hasta que el sensor detecte un choque con algún obstáculo. Un posible programa para lograr esto es el que muestra la figura 4.8. Junto a ella está su equivalente en PASCAL.

```
EscribeMotor(1, 1);
While TRUE do
begin
      if LeeSensor(1) then
            Exit;
end;
EscribeMotor(1, 0);
```

Como podrás ver, Minibloques es un completo lenguaje de programación que te permite controlar todas las capacides de la interfaz didáctica i-723 como si estuvieras usando un lenguaje convencional, pero de manera mucho más fácil.

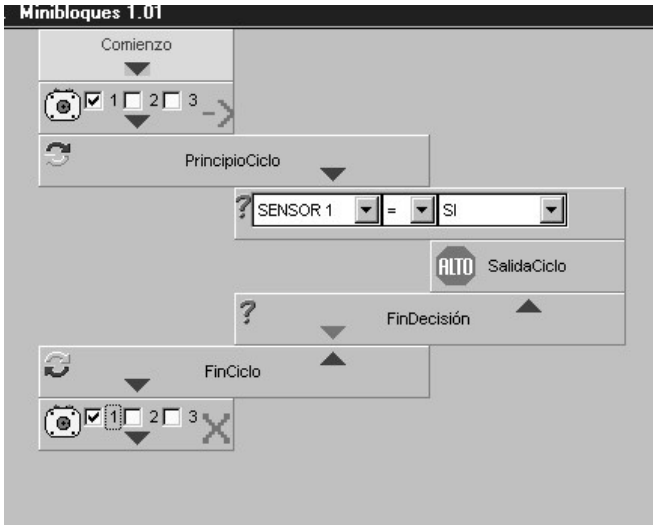


Figura 4.8: Programa donde se detecta un choque con el sensor 1 y se detiene el motor1 tras la detección.

CAPÍTULO V:

PROGRAMACIÓN AVANZADA DE LA INTERFAZ

Una de las características sobresalientes de la i-723, es que la misma puede ser controlada utilizando una gran cantidad de lenguajes de programación o sistemas de autor muy distintos entre sí. En los directorios del Disco de Librerías (disco 2/2) podrás encontrar rutinas de programación para los lenguajes indicados a continuación:

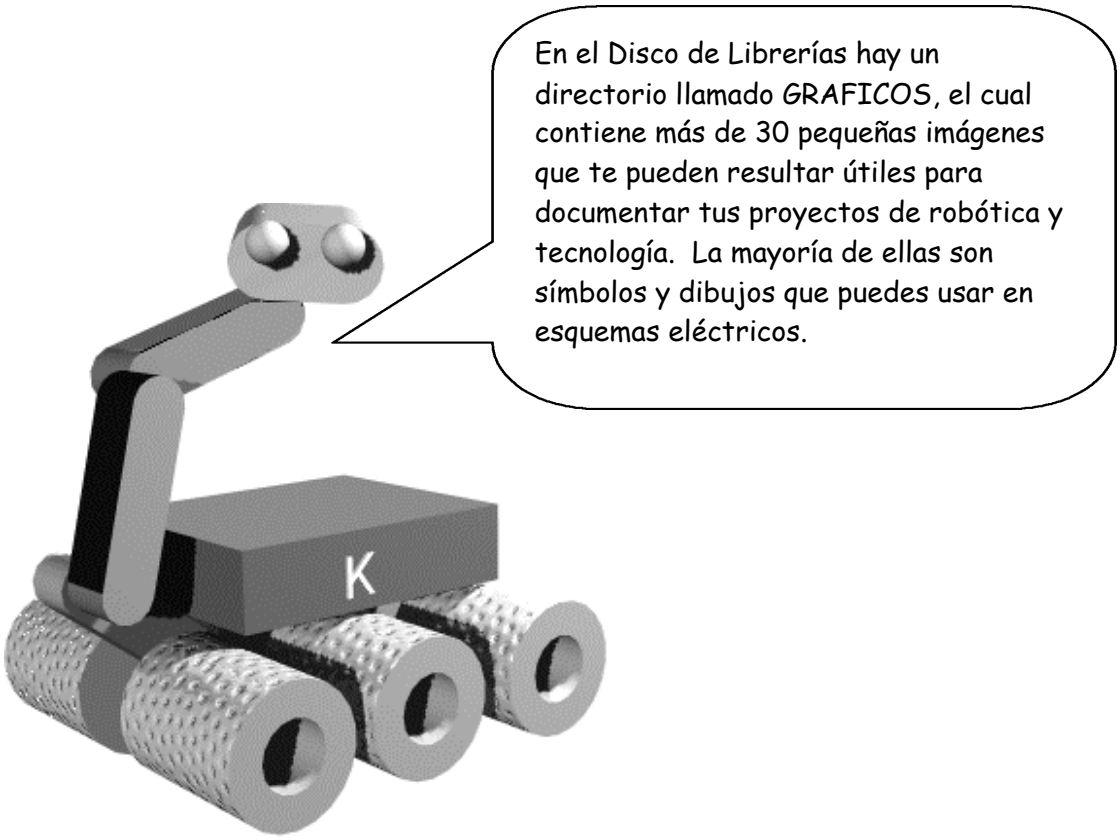
Lenguaje:	Directorio:
Logo Writer	LWR
Turbo Pascal y Object Pascal	TPASCAL
QBasic	QBAS
Lenguaje C	TC
Turbo Prolog	PROLOG
Delphi 1.0 (16 bits)	DELPH16
Delphi 2.0 (32 bits)	DELPH32
Visual Basic 3.0 y 4.0-16 bits	VBAS16
Visual Basic 4.0-32 bits	VBAS32
Word 97 y Visual Basic 5.0	Office97
ToolBook	MTBOOK
Otros lenguajes y sistemas de autor para Windows 3.1x como por ejemplo: C++, Paradox, VBA, dBase, y cualquier lenguaje que soporte llamadas a DLLs de 16 bits.	WIND16
Otros lenguajes y sistemas de autor para Windows 95 ó 98 que soporten llamadas a DLLs de 32 bits.	WIND32

En varios de los lenguajes soportados (como LOGO, QBASIC, PASCAL…), se incluye el **código fuente de todas las rutinas de control** por lo que cualquiera que desee agregar soporte para otros lenguajes, podrá hacerlo convirtiendo las mismas a la sintaxis del lenguaje deseado. Esto agrega flexibilidad a la interfaz, no quedando restringido su uso y programación a un número fijo de lenguajes. El usuario es libre, además, de distribuir sus programas con las rutinas de la interfaz, siempre y cuando respete los derechos de propiedad de los fabricantes del lenguaje utilizado.
A partir de aquí , se describirán las órdenes de control en los distintos lenguajes soportados, comenzando por aquellos que funcionan bajo DOS y continuando con los de Windows. Si utilizas otro lenguaje o sistema de autor que no figura en la lista dada, puedes convertir las funciones y procedimientos aquí provistos. **No dudes en consultar a Simple Informática** , ya que continuamente se incorpora soporte a nuevos lenguajes y sistemas de autor. No dejes de trabajar con tu lenguaje preferido por el mero hecho de que éste no esté incluido entre los soportados en el Disco de Librerías.

Es imporatante mencionar que las órdenes de control de la interfaz son exactamente las mismas para todos los lenguajes, tanto en Windows como en DOS. De todos modos, en algunos lenguajes se agregan algunas facilidades para aprovechar mejor las capacidades del lenguaje en sí, pero esto no quita el hecho de que la sintaxis sea igual, sin importar con qué se esté programando. El siguiente cuadro muestra el conjunto básico de instrucciones para controlar todas las funciones de la i-723:

Función genérica:	Nombre:
Inicialización:	Inicializar
Control de motores:	EscribeMotor LeeMotor
Control de las salidas de relé:	EscribeSalida LeeSalida
Sensores:	LeeSensor
Otras:	ApagaTodo

Lo único que varía en los distintos lenguajes de programación es la forma de llamar a estos comandos y la manera en que las librerías que los contienen son agregadas al mismo. Las particularidades de cada lenguaje de programación se describen a lo largo de todo este capítulo en los apartados correspondientes. Busca entre ellos el lenguaje de tu preferencia y pronto estarás programando la interfaz y automatizando tus proyectos.



V. A) DOS

Logo Writer

En el directorio LWR del Disco de Librerías está el archivo LWR723.LWR. Allí se incluyen las primitivas que posibilitan el manejo de la interfaz desde Logo Writer. Para cargarlo, es necesario copiarlo en el directorio en donde se halla el Logo Writer. De este modo, cuando inicies Logo Writer, aparecerá en la lista de las páginas una nueva, llamada LWR723. Posiciónate sobre ella y presiona la tecla <ENTER> como si cargaras una página cualquiera. En el "Revés" (la pantalla de Logo Writer a la que se accede al presionar las teclas <CTRL> + <D>) encontrarás el código fuente de las primitivas mencionadas. Para utilizarlas, invócalas al igual que lo haces con cualquier primitiva común de Logo.

Con estas órdenes es posible aprovechar todas las capacidades de la i-723. A continuación se explica cada una. Casi todas necesitan de parámetros. En todas las explicaciones se listan los posibles valores de los mismos.

Inicializar

Sintaxis:

Inicializar :puerto

Abreviatura:

No tiene.

Función:

Inicializa el puerto a donde se halla conectada la interfaz, así como las variables globales necesarias para el funcionamiento de la misma. Es **imprescindible** que se la invoque una vez antes de comenzar a trabajar. Es también muy recomendable que se incluya en los programas al principio, para no tener que ejecutarla manualmente.

Ejemplo:

La siguiente línea prepara el entrono de Logo Writer para trabajar con la interfaz conectada al puerto paralelo Lpt 1:

```
Inicializar 1
```

Parámetros:

El parámetro :puerto es un número entre 1 y 3 que indica en qué puerto paralelo está conectada la interfaz. Si no conoces el número de puerto, prueba los 3 valores posibles hasta que la interfaz responda a las órdenes. IMPORTANTE: la interfaz no funciona hasta que la primitiva Inicializar es llamada, ya que el Logo no puede "saber" en qué puerto está conectada la misma.

EscribeMotor

Sintaxis:

EscribeMotor :mot :accion

Abreviatura:

EMot.

Función:

Hace girar el motor indicado por el parámetro :mot en la dirección indicada.

Ejemplo:

En este ejemplo, el motor 1 girará alternativamente hacia la derecha y hacia la izquierda 10 veces y luego se apagará. A modo de aplicación, si el motor 1 de este ejemplo controlara la dirección de un pequeño vehículo, éste describiría un "zigzag" girando 10 veces hacia un lado y el otro.

```
para ZigZag
        repite 10 [EscribeMotor 1, 1 espera 10 EscribeMotor 1, 2 espera 10]
        EscribeMotor 1, 0
Fin
```

Parámetros:

El parámetro :mot es un número del 1 al 3, (o varios números juntos si se quiere controlar más de un motor a la vez). El parámetro :accion es para indicar el sentido de giro o la detención del motor. Este parámetro puede valer 1 (luz de motor encendida en rojo), 2 (luz de motor encendida en verde) o 0 (motor detendido). Las siguientes son formas válidas (nóta que no hay espacio entre los números que identifican los motores a ser accionados):

```
EscribeMotor 12, 1    (Giran los motores 1 y 2)
EscribeMotor 3, 2     (Gira el motor 3 en sentido inverso al del ejemplo anterior)
EscribeMotor 123, 0 (Apaga los tres motores)
EscribeMotor 231, 2 (Giran los tres motores hacia un lado)
EscrtibeMotor 23, 1 (Giran los motores 3 y 2 en sentido inverso al  del ejemplo anterior)
```

LeeMotor

Sintaxis:

LeeMotor :mot

Abreviatura:

LMot.

Función:

Lee el estado actual del motor por el cual se le "pregunta" y devuelve un número que puede ser 0, 1 ó 2 según éste (el motor) se encuentre detenido, girando hacia la derecha o hacia la izquierda, respectivamente.

Ejemplo:

La siguiente línea hará girar el motor hacia en la dirección 1 sólo si éste ya se encontraba girando en la dirección 2:

```
si (LeeMotor 1) =  2  [EscribeMotor 1, 1]
```

Parámetros:

El único parámetro que se le pasa es un número del 1 al 3 indicando el motor a ser evaluado.

Valor devuelto:

La primitiva LeeMotor retorna un número del 0 al 2 siendo:
0 = motor detenido.

1 = motor girando hacia la derecha.
2 = motor girando hacia la izquierda.

EscribeSalida

Sintaxis:

EscribeSalida :sal :accion

Abreviatura:

ESal.

Función:

Encender o apagar la salida que se le pasa como parámetro.

Ejemplo:

Este ejemplo mantiene encendida durante un pequeño lapso de tiempo la salida 1 de la i-723, luego la apaga y enciende la salida 2 manteniéndola prendida durante el mismo tiempo. Finalmente apaga la salida 2 y se vuelve a ejecutar el procedimiento hasta que alguien lo detenga. Suponiendo que haya una luz roja en la salida 1 y una verde en la salida 2, este pequeño programa simularía un semáforo.

```
para semaforo
EscribeSalida 1, 1  espera 30
EscribeSalida 1, 0
EscribeSalida 2, 1  espera 30
EscribeSalida 2, 0
semaforo
fin
```

Parámetros:

El parámetro :sal es un número que indica la/s salida/s a ser accionada/s. El número puede ser 1 ó 2 (para prender o apagar una sola salida) ó 12 (o 21) para accionar ambas. El parámetro :accion puede ser 1 ó 0 según se desee encender o apagar la/s salida/s indicada/s. Algunos valores posibles son:

EscribeSalida 12, 1 (Enciende ambas salidas)
EscribeSalida 21, 1 (Igual al ejemplo anterior, ya que el orden de los números no importa)
EscribeSalida 2, 0 (Apaga la salida 2)

LeeSalida

Sintaxis:

LeeSalida :sal

Abreviatura:

LSal

Función:

Lee el estado actual de la salida por la cual se le "pregunta" y devuelve un número que puede ser 0 (salida activada) ó 1 (salida desactivada).

Ejemplo:

La primer línea del siguiente ejemplo encenderá la salida 1 sólo si ésta se encontraba apagada. La segunda línea hará girar el motor 1 en la dirección 2 sólo si la salida 2 se encuentra encendida:

```
si (LeeSalida 1) = 0  [EscribeSalida 1, 1]
si (LeeSalida 2) =  1 [EscribeMotor 1, 2]
```

Parámetros:

El único parámetro que se le pasa es un número del 1 al 2 indicando la salida a ser evaluada.

Valor devuelto:

La primitiva LeeSalida retorna un número que puede ser 0 ó 1, siendo:
0 = salida desactivada.
1 = salida activada.

LeeSensor

Sintaxis:

LeeSensor :sen

Abreviatura:

LSen

Función:

Lee el estado actual del sensor por el cual se le "pregunta" y devuelve un estado lógico que puede ser FALSO (sensor no excitado) ó CIERTO (sensor excitado).

Ejemplo:

Suponiendo que el motor 1 mueve un vehículo hacia adelante y que el sensor 1 es un sensor de choque colocado en el paragolpes, el programa hará moverse el vehículo hasta que choque con algo.

```
para NoChocar
        EscribeMotor 1, 1
        repite 1000 [si (LeeSensor 1) [EscribeMotor 1, 0]]
fin
```

Parámetros:

El parámetro es un número del 1 al 4 indicando el sensor a ser leído.

Valor devuelto:

Esta primitiva devuelve un estado lógico que puede ser CIERTO ó FALSO, siendo:

CIERTO = sensor excitado.
FALSO = sensor no excitado.

ApagaTodo

Sintaxis:

ApagaTodo

Sintaxis:

ATod

Función:

Como su nombre lo indica, esta primitiva apaga todas las salidas (tanto los motores como las salidas de relé).

Ejemplo:

Los motores 1, 2 y 3 se moverán hasta que termine de ejecutarse la instrucción espera 50. Luego se detendrán (recuérdese que las salidas de relé también serán desactivados por la orden ApagaTodo).

```
para MueveAuto
        EscribeMotor 12, 2
        espera 50
        ApagaTodo
fin
```

Parámetros:

Esta primitiva no recibe parámetros.

QBasic

Las funciones y procedimientos para QBasic se encuentran en el archivo QB723.BAS del directorio QBAS del Disco de Librerías. Para trabajar con ellas, es conveniente que sigas los siguientes pasos:

1) Crea un directorio por cada programa nuevo que realices para la interfaz. Por ejemplo, podrías crear un directorio en el disco rígido C: llamado C:\ROBOTICA y dentro del mismo agregar un directorio por programa, de modo que finalmente quedaría una estructura como la que sigue:

```
C:\ROBOTICA\PROGR1
C:\ROBOTICA\PROGR2
```

En cada uno de estos directorios coloca una copia del archivo QB723.BAS.

2) Para cada nuevo programa, abre este archivo desde el menú de QBASIC y trabaja sobre este archivo. Para grabar tu programa, utiliza "GRABAR COMO" (o "SAVE AS" en inglés) y guarda tu programa con el nombre que más te guste.

Trabajando de esta manera, podrás tener acceso a todas las rutinas de programación para QBASIC desde cada programa que realices, a la vez que mantendrás un orden en tus proyectos. A continuación se da la descripción de las rutinas:

Inicializar

Sintaxis:

Sub Inicializar (PortID As Integer)

Función:

Este procedimiento inicializa el sistema para trabajar con la interfaz. Es **imprescindible** que se lo llame al comienzo de la ejecución del programa.

Ejemplo:

Se inicializa el sistema suponiendo que la interfaz se halla conectada en el puerto paralelo Lpt 1. Si ésta se encontrara en el puerto Lpt 2, se escribiría "Inicializar 2".

```
Inicializar 1
```

Parámetros:

Este procedimiento recibe un solo parámetro, el cual es de tipo integer y es el identificador del puerto al cual se halla conectada la interfaz. Puede ser 1, 2 ó 3.

EscribeMotor

Sintaxis:

Sub EscribeMotor (MotorID As Integer, Dir As integer)

Función:

Accionar el motor indicado en el parámetro que se le pasa. La acción a realizar por el motor es una de las tres siguientes: Girar hacia la derecha, girar hacia la izquierda o detenerse. El segundo parámetro es el que establece la acción a ser realizada.

Ejemplo:

El motor 3 girará hacia la izquierda:

```
EscribeMotor 3, 2
```

Parámetros:

EscribeMotor recibe dos parámetros: el número (integer) identificador del motor (puede ser 1, 2 ó 3) y la orden (integer) a ejecutarse (1 para girar hacia la derecha; 2 para girar hacia la izquierda; 0 para detenerse).

EscribeSalida

Sintaxis:

Sub EscribeSalida (ReleID As Integer, Accion As Integer)

Función:

Este procedimiento enciende o apaga la salida que se le pasa como parámetro.

Ejemplo:

```
EscribeSalida 1, 1
```

Parámetros:

EscribeSalida también recibe dos parámetros: el número de la salida a ser accionada (integer: 1 ó 2) y la acción (integer: 1 para encender y 0 para apagar la salida de relé en cuestión).

ApagaTodo

Sintaxis:

Sub ApagaTodo ()

Función:

Como su nombre lo indica, esta primitiva apaga todas las salidas (tanto los motores como las salidas de relé).

Ejemplo

```
ApagaTodo
```

Parámetros:

Esta primitiva no recibe parámetros.

LeeSalida

Sintaxis:

Function LeeSalida (ReleID As Integer) As Integer

Función:

Esta función lee el estado actual de la salida de relé que se le pasa como parámetro.

Ejemplo:

Se imprime un 1 ó un 0 en la pantalla según esté la salida 2 encendida o apagada:

```
print LeeSalida(2)
```

Parámetros:

LeeSalida recibe un parámetro del tipo integer que identifica a la salida de relé a ser evaluada (éste puede ser 1 ó 2).

Valor de vuelto:

Es retornado un integer cuyo valor puede ser 0 (salida desactivada) ó 1 (salida activada).

LeeMotor

Sintaxis:

Function LeeMotor (MotorID As Integer) As Integer

Función:

Esta función lee el estado actual de la salida de motor que se le pasa como parámetro.

Ejemplo:

Se imprimirá un 0 en caso de que el motor 3 esté detenido, un 1 si éste se halla girando hacia la derecha o un 2 si está girando hacia la izquierda.

```
print LeeMotor(3)
```

Parámetros:

LeeMotor recibe un parámetro del tipo integer el cual identifica a la salida de motor a ser evaluada. El mismo puede ser 1, 2 ó 3.

Valor de vuelto:

Es retornado un integer cuyo valor puede ser 0 (motor desactivado), 1 (motor girando hacia la derecha) ó 2 (motor girando hacia la izquierda) .

LeeSensor

Sintaxis:

Function LeeSensor (Sensor As Integer) As Integer

Función:

Esta función lee el estado actual del sensor que se le pasa como parámetro.

Ejemplo 1:

El estado del sensor 4 es leído, guardado en la variable "retorno" y luego impreso:

```
Dim retorno as integer
retorno = LeeSensor(4)
Print Str$(retorno)
```

Ejemplo 2: Imprime un mensaje según sea el estado del sensor 3:

```
If LeeSensor(3) Then
        Print "El sensor 3 está activado"
Else
        Print "El sensor 3 está desactivado"
End If
```

Parámetros:

Recibe un parámetro también del tipo entero que identifica al sensor. Este número puede ser 1, 2 , 3 ó 4.

Valor devuelto:

Devuelve un valor de tipo integer, el cual puede ser 1 ó 0, e indica el estado del sensor a evaluar.

Pascal

En el directorio TPASCAL del Disco de Librerías están las funciones para Pascal, así también como el código fuente de las mismas. Todas se hallan en el archivo TP723.PAS. El compilador elegido es Turbo Pascal, pero en caso de que desees trabajar con otro compilador Pascal, es relativamente sencillo el pasaje del código. Si lo deseas, puedes encapsular todas estas rutinas en una unidad (unit) para hacer más fácil la inclusión de las mismas en tus programas. Lo que sigue es la descripción de cada una de ellas:

Inicializar

Sintaxis:

Procedure Inicializar(PortID :integer);

Función:

Inicializa el sistema y el puerto de salida pasado como parámetro para trabajar con la i-723.

Ejemplo:

Intenta inicializar la i-723 en el puerto paralelo Lpt 1
Inicializar(1);

Parámetros:

Como parámetro recibe un número entero (integer) que puede ser 1, 2 ó 3 y que identifica al puerto paralelo al que se halla conectada la i-723.

ApagaTodo

Sintaxis:

Procedure ApagaTodo;

Función:

Apaga todas las salidas (tanto las de relé como las de motor).

Ejemplo:

ApagaTodo;

Parámetros:

No recibe parámetros.

EscribeMotor

Sintaxis:

Procedure EscribeMotor(MotorID :integer; Accion :integer);

Función:

Acciona el motor o los motores conectados a la salida cuyo número es pasado como parámetro.

Ejemplo 1:

El motor 1 girará hacia la derecha.
EscribeMotor(1, DA);

Ejemplo 2:

Todos los motores girarán hacia la derecha.

```
EscribeMotor(123, DA);        ó
EscribeMotor(231, DA)    {El orden de los números no altera el funcionamiento}
```

Ejemplo 3:

Los motores 1 y 2 girarán hacia la izquierda.

```
EscribeMotor(21, IZ);
```

Ejemplo 4:

Los motores 1 y 3 se detendrán.

```
EscribeMotor(13, ALTO);
```

Parámetros:

Se le pasan dos parámetros: MotorID y Accion. MotorID es del tipo integer y es el identificador del/los motor/es a ser accionados. Son valores válidos los siguientes:

1 (Acciona el motor 1). 12 ó 21 (Accionan los motores 1 y 2)
2 (Acciona el motor 2). 23 ó 32 (Accionan los motores 2 y 3)
3 (Acciona el motor 3). 13 ó 31 (Accionan los motores 1 y 3)
123, 132, 231, 213, 312 ó 321 (Accionan todos los motores)

Accion es del tipo integer y puede ser 1 (constante DA), 2 (constante IZ) ó 0 (constante ALTO).

LeeMotor

Sintaxis:

Function LeeMotor(MotorID :integer) :integer;

Función:

Evalúa el estado del motor conectado a la salida cuyo número es pasado como parámetro.

Ejemplo:

Lee el estado del motor 2 y lo imprime en pantalla:

```
if  LeeMotor(2) = ALTO then
        writeln('Detenido.');
if  LeeMotor(2) = DA then
        writeln('Girando hacia la derecha.');
if  LeeMotor(2) = IZ then
        writeln('Girando hacia la izquierda.');
```

Parámetros:

El único parámetro que recibe es un integer e identifica al motor a evaluar. Puede ser 1, 2 ó 3.

Valor devuelto:

El valor devuelto es 1 (DA), 2 (IZ) ó 0 (ALTO) .

EscribeSalida

Sintaxis:

Procedure EscribeSalida(ReleID :integer, Accion :integer);

Función:

Acciona la salida o las salidas de relé cuyo número es pasado como parámetro.

Ejemplo 1:

La salida 1 es activada:
EscribeSalida(1, SI);

Ejemplo 2:

Las dos salidas de relé son desactivadas:
EscribeSalida(12, NO);

Parámetros:

Recibe dos parámetros: El primero es un integer que identifica la salida a accionar. Puede ser 1, 2, 12 ó 21 (los dos últimos valores accionan las dos salidas a la vez). El segundo es la acción a ser realizada por esa salida: 1 (SI) ó 0 (NO).

LeeSalida

Sintaxis:

Function LeeSalida(ReleID: integer):boolean;

Función:

Evalúa el estado actual de la salida de relé cuyo número es pasado como parámetro.

Ejemplo:

Lee el estado de la salida 1 y lo imprime en pantalla:

```
if  LeeSalida(1) then
        writeln('Prendida')
else
        writeln('Apagada');
```

Parámetros:

El único parámetro que recibe es un integer e identifica la salida a evaluar. Puede ser 1 ó 2.

Valor devuelto:

El valor devuelto es de tipo boolean: True (salida encendida) ó False (salida apagada).

LeeSensor

Sintaxis:

Function LeeSensor(SensorID: integer):boolean;

Función:

Evalúa el estado del sensor conectado a la entrada cuyo número es pasado como parámetro.

Ejemplo:

Lee el estado del sensor 4 e imprime en pantalla un 1 ó un 0 según esté excitado o no:

```
if  LeeSensor(4) then
        writeln('1')
else
        writeln('0');
```

Parámetros:

El parámetro que recibe es un integer e identifica al sensor a evaluar. Puede ser 1, 2, 3 ó 4.

Valor devuelto:

El valor devuelto es de tipo boolean: TRUE (sensor excitado) o FALSE (sensor no excitado).

Lenguaje C

En el directorio TC del Disco de Librerías están las funciones para C, así también como el código fuente de las mismas. Todas están contenidas en el archivo TC723.C, el cual está escrito como para ser compilado directamente con alguna versión de Turbo C, o portadas para trabajar con otro compilador. Es posible compilarlo en forma de archivo .lib, hacer un header (.h) si se lo desea, o incluirlas directamente en los programas que vallan a trabajar con la interfaz con la directiva INCLUDE. En el archivo TC723.C encontrarás igualmente algunas constantes de utilidad, como ser: SI=1, NO=0, ALTO=0, DA=1, IZ=2. Éstas harán más legibles tus programas de control. A continuación se describen las funciones provistas:

Inicializar

Sintaxis:

void Inicializar(int PortID)

Función:

Inicializa el sistema y el puerto de salida pasado como parámetro para trabajar con la i-723.

Ejemplo 1:

Intenta inicializar la i-723 en el puerto paralelo Lpt 1

```
Inicializar(1);
```

Parámetros:

Como parámetro recibe un número entero (int) que puede ser 1, 2 ó 3 y que identifica al puerto paralelo al que se halla conectada la i-723.

ApagaTodo

Sintaxis:

void ApagaTodo(void);

Función:

Apaga todas las salidas (tanto las de relé como las de motor).

Ejemplo:

```
ApagaTodo();
```

Parámetros:

No recibe parámetros.

EscribeMotor

Sintaxis:

void EscribeMotor(int MotorID, int Accion);

Función:

Acciona el motor o los motores conectados a la salida cuyo número es pasado como parámetro.

Ejemplo 1:

El motor 1 girará hacia la derecha.

```
EscribeMotor(1, DA);
```

Ejemplo 2:

Todos los motores girarán hacia la derecha.

```
EscribeMotor(123, DA);        ó
EscribeMotor(231, DA)    /*El orden de los números no altera el funcionamiento. */
```

Ejemplo 3:

Los motores 1 y 2 girarán hacia la izquierda.

```
EscribeMotor(21, IZ);
```

Ejemplo 4:

Los motores 1 y 3 se detendrán.

```
EscribeMotor(13, ALTO);
```

Parámetros:

Se le pasan dos parámetros: MotorID y Accion. MotorID es de tipo entero y es el identificador del/los motor/es a ser accionado/s. Son valores válidos los siguientes:

1 (Acciona el motor 1). 12 ó 21 (Accionan los motores 1 y 2)
2 (Acciona el motor 2). 13 ó 31 (Accionan los motores 1 y 3)
3 (Acciona el motor 3). 23 ó 32 (Accionan los motores 2 y 3)
123, 132, 231, 213, 312 ó 321 (Accionan todos los motores)

Accion es de tipo entero y puede ser 1 (constante DA), 2 (constante IZ) ó 0 (constante ALTO).

EscribeSalida

Sintaxis:

void EscribeSalida(int ReleID, int Accion);

Función:

Acciona la salida o las salidas de relé cuyo número es pasado como parámetro.

Ejemplo 1:

La salida 1 es activada:

```
EscribeSalida(1, SI);
```

Ejemplo 2:

Las dos salidas de relé son desactivadas:

```
EscribeSalida(12, NO);
```

Parámetros:

Recibe dos parámetros: El primero es un integer que identifica la salida a accionar. Puede ser 1, 2, 12 ó 21 (los dos últimos valores accionan las dos salidas a la vez). El segundo es la acción a ser realizada sobre esa salida: 1 (constante SI) ó 0 (contante NO).

LeeMotor

Sintaxis:

int LeeMotor(int MotorID)

Función:

Evalúa el estado actual de la salida de motor cuyo número es pasado como parámetro.

Ejemplo:

Lee el estado del motor 2 y lo imprime en pantalla:

```
if  (LeeMotor(2) == ALTO) printf("Detenido.");
if  (LeeMotor(2) == DA) printf("Girando hacia la derecha.");
if  (LeeMotor(2) == IZ) printf("Girando hacia la izquierda.");
```

Parámetros:

El parámetro que recibe es de tipo entero e identifica al motor a evaluar. Puede ser 1, 2 ó 3.

Valor devuelto:

El valor devuelto es 1 (DA), 2 (IZ) ó 0 (ALTO) .

LeeSalida

Sintaxis:

int LeeSalida(int ReleID);

Función:

Evalúa el estado actual de la salida de relé cuyo número es pasado como parámetro.

Ejemplo:

Lee el estado de la salida 1 y lo imprime en pantalla:

```
if  LeeSalida(1) {printf("Prendida");}
else {printf('Apagada');}
```

Parámetros:

El único parámetro que recibe es integer e identifica la salida a evaluar. Puede ser 1 ó 2.

Valor devuelto:

El valor devuelto es 1 (SI) ó 0 (NO) .

LeeSensor

Sintaxis:

int LeeSensor(int SensorID);

Función:

Evalúa el estado actual de la entrada de sensor cuyo número es pasado como parámetro.

Ejemplo:

Lee el estado del sensor 4 y activa la salida 2 si el sensor está exitado:

```
if  (LeeSensor(4) ==1) EscribeSalida(2, SI);
```

Parámetros:

El único parámetro que recibe un entero e identifica al sensor a evaluar. Puede ser 1, 2, 3 ó 4.

Valor devuelto:

El valor devuelto es 1 (sensor excitado) ó 0 (sensor no excitado).

Prolog

El Prolog es un lenguaje especialmente indicado para trabajar en Inteligencia Artificial. Quien desee controlar dispositivos mediante la interfaz i-723 utilizándolo, dispone de los predicados necesarios en el archivo PRO723.PRO del directorio PROLOG del Disco de Librerías provisto con la interfaz. A pesar de las diferencias existentes entre este lenguaje y los "lenguajes convencionales", se conserva la sintaxis unificada para el control de la interfaz. Insertando este archivo en tus programas Prolog (puedes hacer esto utilizando, por ejemplo, la directiva INCLUDE) agregas todos los predicados necesarios para el trabajo con la interfaz. El compilador Prolog elegido fue Turbo Prolog, pero como se suministra todo el código fuente no tendrás problemas en la adaptación a otros compiladores, de los predicados provistos. A continuación se describen los mismos:

inicializar

Sintaxis:

inicializar(integer)

Función:

Inicializa el sistema y el puerto de salida pasado como parámetro para trabajar con la i-723.

Ejemplo:

Las siguientes líneas preparan al entorno para trabajar con la i-723 conectada al puerto paralelo Lpt 1.

```
goal: inicializar(1)
yes
```

Parámetros:

Como parámetro recibe un número entero (integer) que puede ser 1, 2 ó 3 y que identifica al puerto paralelo al que se halla conectada la i-723.

apagatodo

Sintaxis:

apagatodo

Función:

Apaga todas las salidas (tanto las de relé como las de motor).

Ejemplo:

```
goal: apagatodo
yes
```

Parámetros:

No recibe parámetros.

escribemotor

Sintaxis:

escribemotor(integer, integer)

Función:

Acciona el motor conectado a la salida cuyo número es pasado como parámetro.

Ejemplo:

El motor 1 girará hacia la derecha, el motor 2 se detendrá y el motor 3 girará hacia la izquierda:

```
goal: escribemotor(1, 1), escribemotor(2, 0), escribemotor(3, 2)
yes
```

Parámetros:

El primer parámetro es el número de motor y el segundo es la acción a realizar por ese motor. Ambos son números enteros (integer). El primero puede ser 1, 2 ó 3 (indicando el número de motor) y el segundo 0 (detención), 1 (giro hacia la derecha) o 2 (giro hacia la izquierda).

escribesalida

Sintaxis:

escribesalida(integer, integer)

Función:

Acciona la salida o las salidas de relé cuyo número es pasado en el primer parámetro.

Ejemplo 1:

La salida 1 es activada y la salida 2 es desactivada:

```
goal: escribesalida(1, 1), escribesalida(2, 0)
yes
```

Ejemplo 2:

Las dos salidas de relé son activadas:

```
goal: escribesalida(12, 1)
yes
```

Parámetros:

Recibe dos parámetros: El primero es un integer que identifica la/s salida/s a accionar. Puede ser 1, 2, 12 ó 21 (los dos últimos valores accionan las dos salidas a la vez sin importar el orden). El segundo es la acción a ser realizada sobre esa salida: 1 (activa salida) o 2 (desactiva salida).

leemotor

Sintaxis:

leemotor(integer, integer)

Función:

Evalúa el estado actual del motor conectado a la salida cuyo número es pasado como primer parámetro.

Ejemplo:

Lee el estado del motor 2 y lo imprime en pantalla. En el ejemplo se supone que el motor estaba girando hacia la derecha, por lo que queda X = 1:

```
goal: leemotor(2, X), write(X)
X = 1
```

yes

Parámetros:

El primer parámetro es un número entero que va de 1 a 3 y que indica el número del motor cuyo estado se desea conocer. El segundo es la variable en la que se obtendrá dicho estado. El valor de la misma luego de la ejecución del predicado será 0 si el motor estaba detenido, 1 si el motor giraba hacia la derecha o 2 si el motor giraba hacia la izquierda.

leesalida

Sintaxis:

leesalida(integer, integer)

Función:

Evalúa el estado actual de la salida de relé cuyo número es pasado como primer parámetro.

Ejemplo:

Lee el estado de la salida 1 y lo imprime en pantalla. En el ejemplo se supone a la salida 1 desactivada, por lo que queda X = 0:

```
goal: leesalida(1, X), write(X)
X = 0
yes
```

Parámetros:

El primer parámetro que recibe es del tipo integer e identifica la salida a evaluar. Puede ser 1 ó 2. El segundo parámetro es la variable en la que se obtiene el estado de la salida evaluada. El valor de la misma, luego de la ejecución de este predicado, podrá ser 0 (salida desactivada) o 1 (salida activada).

leesensor

Sintaxis:

leesensor(integer, integer);

Función:

Evalúa el estado actual del sensor conectado a la entrada cuyo número es pasado como primer parámetro.

Ejemplo:

Lee el estado del sensor 4 y lo imprime en pantalla. En el ejemplo se supone al sensor 4 activado, por lo cual la queda ESTADO = 1:

```
goal: leesensor(4, ESTADO), write(ESTADO)
ESTADO = 1
yes
```

Parámetros:

El primer parámetro que recibe es de tipo entero e identifica al sensor a evaluar. Puede ser 1, 2, 3 ó 4. El segundo es la variable en la que se obtiene el estado del sensor. Puede tomar un valor igual a 0 (sensor desactivado) o igual a 1 (sensor activado).

V. B) WINDOWS 3.1x (16 BITS)

La programación bajo el entorno Windows es mucho más común hoy que algún tiempo atrás. Con la i-723 se provee soporte a varios lenguajes para Windows. En este apartado, se describirá la programación bajo Windows 3.1x con lenguajes de 16 bits. En el apartado IV.C, se hablará del soporte a lenguajes de 32 bits para Windows 95 y Windows 98.

Delphi 1.0

Si programas en Delphi 1.0, te gustará saber que con la i-723 se suministra un componente (para ser agregado a la VCL) para programarla. Esto hará tu trabajo mucho más cómodo a la vez que te permitirá aprovechar al máximo el ambiente orientado a objetos de Delphi. Encontrarás mucha facilidad en el hecho de poder programar la interfaz por medio de un componente, pero si además haz creado alguna vez tus propios componentes, podrás utilizar al que se provee como punto de partida para heredar sus propiedades y modificarlo a tu gusto (incluso podrías agregarle capacidades gráficas o mayor poder de control). En el caso de que quieras agregarle potencia para un mejor control de la i-723, es recomendable que le dés un vistazo al código fuente provisto para Turbo Pascal, ya que el mismo puede ser utilizado en Delphi 1.0.

El componente en cuestión se llama TControl723 y se encuentra en la unidad CTRL723.DCU del subdirectorio VCL16 adentro del directorio DELPH16 del Disco de Librerías. Para instalarlo, debes utilizar la opción INSTALL COMPONENTS del menú OPTIONS de Delphi. Una vez que aparezca el cuadro de diálogo para la instalación de componentes, presiona el botón ADD y en la caja de diálogo que se abrirá, busca el componente del directorio del disco rígido en donde lo hayas copiado (no lo instales directamente desde el Disco de Librerías, sino cópialo primero en tu disco rígido antes; de esta forma evitarás complicaciones futuras).

A continuación de listan las propiedades, métodos y eventos del componente:

Propiedades:

Sensor1	Motor1	Salida1	Name
Sensor2	Motor2	Salida2	Port
Sensor3	Motor3		Tag
Sensor4			

Eventos:

OnChange

Métodos:

Reset

Descripción de las propiedades:

Name y **Tag** son propiedades comunes a todos los componentes y no necesitan descripción.

Motor1, **Motor2** y **Motor3** pueden tomar cualquiera de tres estados: DA, IZ y ALTO. Están disponibles en tiempos de diseño y de ejecución. Cualquier cambio realizado en ellas se reflejará inmediatamente en la interfaz.

Salida1 y **Salida2** también están disponibles en tiempos de diseño y de ejecución y los cambios hechos en ellas se reflejan en la interfaz. Son propiedades booleanas, donde un TRUE significa que la salida está encendida y un FALSE apagada.

Sensor1, **Sensor2**, **Sensor3** y **Sensor4** son propiedades de tiempo de ejecución solamente (Run-Time) y de sólo lectura. Cada una de ellas refleja el estado del sensor correspondiente. Son propiedades booleanas, donde un TRUE significa que el sensor está excitado y un FALSE lo contrario.

Port es la propiedad que le indica al componente en qué puerto de impresora (paralelo) está conectada la i-723. Puede tomar tres valores: Lpt1, Lpt2 o Lpt3. Está disponible en tiempo de diseño y de ejecución. Si está "seteada" incorrectamente ninguno de los cambios hechos en las otras propiedades tendrá efecto sobre la interfaz.

Descripción de los eventos:

El único evento que posee TControl723 es **OnChange**. El mismo se dispara cada vez que ocurre un cambio en alguna de las propiedades relativas al control de la interfaz (Port, MotorX, SalidaX y SensorX). Es un evento ideal para escribir en él el código relativo a los sensores, ya que continuamente está chequeando los cambios en los estados de los mismos.

Descripción de los métodos:

El único método disponible en TControl723 es **Reset**. Como su nombre lo indica, una llamada a este método desactiva las dos salidas de relé y detiene los tres motores.

Ejemplo de utilización:

En el subdirectorio EJEMPLO1 del directorio DELPH16 del Disco de Librerías hay un ejemplo de utilización del componente que puede resultar sumamente útil para comenzar a programar con él. Ten en cuenta que **antes de cargar el ejemplo, deberás instalar el componente TControl723**, como se explicó al principio de este apartado. En el ejemplo se hace uso de todas las propiedades del componente. Pruébalas en tiempo de diseño y verás las ventajas de utilizar componentes en Delphi.

Otras formas de programar la interfaz con Delphi

Los demás lenguajes para Windows soportados por la i-723 no pueden utilizar este componente, por lo que para ellos se provee una librería de enlace dinámico (DLL). Ésta también puede ser utilizada desde Delphi. Su uso es menos cómodo que el del componente TControl723, pero a cambio brinda un poco (y sólo un poco) más de potencia. En el subdirectorio EJEMPLO2 del directorio DELPH16 del Disco de Librerías se encuentra el mismo programa ejemplo utilizado para demostrar el uso de

TControl723, pero esta vez implementado con la DLL provista, llamada I723.DLL. Esta DLL se halla en el directorio WIND16, y su uso se detalla mejor en el próximo apartado (el de Visual Basic). De todos modos, a continuación se listan las declaraciones de funciones y procedimientos de la DLL tal y como aparecen en el programa de EJEMPLO2:

```
{Rutinas para el control específico de la i-723:}
procedure Inicializar(PortID :integer); far; external 'i723';
procedure ApagaTodo; far; external 'i723';
procedure EscribeMotor(MotorID :integer; Accion :integer); far; external 'i723';
procedure EscribeSalida(ReleID, Accion :integer); far; external 'i723';
function LeeMotor(MotorID :integer) :integer; far; external 'i723';
function LeeSalida(ReleID :integer) :WordBool; far; external 'i723';
function LeeSensor(senID :integer) :WordBool; far; external 'i723';

{Rutinas para control en general:}
procedure out(ID, Valor: Integer); far; external 'i723';
function inp(ID: Integer): Integer; far; external 'i723';
```

Object Pascal

Para programar la interfaz desde Object Pascal, existen dos vías: mediante la DLL (la cual puede ser utilizada por casi cualquier lenguaje para Windows) o usando directamente las funciones de Pascal descriptas en el apartado de Turbo Pascal. Para usar estas últimas, abre el archivo TP723.PAS y utiliza las funciones y procedimientos que allí se hallan, ya que el código fuente del archivo es compatible con el de Object Pascal. Si, en cambio, prefieres manejarte con la DLL, tendrás que distribuir ésta con los programas que realices y además deberás incorporar las declaraciones de las funciones externas de la librería de enlace dinámico en cuestión. Para profundizar en el uso de esta DLL, lee el apartado dedicado a Visual Basic en 16 bits, ya que aunque no programes en ese lenguaje, la explicación es comprensible independientemente del lenguaje. A continuación se vuelven a listar (ya que se listaron en el apartado de Delphi 1.0) las declaraciones de las rutinas de la DLL :

```
{Rutinas para el control específico de la i-723:}
procedure Inicializar(PortID :integer); far; external 'i723';
procedure ApagaTodo; far; external 'i723';
procedure EscribeMotor(MotorID :integer; Accion :integer); far; external 'i723';
procedure EscribeSalida(ReleID, Accion :integer); far; external 'i723';
function LeeMotor(MotorID :integer) :integer; far; external 'i723';
function LeeSalida(ReleID :integer) :WordBool; far; external 'i723';
function LeeSensor(senID :integer) :WordBool; far; external 'i723';

{Rutinas para control en general:}
procedure out(ID, Valor: Integer); far; external 'i723';
function inp(ID: Integer): Integer; far; external 'i723';
```

Incluso puedes recortar con el Clip Board (o portapapeles) de Windows estas declaraciones del archivo EJEMP2.PAS del subdirectorio EJEMPLO2 del directorio DELPH16 del Disco de Librerías y pegarlas en el programa que estés haciendo en Object Pascal.

Visual Basic 3.0 y 4.0-16 bits

Las funciones y procedimientos para Visual Basic (y otros lenguajes para Windows) están disponibles en una librería de enlace dinámico (DLL). Para incluirlos en un programa escrito en Visual Basic 3.0 o Visual Basic 4.0 en versión de 16 bits, sólo debes agregar al mismo las siguientes declaraciones en la sección DECLARATIONS del form o módulo de código (archivo .BAS) que las llame en el programa:

```
Declare Sub Inicializar Lib "i723.DLL" (ByVal PortID As Integer)
Declare Sub ApagaTodo Lib "i723.DLL" ()
Declare Sub EscribeMotor Lib "i723.DLL" (ByVal MotorID As Integer, ByVal Accion As Integer)
Declare Sub EscribeSalida Lib "i723.DLL" (ByVal ReleID As Integer, ByVal Accion As Integer)

Declare Function LeeMotor Lib "i723.DLL" (ByVal MotorID As Integer) As Integer
Declare Function LeeSalida Lib "i723.DLL" (ByVal ReleID As Integer) As Integer
Declare Function LeeSensor Lib "i723.DLL" (ByVal senID As Integer) As Integer

Declare Sub out Lib "i723.DLL" (ByVal ID As Integer, ByVal Valor As Integer)
Declare Function inp Lib "i723.DLL" (ByVal ID As Integer) As Integer
```

Igualmente, en el directorio VBAS16 del Disco de Librerías se incluye el módulo VB723.BAS, el cual ya tiene todas esas declaraciones. Para incluirlo en tu proyecto, elije la opción ADD FILE del menú FILE de Visual Basic. Aparecerá un cuadro de diálogo como el de la figura 5.1. Selecciona entonces el archivo VB723.BAS y presiona el botón de OK.

Es importante recordar que el archivo i723.DLL deberá ser distribuido con tus programas hechos en Visual Basic y deberá ser colocado en el directorio WINDOWS\SYSTEM de la PC donde deba funcionar el programa. Este archivo está en el directorio WIND16 del Disco de Librerías.

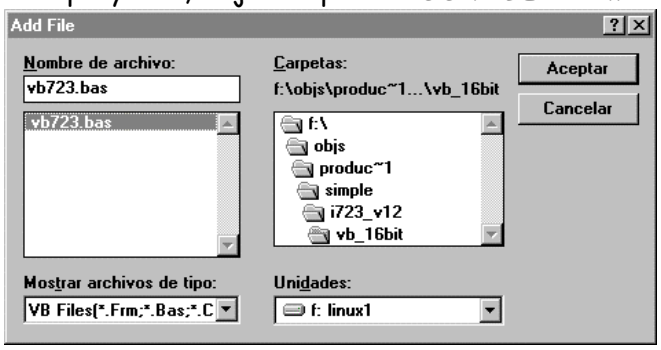


Figura 5.1: Agregado del módulo VB723.BAS a un proyecto de Visual Basic.

Las funciones y procedimientos de la librería son:

Función genérica:	Nombre:	Tipo:
Inicialización:	Inicializar	Procedimiento (Sub)
Control de motores:	EscribeMotor LeeMotor	Procedimiento (Sub) Función
Control de las salidas:	EscribeSalida LeeSalida	Procedimiento (Sub) Función
Sensores:	LeeSensor	Función
Otras:	ApagaTodo Out Inp	Procedimiento (Sub) Procedimiento (Sub) Función

Descripción:

Inicializar

Sintaxis:

Declare Sub Inicializar **Lib** "i723.DLL" (**ByVal** PortID **As Integer**)

Función:

Inicializa el sistema y el puerto de salida pasado como parámetro para trabajar con la i-723.

Ejemplo 1:

Intenta inicializar la i-723 en el puerto paralelo Lpt 1.

```
Inicializar 1
```

Parámetros:

Como parámetro recibe un número entero (integer) que puede ser 1, 2 ó 3 y que identifica al puerto paralelo al que se halla conectada la i-723.

ApagaTodo

Sintaxis:

Declare Sub ApagaTodo **Lib** "i723.DLL" ()

Función:

Apaga todas las salidas (tanto las de relé como las de motor).

Ejemplo:

```
ApagaTodo
```

Parámetros:

No recibe parámetros.

EscribeMotor

Sintaxis:

Declare Sub EscribeMotor **Lib** "i723.DLL" (**ByVal** MotorID **As Integer**, **ByVal** Accion **As Integer**)

Función:

Acciona el motor o los motores conectado a la salida cuyo número es pasado como parámetro.

Ejemplo 1:

El motor 1 girará hacia la derecha:

```
EscribeMotor 1, 1
```

Ejemplo 2:

Todos los motores girarán hacia la izquierda (el orden de los números de motor no altera el funcionamiento):

```
EscribeMotor 123, 2
```

ó

```
EscribeMotor 231, 2
```

Ejemplo 3:

Los motores 1 y 2 girarán hacia la izquierda.

```
EscribeMotor 21, 2
```

Ejemplo 4:

Los motores 1 y 3 se detendrán.

```
EscribeMotor 13, 0
```

Parámetros:

Se le pasan dos parámetros: MotorID y Accion. MotorID es del tipo integer y es el identificador del/los motor/es a ser accionado/s. Son valores válidos de este parámetro los siguientes:

1 (Acciona el motor 1).	12 ó 21 (Accionan los motores 1 y 2)
2 (Acciona el motor 2).	13 ó 31 (Accionan los motores 1 y 3)
3 (Acciona el motor 3).	23 ó 32 (Accionan los motores 2 y 3)

123, 132, 231, 213, 312 ó 321 (Accionan todos los motores)

Accion es del tipo integer y puede ser 1 (constante DA), 2 (constante IZ) ó 0 (constante ALTO).

LeeMotor

Sintaxis:

Declare Function LeeMotor **Lib** "i723.DLL" (**ByVal** MotorID **As Integer**) **As Integer**

Función:

Evalúa el estado actual del motor conectado a la salida cuyo número es pasado como parámetro.

Ejemplo: Lee el estado del motor 2 y lo imprime en pantalla:

```
if  LeeMotor(2) = 0 then
        print “Detenido”
elseif  LeeMotor(2) = 1 then
        print “Girando hacia la derecha”
elseif  LeeMotor(2) = 2 then
        print ”Girando hacia la izquierda.”
end if
```

Parámetros:

El único parámetro que recibe es del tipo integer e identifica al motor a evaluar. Puede ser 1, 2 ó 3.

Valor devuelto:

El valor devuelto es 1 (DA), 2 (IZ) ó 0 (ALTO) .

EscribeSalida

Sintaxis:

Declare Sub EscribeSalida **Lib** "i723.DLL" (**ByVal** ReleID **As Integer**, **ByVal** Accion **As Integer**)

Función:

Acciona la salida o las salidas de relé cuyo número es pasado como parámetro.

Ejemplo 1:

La salida 1 es activada:

```
EscribeSalida 1, 1
```

Ejemplo 2:

Las dos salidas de relé son desactivadas:

```
EscribeSalida 12, 0
```

Parámetros:

Recibe dos parámetros: El primero es un integer que identifca la salida a accionar. Puede ser 1, 2, 12 ó 21 (los dos últimos valores accionan las dos salidas a la vez). El segundo es la acción a ser realizada por la salida: 1 (SI) ó 0 (NO).

LeeSalida

Sintaxis:

Declare Function LeeSalida **Lib** "i723.DLL" (**ByVal** ReleID **As Integer**) **As Integer**

Función:

Evalúa el estado actual de la salida de relé cuyo número es pasado como parámetro.

Ejemplo:

Lee el estado de la salida 1 y lo imprime en pantalla:

```
if  LeeSalida(1) then
        print ”Prendida”
else
        print “Apagada”
end if
```

Parámetros:

El único parámetro que recibe es tipo integer e identifica la salida a evaluar. Puede ser 1 ó 2.

Valor devuelto:

El valor devuelto es 1 (SI) ó 0 (NO) .

LeeSensor

Sintaxis:

Declare Function LeeSensor **Lib** "i723.DLL" (**ByVal** senID **As Integer**) **As Integer**

Función:

Evalúa el estado actual del sensor conectado a la entrada cuyo número es pasado como parámetro.

Ejemplo:

Lee el estado del sensor 4 e imprime en pantalla un 1 ó un 0 según esté excitado o no:

```
if  LeeSensor(4) then
        print “1”
else
```

```
        print "0"
end if
```

Parámetros:

El único parámetro es integer e identifica al sensor a evaluar. Puede ser 1, 2, 3 ó 4.

Valor devuelto:

El valor devuelto es TRUE (sensor excitado) o FALSE (sensor no excitado).

Out (adicional)

Sintaxis:

Declare Sub out **Lib** "i723.DLL" (**ByVal** ID **As Integer**, **ByVal** Valor **As Integer**)

Función:

Este procedimiento y la función que sigue (inp) son rutinas adicionales. Esto quiere decir que no son funciones específicas para el control de la i-723, sino que agregan capacidades de control de puertos de Entrada/Salida (ports) al Visual Basic (y a cualquier lenguaje para Windows que carezca de ellas). Out es el procedimiento encargado de escribir en la dirección de Entrada/Salida (E/S) pasada como primer parámetro, el número pasado como segundo parámetro. Con Out y con Inp, podrás escribir tus propias órdenes para la interfaz, agregando así potencia a la programación bajo Windows. Incluso puedes hacer uso de estas rutinas para controlar desde Visual Basic y otros lenguajes otros dispositivos que no sean la interfaz i-723.

Ejemplo:

Enciende las dos salidas de relé y, si hay algún motor encendido lo apaga (888 es la dirección base de E/S del Lpt 1. A Lpt2 le corresponde 956 y a Lpt3 632):

```
out 888, 3
```

Parámetros:

Recibe dos parámetros del tipo integer: el primero es la dirección de E/S a ser escrita y el segundo es el byte a escribir en ella.

Inp (adicional)

Sintaxis:

Declare Function inp **Lib** "i723.DLL" (**ByVal** ID **As Integer**) **As Integer**

Función:

Esta función es la complementaria de Out, y es la que "lee" en la dirección de (E/S) pasada como parámetro.

Ejemplo:

Chequea la interfaz conectada al Lpt1 tiene todas sus salidas de relé y motor apagadas:

```
if  Inp(888)  = 0 then print "Todas las salidas apagadas"
```

Parámetros:

El único parámetro es la dirección de E/S a ser leída.

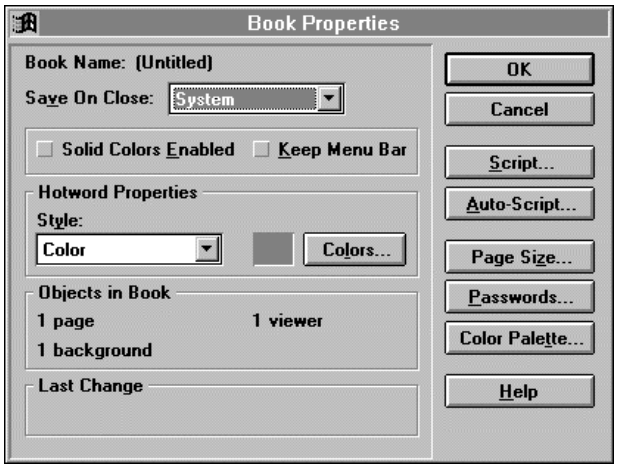
Valor devuelto:

Devuelve un integer que es el valor hallado en la dirección de E/S especificada.

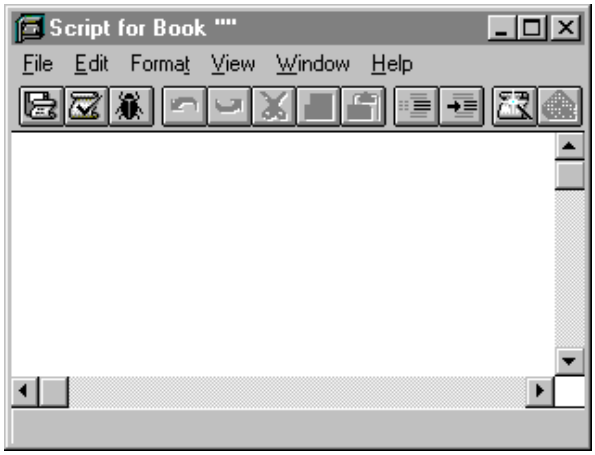
ToolBook

En el directorio MTBOOK del Disco de Librerías, se incluye un archivo de texto para ser "importado" como script y de este modo programar la i-723 desde ToolBook en versiones de 16 bits. Para incluir en tu "Book" o proyecto las órdenes de control de la interfaz sigue los siguientes pasos:

1) Una vez dentro de ToolBook, en el menú OBJECT, selecciona BOOK PROPERTIES y se abrirá la siguiente pantalla.



2) Ahora presiona el botón "Script…" y verás la pantalla de edición de scripts:



3) En esta pantalla, como puedes ver en la figura anterior, encontrarás el menú "FILE". Dentro de él selecciona "IMPORT". Esta opción te permite importar scripts desde un archivo de texto (con extensión .TXT). El archivo que importarás es el I723.TXT, el cual se halla, como ya se explicó, en el directorio MTBOOK del Disco de Librerías.

El script contenido en el archivo mencionado es el siguiente:

```
to handle EnterBook
linkDLL "i723.DLL"
        INT Inicializar(INT)
        INT EscribeMotor (INT, INT)
        INT EscribeSalida (INT, INT)
```

```
        INT LeeMotor (INT)
        INT LeeSalida (INT)
        INT LeeSensor (INT)
end linkDLL
IniRet = Inicializar(1)  --El número que se le pasa a Inicializar es
                                                --el número de puerto paralelo al que está
                                                --conectada la interfaz.
                                                --Ejemplo:
                                                --Inicializar(2) es porque la interfaz está
                                                --conectada en el LPT 2.
end
```

4) Debes tener en cuenta, que este script llama a las rutinas de la librería I723.DLL, la cual es distribuida con la interfaz en el directorio WIND16 del Disco de Librerías. Para que ToolBook tenga acceso a estas rutinas, esta DLL deberá ser copiada en el directorio "SYSTEM" de la PC donde quieras correr tus aplicaciones hechas con ToolBook. Por ejemplo si, tienes una PC con Windows instalado en el directorio WINDOWS, deberás buscar el subdirectorio SYSTEM allí. De esta forma, tendrás que copiar la librería I723.DLL en WINDOWS\SYSTEM. Sin estas precauciones, la interfaz no podrá utilizarse con ToolBook.
Además, ten en cuenta que si distribuyes aplicaciones hechas con ToolBook que hagan uso de la interfaz, tendrás también que distribuir con ellas la DLL y explicar que ésta debe ser copiada en el directorio SYSTEM de WINDOWS.

Ahora ya estás en condiciones de usar las rutinas de la DLL. Estas se describen detalladamente en el apartado de Visual Basic de este manual. A continuación se dan algunos ejemplos breves para que veas cómo se llama a las rutinas desde ToolBook:

Ejemplos de utilización

El siguiente ejemplo es el script de un botón que hace girar el motor 2 hacia la derecha:

```
to handle ButtonClick
        mRet = EscribeMotor(1, 1)
end
```

El ejemplo dado a continuación es también un script para un botón. Cuando hagas click sobre ese botón, su título (Caption) cambiará entre 0 y 1 dependiendo del estado del sensor 1 de la interfaz:

```
to handle ButtonClick
        Caption of self = LeeSensor(1)
end
```

Por último, las líneas que siguen harán que los tres motores de la i-723 giren hacia la izquierda al presionar el botón en el que se coloque este script:

```
to handle ButtonClick
        mRet = EscribeMotor(123, 2)
end
```

V.C) WINDOWS 95 Y WINDOWS 98 (32 BITS)

Para aquellos lenguajes de 32 bits, se provee una librería de enlace dinámico similar a la disponible en versión de 16 bits. Ésta puede encontrarse en el directorio WIND32 del Disco de Librerías.

Delphi 2.0

Delphi 2.0 es uno de los lenguajes de 32 bits soportados directamente por la i-723. En el directorio DELPH32 del Disco de Librerías se proveen un componente (VCL) y dos programas de ejemplo.

Así como con Delphi 1.0, es posible programar a la interfaz con la librería de enlace dinámico (DLL) o mediante el componente para Delphi. La programación es exactamente igual a la de Delphi 1.0, por lo que consultando ese apartado de este manual podrás comenzar a trabajar con esta versión del lenguaje.

Si utilizas el componente, deberás instalar el que se encuentra en el directorio DELPH32\VCL32 del Disco de Librerías (cópialo primero a tu disco rígido). Una vez instalado (utilizando la opción INSTALL del menú COMPONENT del entorno de Delphi 2.0), deberás ver en tu barra de componentes la nueva sección "Simple Informática". Allí habrá sólo un componente: TCONTROL723. Si has escrito programas con Delphi 1.0 para la interfaz utilizando este componente, estos serán compatibles con Delphi 2.0 directamente, una vez que haya instalado TControl723 en versión 32 bits. La figura 5.2 muestra la barra de componentes de Delphi 2.0 después de la instalación de Tcontrol723:

Figura 5.2: Barra de componentes de Delphi 2.0 después de instalar Tcontrol723.

Por supuesto que si no quieres usar el componente TControl723, puedes programar la interfaz con las rutinas provistas en la DLL de 32 bits. Todas las rutinas contenidas en ella son equivalentes a las de la DLL de 16 bits.

En el directorio DELPH32\EJEMPL2 del Disco de Librerías, hay un programa de ejemplo mediante el cual podrás comprender fácilmente el uso de esta DLL.

Una única salvedad: en Windows 95 ó 98, no es única la manera de llamar DLLs desde lenguajes de alto nivel como Delphi 2,0, sino que las formas de llamada son varias y difieren unas de otras en la manera de pasar los parámetros a las rutinas. Para que la DLL de 32 bits funcione con tus programas hechos en Delphi 2.0, deberás agregar la palabra **stdcall** a las declaraciones de las funciones y procedimientos de la DLL. Para ver cómo se hace esto, puedes observar las declaraciones del programa de ejemplo del subdirectorio EJEMPLO2 del directorio DELPH32 del Disco de Librerías. En él se hace uso de la mayoría las funciones y procedimientos de la DLL de 32 bits.

Visual Basic 4.0- 32 bits

La programación con la versión de 32 bits de Visual Basic 4.0 es prácticamente idéntica a la de la versión de 16 bits. Debes utilizar las funciones y procedimientos (subs) de la librería de enlace dinámico (DLL) de 32 bits provista con la interfaz. La misma se encuentra en el directorio Wind32 del Disco de Librerías y se llama i723_32.DLL. En el directorio VBAS32 del mismo disco, encontrarás un ejemplo hecho con Visual Basic 4.0 en versión 32 bits. Para que éste ejemplo funcione, deberás primero copiar la DLL en el directorio SYSTEM de WINDOWS.

Al igual que ocurre con la versión de 16 bits, tendrás que distribuir esta DLL con todos tus programas hechos en Visual Basic que hagan uso de la interfaz y explicarle al usuario de los mismos que debe copiar la DLL en el directorio SYSTEM de su PC.

En cuanto a las funciones y procedimientos que la DLL contiene, son **exactamente iguales** a los de la versión de 16 bits, por lo que las encontrarás documentadas en el apartado didicado a las versiones 3.0 y 4.0-16 bits de Visual Basic. De todos modos, a continuación se muestran las declaraciones de las mismas, las cuales difieren de las de la versión de 16 bits sólo en el hecho de que hacen referencia al archivo "i723_32.DLL" en lugar del "i723.DLL", el cual es de 16 bits:

```
Declare Sub Inicializar Lib "i723_32.DLL" (ByVal PortID As Integer)
Declare Sub ApagaTodo Lib "i723_32.DLL" ()
Declare Sub EscribeMotor Lib "i723_32.DLL" (ByVal MotorID As Integer, ByVal Accion As Integer)
Declare Sub EscribeSalida Lib "i723_32.DLL" (ByVal ReleID As Integer, ByVal Accion As Integer)

Declare Function LeeMotor Lib "i723_32.DLL" (ByVal MotorID As Integer) As Integer
Declare Function LeeSalida Lib "i723_32.DLL" (ByVal ReleID As Integer) As Integer
Declare Function LeeSensor Lib "i723_32.DLL" (ByVal senID As Integer) As Integer

Declare Sub out Lib "i723_32.DLL" (ByVal ID As Integer, ByVal Valor As Integer)
Declare Function inp Lib "i723_32.DLL" (ByVal ID As Integer) As Integer
```

Igualmente, en el directorio VBAS32 del Disco de Librerías se incluye el módulo VB723.BAS, el cual ya contiene todas estas declaraciones, por lo que agregándo el módulo a tus proyectos te ahorrarás el trabajo de copiarlas a mano.

Word 97, PowerPoint y Visual Basic 5.0

Gracias a la tecnología abierta de Simple Informática, ahora es posible hacer programas de control para la interfaz i-723 con herramientas como Word 97 o PowerPoint. Esto se logra utilizando los macros de Word (o de PowerPoint), los cuales se basan en la sintaxis de Visual Basic 5.0 (VB 5.0). Es por esto que este apartado se titula "Word 97, PoerPoint y Visual Basic 5.0", pues todo lo que aquí se exprese vale tanto para los macros de Word y PowerPoint como para VB 5.0.

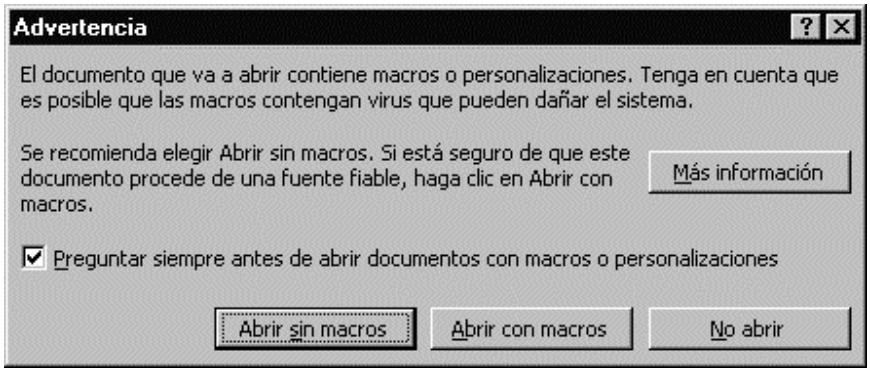
En realidad, la manera de programar con VB 5.0 es idéntica a la de la versión de 32 bits de Visual Basic 4.0.

En el directorio OFFICE97 del Disco de Librerías encontrarás tres archivos: i723.DOC, i723.PPt y VB723.BAS. El primero es un documento de Word 97 que contiene macros para controlar la interfaz. El segundo es una presentación de PowerPoint similar. El tercero es el archivo con las declaraciones de funciones (functions) y procedimientos (subs) de control, los cuales se hallan en una librería de enlace dinámico (DLL).

Si vas a utilizar VB 5.0, simplemente agrega a tu proyecto el archivo VB723.BAS y copia en el directorio SYSTEM de WINDOWS la DLL llamada I723_32.DLL, la cual encontrarás en el directorio WIND32 del Disco de Librerías. Ésta deberá ser distribuída con todos tus programas hechos en VB 5.0 que hagan uso de la interfaz. Como todas las rutinas de la DLL son exactamente iguales a las de la DLL de 16 bits utilizada por Visual Basic 3.0, encontrarás la documentación de las mismas, de manera completa, en el apartado dedicado a ese lenguaje en este manual.

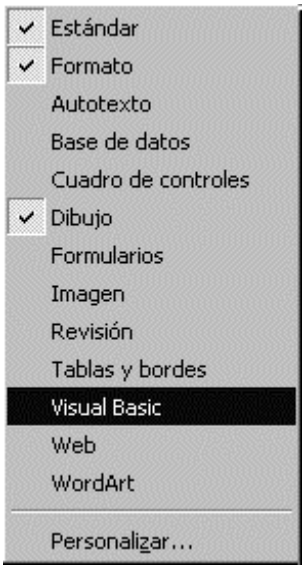
En cuanto a Word y PowerPoint, también tendrás que hacer uso de la DLL, por lo que deberás recurrir al apartado mencionado. Además, también deberás copiarla en tu directorio SYSTEM de WINDOWS.

Si quieres ver un ejemplo de uso con Word 97, abre el archivo i723.DOC del directorio OFFICE97 del Disco de Librerías con Word. Al intentar abrirlo, Word presentará el siguiente mensaje:



Presiona el botón que dice "Abrir con macros". Si tienes conectada la interfaz a la PC, notarás que se encienden las tres salidas de motor con sus indicadores en rojo. Esto es señal de que los macros del documento están funcionando correctamente. Lee el documento con cuidado y éste te guiará a modo de demostración para que veas cómo se controla la interfaz desde un documento de Word 97.

Para ver los macros que dan su funcionalidad al documento, da click con el botón derecho del mouse sobre cualquier barra de herramientas del Word para hacer que se despliegue el siguiente menú:



En este menú, selecciona la opción titulada "Visual Basic" (si no está seleccionada aún). Deberá aparecer una barra de herramientas con los siguientes íconos:

En esa nueva barra, haz clik en el botón que tiene el ícono con la escuadra para pasar al "modo de diseño". Ahora, dando doble click sobre los botones y demás controles que haya en el documento, se abrirá una ventana conteniendo al macro correspondiente a ese control.
A continuación se listan las declaraciones de las funciones y procedimientos que puedes utilizar en los macros para controlar la i-723:

```
Declare Sub Inicializar Lib "i723_32.DLL" (ByVal PortID As Integer)
Declare Sub ApagaTodo Lib "i723_32.DLL" ()
Declare Sub EscribeMotor Lib "i723_32.DLL" (ByVal MotorID As Integer, ByVal Accion As Integer)
Declare Sub EscribeSalida Lib "i723_32.DLL" (ByVal ReleID As Integer, ByVal Accion As Integer)

Declare Function LeeMotor Lib "i723_32.DLL" (ByVal MotorID As Integer) As Integer
Declare Function LeeSalida Lib "i723_32.DLL" (ByVal ReleID As Integer) As Integer
Declare Function LeeSensor Lib "i723_32.DLL" (ByVal senID As Integer) As Integer

Declare Sub out Lib "i723_32.DLL" (ByVal ID As Integer, ByVal Valor As Integer)
Declare Function inp Lib "i723_32.DLL" (ByVal ID As Integer) As Integer
```

Para más información sobre estas rutinas, debes ver el apartado dedicado a Visual Basic 3.0 y 4.0-16 bits. Allí encontrarás una completa descripción de cada una con ejemplos y explicaciones.

V. D) OTROS LENGUAJES Y SISTEMAS DE AUTOR PARA WINDOWS EN 16 Y 32 BITS

Windows 3.1x (16 bits)

Cualquier lenguaje o sistema de autor que tenga capacidad para cargar DLLs de 16 bits podrá ser utilizado como herramienta de programación para la interfaz i-723 bajo Windows 3.1x. Las funciones incluidas en la librería i723.DLL ya fueron descriptas en el apartado de Visual Basic 3.0 y 4.0-16 bits. Además, en el Disco de Librerías hay dos programas en los que se puede ver el uso de esta DLL en dos lenguajes distintos: Delphi y Visual Basic (directorios DELPH16\EJEMPLO2 y VBAS\EJEMPLO, respectivamente).

En cuanto a los sistemas de autor para Windows, algunos pueden utilizar DLLs. Consulta los manuales de tu herramienta de programación para ver cuál es la forma de uso de las librerías de enlace dinámico (Dynamic Link Libraries o DLLs) en ella. En este manual ya se mostró la forma en que se puede utilizar a ToolBook para programar la i-723.

Prácticamente cualquier sistema de programación bajo Windows soporta DLLs. Visual C++, Borland C/C++, Turbo Pascal para Windows, Paradox, dBase, algunas versiones de LOGO para Windows y varios sistemas de autor, son algunas de las posibilidades.

Windows 95 y 98 (32 bits)

Como ya se dijo, con la DLL de 32 bits, se amplía en gran manera el rango de posibilidades para la programación de la i-723.

La DLL de 32 bits puede ser hallada en el directorio WIND32 del Disco de Librerías y las rutinas que contiene están descriptas con detalle en el apartado dedicado a Visual Basic 3.0 y 4.0-16 bits.

Además, en el disco hay tres ejemplos del uso de estas rutinas: uno en Delphi 2.0 (directorio DELPH32\EJEMPLO2), otro en Visual Basic 4.0-32 bits (directorio VBAS32\EJEMPLO) y uno en forma de macros para Word 97 (archivo i723.DOC del directorio OFFICE97). En suma, el hecho de que las rutinas de control se hallen en una DLL deja el camino abierto para programar a la i-723 desde casi cualquier lenguaje para Windows que se te ocurra.

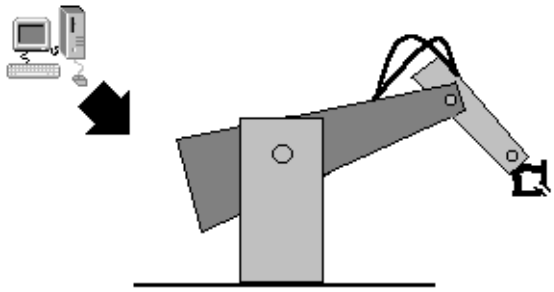
CAPÍTULO VI:

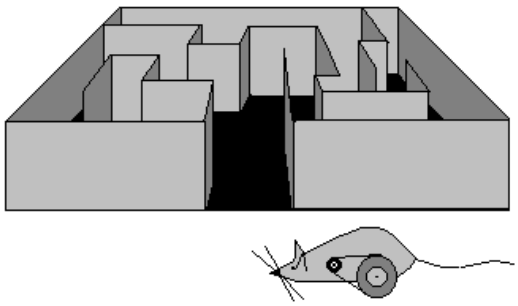
ALGUNAS ACTIVIDADES POSIBLES

Las aplicaciones de la interfaz son muchísimas. Algunas pueden ser las siguientes:

Control de un brazo robot

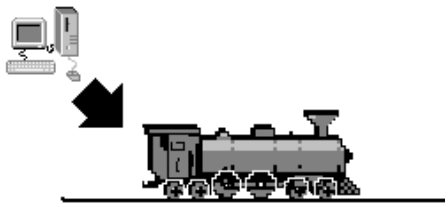
Esta es una aplicación típica. Los brazos pueden estar construidos de diferentes formas y con distintos elementos, desde modernos juegos de construcción hasta materiales de descarte. Cada movimiento de un brazo robot o manipulador robótico es llamado **grado de libertad**. Típicamente se utiliza un motor por cada grado de libertad. La i-723 puede controlar tres motores directamente más los que se conecten a las salidas de relé. Si se deseas utilizar algún motor de mayor potencia que los que se conectan directamente a la interfaz, debes conectarlo a las salidas de relé. Sensores de choque o fin de carrera pueden colocarse en las pinzas del robot para detectar objetos agarrados por el mismo. Sensores de luz pueden ayudar al robot a distinguir entre piezas claras y oscuras. Los manipuladores robóticos son sumamente interesantes y ofrecen grandes posibilidades de desarrollo.





Robots que resuelven situaciones tomando decisiones

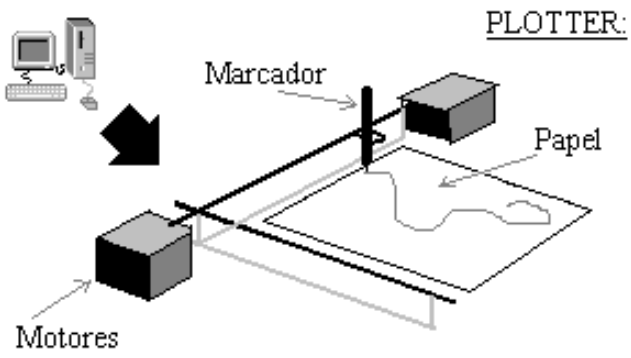
Si bien los manipuladores robóticos descriptos arriba pueden estar programados para tomar ciertas decisiones (como por ejemplo qué piezas agarrar, o a dónde colocarlas según su color, forma, etc.), existen otros tipos de robots donde la programación de los mismos para la resolución de un problema es una actividad especialmente interesante. Un ejemplo de ello son los "ratones robot" que deben salir de laberintos. Con dos motores y uno o dos sensores de choque es posible armar uno de ellos. Los laberintos pueden construirse con cartón, madera o cualquier material medianamente resistente, y la programación puede hasta presentar el recorrido en pantalla del robot (la tortuga del LOGO es ideal para este fin).



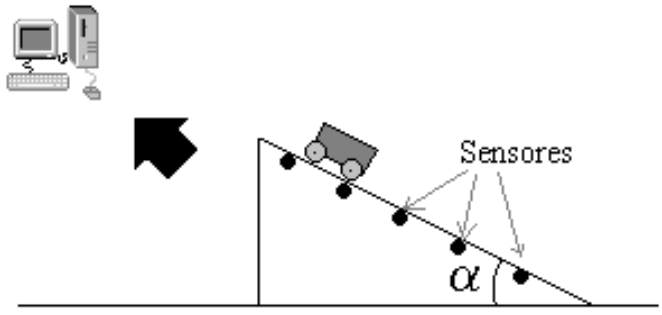
Maquetas, dioramas, trenes y autos que funcionan solos

No necesariamente los sistemas a controlar deben ser manipuladores mecánicos o robots móviles. Casi cualquier cosa que tenga movimiento accionado por motores eléctricos es controlable por medio de la interfaz. De este modo, se pueden conectar trenes eléctricos, autos de pistas de carreras, viejos juguetes a pilas, etc. Es posible armar maquetas de ciudades con material de descarte, cartón y otros y hacerlas funcionar de forma automática. Los sistemas de semáforos pueden estar controlados por sensores que detecten el paso del tren, o de los autos, y son numerosos los pequeños automatismos que pueden emplearse en un diorama o maqueta. Con dos sensores, se pueden también construir "cuenta-vueltas" para los autos de carreras. En la ciudad puede haber más de un edificio con luces en su interior y por medio de la PC se pueden establecer los tiempos que cada luz permanece encendida o apagada. Un sensor que detecte la luz del ambiente podría prender todas las luces de la maqueta al oscurecerse la habitación, dando de este modo la sensación de "día y noche". Otros accesorios a tener en cuenta en estas construcciones son sirenas, molinos, grúas con electroimanes para levantar "clavitos" y cargar vagones de tren, y muchos otros.

Robots tortuga y plotters



Un sistema sencillo y con muchas posibilidades y variantes es un pequeño robot móvil que haga las veces de tortuga. Con dos motores es posible su implementación, pudiéndose ampliar este número para agregar grados de libertad y funciones al robot. Colocando un marcador en el robot, se puede programar al mismo para que siga a la tortuga de la pantalla del LOGO, y generar dibujos en papel. Una variante de estos robots son los PLOTTERS, los cuales se utilizan profesionalmente en talleres gráficos. Un plotter es básicamente un marcador (o varios marcadores de distintos colores que se elevan o se bajan para que tomen contacto con el papel según el color seleccionado) que se puede mover sobre un plano a lo largo de ejes cartesianos, trazando así el dibujo requerido. Algunos juegos de construcción para niños traen en sus manuales variantes de plotters para ser montados.



Aplicaciones en física y otras ciencias

Las interfaces de robótica son un excelente complemento para los laboratorios escolares de física y otras ciencias, ya que por medio de ellas es posible utilizar a la computadora como herramienta de medición en distintos experimentos.

Esto tiene una ventaja adicional muy importante: los datos así adquiridos por la PC pueden ser analizados directamente con distintos programas (por ejemplo programas de matemática, estadística, planillas de cálculo, aplicaciones para ciencias específicas...), almacenados en bases de datos para su posterior uso, adaptados para formar parte de informes escritos y graficados ya sea generando curvas de función, gráficos de barras, torta, y otros. Mediciones de velocidades media e instantánea, aceleraciones, distancias recorridas, ángulos, períodos de oscilación (péndulos), velocidades angulares, temperaturas, nivel de líquido, y muchas otras son posibles.

Un buen ejemplo de ellos es la medición de tiempos en la caída de un objeto en un plano inclinado. Colocando sensores a lo largo del recorrido se puede medir el tiempo entre la excitación de cada sensor. De este modo se calcula la velocidad promedio y, si los sensores están próximos, también se aproxima la velocidad instantánea. Variando el ángulo del plano, midiendo nuevamente y graficando las curvas de función representativas de las velocidades, se consigue una interesante actividad, donde se podrían calcular aceleraciones analíticamente a partir de los datos tomados, hacer predicciones de comportamiento basadas en la cinemática del problema, y así comparar todos los resultados e integrarlos a un informe escrito.

APÉNDICE A:
CONEXIONES ELÉCTRICAS COMUNES

Este apéndice presenta algunas formas de conexión comunes para motores eléctricos y luces. Quienes no estén muy familiarizados con estos dispositivos encontrarán estas páginas de utilidad.

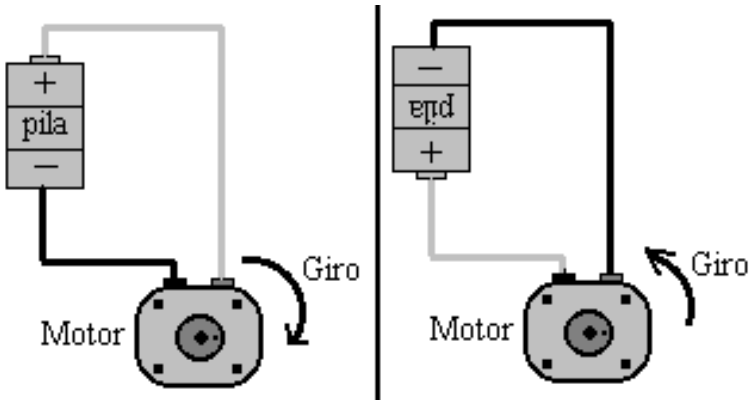


Figura A.1:Inversión del sentido de giro en un motor de corriente continua

Inversión del giro de un motor de corriente continua

Lo primero que se debe saber acerca de los motores utilizados generalmente en juguetes y aplicaciones de robótica educativa es el tipo de fuente de alimentación que se debe utilizar para que funcionen. La mayoría de los pequeños motores empleados en robótica son de **corriente continua**. Ejemplos típicos de fuentes de corriente continua son las pilas, las baterías, las propias SALIDAS DE MOTOR de la i-723, y la fuente de alimentación provista con esta interfaz, entre otros. Existe otro tipo de motores que se alimentan con **corriente alterna**. Estos son menos comunes y no tan prácticos para las aplicaciones de robótica educativa en general, por lo que aquí se describirán motores de continua. Estos motores poseen dos terminales. En uno de ellos debe conectarse el **polo negativo** y en el otro el **polo positivo**. Lo principal a saber es que si se invierte la polaridad, el motor invierte el sentido de giro de su eje. La figura A.1 ilustra esta situación. Si se conecta el polo positivo en ambos terminales del motor, éste no funcionará (lo mismo ocurre con el polo negativo). Una manera práctica de invertir la marcha de un motor es por medio de **llaves inversoras**. Estas son interruptores que poseen tres conectores en lugar de dos (recuérdese que un interruptor o llave eléctrica posee sólo dos terminales: uno al que "entra" corriente eléctrica y otro del que "sale" si la llave está "activada". El ejemplo más típico de esto son las llaves de luz de una casa). La llave inversora tiene un punto medio y dos contactos más. El punto medio siempre está conectado a uno de ellos. Al mover la palanca de la llave, el punto medio se conecta al otro contacto, quedando desconectado el anterior. La figura A.2 muestra el símbolo esquemático de una llave inversora, mientras que la figuras A.3 y A.4 esquematizan el uso de estas llaves para controlar un motor.

Figura A.2: Símbolo de una llave inversora

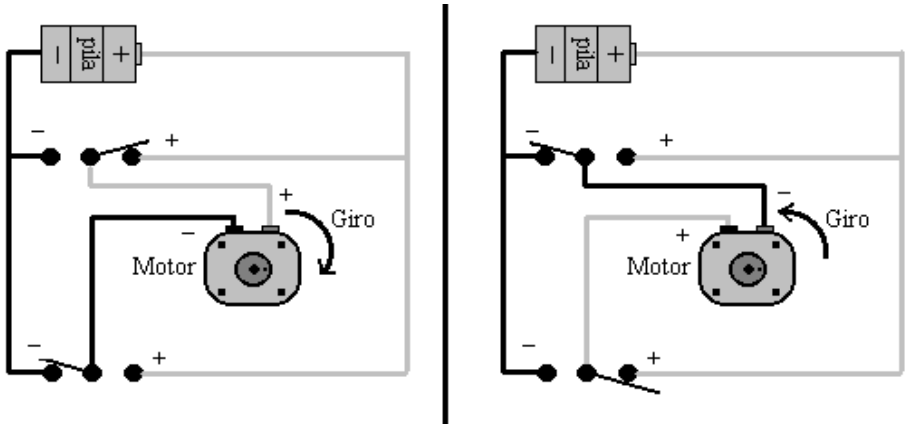


Figura A.3: Inversión del sentido de giro con llaves inversoras.

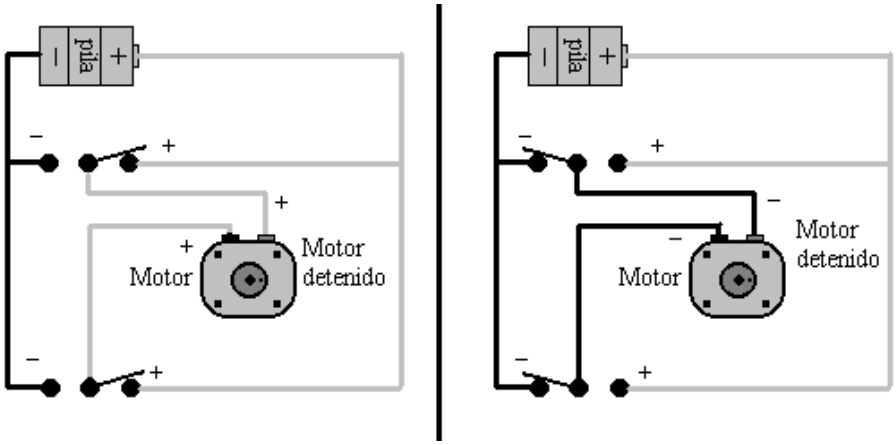


Figura A.4: Motor detenido.

Implementación de un semáforo con una llave inversora

Las llaves inversoras tienen otros posibles usos. Uno de ellos es el de hacer conmutaciones de modo de activar un dispositivo desactivando a la vez otro. La figura A.5 es el esquema de conexiones de un posible semáforo de dos luces (pueden ser la verde y la roja, por ejemplo) utilizando una llave inversora. Como idea final téngase en cuenta que las salidas de relé de la interfaz (descriptas en el Capítulo II) son exactamente iguales a una llave inversora, sólo que están controladas por la computadora (Ver figura A.6).

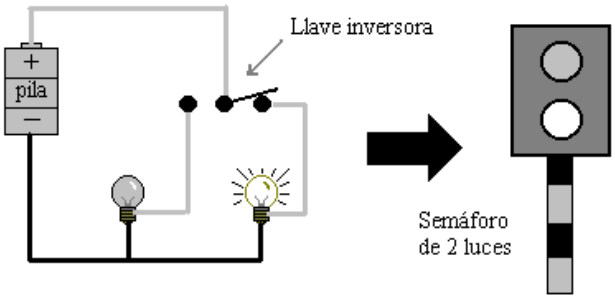


Figura A.5: Circuito del semáforo de dos luces

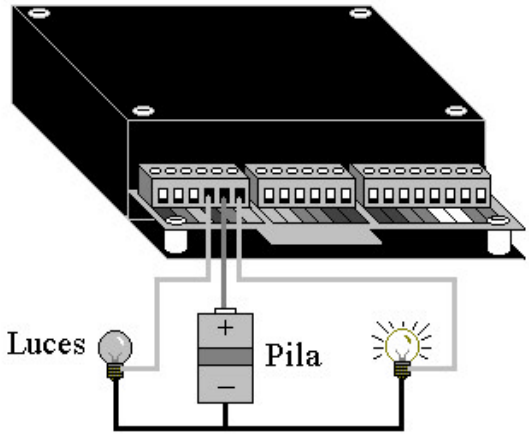


Figura A.6: Circuito del semáforo de dos luces controlado mediante la interfaz

APÉNDICE B:
ESPECIFICACIONES TÉCNICAS

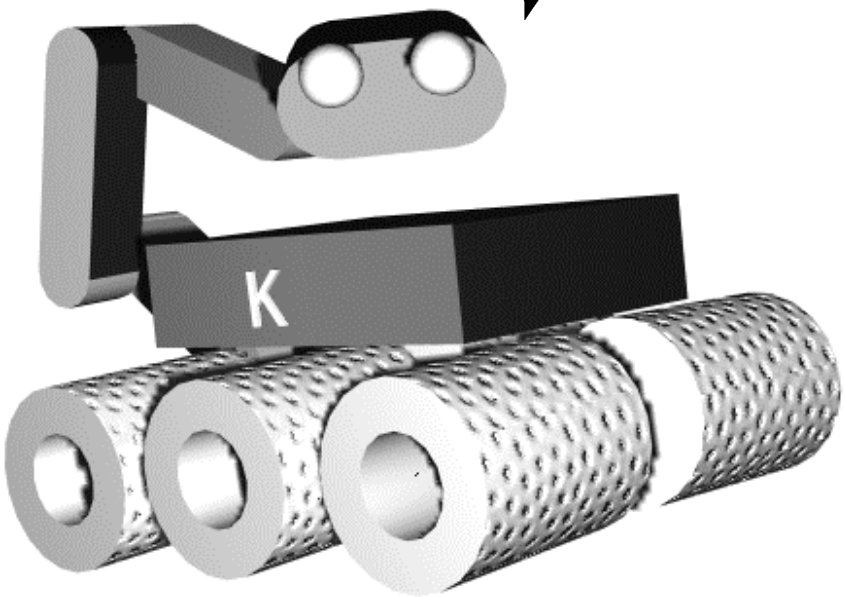
La siguiente tabla presenta las características técnicas de la i-723:

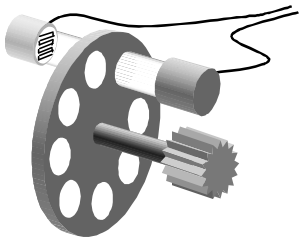
Alimentación:	220 - 230 Volts AC @ 50 Hz
Protección:	• Circuito interno: Protección electrónica activa. • Salidas: Protección electrónica por switch de polímero, configurable por "jumper". (Consulta a tu distribuidor para realizar intercambios de fuente de alimentación y modificaciones en la configuración de protección).
Entradas:	• 4 entradas digitales de sensor (nivel: 5 VDC aprox.). Conectores universales. • Entrada para fuente de alimentación externa intercambiable.
Salidas:	• 3 salidas con alimentación e inversión de polaridad (9 VDC, 0,9 Amperes de consumo constante total* con la fuente provista). Conectores universales. • 2 salidas de relé simple inversor (alimentación externa: hasta 48 VAC, 2 Amperes ó 24 VDC, 2 A). Conectores universales.
Conexión a la Computadora:	Puerto paralelo (DB-25 en la computadora, conector de 36 pines en la interfaz). Cable no incluido con la interfaz.

* El consumo continuo total entre las 3 salidas inversoras es de 0,9 Amperes con la fuente de alimentación provista. Es posible utilizar fuentes de mayor capacidad conectándolas a la entrada de alimentación según las especificaciones dadas y configurando el sistema de protección electrócica mediante el respectivo "jumper". Para esta operación consulta a tu distribuidor.

APÉNDICE C:
FICHAS DIDÁCTICAS

Las Fichas Didácticas son breves explicaciones sobre determinados temas relacionados con la tecnología y la robótica educativa. En este manual se incluyen tres fichas, las cuales describen el funcionamiento, uso y aplicaciones de algunos de los sensores más utilizados como complemento de la interfaz i-723.





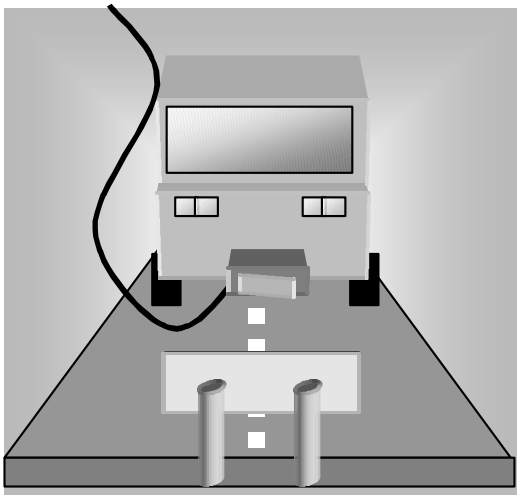
Sensores de Choque

Fáciles de usar, de pequeño tamaño y de bajo costo, estos sensores son muy utilizados en robótica educativa. También llamadas sensores de fin de carrera, son en verdad interruptores eléctricos que abren o cierran un circuito al apl icárseles una pequeña fuerza.

Funcionamiento

Un sensor de este tipo no es ni más ni menos que un interruptor eléctrico que se acciona presionándolo como se lo haría con una llave de luz doméstica. La diferencia es que para que el sensor siga activado, hay que mantener la fuerza aplicada sobre él, ya que al soltarlo se desactiva. De esta forma se puede medir el tiempo que duró la presión sobre el mismo.

Los sensores de choque pueden ser de dos tipos: normal abierto y normal cerrado . Externamente son iguales. La diferencia radica en que los primeros (normales abiertos) no conducen la corriente eléctrica **hasta** que se los presiona, mientras que los segundos (normales cerrados), conducen siempre la corriente eléctrica **salvo** cuando se los presiona. Los más comunes son los normales abiertos.

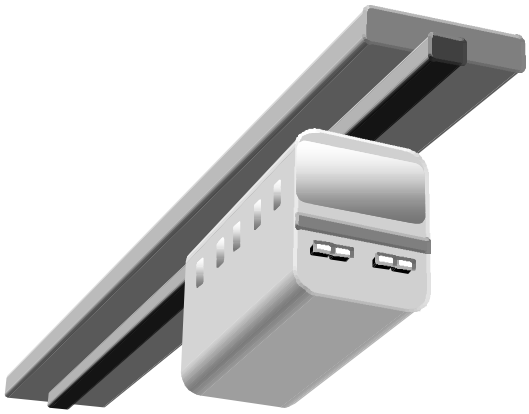


Uso

Del sensor sale un cable terminado en una pequeña ficha hembra de dos vías. Por medio de un cable de la longitud que se necesite en cada caso, se conecta la ficha a una entrada de sensor de la interfaz (puedes ver el Capítulo II: "Trabajando con la interfaz").

Al presionar la "lengüeta" externa del sensor, se encenderá la luz verde de la entrada de la interfaz a la que está conectado. El mismo puede utilizarse como interruptor en un circuito eléctrico, aún sin interfaz. Puede usarse para reemplazar un interruptor y así abrir o

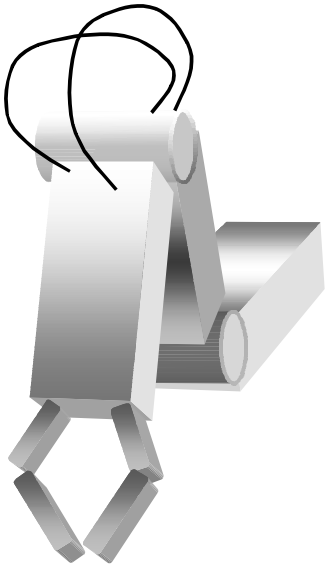
cerrar el circuito en cuestión. Lo único que hay que tener en cuenta es que no se deben sobrepasar los límites absolutos de 24 voltios en corriente continua y 48 voltios en corriente alterna en cuanto a la tensión, y 900 miliamperes en lo que respecta a la corriente.

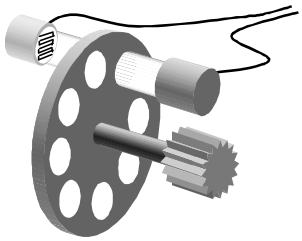


Aplicaciones prácticas

Típicamente estos sensores son colocados en los paragolpes de pequeños vehículos para detectar obstáculos. Pueden colocarse sensores tanto en el paragolpes trasero como en el delantero. Cuando el vehículo choca, el sensor informa a la computadora, la cual, con un programa adecuado puede hacer que el vehículo retroceda e intente otro camino. Se los suele utilizar también para detectar el paso de objetos móviles por peso (por ejemplo esferas de metal, autos pequeños, etc.). Esto es: se coloca el sensor en un orificio practicado a tal fin en el posible recorrido del móvil. Sobre el sensor se puede poner una pequeña plancha de cartulina, por ejemplo. Al pasar el objeto por encima de ésta, su peso activará al sensor.

Estos sensores también son conocidos como "sensores de fin de carrera", ya que con frecuencia se los pone en el final del recorrido de algún objeto para detenerlo. Ejemplos diversos de esto pueden ser: brazos mecánicos, ascensores, trenes, etc..





Sensores Magnéticos

Los Sensores Magnéticos, como su nombre lo indica, pueden detectar la proximidad de un campo magnético. Esto quiere decir que, si se le acerca un imán, el sensor envía una señal a la interfaz a la cual se encuentra conectado y ésta última da aviso a la computadora.

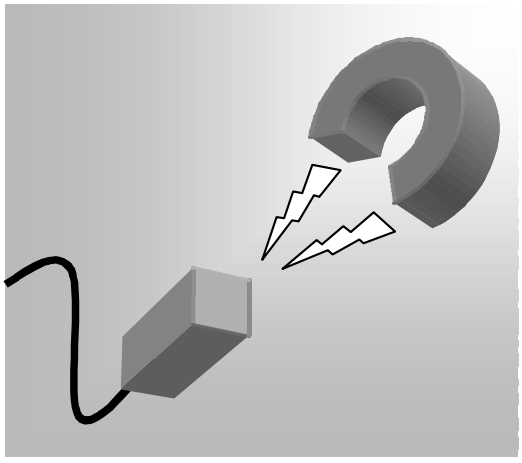
Ventajas

Las principales ventajas de estos sensores son su uso sencillo , su bajo costo y el hecho de que realizan la detección de ciertos eventos sin contacto físico . Esto último es importantísimo, ya que todo contacto físico produce un desgaste del dispositivo que lo recibe, además de que el sistema físico que se está controlando es más propenso a fallar cuando hay choques, roces, etc..

Uso

Son muchísimos los usos posibles de este tipo de sensores. Piénsa que con sólo acercarle el pequeño imán incluido con el sensor, éste se activa y envía la señal a la interfaz.

Exteriormente, un sensor magnético es una cajita cerrada, de la cual sale un cable con una ficha hembra de dos vías. La ficha debe ser conectada a una entrada de sensor de la interfaz por medio de un cable de la longtud que desees (para más información acerca del **sistema de conexiones**, lee el Capítulo II)



Internamente, estos sensores no son ni más ni menos que un interruptor eléctrico. Al acercarle el imán a una distancia de algunos centímetros, la interfaz deberá indicar que la entrada de sensor a la que está conectado recibe señal (se encenderá la luz verde de la entrada en cuestión). Cambiando el ángulo en el que se acerca el imán, se lograrán distancias de excitación del sensor mayores (o sea que el mismo detectará al imán a distancias más

grandes, dependiendo de la posición de ambos). Por lo general, una de las caras del sensor será más sensible que las demás.

Aplicaciones prácticas

Las aplicaciones de estos sensores van desde el conteo de revoluciones hasta el control de trenes en miniatura. La figura C.1 muestra una rueda con un imán pegado a ella. El sensor magnético se activa cada vez que la rueda da una vuelta. De esta forma se pueden contar las vueltas de la misma. Esta configuración puede ser utilizada, por ejemplo, en pequeños vehículos para informar a la computadora de su recorrido.

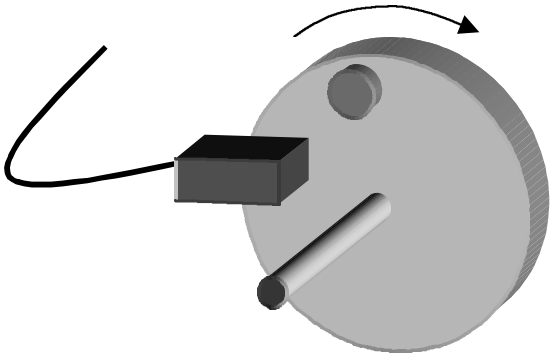


Figura C.1: Conteo de revlouciones con un sensor magnético.

En la figura C.2 se puede ver un ascensor que tiene también un imán, habiendo sensores magnéticos en los distintos "pisos" donde el ascensor debe detenerse. Este mismo principio es aplicable a un tren (colocando sensores en las estaciones). También es posible utilizar sensores magnéticos para detectar fin de carrera o de recorrido, para controlar semáforos cuando pasan vehículos, etc..

Además de todo esto, un sensor magnético es, como ya fue mencionado, un interruptor eléctrico. Por lo tanto, puede utilizarse para operar sobre circuitos eléctricos sin interfaz. Ten en cuenta para esto que el sensor simplemente cierra o abre el circuito entre las dos "patitas" de su ficha de conexión, por lo que puede reemplazar directamente a una llave interruptora común. Recuerda siempre que no debes trabajar con tensiones mayores de 24 voltios en corriente continua y 48 voltios en corriente alterna . En cuanto a la corriente soportada por estos sensores magnéticos, el máximo es de 900 miliamperes de consumo constante

.

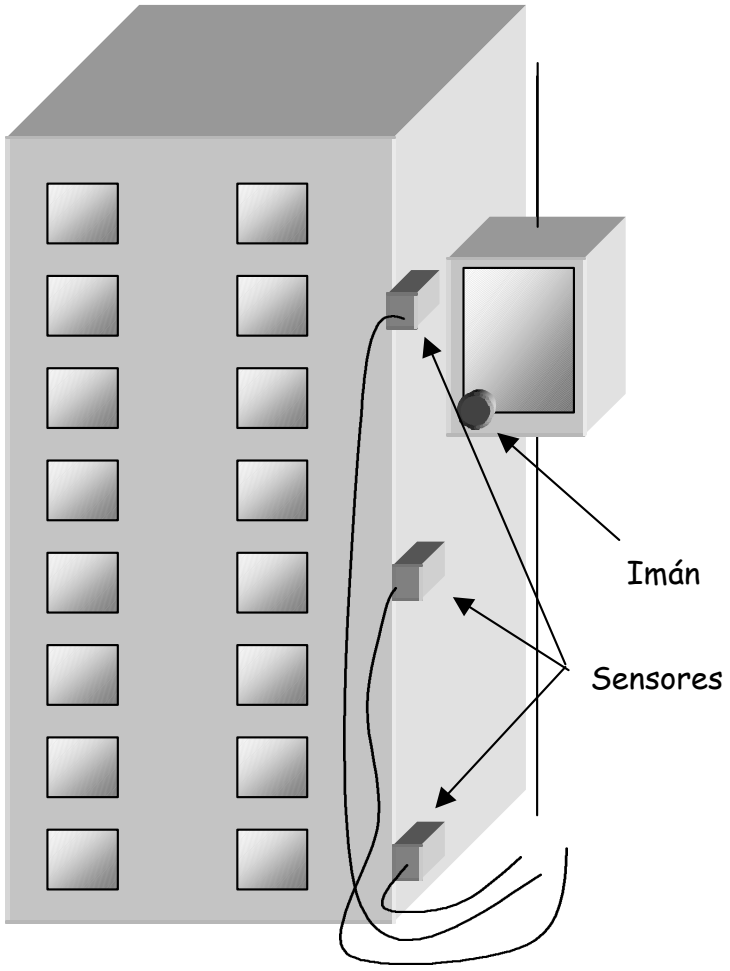
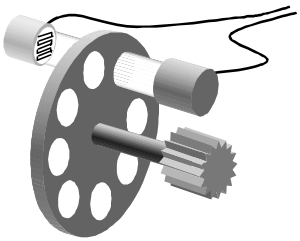


Figura C.2: Sensado de la posición de un ascensor.



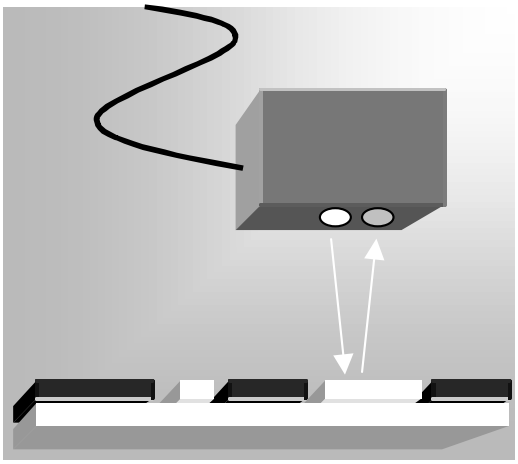
Sensores Ópticos

Estos sensores, capaces de "ver", poseen una gran variedad de aplicaciones en robótica educativa, que pueden ir desde la distinción entre objetos claros y oscuros hasta el control de la iluminación en una maqueta, pasando por la detección de obstáculos, etc.. Veamos un poco más.

Funcionamiento

Estos sensores poseen en su parte externa dos pequeños "ojos". Uno de ellos es un emisor de luz infrarroja. Este tipo de luz es invisible al ojo humano y tiene muchos usos en la vida cotidiana. Se utiliza, por ejemplo, en los controles remotos de televisores y otros electrodomésticos. El otro "ojo" del sensor es un receptor de luz infrarroja. Lo que el sensor hace, es enviar, mediante su emisor, un haz de esta luz y detectar, mediante su receptor, si la luz "regresa". Si esto ocurre, es porque el haz de luz ha rebotado en algún objeto que se cruzó en su camino. Hay que tener en cuenta que los objetos oscuros (sobre todo los pintados de color negro), prácticamente no reflejan la luz, por lo que el haz no regresa al sensor si el objeto con que se encontró era oscuro.

Esto resulta ser ventajoso, ya que hace que el sensor sirva para diferenciar entre objetos claros y oscuros.



Los colores que mejor reflejan este tipo de luz son el blanco, y el rojo, pero si el objeto es brillante, plateado o espejado, la luz es reflejada con mayor intensidad aún. Es por esto que para detectar un objeto blanco, hay que acercarlo más al sensor que para detectar un objeto espejado.

Si se recorre un tablero de ajedrez con un sensor de estos a la altura apropiada, el mismo se activará y desactivará dependiendo del color sobre el cual se encuentre posicionado.

Del sensor sale un cable que termina en dos fichas de

conexión. Una es una pequeña ficha de dos vías, para conectar el sensor a la entrada de la interfaz. La otra es una ficha que tiene el aspecto de las utilizadas en las salidas de audio y video de las videograbadoras. La misma es para la alimentación del sensor. Si ya haz trabajado con sensores de choque o magnéticos, te preguntarás por qué aquellos no tenían esta ficha extra. La respuesta es fácil: el sensor de luz debe emitir un haz de luz infrarroja, y esto requiere de energía, cosa que no precisaban los otros sensores. Además, por la forma en que está construído, también necesita la alimentación para que el receptor funcione correctamente. Para alimentar estos sensores, se requiere un accesorio de alimentación. Este se conecta a la fuente de alimentación de la interfaz. Del mismo sale un cable con un conector igual al de la fuente de alimentación de la interfaz y debe ser conectado a la entrada de alimentación de la i-723. Dicho de otro modo: el accesorio de alimentación se intercala entre la interfaz y su fuente de alimentación. Cada accesorio de estos puede alimentar hasta cuatro sensores ópticos.

Aplicaciones prácticas

Como ya se dijo, las aplicaciones son numerosas. Ejemplos típicos son el robot que sigue una línea negra sobre una superficie blanca; el vehículo que detecta obstáculos sin chocar con ellos y los esquiva (mediante un adecuado programa en la computadora que lo controla); la diferenciación entre objetos oscuros y claros en cintas de transporte (como las que se usan en las líneas de producción de las fábricas reales); la detección del paso de un pequeño tren (u otro móvil) en miniatura… Además, teniendo dos sensores, se pueden medir velocidades (por ejemplo velocidades de móviles en planos inclinados) y teniendo tres, aceleraciones.

Otra particularidad de estos sensores es que, como en la luz del ambiente (incluso en la luz generada por lámparas incandescentes de las comunes) hay cierta "cantidad de luz infrarroja", si se coloca al sensor apuntando hacia una lámpara (u otra fuente de luz), se podrá comprobar que el mismo se activa o se desactiva según esté oscura o iluminada la habitación. Con esto en mente, es fácil imaginar una forma de hacer que una maqueta de una ciudad se ilumine si "se hace de noche" Otros usos pueden ser el conteo de vueltas utilizando ruedas pintadas de negro con zonas blancas: cada vez que la rueda gira y la franja blanca pasa por el sensor, este se activa. De esta forma la computadora puede saber cuántas revoluciones dio una rueda (de más está decir que si se conoce el perímetro de la rueda, se puede obtener la distancia recorrida por ella). Como última aclaración, vale decir que estos sensores pueden servir como simples interruptores eléctricos, al igual que los sensores magnéticos y los de choque y así controlar circuitos en los que no hay ninguna interfaz conectada. Por ellos puede circular tanto corriente continua como alterna, siempre y cuando no se sobrepasen los 24 voltios y 900 miliamperes en continua y 48 voltios y 9 00 miliamperes alterna. En cuanto a la alimentación, es de 9 voltios.

Si el sensor comienza a oscilar (activarse y desactivarse rápidamente, como si estuviera loco), debe cambiarse la distancia al objeto que se está sensando, ya que hay una distancia crítica en la que esta situación, llamada histéresis, se puede dar.