

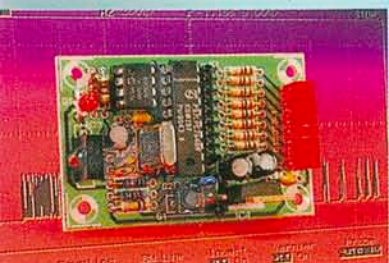
elektor

Nº 265
3,60 €

REVISTA INTERNACIONAL DE ELECTRONICA Y ORDENADORES



Interface
LPT/DMX



Receptor de
Infrarrojos
Multi-estándar

El Puerto Serie
Bajo Windows

Protocolo para
la Unidad
de Control
del EEDTS Pro

¡INTERFACES!

CompactFlash a IDE

I²C a RCX

1 Wire a RS232



Director
Eduardo Corral

Colaboradores
Jose M^a Viloch, Pablo de la Muñoza, Andrés Ferrer.

Redacción
VIDELEC, S.L.
Santa Leonor, 61 4^o-I
28037 MADRID
Tel.: 91 375 02 70
Fax: 91 375 61 42

Publicidad
Director de Publicidad: Julio Mollejo
julio.mollejo@larpress.com
Coordinadora de publicidad: Gema Sustaeta
C/ Medea, 4 5^a planta (Edificio ECU)
Tel.: 91 754 32 88
Fax: 91 754 18 58
28037 MADRID
email: publicidad@lar.es

Delegación Cataluña

ad press

Delegado: Isidro Ángel Iglesias López
iai_ad_press@infonegocio.com
Jefe de publicidad: Eva Matute Calvo
emc_ad_press@infonegocio.com
Comte d'Urgell, 165-167, B-1^o-3^a
08036 BARCELONA
Tel.: 93 451 89 07
Fax: 93 451 83 23
email: ad_press@infonegocio.com

Suscripciones
C/ La Forja, 27
28850 Torrejón de Ardoz (Madrid).
Tels.: 91 677 70 75 - Fax: 91 676 76 65

Edita

LAR
LARPRESS, S.A.

Director Editor
Julio Rodríguez

Director de Producción
Gregorio Gohí
Director Comercial
Eloy Zamanillo

Distribución en España
COEDIS, S.A.
Ctra. Nacional II Km. 602,5
08750 Molins de Rei - Barcelona
Tel.: 93 680 03 60

Importador para Chile:
Iberoamericana de Ediciones, S.A.
C/ Leonor de la Corte, 6035. Quinta Normal
Santiago de Chile
Tel.: 774 82 87 - 774 82 88

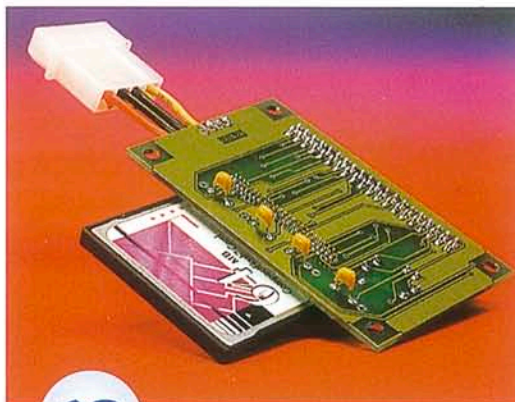
Distribución en Chile:
Alfa, S.A.
Distribución México:
Importador exclusivo Cade, S.A. de C.V.
C/ Lago Ladoga, 216
Colonia Anahuac-Delegación
Manuel Hidalgo, 11320 Mexico D.F.
Tel: 5254-2999 Fax: 5545-6879
Distribución Estados: Citem
Distribución D.F.: Unión de Voceadores
Distribución en Venezuela:
Distribuidora Continental
Distribución en Colombia:
Disunidas, S.A.
Distribución en Ecuador:
Disandes
PVP en Canarias, Ceuta y Melilla 3,61 €

Imprime
IBERGRAPHI 2002 S.L.L.
C/ Mar Tirreno, 7 Bis. Polígono Industrial San Fernando.
28830 San Fernando de Henares, Madrid.
Depósito legal: GU.3-1980
ISSN 0211-397X
31/junio/2.002

Preimpresión
Videlec, S.L.
C/ Sta. Leonor, 61 4^o local I

Reservados todos los derechos de edición.
Se prohíbe la reproducción total o parcial del contenido de este número, ya sea por medio electrónico o mecánico de fotocopia, grabación u otro sistema de reproducción, sin la autorización expresa del editor.

Las opiniones expresadas a lo largo de los distintos artículos, así como el contenido de los mismos, son responsabilidad exclusiva de los autores. Así mismo, del contenido de los mensajes publicitarios son responsables únicamente los anunciantes.
Copyright = 1996 Segment BV



68

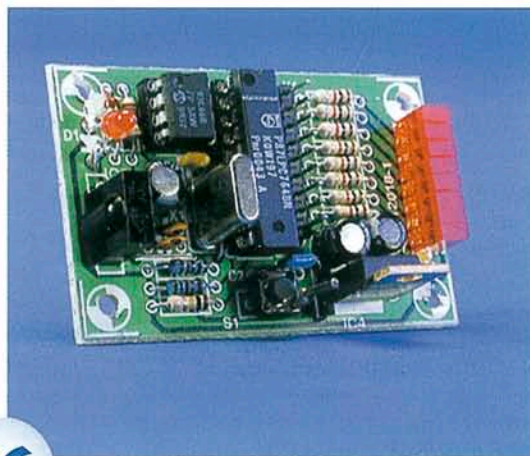
Controlador de Compactflash para bus Ide

Las tarjetas CompactFlash son un dispositivo de memoria que retienen su contenido sin necesidad de que esté presente una ten-

sión de alimentación. Son utilizadas, sobre todo, en cámaras digitales, entre otros equipos. Gracias a su "inteligencia", presente en este tipo de tarjetas, también pueden conectarse, fácilmente, a un ordenador, por ejemplo, para usarlas como "disco de estado sólido". El sencillo adaptador que se presenta en este proyecto facilita la conexión de todo tipo de tarjetas CompactFlash a un ordenador.

Receptor de infrarrojos multi-estándar

Este circuito complementa el circuito de control remoto con el que prácticamente todos los equipos electrónicos de gran consumo están equipados. Este receptor puede funcionar con una amplia gama de transmisores de infrarrojos.

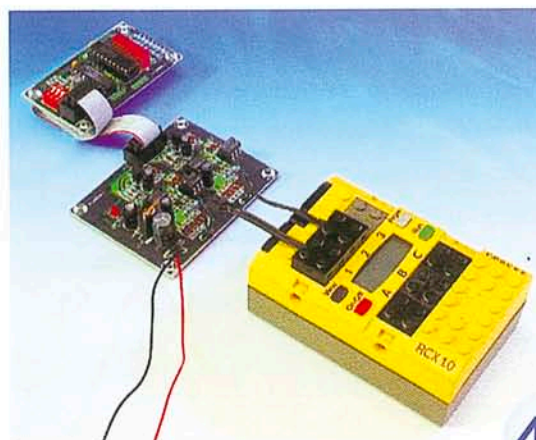


36

Interfaz I²C para el módulo RCX de Lego

Es conocido que el módulo RCX de la casa LEGO es adecuado, principalmente, para experimentos relacionados con el mundo de la robótica, pero una vez que el diseño crece más allá de la simple experimentación, rápidamente observamos que el número de entradas y de salidas que proporciona este módulo de control es insuficiente. Éste es el motivo por el que Elektor Electronics presenta el Interfaz I²C para el módulo de Lego. Con este dispositivo

se nos abre un nuevo mundo ante nuestros ojos: podemos conectar a este bus un mínimo de 128 dispositivos I²C.



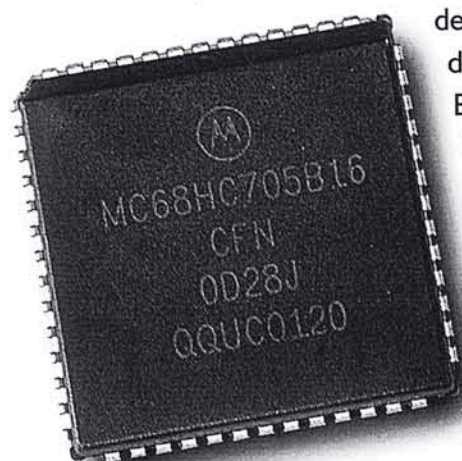
42

ARTÍCULOS INFORMATIVOS

Protocolo para la unidad de control del EEDTs Pro

Sin lugar a dudas, el protocolo para el EEDTs Pro ha sido el tema de mayor discusión relacionado con el EEDTs Pro,

debido al hecho de que es diferente de los viejos EEDTs y del protocolo de la casa Motorola. Así, disponemos de un nuevo microcontrolador para nuevos proyectos.



26

Montaje de Proyectos

- 6 Interfaz serie para el bus I-Wire
- 36 Receptor de infrarrojos Multi-Estándar
- 42 Interfaz I²C para el módulo RCX de Lego
- 48 Interfaz LPT/DMX
- 68 CompactFlash para bus IDE

Artículos Informativos

- 26 Protocolo para la unidad de control del EEDTs Pro
- 30 El puerto serie bajo Windows
- 54 Curso básico de microcontroladores (IV)
- 64 Medida de distancias sin contactos

Regulares

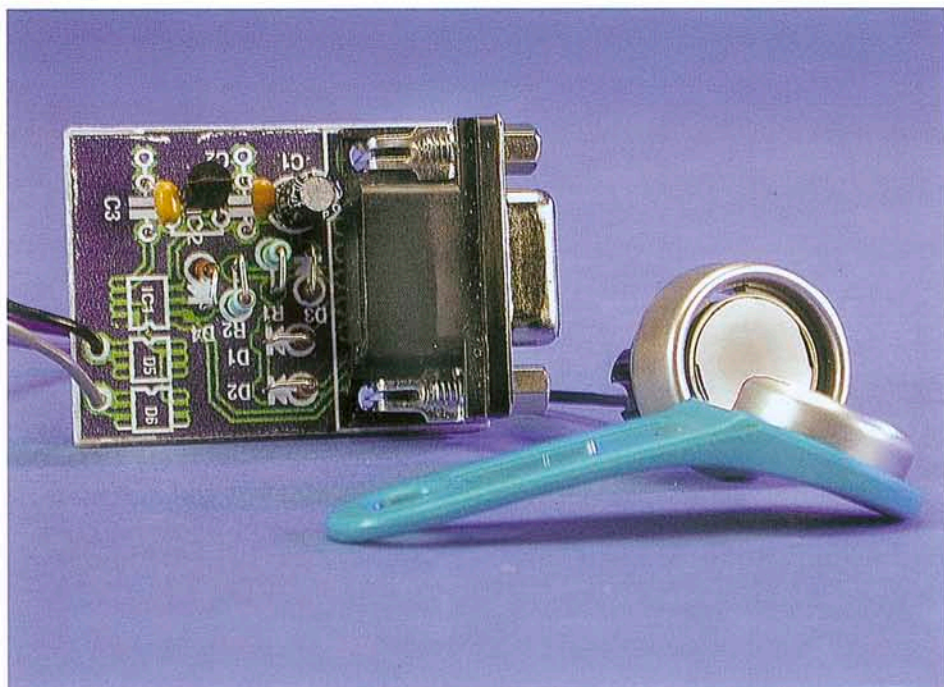
- 3 Sumario
- 14 Teletipo
- 40 Ojeada al próximo número
- 41 Nuevos Libros
- 47 Libros
- 60 EPS

Interfaz Serie para el Bus 1-Wire de Dallas

sencillo de trabajar desde Windows

Por Luc Lemmens

Los dispositivos 1-Wire de la casa Dallas Semiconductor son componentes que a través de una sencilla conexión serie pueden conectarse entre sí, a un circuito o a un ordenador. La interfaz RS 232 que se describe en este artículo hace que la conexión a un ordenador sea muy sencilla. Dallas proporciona los programas necesarios gratuitamente.



La serie 1-Wire de la casa Dallas ya ha recibido nuestra atención en ocasiones anteriores en esta revista, como es el caso del proyecto E-Key, y el Espía 1-Wire publicado. En estos proyectos la comunicación entre el bus 1-wire y el ordenador estaba controlada por un microcontrolador que había sido programado específicamente

para este propósito. La casa Dallas también fabrica circuitos integrados que se encargan de realizar el interfaz entre los microcontroladores y los ordenadores. Tanto los programas de evaluación y de desarrollo como los controladores pueden bajarse gratuitamente de la

página web de la casa Dallas. Esto nos puede aclarar la mayoría de las lagunas que nos encontramos cuando comenzamos a trabajar con estos dispositivos. El Kit de Desarrollo de Programas (SDK) del bus 1-Wire para Windows Me, 2000, 98, 95, NT 4.00 y NT 3.51, contiene ejemplos de programación en lenguajes C, C++, Pascal (Borland Delphi), Microsoft Access y Microsoft Visual Basic. Las herramientas de desarrollo para aplicaciones de 16 bits (Windows 3.1 y MS-DOS), también están disponibles.

Nuevos componentes

El número de componentes 1-Wire diferentes es bastante extenso. En publicaciones anteriores de la revista Elektor la mayor parte de nuestra atención estaba dirigida directamente a los denominados "iButtons". Estos dispositivos son componentes que están alojados de tal manera que se asemejan a un botón. Pero también existen componentes de este tipo tales como potenciómetros digitales, sensores de temperatura, relojes,

conmutadores, conversores A/D, IDs electrónicos y memorias, que están disponibles en encapsulados TO-92 o en versión SMD. Cada dispositivos tiene su propio código de familia (dependiendo del tipo de dispositivo) y un único número de serie, el cual permite que cada dispositivo que se conecta al bus pueda identificarse y direccionarse. La **Tabla 1** muestra una vista general de los componentes 1-Wire que están disponibles actualmente en el mercado.

El bus 1-Wire hace muy fácil construir pequeñas redes que, por ejemplo, puedan medir la temperatura en diferentes habitaciones de un edificio, o activar conmutadores, o reguladores de luz, etc.

El circuito es muy sencillo: dos hilos proporcionan alimentación y comunicación. Además, la casa Dallas ha puesto a disposición de los usuarios, por medio de Internet, un completo paquete de programas, con los que cualquiera puede comenzar a trabajar fácilmente. Por supuesto, el bus 1-Wire tiene

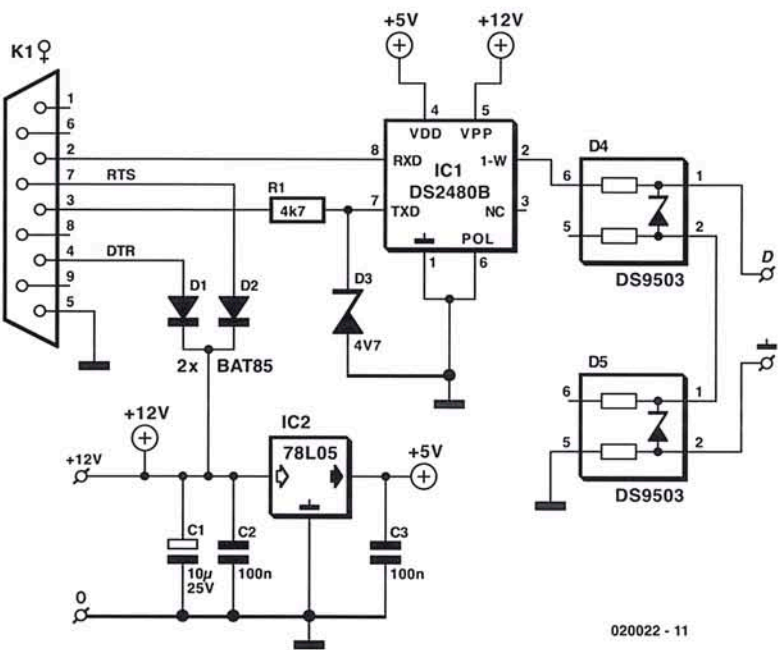


Figura 1. El circuito interfaz está formado, casi enteramente, por dos circuitos integrados de la casa Dallas.

I-Wire-Device	Función	Memoria
DS1820	Termómetro Digital	16 Bits de EEPROM
DS18B20	Termómetro Digital con Resolución Programable	16 Bits de EEPROM
DS18S20	Termómetro Digital de Alta Precisión	16 Bits de EEPROM
DS1821	Termostato Independiente	2 bytes de NV
DS1822	Termómetro Digital con Resolución Programable	Sin NV
DS2401	Número de Serie del Silicio	Sin Memoria Adicional
DS2404	Circuito Temporizador EconoRAM	4.096 bits de RAM
DS2404S-C01	Puerto Dual de Memoria más Temporizador	4.096 bits de RAM
DS2405	Conmutador Direccionable	Sin Memoria Adicional
DS2406	Doble Conmutador Direccionable	1.024 Bits de EPROM
DS2409	Acoplador MicroLAN	Sin Memoria Adicional
DS2417	Circuito Temporizador con Interrupción	32-bit Reloj Contador en Tiempo Real
DS2423	I-Wire RAM con Contadores	4.096 bits de RAM
DS2430A	I-Wire EEPROM	256+64 bits de EEPROM
DS2433	I-Wire EEPROM	4.096 bits de EEPROM
DS2434	Termómetro	32 bytes EEPROM, 32 bytes SRAM
DS2435	Termómetro /Histograma de Tiempo-Temperatura	32 bytes EEPROM, 32 bytes SRAM
DS2436	Termómetro, Conversor A/D de Tensión	32 bytes EEPROM, 8 bytes SRAM
DS2437	Medida Combustible, A/D Tensión, Reloj Tiempo Real, Temperatura	40 bytes de EEPROM
DS2438	Medida Combustible, A/D Tensión, Tiempo Transcurrido, Temperatura	40 bytes de EEPROM
DS2450	I-Wire Cuádruple Conversor A/D r	Solo Memoria de Control de Estado
DS2480B	I-Wire Controlador de Línea	Solo Memoria de Control de Estado
DS2490	Puente USB a I-Wire	Circuito de Control de Modo y FIFOs de E/S
DS2502	Sólo Añade Memoria	1.024 Bits de EPROM
DS2502-UNW	Ware Único	1.024 Bits de EPROM
DS2502-E64	IEEE EUI-64 Circuito de Nodo de Direcciones	256 bits pre-programado, 768 bits user-programmable
DS2505	Solo Añade Memoria	16,384 Bits EPROM
DS2505-UNW	Ware Único	16,384 Bits EPROM
DS2506	Solo Añade Memoria	65,536 Bits EPROM
DS2506-UNW	Ware Único	65,536 Bits EPROM
DS2890	I-Wire Potenciómetro Digital	Prestaciones y Control de Memoria
DS9502	ESD Diodo de Protección	-
DS9503	ESD Diodo de Protección con Resistencia	-

Tabla 1. Conjunto general de los dispositivos I-Wire disponibles en la actualidad.

que estar conectado a un ordenador a través de algún dispositivo. Pero esto no es un problema insalvable, ya que existen en el mercado circuitos integrados interfaces que han sido diseñados para este bus. Así, en este artículo presentamos una interfaz serie RS 232. La interfaz USB será el próximo dispositivo a publicar.

El corazón del interfaz: el DS 2480B

Este circuito integrado es pequeño con respecto a sus dimensiones (encapsulado SO-8), pero grande con respecto a sus prestaciones y opciones. Este componente puede parecer un dispositivo bastante complicado, pero gracias a los programas proporcionados por la casa Dallas no necesitamos saber demasiadas cosas sobre cómo trabajan estos circuitos integrados. Los lectores que realmente deseen saber cómo trabaja exactamente y qué sucede en el interior del DS 2480B, pueden estudiarse las 30 páginas de hojas de características de este componente. En este artículo haremos un gran uso del hecho de que este circuito integrado puede tratarse como una especie de "caja negra". Sólo nos interesa saber los dispositivos que trabajan con el bus 1-Wire y los que están conectados al mismo. Así, el modo en que se comunican con el ordenador es de un interés secundario.

El circuito

La interfaz serie entre el ordenador y el bus 1-Wire no requiere más de algunos componentes. Así, el DS 2480B hace todo el trabajo, en cooperación con el controlador que forma parte del programa. El esquema eléctrico del circuito se muestra en la **Figura 1**, y es la versión más completa que se ha podido conseguir. Para aplicaciones más básicas pueden omitirse algunos componentes del circuito.

Interfaz serie

El circuito integrado interfaz se comunica a través de las líneas Rx/D y Tx/D del puerto RS 232 del ordenador. Como todos sabemos, de acuerdo con el esquema estándar, un nivel lógico "uno" se corresponde con una tensión de -12 V y un nivel lógico "cero" con +12 V, mientras que el protocolo serie en el microcontrolador normalmente es de +5 V para nivel lógico "uno" y de 0 V para nivel lógico "cero". Cuando el terminal 6 del circuito integrado (POLARIDAD) se conecta a masa, las señales serie se invierten

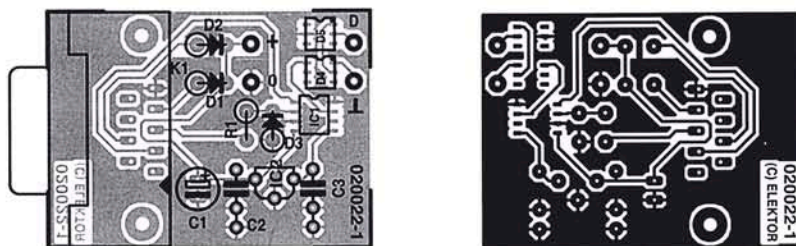


Figura 2. La placa de circuito impreso no es demasiado grande ya que sus circuitos integrados vienen en formato SMD. Debemos tener cuidado en el proceso de soldadura.

LISTA DE MATERIALES

Resistencias

R1 = 4,7 K Ω

Condensadores

C1 = 10 μ F, electrolítico de 25 V, radial
C2, C3 = 100 nF

Semiconductores

D1, D2 = BAT 85
D3 = diodo zéner de 4,7 V, 400 mW

D4, D5 = DS 9503 (6 terminales TSOC)
IC1 = DS 2480BS (8 terminales SOIC)
IC2 = 78L05

Varios

K1 = Conector "sub-D" de 9 terminales hembra en ángulo recto para montaje en placa de circuito impreso
Placa PCB, Código de pedido (a través del Servicio de Lectores), N° 020022-1.

en el interior del DS 2480B. La interfaz serie RS 232 es tan sencilla que sólo necesita unos componentes pasivos (la resistencia R1 y el diodo D3) para estabilizar la señal

en el terminal 7 (Tx/D). La resistencia y el diodo zéner aseguran que la tensión en este terminal esté limitada en el rango comprendido entre 0 y 4,7 V.

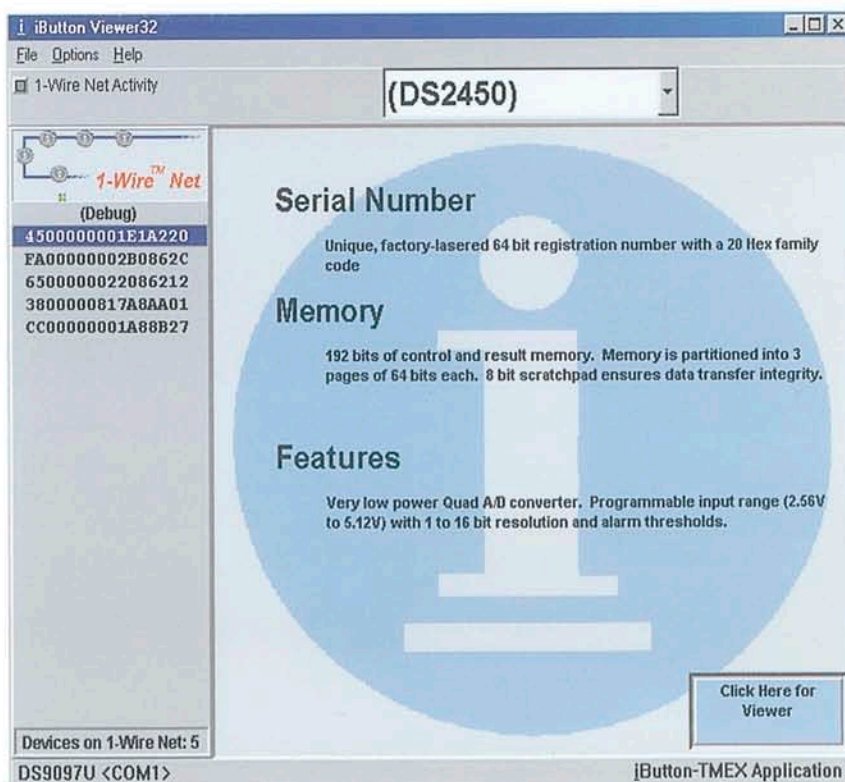


Figura 3. La ventana principal del visualizador iButton.

La mayoría (¡pero no todos!) de los modernos puertos RS 232 aceptan una tensión de 0 V para un nivel lógico "1" y + 5 V para un nivel lógico "0". Por este motivo no se ha añadido ningún circuito adicional para hacer que esta señal sea conforme con el estándar (- 12 y + 12 V, respectivamente).

Bus 1-Wire

Como su nombre indica, sólo hay un hilo en el DS 2480B dedicado a las comunicaciones para el bus. La mayoría de los circuitos integrados 1-Wire pueden estar alimentados directamente de este bus. Además, hay componentes con una memoria programable (EPROM, ROM Programable Eléctricamente, del tipo OTP, ya que no tienen ventana de borrado), que requieren pulsos de programación de + 12 V para poder almacenar los datos. Esta tensión de programación está disponible en la interfaz, sobre el terminal VPP (terminal 5) del DS 2480B y se aplica automáticamente en el bus cuando se ha suministrado el comando de programación.

La casa Dallas Semiconductor advierte explícitamente al usuario de que cuando se está programando sólo puede haber un circuito integrado EPROM conectado a la interfaz, con un trozo muy pequeño de hilo. Además, nunca debemos ejecutar los comandos de programación en el bus si existen componentes conectados al mismo que no contienen una memoria EEPROM. Si hacemos esto podemos causar daños a los componentes o al DS 2480B.

Los diodos D4 y D5 proporcionan una protección ESD (contra Descargas Electroestáticas) en la interfaz, de manera que no puede dañarse por descargas electrostáticas procedentes del bus 1-Wire. El circuito integrado DS 9503 dispone de un diodo interno muy rápido, con una tensión zéner de unos 7,5 V. El efecto de la resistencia interna de 5 W puede ser ignorado durante el proceso de comunicación normal del componente, pero se convierte en una alta impedancia con respecto al diodo zéner, si se produce una descarga en el bus. Esta resistencia asegura que

la corriente de descarga fluya a través de los diodos y no a través del terminal 2 del circuito integrado interfaz. Si la interfaz no se está utilizando con memorias EEPROM (las cuales requieren pulsos de programación de + 12 V), la tensión normal de funcionamiento en el bus no deberá exceder los + 5 V, por lo que el diodo D5 puede omitirse. El terminal 5 de D4 se conecta entonces directamente al plano de masa en la placa del circuito impreso y la conexión de masa de K2 al ánodo (terminal 2 de D4).

La fuente de alimentación

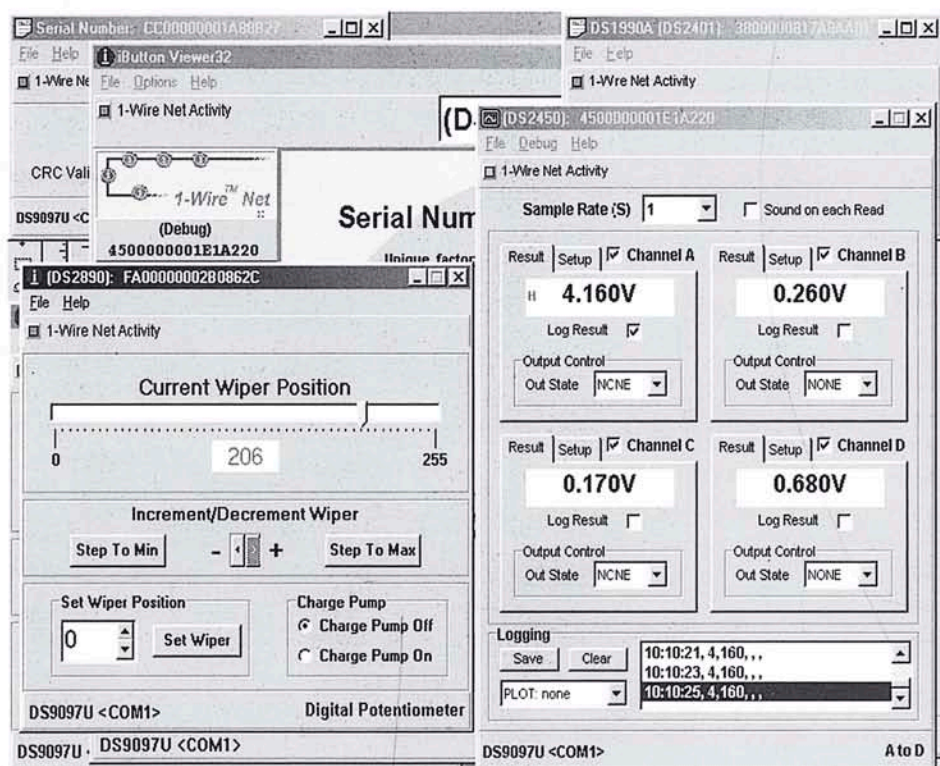
La tensión de alimentación se puede obtener a través del puerto RS 232, pero será insuficiente para

En circunstancias normales de trabajo, las señales DTR y RTS pueden proporcionar suficiente potencia para el interfaz. Cuando una de las líneas está a nivel alto (+ 12 V) disponemos de suficiente corriente para que IC2 pueda proporcionar una adecuada tensión de alimentación regulada de + 5 V.

Montaje y verificación

La **Figura 2** muestra la serigrafía de la placa del circuito impreso que ha sido diseñada para esta interfaz.

Los circuitos integrados DS 2480B y DS 9503 son componentes en formato SMD y se recomienda montar en primer lugar estos componentes en el lado de las pistas de cobre de la placa del circuito impreso. Antes de nada, deberemos poner atención en la marca que nos indica el terminal 1 (una



los componentes con memoria instalados en el bus 1-Wire, que requieren una tensión de programación de + 12 V. Para poder programar memorias se debe conectar una tensión de alimentación externa regulada de + 12 V a K3. Algo de lo que siempre debemos asegurarnos es de tomar nota de los avisos con respecto a la programación de las memorias mencionados anteriormente.

pequeña hendidura). Una vez localizado el terminal 1 pondremos una pequeña cantidad de estaño en la esquina del mismo, colocaremos el circuito integrado correctamente sobre su posición y soldaremos en primer lugar dicho terminal. Seguidamente se procederá a soldar el terminal de la esquina opuesta diagonalmente. A continuación, verificaremos que el resto de los terminales están alineados adecuadamente con sus correspondientes "pads" del circuito impreso, antes del soldar los otros terminales.

Si en algún momento colocamos demasiado estaño en los terminales podemos retirarlo con una malla de soldadura.

Continuaremos montando las resistencias y los diodos de forma vertical en la cara de componentes de la placa del circuito impreso y, por último, pero no menos importante, el circuito integrado regulador de tensión IC2, los condensadores y los conectores.

Cuando hayamos acabado este proceso inspeccionaremos cuidadosamente el trabajo realizado. Pondremos especial atención a los posibles cortocircuitos entre los terminales de los componentes SMD.

Antes de verificar el circuito deberemos bajarnos primero el visualizador iButton y los controladores TMEX de la página web:

www.ibutton.com/software/tmex/index.html

La versión más reciente en la actualidad es la 3.20. El fichero *TM320_32.EXE* instala los controladores y el programa y, cuando ha finalizado, inicia inmediatamente el visualizador iButton. Lo primero que hace este programa es buscar la interfaz 1-Wire.

El siguiente paso será conectar el circuito a un cable serie normal de 9 hilos (¡NO un cable null módem!) por un extremo y por el otro a un puerto COM libre de nuestro ordenador. Si aún no tenemos componentes conectados al bus 1-Wire la tensión de alimentación externa en K3 no es necesaria. Si todo está bien nuestra interfaz será detectada después de haber actuado sobre el botón "Auto-Detect". Seguidamente el programa de configuración se cerrará y el visualizador iButton se reiniciará de nuevo. Si el programa es incapaz de comunicarse con la interfaz aparecerá inmediatamente un mensaje de error. Si éste es nuestro caso, el fallo estará relacionado casi siempre con el circuito. Por ello, debemos verificar cuidadosamente nuestras soldaduras, la orientación de los circuitos integrados, de

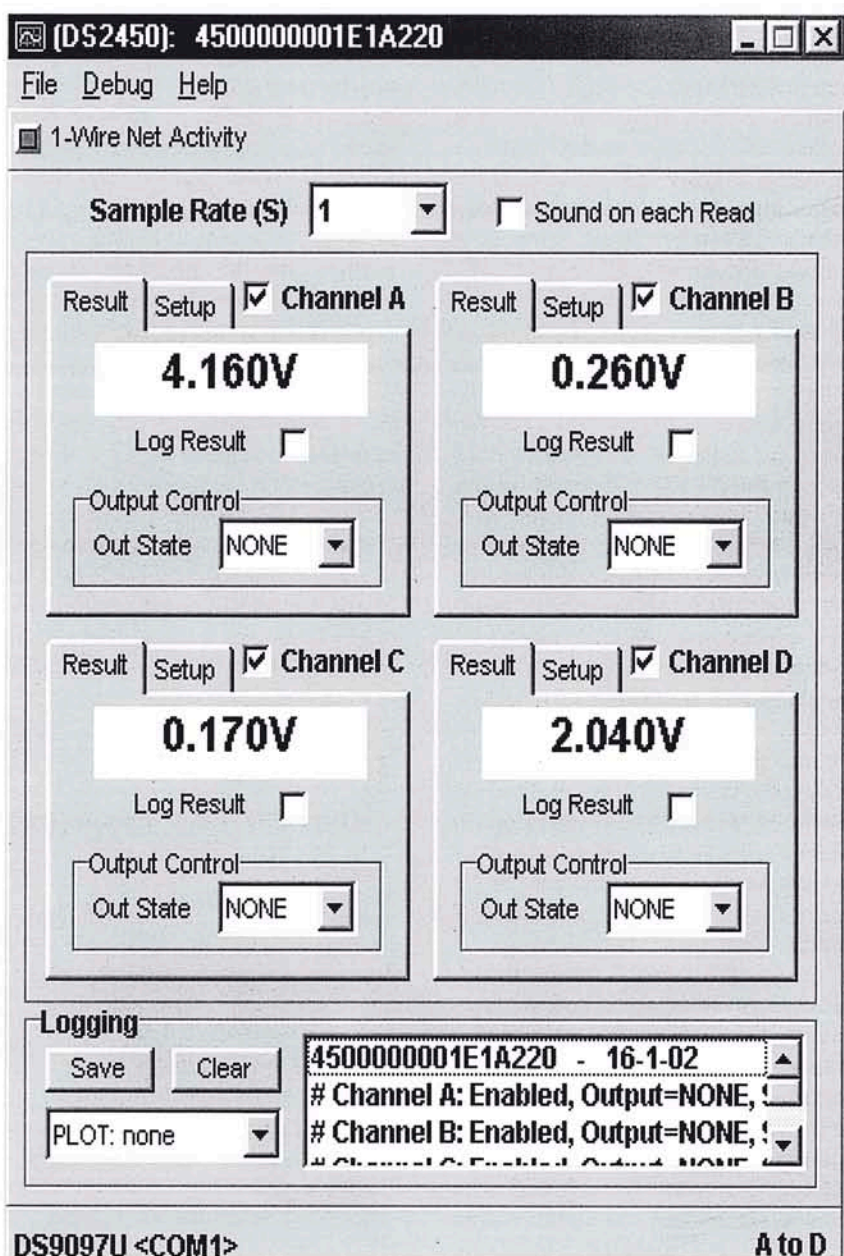


Figura 4. Esta ventana muestra los cuatro resultados de las medidas y varias configuraciones del conversor A/D de cuatro canales DS 2450.

Direcciones de Internet:

SDK para I-Wire:

www.ibutton.com/software/tmex/

Notas de Aplicación para I-Wire:

http://dbserv.maxim-ic.com/an_prodline2.cfm?prodline=21

los condensadores electrolíticos y de los diodos.

Si todo está correctamente el programa nos indicará que tenemos un adaptador DS 9097U conectado al puerto COM correspondiente. Ésta es la interfaz que hemos construido con el DS 2480B. Llegados a este punto, la comunicación entre el ordenador y la interfaz estará funcionando adecuadamente. En este momento podemos proceder a trabajar con el bus 1-Wire, el cual ha sido el primero en iniciarse.

Dispositivos en el bus 1-Wire

Una vez que tenemos funcionando totalmente el interfaz, es el momento de conectar uno o más dispositivos al bus 1-Wire. Para ello leeremos cuidadosamente las hojas de características correspondientes a cada dispositivo. Algunos dispositivos pueden conectarse directamente con dos hilos al conector K2 del interfaz, pero otros pueden requerir componentes externos (por

ejemplo, un circuito integrado con cristal de oscilación DS 2417), o su propia tensión de alimentación estabilizada.

Aquellos a los que no les guste programar pueden comenzar a trabajar inmediatamente con el visualizador iButton. Este programa muestra automáticamente los dispositivos que están presentes en el bus 1-Wire. La **Figura 3** representa la ventana principal del visualizador. En la columna de la izquierda están los números de identificación única de los cinco dispositivos que han sido encontrados en el bus. A la derecha de la pantalla tenemos una breve descripción de los componentes que han sido señalados en la izquierda, en este caso un DS 2450 (un conversor A/D de cuatro canales).

Al actuar sobre el botón "Click Here for Viewr" (es decir, "Pulsar aquí para el Visualizador"), se abre una ventana que nos muestra cuatro valores medidos y varias configuraciones para este circuito integrado (ver **Figura 4**). La configuración puede modificarse. Por ejemplo, se puede cambiar el intervalo de tiempo entre dos medidas, pueden activarse o desactivarse uno o más canales, o ajustar el rango de medida de un canal, etc.

También es posible abrir múltiples ventanas al mismo tiempo, de manera que se puedan mostrar varios dispositivos en la pantalla a la vez. De este modo, es fácil obtener una vista general de la red y realizar experimentos fácilmente. Un modo de realizar experimentos es interconectando diferentes dispositi-

tivos 1-Wire. Por ejemplo, podemos conectar un potenciómetro 1-Wire a un canal del conversor A/D y observar el cambio en la medida resultante de este canal.

Todo esto es muy sencillo y muy entretenido y puede ser suficiente para una sencilla aplicación, aunque el visualizador iButton no permite que varios dispositivos estén interconectados en el programa. Si deseamos realizar más interconexiones, como por ejemplo que el circuito active automáticamente un conmutador conectado al bus cuando el resultado de la medida en el canal A del conversor A/D exceda de un cierto valor, deberemos realizar la programación por nosotros mismos. El Kit de Desarrollo de Programas (SDK), disponible en la página web de la casa Dallas, nos facilita esta tarea.

SKD 1-WIRE

Este paquete de programas podemos obtenerlo de la misma página web de la que bajamos el visualizador iButton. Por el momento, la versión más actual del SDK es la V4.0 ALPHA. Esta versión contiene, entre otras cosas, la documentación para el TMEX API (Interfaz de Programas de Aplicación) y descripciones de las funciones que pueden ser llamadas desde el interior de nuestro programa para comunicarnos con los dispositivos que están conectados al bus 1-Wire. También contiene controladores para la interfaz serie y el interfaz USB de un ordenador, así como ejemplos de programación en

Pascal, Delphi, C y Visual Basic. Esto es más que suficiente información y material de lectura para permitirnos comenzar con nuestra propia programación.

El bus 1-Wire en la práctica

Por desgracia, no podemos conectar un número ilimitado de dispositivos 1-Wire a la interfaz descrita en este artículo. Además, la arquitectura física de la red total, así como las longitudes de los cables y la topología de los mismos en el bus, están limitados en la práctica. Siempre y cuando respetemos que la interfaz y los dispositivos conectados al mismo estén diseminados por nuestro banco de trabajo y que la longitud del bus sea menor de un metro, no tendremos demasiados problemas. Pero antes de que comencemos a realizar taldros de forma aleatoria y tirar cables a través de la casa o entre otros edificios, recomendamos que se lean en primer lugar las notas de aplicación AN 148, "Guidelines for Reliable 1-Wire Networks" (es decir, "Guía para la realización de redes con el bus 1-Wire"). Esta información contiene una clara y precisa descripción de las reglas que deberemos respetar y los pasos a seguir para conseguir una buena comunicación a través de la red. Esta nota de aplicación, que también contiene información adicional sobre el DS 2480B, podemos encontrarla con otras notas de aplicación del bus 1-Wire.

(020022-1)

**CONSULTE
NUESTRAS
OFERTAS EN
ESPACIOS
PUBLICITARIOS**

**91 754 32 88
93 451 89 07**

**ANÚNCIESE
CON NOSOTROS
LE VERÁN
MÁS DE
70.000
POTENCIALES
CLIENTES**

Circuitos Impresos



**Diseño y Fabricación
Prototipos y Series**

ELECTRONICA INDUSTRIAL, S.A.

OFICINAS Y TALLERES

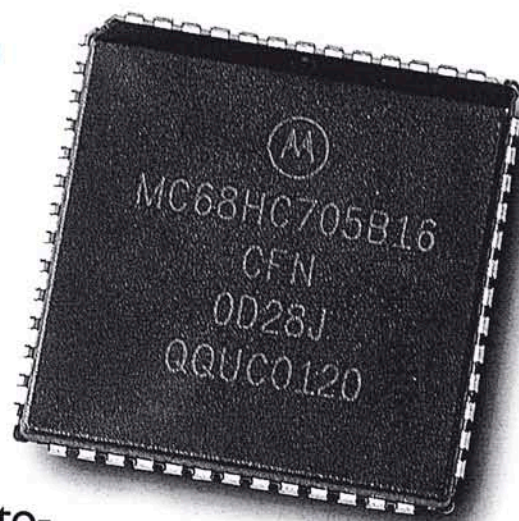
C/MOLINA, 39 - 28029 MADRID
Tel. 91 315 18 54 - Fax 91 315 18 95
E-mail: roan@solitel.es

Protocolo para la Unidad de Control del EEDTS Pro

ampliaciones del protocolo

Por S. van de Vries

Sin lugar a dudas, el protocolo para el EEDTS Pro ha sido el tema de mayor discusión relacionado con el EEDTS Pro, debido al hecho de que es diferente de los viejos EEDTS y del protocolo de la casa Motorola. Así, tenemos un nuevo microcontrolador para nuevos proyectos...



Además, muchos usuarios y suministradores de programas han encontrado bastantes dificultades en convertir direcciones y datos en la información codificada requerida por el controlador EDTs.

En consecuencia, el protocolo se ha ampliado para incluir el conjunto de comandos que se muestra en la **Tabla 1**.

Naturalmente, el conjunto de comandos existente (como se describe en el libro EEDTS Pro), continúa siendo soportado en su totalidad, y la nueva unidad de control es totalmente compatible hacia abajo con las viejas versiones.

El protocolo Märklin se creó como punto de partida para comenzar a andar, pero como el protocolo no dispone de ningún comando para funciones adicionales, se ha añadido, de forma específica, un byte extra (en particular, para los comandos de control de locomotoras).

Estructura general de comandos

La unidad de control EEDTS Pro se comunica con el ordenador a través de un puerto serie RS 232 que utiliza una configuración fija de 9.600 baudios, sin paridad, con 8 bits de datos y un bit de parada. Para asegurar la correcta transferencia de datos se ha utilizado un diseño en el que se devuelve un byte de respuesta por cada byte que envía la unidad de control.

Así, una instrucción de la unidad de control puede estar formada por 1, 2, 3 ó 4 bytes, en la que el primer byte es el byte de comando.

En general, el byte de comando se devuelve de forma literal como una indicación de que la instrucción ha sido recibida correctamente, excepto en el caso de los comandos de retorno (instrucciones de la unidad de control de un único byte), en cuyo caso la información de retorno se devuelve directamente para poder conseguir una alta velocidad de transferencia.

Si la unidad de control no envía la información hacia la vía (por ejemplo, si el repetidor está fuera de servicio debido a un cortocircuito), se devuelve el valor "65" como byte de comando para informar al ordenador de esta situación.

Para poder informar a la unidad de control en funcionamiento que el repetidor está fuera de servicio, se necesita realizar una pequeña modificación en el interfaz del repetidor. Esta modificación será descrita en la siguiente entrega, junto con el direccionamiento del teclado y del controlador independiente.

Comandos de movimiento

El primer byte del comando de la locomotora puede tener un valor comprendido entre 0 y 15. Este valor representa la velocidad, teniendo en cuenta que en el antiguo formato, el nivel de velocidad "1" es el comando "inverso" (en el protocolo Märklin este nivel de velocidad es de "15").

El segundo byte que se envía representa la dirección de la locomotora. Las primeras 80 direcciones se corresponden con las 80 direcciones disponibles en el formato Märklin. Sin embargo, como en el nuevo formato hay espacio para 256 direcciones, el comando se ha ampliado hasta las 256 direcciones.

El único "estándar" existente para esta secuencia lo hemos encontrado en Uhlenbock, motivo por el que se ha elegido esta secuencia.

El tercer byte proporciona información acerca del formato, la dirección de viaje y de funciones adicionales. En la **Tabla 2** se muestra el significado de los bits individuales del viejo formato, mientras que en la **Tabla 3** se muestra la misma información para el nuevo formato. Por conveniencia, los valores también se muestran en notación decimal.

De este modo es posible generar funciones en el viejo formato. Una impor-

Tabla 1. Conjunto de comandos ampliados.

	Byte de Comando	Respuesta Normal	Corto-circuito	Segundo Byte	Respuesta	Tercer Byte	Respuesta	Cuarto Byte	Respuesta
Comando de Locomoción	0 - 15	0 - 15	65	0 - 255	0 - 255	Ver Tablas 2 & 3	Tercer Byte		
Último cambiado	190	1..64	65						
Módulo de vuelta a señalización	192 - 255	0 - 255	65						
Reset (conmutador) de giro	32	32	65						
Girar (cambio) hacia la izquierda	33	33	65	0 - 255	0 - 255				
Girar (cambio) hacia la derecha	34	34	65	0 - 255	0 - 255				
Programa manual del controlador	40	40	65	16 - 23 24 - 31	0 - 255 Velocidad	0 - 255	Dirección Previa	0 - 255	Estado previo
Lectura manual del controlador	41	41	65	16 - 23 24 - 31	0 - 255 Velocidad	0	Dirección	0	Estado

tante diferencia entre las dos tablas es que en el viejo formato el total de las cuatro funciones es modificado por cada byte, mientras que en la tabla del nuevo formato se utiliza un byte independiente para seleccionar cada función.

En el viejo formato, si no desea modificar las funciones adicionales se envía un "0" o un "3" (el "n" en la tabla indica que F1 - F4 no cambian). El nuevo formato también incluye los comandos de "hacia atrás" y de "invertir".

Los lectores más observadores pueden haber notado ya que se han reservado dos bits (b1 y b2) para la función F0 y que estos bits siempre tienen el mismo valor en la tabla ("00" ó "01"). Los valores "01" y "10" también están permitidos, pero están reservados para la asignación de futuras funciones.

Comandos de retorno

Último cambiado

El comando "último cambiado" ("190") nos demostrará, sin lugar a dudas, su utilidad para controles en tiempo real en montajes que tengan un gran número de unidades de señalización de retorno. El byte de respuesta contiene la dirección del señalizador de retorno cuyo estado de entrada ha cambiado (con respecto al momento en el que la unidad había sido leída). Esto hace posible determinar rápidamente qué unidad debe ser leída.

Módulos de señalización de retorno

Las unidades de señalización de retorno pueden leerse utilizando los comandos comprendidos entre el 192 y el 255.

El comando "192" se usa para leer la primera unidad, el "193" para la segunda unidad y así sucesivamente hasta el comando "255" para la 64ª unidad de

señalización de retorno del EEDTS (o 9 - 16 del 32ª S88 de Märklin).

El valor que la unidad de control devuelve proporciona una representación binaria de las entradas (en el caso de un módulo esto sería la dirección de la locomotora).

Comandos de giro (cambio)

Los comandos de giro (conmutación) son exactamente igual que los de Mär-

klin. Por ejemplo, el comando "33 - (1 - 4)" activa la salida "verde" de un k73, mientras que el comando "34 - (1 - 4)" activa la salida "roja". Incluso la peculiaridad de "33 - 0" y "34 - 0" representa el final de dos salidas del decodificador 64 que se ha implementado. La activación de giro queda anulada usando el comando "32".

Estos comandos limitan el número de decodificadores a 64. Si nuestro proyecto necesita soportar un mayor número podemos recurrir al conjunto de comandos normales del EEDTS, para direccionar hasta un total de 240 decodificadores.

Tabla 2. El viejo formato de Motorola

b8	b7	b6	b5	b4	b3	b2	b1	Decimal	F0	F1	F2	F3	F4
0	0	0	0	0	0	0	0	0	0	n	n	n	n
0	0	0	0	0	0	1	1	3	1	n	n	n	n
0	1	0	0	0	1	0	0	68	0	0	0	0	0
0	1	0	0	0	1	1	1	71	1	0	0	0	0
0	1	0	0	1	0	0	0	72	0	1	0	0	0
0	1	0	0	1	0	1	1	75	1	1	0	0	0
0	1	0	0	1	1	0	0	76	0	1	1	0	0
0	1	0	0	1	1	1	1	79	1	1	1	0	0
0	1	0	1	0	0	0	0	80	0	0	0	1	0
0	1	0	1	0	0	1	1	83	1	0	0	1	0
0	1	0	1	0	1	0	0	84	0	1	0	1	0
0	1	0	1	0	1	1	1	87	1	1	0	1	0
0	1	0	1	1	0	0	0	88	0	1	1	1	0
0	1	0	1	1	0	1	1	91	1	1	1	1	0
0	1	0	1	1	1	0	0	92	0	0	0	0	1
0	1	0	1	1	1	1	1	95	1	0	0	0	1
0	1	1	0	0	0	0	0	96	0	1	0	0	1
0	1	1	0	0	0	1	1	99	1	1	0	0	1
0	1	1	0	0	1	0	0	100	0	0	1	0	1
0	1	1	0	0	1	1	1	103	1	0	1	0	1
0	1	1	0	1	0	0	0	104	0	1	1	0	1
0	1	1	0	1	0	1	1	107	1	1	1	0	1
0	1	1	0	0	1	0	0	108	0	0	0	1	1
0	1	1	0	0	1	1	1	111	1	0	0	1	1
0	1	1	0	1	1	0	0	112	0	1	0	1	1
0	1	1	0	1	1	1	1	115	1	1	0	1	1
0	1	1	1	1	1	0	0	116	0	1	1	1	1
0	1	1	1	1	1	1	1	119	1	1	1	1	1

Controladores de programación manual

Programación manual de los controladores

Se puede utilizar el comando "40" para programar de forma manual los controladores. El byte de comando ("40") está seguido por un segundo byte que puede tomar un valor comprendido entre 16 y 23. El valor "16" selecciona al controlador 1, el valor "17" al controlador 2 y así hasta llegar al valor "23" que selecciona al controlador 8.

Si se elige un valor comprendido en el rango de 16 a 23, la modificación de las direcciones será temporal y no será válida después de un apagado del circuito. Para almacenar permanentemente la modificación de las direcciones debe elegirse un rango comprendido entre 24 y 31 ("24" para el controlador 1... y "31" para el controlador 8).

Una vez que la unidad de control ha recibido el segundo byte, devolverá el estado del controlador. Este estado puede estar comprendido en el rango de "0" (control girado totalmente a la izquierda) y "255" (control girado totalmente a la derecha).

El tercer byte determina la dirección que se debe configurar para el controlador, mientras que el cuarto byte indica el formato del controlador y las funciones que están activadas o desactivadas en la locomotora (o en el programa que corre en el ordenador). De esta forma, por ejemplo, la unidad de control sabe que si la función F3 está habilitada, debe enviar un comando "F0 off" a la locomotora cuando se haya pulsado el botón F0 en el controlador manual.

Los bits del cuarto byte tienen los siguientes significados:

b8 b7 b6 b5 b4 b3 b2 b1
s/h o/n f/r F4 F3 F2 F1 F0

Por ejemplo, si el cuarto octeto tiene un valor asignado de "36" (en binario '00100100'), el controlador se configurará de la siguiente manera:

- s/h = 0 Selección de formato:
1 = formato seleccionado por programa
0 = formato seleccionado por circuito
o/n = 0 formato:
1 = viejo formato
0 = nuevo formato
f/r = 0 dirección de trayecto:
1 = hacia atrás
0 = invertir
F4 = 0 (F4 off)
F3 = 0 (F3 off)
F2 = 1 (F2 on)
F1 = 0 (F1 off)
F0 = 0 (F0 off)

Si el bit s/h está a nivel lógico "1" significa que el formato (nuevo o viejo) no está determinado por

Tabla 3. El nuevo formato de Motorola.

b8	b7	b6	b5	b4	b3	b2	b1	Decimal	Descripción	
0	0	0	0	0	1	0	0	4	F1 off	F0 off
0	0	0	0	0	1	1	1	7	F1 off	F0 on
0	0	0	0	1	0	0	0	8	F1 on	F0 off
0	0	0	0	1	0	1	1	11	F1 on	F0 on
0	0	0	0	1	1	0	0	12	F2 off	F0 off
0	0	0	0	1	1	1	1	15	F2 off	F0 on
0	0	0	1	0	0	0	0	16	F2 on	F0 off
0	0	0	1	0	0	1	1	19	F2 on	F0 on
0	0	0	1	0	1	0	0	20	F3 off	F0 off
0	0	0	1	0	1	1	1	23	F3 off	F0 on
0	0	0	1	1	0	0	0	24	F3 on	F0 off
0	0	0	1	1	0	1	1	27	F3 on	F0 on
0	0	0	1	1	1	0	0	28	F4 off	F0 off
0	0	0	1	1	1	1	1	31	F4 off	F0 on
0	0	1	0	0	0	0	0	32	F4 on	F0 off
0	0	1	0	0	0	1	1	35	F4 on	F0 on
0	0	1	0	0	1	0	0	36	Reservado	F0 off
0	0	1	0	0	1	1	1	39	Reservado	F0 on
0	0	1	0	1	0	0	0	40	Atrás	F0 off
0	0	1	0	1	0	1	1	43	Atrás	F0 on

el diodo de selección en el controlador manual sino que, en su lugar, está controlado por el bit o/n. Si el bit s/h está a nivel lógico "0", el formato está determinado por la selección del diodo en el controlador manual.

Después de enviar el tercer byte (de direcciones) a la unidad de control, ésta devuelve el conjunto de direcciones para el controlador manual (en el momento en que se envió el comando). A continuación del cuarto byte la unidad de control devuelve el estado previo leído en el controlador, en un byte de respuesta. Este dato de respuesta puede usarlo el programa del EEDTS Pro para, por ejemplo, restablecer los valores originales cuando el control se pasó de un control por programa a un control manual.

El octavo controlador manual no puede ser inhabilitado y enviará información continuamente hacia las vías. Por otro lado, debemos evitar tener más de un controlador seleccionado en la misma dirección (excepto para la "0") o enviar comandos con el ordenador a las direcciones que están usando los controladores manuales.

Para evitar que aparezcan problemas con el uso de controladores manuales, es una buena idea seleccionar, en primer lugar, todos los controladores manuales (o todos aquellos controladores manua-

les que no están en uso de forma activa) en la dirección "0" cuando comenzamos a ejecutar nuestro programa.

Lectura manual del controlador

El comando "41", que se emplea para leer el estado de un controlador manual, está muy estrechamente relacionado con el comando "programación manual del controlador". Sin embargo, el comando "lectura manual del controlador" sólo realiza tareas de lectura de datos y no cambia ninguna configuración del controlador. En este caso también se devuelve el estado del controlador a continuación del segundo byte.

Los valores que siguen al tercer y al cuarto byte son los mismos que los del comando "programación manual del controlador", con la diferencia de que no es importante el valor enviado por el ordenador, ya que la unidad de control no hace nada con estos valores.

Esto completa nuestra presentación de los cambios más importantes en el nuevo protocolo de la unidad de control. El nuevo microcontrolador está disponible a través de nuestro Servicios de Lectores bajo el código de pedido N° 010088-4 y se puede montar fácilmente en la placa de circuito impreso de la unidad de control existente.

El Puerto Serie Bajo Windows

con un ejemplo en Delphi

Por Paul Goossens

El uso de las funciones de E/S en programas que corren bajo Windows es aún una “zona gris” para la mayoría de los programadores. En este artículo intentamos arrojar alguna luz sobre el uso de los puertos serie RS 232 de nuestro ordenador, de manera que sean compatibles con Windows.

Cuando deseamos utilizar ciertos dispositivos dentro de un ordenador tenemos que pensar en que necesitaremos programas que los controlen. Con los viejos ordenadores domésticos la creación de esos controladores y de las rutinas de control era relativamente sencilla. Con aquellos equipos sólo había un programa ejecutándose al mismo tiempo, por lo que no teníamos que tener en cuenta si había otros programas que desearan tener acceso al mismo dispositivo. Un programa bien escrito y diseñado debería también asegurarse que el dispositivo que está controlando vuelve a su configuración inicial al final de la ejecución de dicho programa de control.

Con la introducción del sistema operativo Windows el mundo del ordenador cambió de la noche a la mañana. Así, fue posible ejecutar varios programas al mismo tiempo. Aunque en aquellos momentos esto ya era una práctica común en el mundo de los ordenadores profesionales, para el usuario doméstico este paso adelante parecía algo mágico.

Estos usuarios, que comenzaban a utilizar el control de los dispositivos directamente en sus programas, se vieron enfrentados, por desgracia, con un nuevo problema. ¿Qué sucedía cuando dos programas intentaban acceder al mismo dispositivo en el mismo momento? Cuando estábamos completamente seguros de que esta situación nunca podría ocurrir, esto no era ningún problema. La cosa podía llegar a ser aún peor, pues en algunos lenguajes de programación la facilidad de acceder directamente a los dispositivos del ordenador era prácticamente imposible. En las situaciones más ventajosas

existía la posibilidad de que aún era posible utilizar las instrucciones de E/S del procesador por medio de ciertos trucos de programación.

En este artículo intentaremos mostrarles que la programación correcta del uso del interfaz serie bajo el entorno Windows no es tan difícil como la mayoría de la gente imagina. También intentaremos proporcionarles algún programa con apariencia profesional que también se ejecutará en este entorno sin ningún problema, incluso en las más recientes versiones de Windows.

Sistemas operativos

Una corta pero sencilla descripción de un sistema operativo es que no se trata más que de un gran programa que gestiona y controla todos los componentes instalados en el ordenador. Algunos de estos componentes pueden ser el ratón, el teclado, la pantalla del ordenador, la memoria, el sonido y muchos otros.

La tarea del sistema operativo es aislar el programa que está ejecutando el usuario de la necesidad de conocer exactamente la configuración

Tabla I

Funciones útiles en las API's de Windows para el control de interfaz serie.

FileCreate	Abre un fichero
EscapeCommFunction	Control de las señales DTR y RTS
GetCommStatus	Solicita el estado de los terminales de entrada y del buffer de recepción.
ReadFile	Lee el buffer de recepción.
WriteFile	Escribe en el buffer de salida del puerto serie correspondiente.
GetCommState	Interroga el estado actual del puerto serie.
SetCommState	Configura el estado del puerto serie.
GetCommMask	Lee la máscara de eventos. Esta máscara determina qué incidentes genera un “evento” en Windows.
SetCommMask	Selecciona la máscara de eventos.
ClearCommError	Obtiene información teniendo en cuenta el último error en la interfaz serie.

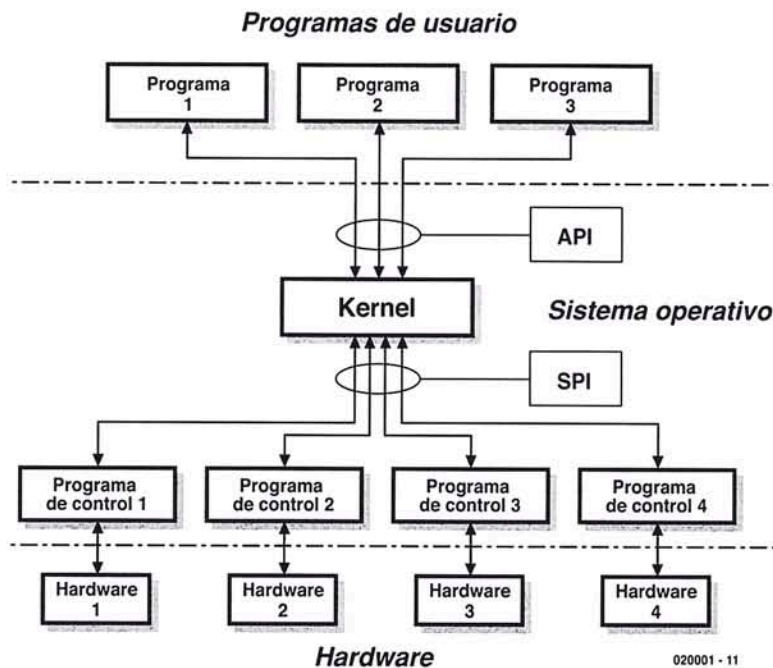


Figura 1. Estructura simplificada de un sistema con ordenador.

DLLs

Windows hace un uso extensivo de los ficheros DLL. Estos ficheros son librerías que pueden ser utilizadas por cualquier programa. La mayoría de las funciones de las APIs de Windows son accesibles por medio de estos ficheros DLLs. Cuando un programa hace uso de una DLL se dice que dicha librería está enlazada dinámicamente con este programa (de aquí el nombre: Dynamically Linked Library, es decir, Librería Enlazada Dinámicamente).

Para poder saber las funciones que pueden ser exportadas por las DLLs de Windows podemos utilizar el programa (al que generalmente se le da poca importancia) "quickview.exe", que está incluido con la mayoría de las versiones del sistema operativo Windows. Este programa puede ser utilizado para abrir DLLs. El programa Quickview presenta varias informaciones sobre la DLL abierta, incluyendo el nombre de la misma y sus funciones. De esta manera podemos tener una visión general bastante buena de las posibilidades de dicha DLL.

El programa Quickview también se puede ejecutar pulsando el botón derecho del ratón sobre un fichero DLL.

Tabla 2

Tabla 2. Los campos más significativos de un registro DCB.

DCBLength	La longitud del registro DCB.
BaudRate	Velocidad de la conexión en baudios.
Binary	Conexión binaria (tiene que estar a "True" en Windows).
Parity	Verificación de la paridad (activada o desactivada).
fOutxCtsFlow	Habilita o inhabilita el control de flujo en el terminal CTS cuando se envía información. Cuando está habilitado el controlador detiene la transmisión desde el puerto serie cuando la señal CTS está inactiva.
fOutxDsrFlow	Lo mismo que fOutxCtsFlow pero para la señal DSR.
fDtrControl	Configuración de la señal DTR. Esta configuración puede tomar los siguientes valores*: DTR_CONTROL_DISABLE: DTR pasa a nivel lógico bajo cuando se abre el puerto. DTR_CONTROL_ENABLE: DTR pasa a nivel lógico alto cuando se abre el puerto. DTR_CONTROL_HANDSHAKE: la señal DTR se utiliza para el "handshake".
FDsrSensitivity	Cuando está activa, el controlador ignorará todos los datos recibidos si la señal DSR está a nivel bajo.

* La señal DTR es controlada por la función EscapeCommFunction, excepto cuando la opción DTR_CONTROL_HANDSHAKE está activa.

interna del ordenador en el que se está ejecutando. El programa del usuario envía instrucciones al sistema operativo que se encarga de realizar el resto del trabajo. Esto hace posible que un programa pueda ejecutarse en ordenadores con diferentes configuraciones internas sin tener que adaptar y modificar el programa a cada configuración. El único requerimiento que se necesita es que tengamos siempre el mismo sistema operativo instalado en todos los ordenadores y que los componentes instalados acepten las tareas encomendadas.

Controladores

El control actual de los componentes del ordenador se realiza a través de los denominados controladores. Para cada uno de los componentes del ordenador el sistema operativo requiere un controlador específico. De este modo el sistema operativo puede hacer uso de los dispositivos más nuevos existentes en el mercado, con la única condición de que el fabricante incluya el controlador actualizado.

API's

El modo en el que los programas de los usuarios se comunican con el sistema operativo está establecido por las APIs de Windows (Application Programming Interface, es decir, Interfaz de Programación de Aplicaciones). Como su nombre indica, estas herramientas son una interfaz entre el programa del usuario y el sistema operativo. Normalmente, los compiladores disponen de librerías y de ficheros cabecera incluidos en sus características que hacen uso y llamadas a las APIs de Windows. Cuando éste no es el caso, tendremos que dirigirnos y consultar la documentación interna del sistema operativo. Las APIs están incluidas normalmente en el SDK (Software Development Kit, es decir, Conjunto de Desarrollo de Programas), del sistema operativo. Este SDK, normalmente, está disponible para bajarlo de forma gratuita de Internet en nuestro proveedor del sistema operativo.

Como podemos ver en la **Figura 1**, un programa de usuario sólo se comunica con las APIs del sistema operativo. Al programador no le concierne cómo trabaja internamente el sistema operativo, sólo le interesa que las APIs aseguran que el sistema operativo hará lo que el programa le está mandando. Es a un nivel más inferior cuando la API interroga al núcleo (kernel) del sistema operativo para que realice el procesamiento posterior. Cuando un programa requiere que se utilice un dispositivo, el núcleo del sistema se ayuda del programa controlador que acompaña a dicho dispositivo. En ese momento, el sistema operativo se comunica directamente con el circuito en cuestión.

Listado I

```
unit Unit1;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls;

type
  TForm1 = class(TForm)
    GroupBox1: TGroupBox;
    RadioButton1: TRadioButton;
    RadioButton2: TRadioButton;
    RadioButton3: TRadioButton;
    RadioButton4: TRadioButton;
    GroupBox2: TGroupBox;
    RadioButton5: TRadioButton;
    RadioButton6: TRadioButton;
    RadioButton7: TRadioButton;
    RadioButton8: TRadioButton;
    RadioButton9: TRadioButton;
    Button1: TButton;
    Button2: TButton;
    Label1: TLabel;
    Label2: TLabel;
    CheckBox1: TCheckBox;
    CheckBox2: TCheckBox;
    CheckBox3: TCheckBox;
    CheckBox4: TCheckBox;
    CheckBox5: TCheckBox;
    CheckBox6: TCheckBox;
    Button3: TButton;
    Label3: TLabel;
    Label4: TLabel;
    Label5: TLabel;
    Label6: TLabel;
    GroupBox3: TGroupBox;
    RadioButton10: TRadioButton;
    RadioButton11: TRadioButton;
    RadioButton12: TRadioButton;
    GroupBox4: TGroupBox;
    RadioButton13: TRadioButton;
    RadioButton14: TRadioButton;
    RadioButton15: TRadioButton;
    Label7: TLabel;
    Edit1: TEdit;
    Label8: TLabel;
    Button4: TButton;
    Edit2: TEdit;
    procedure FormActivate(Sender: TObject);
    procedure Button1Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    procedure RadioButtonComPort(Sender: TObject);
    procedure RadioButtonBaudRate(Sender: TObject);
    procedure CheckBox1Click(Sender: TObject);
    procedure CheckBox2Click(Sender: TObject);
    procedure UpdateClick(Sender: TObject);
    procedure ComboBox1KeyPress(Sender: TObject; var Key: Char);
    procedure RadioButtonParityClick(Sender: TObject);
    procedure RadioButtonStopBitsClick(Sender: TObject);
    procedure Button4Click(Sender: TObject);
    procedure Edit2KeyPress(Sender: TObject; var Key: Char);
  private
    { Declaraciones Privadas }
    hPort      : LongInt;      {controlador para el puerto serie}
    dcbCom      : TDCB;        {registro que almacena las propiedades}
                                {del puerto COM abierto rt}
    Open        : Boolean;      {¿Está el puerto COM abierto o cerrado?}
    ComPort     : Integer;      {Número del puerto COM}
    BaudRate    : LongInt;      {Velocidad en baudios deseada}
    Parity      : Byte;         {Paridad}
```

La comunicación entre el controlador y el núcleo del sistema no se realiza a través de una API sino del SPI (System Programming Interface, es decir, Interfaz de Programación del Sistema).

El Interfaz serie

El interfaz serie se utiliza más comúnmente en circuitos hechos en casa. Estos circuitos contienen habitualmente un microprocesador con un puerto serie interno. Es una tarea sencilla proporcionar los comandos para que el circuito envíe información a través de este puerto o la reciba por el mismo. Incluso cuando no se hace uso de un controlador, es todavía posible utilizar circuitos lógicos discretos para incorporar un puerto serie de E/S (un ejemplo de esto lo podemos ver en el artículo DCI-PLC del número de julio de 2001 de Elektor Electronics).

Los dispositivos del ordenador tales como el puerto de impresora, los interfaces serie, los dispositivos de E/S, etc., son tratados por el sistema operativo Windows (y por la mayoría del resto de sistemas operativos), como tipos especiales de ficheros. Esto significa que un programa tiene que abrir en primer lugar este "fichero" antes de ser capaz de utilizar el dispositivo. Una vez abierto este "fichero" existen una gran variedad de propiedades y funciones que pueden ser aplicadas y modificadas. Las funciones más relevantes que hacen relación a la interfaz serie se muestran en la **Tabla 1**.

Antes de comenzar nuestra programación para el puerto serie tenemos que tomar una decisión importante. La interfaz serie se puede abrir de modos muy diferentes.

El fichero puede abrirse en modo OVERLAPPED (SOLAPADO) o NON-OVERLAPPED (NO SOLAPADO). El modo SOLAPADO permite que el sistema operativo Windows procese cualquier acción de fondo, mientras el programa principal continúa su ejecución. Con el modo NO SOLAPADO, el programa en ejecución tiene que esperar a que el sistema operativo Windows complete cualquier acción sobre el puerto serie antes de continuar con la propia ejecución de dicho programa.

La ventaja del modo SOLAPADO es que el programa no se tiene que interrumpir de forma innecesaria

cada vez que se utiliza el puerto serie. La gran desventaja de este modo es que no podemos saber inmediatamente si el último comando se ha completado con éxito. Por lo tanto es necesario escribir en nuestro programa un "gestor de eventos", que será llamado cada vez que Windows ejecute un comando hacia el puerto serie. Por desgracia, es bastante más fácil cometer errores en este tipo de procesos, que, además, son bastante más difíciles de encontrar (tales como falta de señal de reloj y condiciones de transmisión). De hecho, sólo los programadores con más experiencia en la programación del puerto paralelo deberían utilizar este modo.

La gran ventaja del modo NO SOLAPADO es que el resultado del comando enviado al puerto serie está disponible una vez que éste se ha completado, sin tener que preocuparnos por si la operación se ha completado. En nuestro ejemplo, con un programa en Delphi, hemos utilizado el modo NO SOLAPADO, en parte para evitar procedimientos demasiado complejos.

Delphi

Como ejemplo, hemos escrito un programa sencillo en Delphi (ver **Listado 1**). Este programa hace uso de las llamadas más importantes a las APIs de Windows que utilizan el puerto serie.

La rutina "Button1Click" se ejecuta cuando el usuario hace "clic" sobre el botón "OPEN" del programa. Esta rutina abre un fichero con un nombre que se corresponde con el puerto COM seleccionado (COM1, COM2...).

La apertura del fichero es sólo la mitad del trabajo. El siguiente paso es configurar el puerto serie. En primer lugar se interroga sobre cuál es la configuración actual, utilizando el comando "GetCommState". Esta rutina de Windows rellena el registro "DCBCom" con el estado actual del puerto serie. En la **Tabla 2** se muestra una lista con los campos más interesantes de este registro. Seguidamente tienen que configurarse los campos más significativos del registro. Por último, los cambios de configuración efectuados deben activarse haciendo uso de la rutina "SetCommState". A partir de este momento el puerto serie está abierto con la configuración que hemos determinado.

```

        {usa sólo la constante declarada en Windows}
        {!SIN PARIDAD PARIDAD PAR PARIDAD IMPAR! }
        {Número de bits de parada: StopBits}
        {usa sólo la constante declarada en Windows}
        {!1 BIT STOP, 1,5 BITS STOP o 2 BITS STOP!}

        StopBits      : Byte;

        ReadBuffer    : string;

    public
        { Declaraciones Públicas }
    end;

var
    Form1: TForm1;

implementation

{$R *.DFM}

procedure TForm1.FormActivate(Sender: TObject);
begin
    ComPort:=1;                                {Selecciona la configuración de inicio}
    RadioButton1.Checked:=true;
    BaudRate:=9600;
    RadioButton6.Checked:=true;
    Parity:=NOPARITY;
    RadioButton10.Checked:=true;
    StopBits:=ONESTOPBIT;
    Radiobutton13.Checked:=true;

    Open:=false;
end;

procedure TForm1.Button1Click(Sender: TObject);
begin
    hPort:=CreateFile (PChar('COM'+IntToStr(ComPort)),
        GENERIC_READ or GENERIC_WRITE,0,nil,OPEN_EXISTING,
        FILE_ATTRIBUTE_NORMAL,LongInt(0));
    if (hPort = LongInt(INVALID_HANDLE_VALUE)) then
        MessageDlg ('Error opening port COM'+IntToStr(ComPort)+' : '+
            #13+#10+SysErrorMessage(GetLastError), mtError,[mbOk],0);

    if (hPort = LongInt(INVALID_HANDLE_VALUE)) then
    begin
        if GetCommState (hPort,dcbCom) then
        begin
            dcbCom.Baudrate:=BaudRate;
            dcbCom.ByteSize:=8;
            dcbCom.Parity:=Byte(Parity);
            dcbCom.Flags:=0;
            SetCommState (hPort, dcbCom);
        end;
    end;

    if (hPort = LongInt(INVALID_HANDLE_VALUE)) then
    begin
        Open:=true;
        CheckBox1Click(Self);
        CheckBox2Click(Self);
        UpdateClick (Self);
        Button1.Enabled:=false;           { Botón de apertura inhabilitado }
        Button2.Enabled:=true;           { Botón de cierre habilitado }
    end;
end;

procedure TForm1.Button2Click(Sender: TObject);
begin
    CloseHandle (hPort);                { Cierra el manejador de ficheros }
    Button2.Enabled:=false;             { Botón de cierre inhabilitado }
    Button1.Enabled:=True;              { Botón de apertura habilitado }
end;

procedure TForm1.RadioButtonComPort(Sender: TObject);
begin
    if (Open=true) then Button2Click(Self); { Cierra el manejador }

```



```

with Sender as TRadioButton do
begin
  ComPort:=(Sender as TRadioButton).Tag;
end;
end;

procedure TForm1.RadioButtonBaudRate(Sender: TObject);
begin
  if (Open=true) then Button2Click(Self); {Close handle }
  with Sender as TRadioButton do
  begin
    BaudRate:=(Sender as TRadioButton).Tag;
  end;
end;

procedure TForm1.CheckBox1Click(Sender: TObject);
var command : integer;
begin
  if CheckBox1.Checked=true then
    command:=SETDTR
  else
    command:=CLRDTR;
  if (EscapeCommFunction (hPort,command)=false) then
    MessageDlg ('Error changing signal : '+'
      #13+#10+SysErrorMessage(GetLastError), mtError,[mbOk],0);
end;

procedure TForm1.CheckBox2Click(Sender: TObject);
var command : integer;
begin
  if CheckBox2.Checked=true then
    command:=SETRTS
  else
    command:=CLRRTS;
  if (EscapeCommFunction (hPort,command)=false) then
    MessageDlg ('Error changing signal : '+'
      #13+#10+SysErrorMessage(GetLastError), mtError,[mbOk],0);
end;

procedure TForm1.UpdateClick(Sender: TObject);
var State : Cardinal;
    State2 : TCOMSTAT;
    Error : DWord;
    chRead : DWord;
    avail : DWord;
begin
  ReadBuffer:='';
  if (GetCommModemStatus (hPort,State)=false) then
    MessageDlg ('Error retrieving ModemStatus : '+'
      #13+#10+SysErrorMessage(GetLastError), mtError,[mbOk],0)
  else
    begin
      if ( (state and MS_CTS_ON)0) then
        CheckBox3.Checked:=True
      else
        CheckBox3.Checked:=false;
      if ( (state and MS_DSR_ON)0) then
        CheckBox4.Checked:=True
      else
        CheckBox4.Checked:=false;
      if ( (state and MS_RLSD_ON)0) then
        CheckBox5.Checked:=True
      else
        CheckBox5.Checked:=false;
      if ( (state and MS_RING_ON)0) then
        CheckBox6.Checked:=True
      else
        CheckBox6.Checked:=false;
    end;
  ClearCommError (hPort,Error,@State2);
  if (Error 0) then
    MessageDlg ('Error retrieving CommError : '+'
      #13+#10+SysErrorMessage(GetLastError), mtError,[mbOk],0);
  Avail:=State2.cbInQue;

```

En el programa ejemplo es posible introducir algún texto y transmitirlo con un simple "clic" de ratón. La rutina que se encarga de realizar la transmisión del texto se llama "Button4Click", la cual es bastante fácil de comprender. Esta rutina transmite el texto utilizando la rutina de Windows "Writefile".

El programa también puede mostrar cualquier dato serie recibido, así como el estado de todos los terminales de entrada. Esto se puede conseguir pulsando el botón "Update input". La rutina utilizada para este propósito es la llamada "UpdateClick". Así, lo primero que se ejecuta es la rutina de Windows "GetModemStatus", la cual configura una variable (llamada "State" en nuestro ejemplo), con el estado actual de los terminales de entrada. Las líneas que siguen tan sólo actualizan la pantalla con los datos del estado del puerto que acabamos de leer.

La segunda parte de esta rutina se encarga de tratar cualquier carácter que se haya recibido por el puerto. En primer lugar se hace llamada a la rutina "ClearCommError". Esta rutina sólo se encarga de limpiar cualquier aviso de error anterior. También rellena el registro TCOMSTAT con una variada información. La información más importante a tener en cuenta de este registro es el número de caracteres que están presentes en el buffer de entrada. Si este número es mayor de cero, el programa hace llamada a la rutina "ReadFile", con lo que el carácter recibido se puede mostrar en la pantalla.

El resto del programa está basado en proporcionar un interfaz agradable al usuario, permitiendo que éste pueda cambiar la configuración del puerto serie, elegir un puerto diferente, etc.

HTML y E/S

¿HTML y E/S? ¿Qué tiene que ver este artículo con este tipo de programación? La ejecución de programas también es posible dentro de las páginas HTML. Los lenguajes de sentencias (Scripts) han sido creados especialmente para este propósito. Los dos mejores lenguajes conocidos de sentencias son JavaScript y VBScript. Les recomendamos que utilicen JavaScript, ya que la mayoría de los navegadores y visualizadores de Internet

soportan estos lenguajes. Para comenzar con este tipo de lenguajes debemos señalar que sólo tienen una funcionalidad limitada. Después de la introducción de los componentes ActiveX de Microsoft, ha sido posible crear aplicaciones mucho más agradables para el usuario, haciendo uso de los lenguajes JavaScript o VBScript en una página web.

Un componente ActiveX muy interesante de Microsoft es el "Communications Control". Este componente se encarga de controlar un puerto serie. ¡Como Internet Explorer puede trabajar con componentes ActiveX, es posible controlar los puertos serie desde una página web!

Para utilizar estos componentes en una página web se debe incluir la siguiente declaración:

```
<OBJECT
classid=clsid:648A5600-
2C6E-101B-82B6-
000000000014 id=MSComm1>
</OBJECT>
```

Esta declaración incluye componentes ActiveX en la página web. Obviamente, no seremos capaces de ver este componente en nuestra página web ya que no es un componente gráfico. Las propiedades de este componente se configuran a través de un comando PARAM o directamente con las sentencias (script).

Las configuraciones realizadas con los comandos PARAM pasarán a ser activas cuando el visualizador coloque el componente en la página web. Sin embargo, estos comandos PARAM no son obligatorios. Cuando las propiedades se configuran por medio de los parámetros PARAM, se aplican los valores por defecto. El único problema es que no siempre está claro cuáles son las propiedades por defecto.

Por el contrario, si utilizamos JavaScript en nuestra programación de los puertos, podemos leer y/o cambiar estas propiedades a partir del contenido de una sentencia (script). Los programadores con alguna experiencia en lenguajes de programación orientados a objetos apreciarán estas propiedades. En la aplicación HTML del **Ejemplo 1** se han utilizado ambos métodos.

Otros entornos y lenguajes de programación

El programa Delphi descrito en este artículo podría haber estado tam-

```
chRead:=0;
if (avail>20) then avail:=20;
begin
  setLength (ReadBuffer,avail+1);
  ReadFile (hPort,PChar (ReadBuffer)^,avail,chRead,nil);
  Edit2.Text:=ReadBuffer;
end;
end;

procedure TForm1.ComboBox1KeyPress(Sender: TObject; var Key: Char);
begin
  Key:=Chr(0);
end;

procedure TForm1.RadioButtonParityClick(Sender: TObject);
begin
  if (Open=true) then Button2Click(Self); {Cierra el manejador }
  case (Sender as TRadioButton).Tag of
    1 : Parity:=NOPARITY;
    2 : Parity:=EVENPARITY;
    3 : Parity:=ODDPARITY;
  else
    end;
end;

procedure TForm1.RadioButtonStopBitsClick(Sender: TObject);
begin
  if (Open=true) then Button2Click(Self); {Cierra el manejador }
  case (Sender as TRadioButton).Tag of
    1 : StopBits := ONESTOPBIT;
    2 : StopBits := ONE5STOPBITS;
    3 : StopBits := TWOSTOPBITS;
  end;
end;

procedure TForm1.Button4Click(Sender: TObject);
var Written : DWord;
begin
  WriteFile(hPort, PChar (Edit1.Text)^, Length (Edit1.Text), Written, nil);
  if (WrittenLength(Edit1.Text)) then
    MessageDlg ('Error writing : '+#13+#10+SysErrorMessage(GetLastError),
      mtError,[mbOk],0);
end;

procedure TForm1.Edit2KeyPress(Sender: TObject; var Key: Char);
begin
  Key:=#0;
end;

end.
```

bién escrito en C++ Builder, Visual Basic, etc. Cuando se escribe un programa en estos lenguajes y se intenta utilizar el puerto serie, se pueden usar las mismas funciones que las que hemos definido para Delphi.

Hay que señalar que en el lenguaje de programación Delphi las APIs de Windows pueden usarse directamente, incluso si no están descritas en las páginas de ayuda. De este modo, incluso cuando nuestro entorno de programación (sistema operativo Windows) no tenga una descripción de estas rutinas en sus páginas de ayuda, es una suerte

que las APIs de Windows puedan emplearse directamente, con la única condición de que el nombre de las rutinas sea conocido. También es posible utilizar componentes ActiveX, algo que ya se ha usado en el ejemplo de la página HTML.

(020001-1)

Enlaces Útiles:

www.codeguru.com
www.programmersheaven.com
www.microsoft.com

Literatura Útil:

'Advanced Windows',
 por Jeffrey Richter, ISBN 1572315842.

Receptor de Infrarrojos Multi-estándar

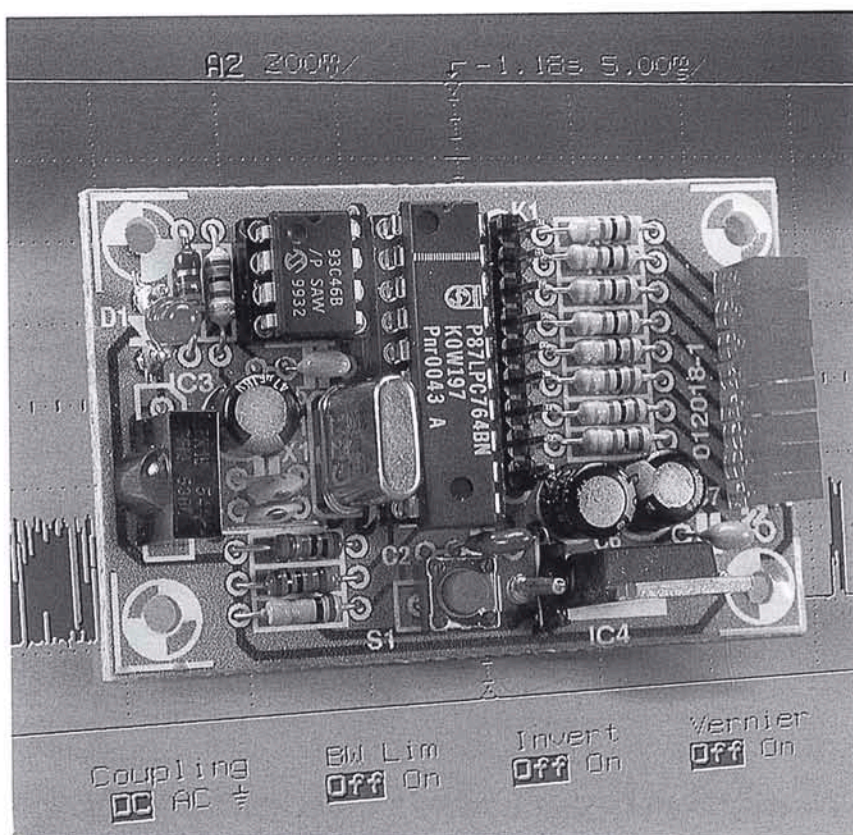
compatible con casi todos los transmisores de control remoto...

Este circuito complementa al de control remoto con el que prácticamente todos los equipos electrónicos de gran consumo están equipados. Este receptor puede funcionar con una amplia gama de transmisores de infrarrojos.

De acuerdo con la información aparecida en los controles remotos publicados anteriormente en nuestra revista Elektor Electronics, lo único que nos quedaba era construir un receptor de infrarrojos genuino y universal multi-estándar. Usando este montaje se puede construir un sistema controlado remotamente que utilice prácticamente cualquier transmisor disponible en el mercado ya montado, de modo que los controles remotos, que de otra manera podrían quedar anulados, nos proporcionen otra calidad de vida. Así, aquellos botones del teclado que no se utilizan frecuentemente (como los de la programación automática en un control remoto de televisión), pueden tener nuevas funciones y usarse, por ejemplo, para controlar la luz de la habitación. Por desgracia, con este receptor no pueden utilizarse los controles remotos que transmiten utilizando el modo "flash", debido a la manera en que dicho receptor de infrarrojos trabaja internamente.

El diagrama de bloques de la **Figura 1** nos muestra el receptor de nuestro montaje. Cualquier tecla del control remoto puede usarse para el circuito: si presionamos el botón la salida seleccionada se activará o se desactivará. Si se monta sobre un terminal de extensión con múltiples líneas se pueden controlar hasta un total de ocho canales independientemente.

Los aficionados al modelismo de trenes seguramente apreciarán esta posibilidad para controlar fácilmente las luces u otras funciones especiales que utilicen luz infrarroja, evitando de este modo el pesado sistema de cableado. Puede incluso que ya dispongamos del transmisor adecuado. Si es



necesario pueden conectarse varios receptores en paralelo para llegar a decodificar todas las funciones disponibles desde un único transmisor. El número de combinaciones de código es prácticamente ilimitado, permitiendo un amplio rango de aplicaciones de control remoto.

El circuito que se muestra en la **Figura 2** es muy sencillo y no debe presentar dificultades especiales en su montaje. Se recomienda montar los circuitos integrados IC1 e IC2 sobre zócalos.

Características técnicas:

- Tensión de alimentación: + 5 V.
- Capaz de aprender los siguientes códigos de infrarrojos: "Japonés", NEC, RC5, RECS80, SIRCS, Denon y Motorola, así como "far east" ("lejano este", utilizado por Daewoo, por ejemplo).
- 8 Salidas libres programables.
- Todos los datos están almacenados en una memoria EEPROM.
- Optimizado para una frecuencia portadora de 36 KHz.
- Confirmación de programación visual.
- Indicación visual de los estados de salida.

Recepción de infrarrojos

El circuito integrado receptor IC3 es un dispositivo con un alto grado de integración que puede descodificar señales infrarrojas desde un conjunto de frecuencias portadoras. El circuito incluye un receptor de infrarrojos de alta sensibilidad que funciona utilizando una frecuencia portadora de 36 kHz. También están incluidos un fotodiodo con filtro para luz diurna, unas etapas amplificadoras, un filtro y un demodulador, de manera que no se puede decir que haya componentes externos que funcionen mal. La resistencia R1 y el condensador C1 forman un filtro paso/bajo que elimina las interferencias de la línea de alimentación. En la lista de materiales se han proporcionado diferentes alternativas para este circuito integrado. Debemos tener especial cuidado en la distribución de terminales del modelo elegido.

El diodo LED D1 indica cuándo está el circuito en funcionamiento, parpadeando rítmicamente ante la recepción de datos.

El microcontrolador

El microcontrolador utilizado en este circuito ya se ha usado en distintos proyectos publicados en Elektor Electronics, tales como el Analizador de Códigos de Infrarrojos, publicado en el número de Noviembre de 2001. Dicho microcontrolador dispone de las siguientes características:

- 4 kbytes de memoria ROM.
- 128 bytes de memoria RAM.
- 32 bytes de memoria EEPROM para código de programa del cliente.
- Tensión de alimentación comprendida entre 2,7 y 6 V.
- Dos contadores/temporizadores de 16 bits.
- Un circuito de reset interno.
- Un oscilador RC interno seleccionable.
- Un controlador de 20 mA para todos los terminales del puerto.
- Un máximo de 18 terminales de E/S, si el reset interno y el oscilador RC se han seleccionado.
- Dos comparadores analógicos.
- Un interfaz I²C.
- Una UART "full-duplex".

- Opción de programación en el propio circuito por medio de una interfaz serie.

Utilizando una frecuencia de reloj de 16 MHz y el divisor interno (como una relación de división de 6), tenemos un ciclo de reloj de 1 ms. Esto nos permite conseguir mayor precisión en las medidas de los anchos de los pulsos, algo que es esencial cuando tenemos que decidir entre distintos códigos. El ancho de pulso se mide utilizando uno de los dos temporizadores internos y comparándolo contra los valores umbrales preseleccionados.

La memoria EEPROM

La memoria serie EEPROM, incluida en el circuito integrado IC2, tiene una capacidad de almacenamiento de 1.024 bits, organizada en una matriz de 64 por 16. Todas las operaciones de escritura y de lectura se realizan sobre la interfaz compatible "Microwire", en bloques de 16 bits. Una vez escrito el dato se puede almacenar durante más de 40 años, de acuerdo con las especificaciones del fabricante.

Etapas de conmutación

El microcontrolador dispone de controladores capaces de proporcionar hasta un total de 20 mA por terminal de salida, cuando están conectados a masa. Sin embargo, cuando los terminales están seleccionados a nivel alto sólo pueden proporcionar una corriente de 1 mA, por lo que se requiere un transistor para conmutar una carga. Para conseguir esto se proporcionan dos soluciones en el esquema eléctrico del circuito:

Opción 1:

El terminal del puerto controla la etapa de salida a través de un transistor MOSFET BUZ11. Como este transistor dispone de una $R_{DS(on)}$ de 0,04 Ω , puede manejar cargas de hasta 5 A continuamente. El diodo protege al transistor contra picos de tensión si, por ejemplo, ésta utilizándose una carga inductiva. El diodo puede suministrarse formando parte de una carga resistiva. Esta opción sólo se puede utilizar si tienen que conmutarse corrientes directas.

Opción 2:

En esta variación del montaje se ha utilizado un relé de 12 V con una resistencia de bobina de 400 Ω (o lo que es lo mismo, con una corriente de bobina de 30 mA), que se conmuta mediante un transistor BC 548. La carga deseada puede conmutarse utilizando un pequeño relé de potencia. En este montaje es esencial montar el diodo antirrebote.

Fuente de alimentación

El regulador de tensión IC4 produce la tensión de alimentación de + 5 V, que también necesita el circuito receptor IC3. La tensión de entrada del regulador debe ser de, al menos, 9 V, ya que el

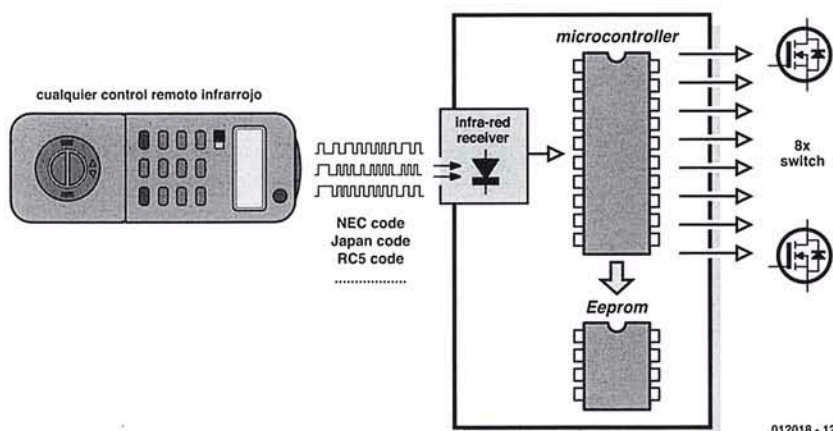


Figura 1. Diagrama de bloques del receptor multi-estándar.

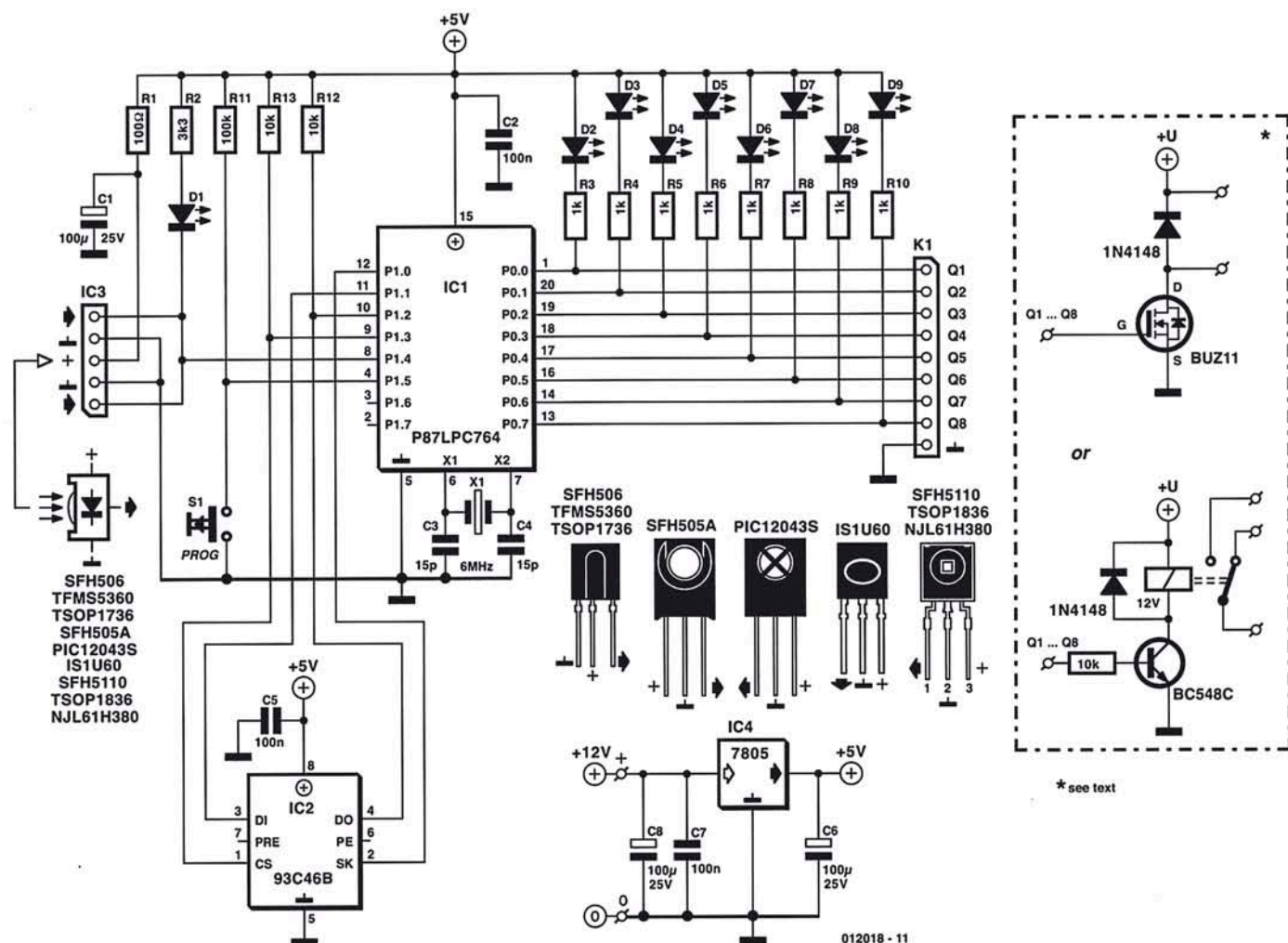


Figura 2. El circuito está formado tan sólo por un módulo receptor de infrarrojos, un microcontrolador y una EEPROM.

regulador tiene una caída interna de unos 3 V. En aplicaciones donde ya se dispone de una tensión de alimentación de + 5 V, el circuito integrado IC4 puede evitarse. Debemos observar que se requiere un radiador de calor cuando se pretende que el regulador proporcione una corriente de 1 A, aunque este radiador no será necesario para corrientes de 100 mA o inferiores. Si en nuestro montaje estamos utilizando relés de 12 V, es recomendable usar una entrada de tensión de 12 V.

El programa

El programa está basado en el Analizador de Códigos de Infrarrojos, ampliado de manera que los códigos recibidos sean almacenados y gestionen los conmutadores de salida. Nuestro lector de códigos hace algo más que un simple muestreo de la señal y almacenamiento de las muestras: por ejemplo, compara las longitudes de los

pulsos individuales con ciertos valores establecidos como de referencia. Este funcionamiento utiliza menos cantidad de memoria, pero tiene la desventaja de que sólo detecta formatos que previamente hayan sido reconocidos. Los controles remotos que "aprenden" funcionan generalmente por el simple hecho de almacenar las muestras de la señal, lo que les hace requerir una gran cantidad de memoria estática para almacenar tal cantidad de datos.

El programa muestrea continuamente la señal (más frecuentemente que lo que se ve en la **Figura 3**), cambiando el estado lógico del terminal de entrada P1.4. Esto se produce a una velocidad que viene determinada por el ciclo de tiempo de programa. Durante el proceso de almacenamiento, un módulo cuenta el número de muestras durante las que la entrada permanece en cada estado lógico. El conteo es una medida digital de la longitud de pulso. Pueden ignorarse pequeñas interrupciones o picos en la señal, con la con-

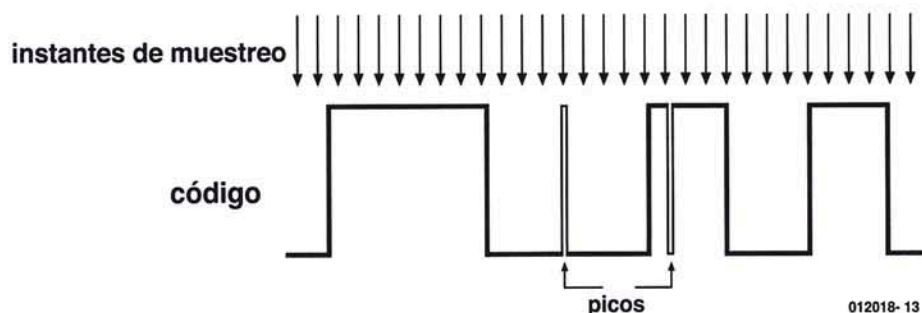


Figura 3. Sistema de muestreo por "almacenamiento".

LISTA DE MATERIALES

Resistencias

R1 = 100 Ω
R2 = 3,3 K Ω
R3 - R10 = 1 K Ω
R11 = 100 K Ω
R12, R13 = 10 K Ω

Condensadores

C1, C2, C8 = 100 μ F, electrolítico de 10 V, radial
C2, C5, C7 = 100 nF, cerámico
C3, C4 = 15 pF

Semiconductores

D1 = diodo LED rojo de alta eficiencia
D2 - D9 = diodo LED rojo con encapsulado rectangular
IC1 = P87LPC764BN (Philips), programado; bajo código de pedido N° 012018-41.

IC2 = 93C46 (Microchip 93C46B/P)
IC3 = TSOP 1736 (alternativas : SFH 5110-36, ISIU60, TMFS 5360, PIC 26043SM, TSOP 1836)
IC4 = 7805

Varios

X1 = Cristal de cuarzo de 6 MHz
K1 = Conector "pinheader" SIL de 9 terminales
S1 = Pulsador de un circuito
Z1 = Zócalo de 20 terminales para circuito integrado
Z2 = Zócalo de 8 terminales para circuito integrado
Disco, contiene los ficheros del proyecto. Código de pedido (a través del Servicio de Lectores), N° 012018-11. Puede bajarse gratuitamente de la página web www.elektor-electronics.co.uk

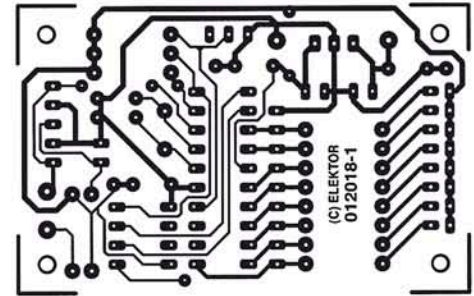
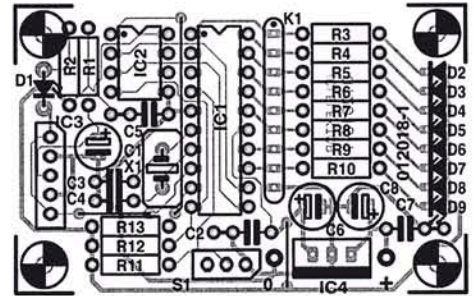


Figura 4. Diagramas de pistas y de implantación de componentes de la placa de circuito impreso.

dición de aceptar un nivel lógico sólo cuando dicho nivel se ha mantenido estable durante un cierto tiempo, que suele coincidir con un número mínimo de muestras. Si este umbral mínimo no se alcanza, se asume que se trata de una interferencia y el valor actual de conteo se desestima, iniciando de nuevo el conteo a partir del valor previamente almacenado.

Al igual que sucede con todas las cosas, este procedimiento tiene sus ventajas y sus desventajas. Por un lado es fácil de programar, es elástico y resistente (las interferencias entre muestras se ignoran cuando se utiliza un lazo de retardo), y es exportable a otros tipos de micro-

controladores, ya que no está utilizando recursos del propio circuito. Por otro lado, este sistema proporciona una gran cantidad de trabajo al procesador: el programa está procesando exclusivamente las muestras y no puede atender otras tareas cuando se está recibiendo un mensaje. Además, las interrupciones que se producen durante la medida pueden llevarnos a unas lecturas bastante imprecisas, dependiendo del tiempo que las usemos.

Los códigos recibidos se almacenan en formato hexadecimal en la memoria RAM interna del microcontrolador y en la memoria EEPROM. No se ha hecho distinción entre las direcciones y los bits de

comandos, por lo que el mensaje completo recibido se compara con el que está almacenado en memoria. Los bits de conmutación, como los utilizados en el código RC5, los cuales cambian su estado cada vez que se pulsa un botón, son ignorados por el programa, ya que si no fuese así nos proporcionarían dos valores hexadecimales diferentes para el código

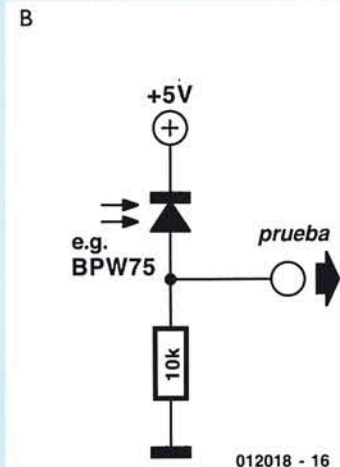
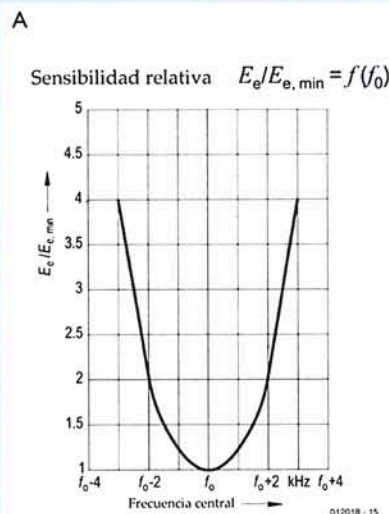
En funcionamiento

El montaje del circuito receptor en la placa de circuito impreso, tal y como se muestra en la **Figura**

Frecuencia Portadora

En la **Figura A** puede verse claramente que, incluso con un transmisor que funcione con una portadora de 34 ó 38 KHz, ya tenemos un factor de pérdida de sensibilidad de $\times 2$. Si esto supone un problema, puede ser de gran ayuda utilizar un receptor de infrarrojos diferente. El circuito integrado está disponible para, prácticamente, la totalidad de las frecuencias centrales, en pasos de 2 KHz, desde 30 hasta 40 KHz.

Un método sencillo para determinar la frecuencia portadora del transmisor es utilizar un osciloscopio. Un fotodiodo ordinario de infrarrojos, como el BPW 75, se puede conectar a una tensión de + 5 V a través de una resistencia, tal y como se muestra en la **Figura B**. Si un dispositivo control remoto nos mantiene activos cerca del fotodiodo, la señal que se presenta en el osciloscopio puede ser analizada y obtener así la frecuencia portadora medida.



4, no debe presentar mayores problemas. Además, también se muestran las distribuciones de terminales de los distintos receptores de infrarrojos adecuados para este montaje.

Antes de instalar la unidad debemos programarla, de manera que el microcontrolador pueda aprender qué comandos va a utilizar para controlar, así como las distintas salidas que debe controlar. La programación se inicia presionando brevemente sobre el botón S1. El diodo LED D2 comienza a parpadear inmediatamente. A partir de este momento, cada comando que se recibe utilizando uno de los códigos interpretables por el microcontrolador es almacenado y asignado a un terminal del puerto. Este terminal del puerto se programa en modo entrada, iniciándose el proceso de programación automático para el siguiente terminal: el diodo LED D3 comienza a parpadear y... Este terminal del puerto puede ser asignado a un comando exactamente de la misma manera. Por lo tanto, es irrelevante que para cada salida se utilice un transmisor de control remoto diferente con un código distinto.

El proceso continúa hasta que se han programado las ocho salidas, después de lo cual los diodos LEDs dejan de parpadear. Si, unos días más tarde, queremos cambiar los comandos asociados a uno de los canales, los terminales que no son requeridos para la programación pueden ser "saltados" pulsando brevemente el botón.

Debemos señalar que mientras el diodo LED está parpadeando, cualquier elemento conectado al equipo estará conectándose y desconectándose. Todas las configuraciones son almacenadas de forma permanente en la memoria EEPROM, IC2, de manera que cuando el microcontrolador es reseteado, la configuración programada más recientemente será la que esté activa.

Cuando se recibe un código previamente almacenado, la salida correspondiente cambia de estado. Como muchos controles remotos transmisores envían el mismo mensaje repetidamente cuando un botón se mantiene pulsado, se ha creado un dispositivo monoestable por programa que asegura un funcionamiento limpio. Sólo después de un retardo de aproximadamente un segundo después del último mensaje válido, una salida puede cambiar su estado de nuevo.

Como el programa sólo puede interpretar formatos de códigos que conoce, el usuario puede utilizar el hecho de que los diodos LEDs que parpadean se desplazan cuando se está programando cada salida, como confirmación de que el código se está leyendo correctamente. Si

el código no es leído, el diodo LED que parpadea no avanza.

Como hemos utilizado un circuito integrado receptor de infrarrojos, IC3, optimizado para una frecuencia portadora de 36 kHz, podemos obtener el rango más amplio utilizando transmisores que funcionen en esta frecuencia. Si no se ha conseguido el rango adecuado, lo más probable es que sea debido a que el transmisor utiliza una frecuencia portadora diferente.

(012018-1)

Direcciones de Internet:

EEPROM:

www.fairchildsemi.com/pdf/FM/FM93C46.html

Microcontroller:

www.semiconductors.philips.com/pip/p871pc764bd

Circuito Integrado Receptor de Infrarrojos:

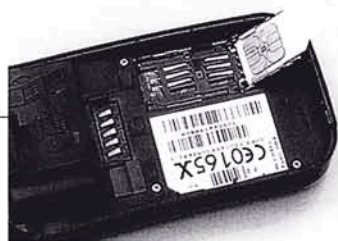
www.infineon.com/cmc_upload/0/000/008/562/sfh5110.pdf

www.vishay.com/products/optoelectronics/IRMail.html

PRÓXIMO NÚMERO

TARJETA BASIC PARA TELÉFONO GSM

En este artículo nuestro experto en tarjetas chip describe como sustituir la tarjeta SIM en nuestro teléfono GSM por una Tarjeta Basic y así forzar al móvil a darnos muchos de sus secretos.



REGULADOR DMX

Para el control de luces usamos proyectos DMX ya publicados, siendo necesario para un regulador usarlo en combinación con una señal DMX-512 o, como mínimo, la tensión de control estándar de 0 a 10 V. El proyecto que describiremos permite cargas de salida de hasta 1.000 W por canal.



ESPÍA DE DATOS PARA SISTEMA MÄRKLIN DE MODELISMO FERROVIARIO

Locomotora, medida de señales de control en las vías que serán normalmente en formato Motorola y se pueden suministrar por un control DELTA, un controlador digital o un elevador EEDTS Pro. El montaje permite monitorizar en el PC todos los datos de este formato.

TAMBIÉN...

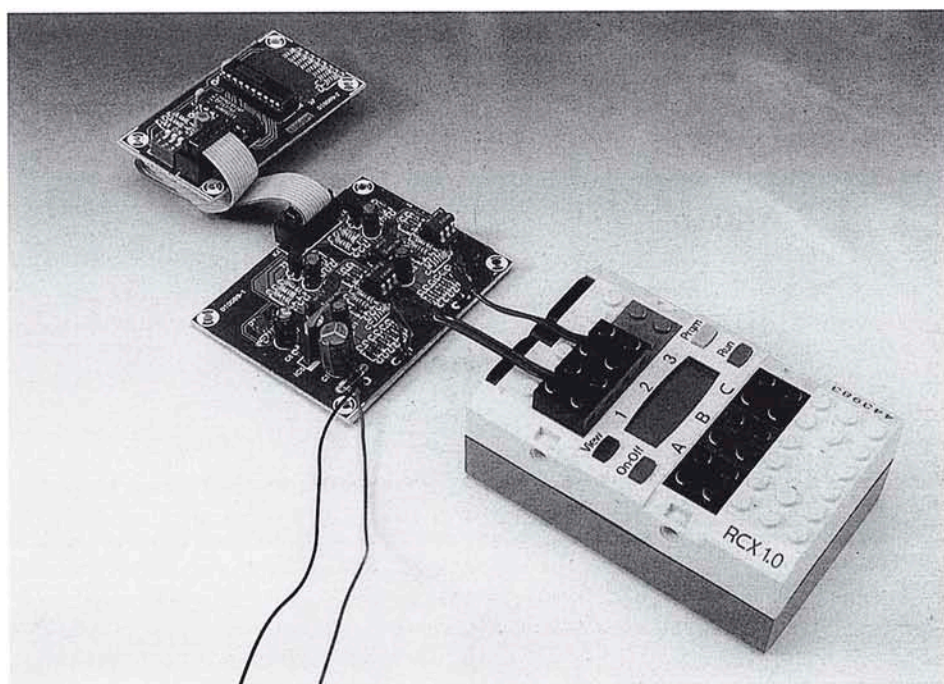
Añadir un silenciador a un receptor de banda VHF, medida a distancia por infrarrojos, chip sintonizador en coches, receptor Atmel TRX01 para banda-ISM, juego de dados con microcontrolador AVR.

Interface I²C para Bloque Lego RCX

hace posible una ampliación sin precedentes

Diseñado por W. Huiskamp

Es conocido el hecho de que los módulos RCX de Lego son adecuados para experimentar en robótica, pero una vez que el diseño crece y deja de ser un simple experimento el número de entradas y salidas que proporciona este módulo de control es insuficiente. Por ello, este mes presentamos en Elektor la interface I²C para este modelo de Lego. Un nuevo mundo se nos abre: en principio se podrán conectar al bus al menos 128 dispositivos I²C.



El RCX, es el componente inteligente del Sistema Robótico Inventado por Lego, el cual forma parte del programa Lego Mindstorms y

es eminentemente adecuado para servir como base de varias aplicaciones automáticas. Aparte de la pro-

gramación, que aún requiere, es muy fácil de ensamblar, por ejemplo, un pequeño robot junto con las partes normales de Lego. Sin embargo, el módulo RCX sólo es útil para controlar o medir cosas que necesiten tres entradas y tres salidas, pero esto va a cambiar, ¡estamos de suerte!

No es la primera vez que hemos puesto esta pequeña entrada en el diseño. En una serie de cinco artículos, que comenzó en el mes de Abril del año 2000, presentamos el sistema Robotics de Lego. Incluso entonces conectamos múltiples sensores a una entrada. Los circuitos de verano del mismo año también contenían un truco que permitía conectar múltiples interruptores a una sola entrada.

Naturalmente, el personal de diseño y editores de Elektor vieron que la capacidad de I/O del módulo RCX era limitada. Varios aficionados inventaron circuitos que, utilizando técnicas de multiplexado, permiten conectar más periféricos. Algunos de ellos son realmente ingeniosos y en

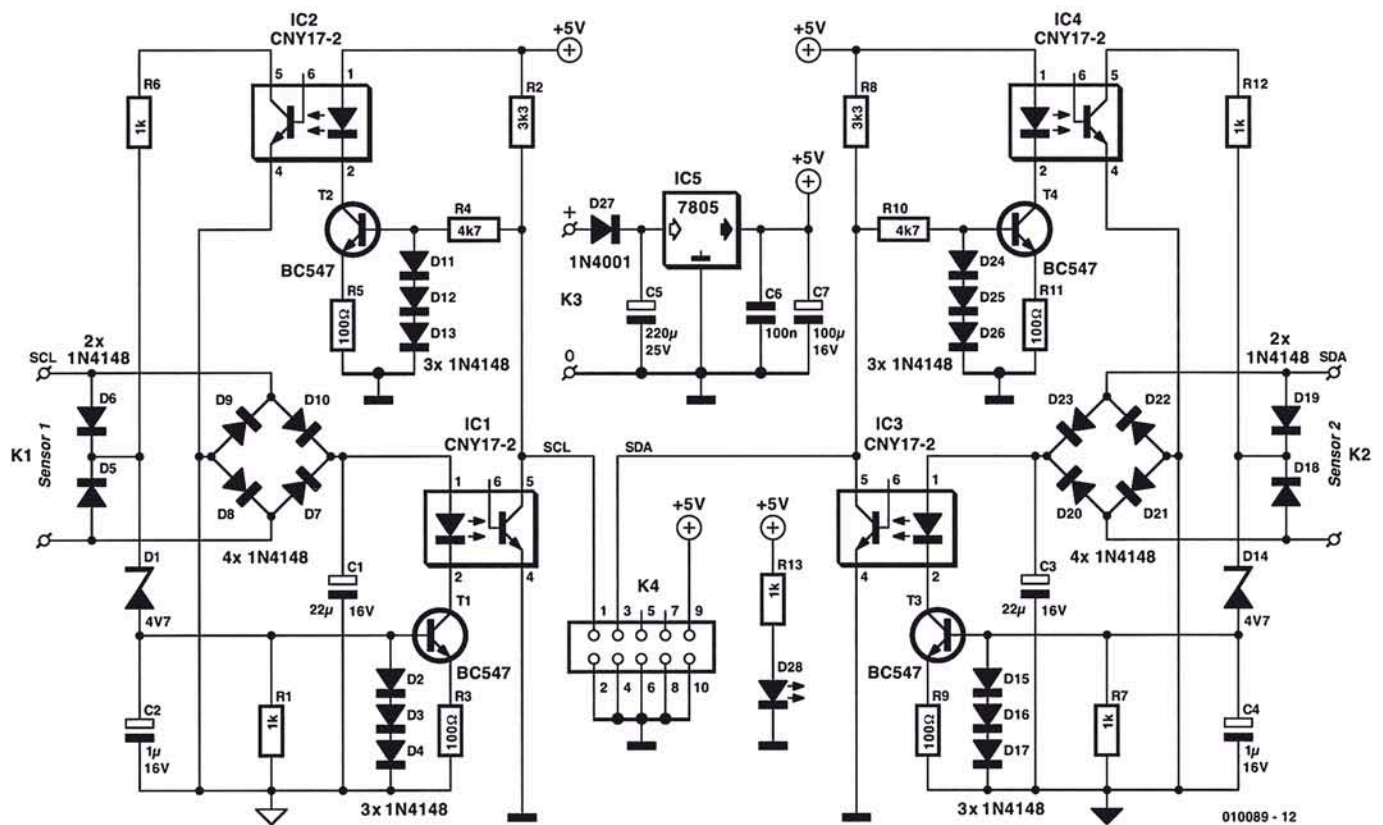


Figura 1. La Interface I2C para el Lego RCX consta de dos partes idénticas para la señal SCL y SDA.

muchas ocasiones es suficiente. El circuito que presentamos aquí, sin embargo, es probablemente una fracción aún más pequeña. Con ello, el módulo RCX se puede comunicar (teóricamente) hasta con 128 dispositivos en el bus I2C.

Operación de la interface

El bus Inter-IC (I2C) es un bus de dos cables con una línea de datos bidireccional (SDA) y una señal de reloj (SCL). El bus está sujeto a un número de reglas. La transferencia de datos tiene lugar entre el llamado master y uno o más esclavos. El master es un dispositivo que inicializa las comunicaciones y también genera la señal de reloj. Además, los dispositivos no sólo permiten conexiones activas a nivel alto al bus. Un nivel alto se obtiene haciendo una conexión de alta impedancia, se conecta una resistencia de valor 3K3W a la línea de alimentación, proporcionando un nivel alto en el bus. Está permitida la colocación de un nivel bajo para producir un cero. En este caso, la resistencia de 3K3 se debe conectar a masa.

Los módulos RCX tienen que poder generar tanto unos como ceros en el bus, y también poder leer unos y ceros. Esto se puede conseguir con un simple sensor de entrada y un truco bien conocido por todos. Bajo el software de control podemos configurar un sensor de entrada como activo o como pasivo. En modo pasivo, el valor medido (= la tensión de entrada) se determina por medio de una resistencia interna y la resistencia colocada en el sensor. La tensión de este divisor de tensión puede variar entre 0 y 5 V. Cuando el sensor de entrada se configura en modo activo, la tensión de la batería (7 a 9 V) se aplica directamente a las conexiones del sensor entre medidas. Esta tensión es periódica, aproximadamente cada 3 ms, se desconecta y el valor de entrada se determina de forma normal. Una medida tarda, aproximadamente, 0,1 ms en salir. De esta forma, conmutando entre sensores activos y pasivos bajo control software, es posible generar una señal desde el RCX para el bus I2C. Además, es posible leer señales originadas en el bus a través del mismo sensor de entrada.

El circuito

La interface real para las señales SDA y SCL es idéntica. Como consecuencia, en la **Figura 1** se puede reconocer el circuito dos veces. Nosotros sólo discutiremos la operación para una señal. El circuito se divide en dos partes, cada una con optoacoplador. Esto evita daños a cada módulo RCX o los aísla cuando la tensión es demasiado elevada (por ejemplo, cuando sólo está conectada una de las tensiones de alimentación). Esto se debe a que la sección del circuito del lado I2C está alimentada de forma separada. De esto se encarga IC5. El diodo D27 protege contra cambios de polaridad si la tensión de alimentación se conecta de forma incorrecta. El LED D28 es el indicador de tensión de alimentación.

En el RCX parte del circuito se alimenta a través del diodo D7 a D10. Los diodos están conectados como un puente rectificador, por lo que no necesitan considerar la polaridad cuando los conectamos al módulo RCX. El condensador C1 almacena la tensión de alimentación durante esos periodos, cuando tiene lugar la medida. Esto es necesario porque en el modo activo la batería se conecta periódicamente.

La tensión en la conexión del sensor se aplica al circuito a través de los diodos D5 y D6 (juntos aseguran la polaridad). La combinación

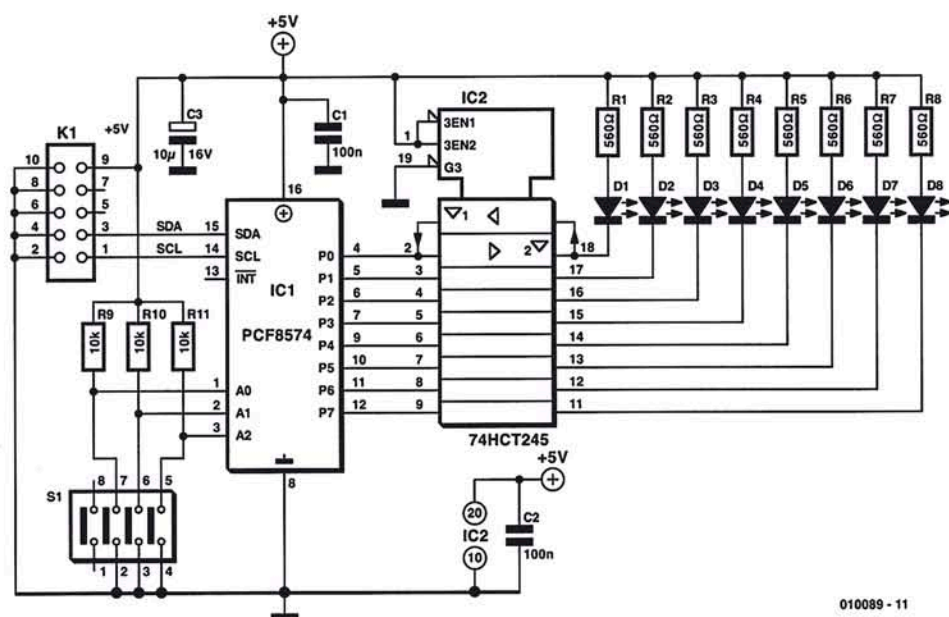


Figura 2. Circuito de prueba con el expansor de puerto y el driver para ocho LEDs.

de los diodos zéner D1 y D2, D3 y D4, hace posible establecer la distinción entre el modo activo y pasivo. Los zéner no conducen en modo pasivo. Una vez que la tensión de batería se conecta, el condensador C2 se cargará y causará una caída de tensión de hasta 1,8 V en D2, D3 y D4. Cuando estén presentes los 1,8 V, el transistor T1 conducirá y circulará una corriente a través del LED del optoacoplador. La resistencia R1 nos asegura que el condensador se descargue rápidamente, lo cual sucede cuando la entrada se conmuta a modo pasivo. Este método, con tensiones de alimentación separadas y circuito conmutado, proporciona un nivel preciso de '1' ó '0' en el bus.

El circuito en la otra parte de IC1 es muy sencillo y realmente consta sólo de R2. Esta resistencia proporciona una señal a nivel alto cuando la línea está en alta impedancia. Cuando el LED está activo, parte del transistor del optoacoplador conduce y el nivel de salida en el bus es bajo. Por lo tanto, el circuito actúa como un inversor. Esto es algo que tendremos que tener en cuenta en el software.

El propósito de R4, D11, D12, D13, T2 y R5, junto con IC2 y R6, es poder leer la señal procedente del bus en el módulo RCX. El nivel lógico en la línea SCL o SDA se aplica a través de la resistencia R4 a la base de T2 y la cadena de diodos D11, D12 y D13. Los diodos limitan la tensión en la base de T2 a 1,8 V. Cuando el nivel del bus es alto, T2 conducirá y el LED en el optoacoplador IC2 se encenderá. El LED obtiene su tensión de la tensión de alimentación independiente para la sección I2C.

En la parte I2C del módulo RCX, el transistor en el optoacoplador asegura que, a través

de R6, la tensión de entrada en la entrada del sensor será un nivel bajo cuando el LED esté iluminado. D5 y D6 aseguran que, una vez más, la polaridad no es importante cuando hacemos la conexión.

Nuestro primer dispositivo

En lo que se refiere a la comprobación de la interface, por supuesto, es necesario conectar un dispositivo al bus. Para este propósito añadimos el circuito de la **Figura 2**. En el centro del circuito tenemos IC1, el llamado puerto expansor I2C. Este dispositivo está programado a través del bus con un valor de 8 bits que se presenta en binario en las salidas. Si enviamos, por ejemplo, el número 1, entonces P0 pasará a nivel alto; el valor 17 hará que P4 y P0 pasen a nivel alto. Las salidas de IC1 se conectan a través de IC2, un integrado con 8 drivers, a los ocho LEDs. En una aplicación real éstas pueden tener otros indicadores arbitrarios o actuadores, tales como zumbadores o (a través de etapas buffer) relés. También debemos de señalar que la expansión de puerto puede leer un valor de 8 bits a través de P0 a P7. Para este propósito, el master primero tiene que configurar todas las salidas a nivel alto, entonces las conexiones se pueden poner a nivel bajo. Un comando de lectura proporcionará el valor de cada pin. Se

pueden conectar al bus hasta ocho diferentes puertos de expansión. Esto se debe a que la dirección para este circuito integrado tiene la forma 0100xxx0 donde xxx se puede configurar con el microrruptor DIP S1.

Construcción

La construcción de cada circuito no debería causar ningún tipo de dificultad, particularmente si hacemos nuestro circuito impreso tal y como se puede ver en las **Figuras 3 y 4**. Sin embargo, vamos a ver algunos asuntos a considerar durante el ensamblaje.

Tenga en cuenta la polaridad de los diodos, condensadores, transistores y circuitos integrados. Podemos utilizar zócalos para los optoacopladores. También podemos ver que en la lista de componentes hemos seleccionado un LED de bajo consumo para D28, en la PCB de la interface. Esto es importante, porque el LED está alimentado directamente desde el módulo RCX. Todo ahorro de energía ayuda a prolongar la vida de las baterías.

Los pines de la PCB y los conectores proporcionan las conexiones con el mundo exterior. Se pueden soldar cables estándar de Lego a K1 y K2. Conectaremos una pila de 9 V a K3. El conector K4 lleva el bus I2C al otro mundo exterior. No sólo tenemos disponibles las señales SCL y SDA (pines 1 y 3 respectivamente), sino también la tensión de alimentación (pin 9) y masa (pines 2, 4, 6, 8 y 10). Puesto que la tensión de alimentación está disponible en este conector, la PCB de prueba se puede conectar sin necesidad de tener su propia alimentación, siempre que la carga sea razonable. Los ocho diodos LED del ejemplo no son un problema. La placa de prueba se coloca con el mismo conector de 10 pines y se puede conectar con un trozo de cable plano.

Software

El software, como no, es una parte esencial de la interface. Se implementa una serie de funciones que hacen posible la comunicación de datos con los dispositivos I2C a un gran nivel. Esas funciones se han escrito en NQC 2.2 (Not Quite C). La versión microprogramada del módulo RCX para comunicarse con el hardware es la 2.0, y no funcionan

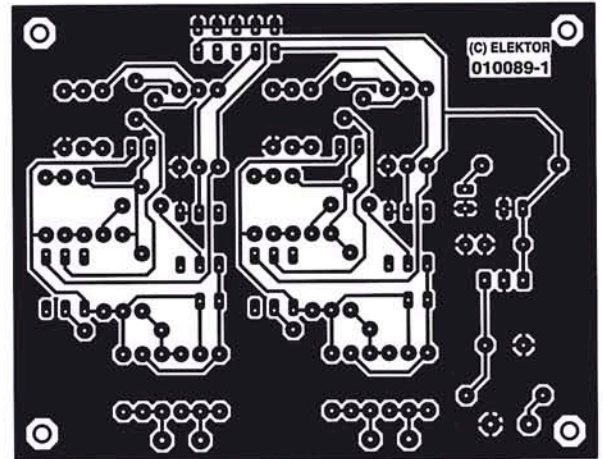
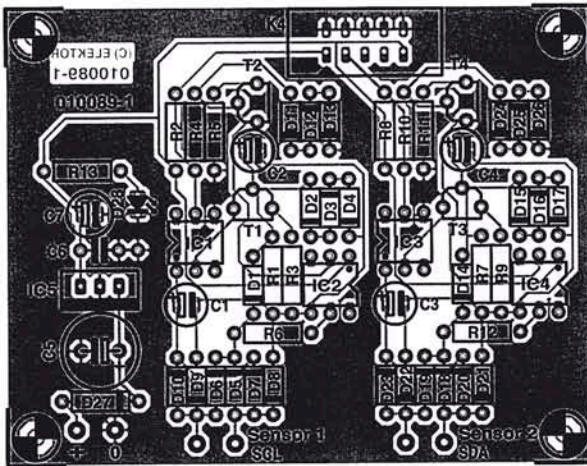


Figura 3. Placa de circuito impreso para la interface I2C.

LISTADO DE COMPONENTES (placa del interface)

Resistencias:

R1, R6, R7, R12, R13 = 1 k Ω
R2, R8 = 3k3
R3, R5, R9, R11 = 100 Ω
R4, R10 = 4k Ω 27

Condensadores:

C1, C3 = 22 μ F, 16 V, radial
C2, C4 = 1 μ F, 16 V, radial
C5 = 220 μ F, 25 V, radial
C6 = 100 nF
C7 = 1.000 μ F, 16 V, radial

Semiconductores:

D1, D14 = diodo zéner 4,7 V, 400 mW
D2-D13, D15-D26 = 1N4148
D27 = LED de baja corriente
T1-T4 = BC547
IC1-IC4 = CNY17-2
IC5 = 7805

Varios:

K1, K2, K3 = 6 espadines
K4 = conector de 10 contactos
Disco, software del proyecto:
código de pedido 010089-11 (ver
página del Servicio de Lectores).
Descarga gratuita desde
www.elektor-electronics.co.uk

LISTADO DE COMPONENTES (placa de prueba)

Resistencias:

R1-R8 = 560 W
R9, R10, R11 = 10 kW

Condensadores:

C1, C2 = 100 nF
C3 = 10 μ F, 16 V, radial

Semiconductores:

D1-D8 = LED rectangular
IC1 = PCF8574
IC2 = 74HCT245

Varios:

K1 = Conector de 10 contactos
S1 = Microrruptores DIP de 3
contactos (también se podría
colocar de 4)

las versiones anteriores. Si nuestro módulo RCX no tiene esta versión, deberíamos de hacer uso de las funciones I2C que mostramos aquí, grabando primero el módulo. Esto se hace de la siguiente forma: primero descargaremos la versión microprogramada 2.0 (el sistema software) para el módulo RCX (la dirección viene al final del artículo). Ahora lo cargamos en el módulo RCX. Se puede utilizar el programa BricxCC para esta operación, pero antes debemos configurarlo para RCX2, no RCX. El programa lo pedirá la primera vez para arrancar. Después podemos llegar a esta configuración entrando por Edit. Las preferencias están bajo el tabulador. Obsérvese que BricxCC no sólo se utiliza para descargar la nueva versión microprogramada, sino que también proporciona comunicación entre un PC y el RCX cuando enviamos programas NQC. NQC es un compilador que puede traducir programas desde código C a código para el módulo RCX.

Las funciones que dan el soporte de comunicaciones con el bus se ilustran mejor con la ayuda de dos programas de ejemplo.

Primero veremos el `i2c_test_wr.nqc`, el cual se puede encontrar con los otros ejemplos en el disco que pertenece a este proyecto (ver

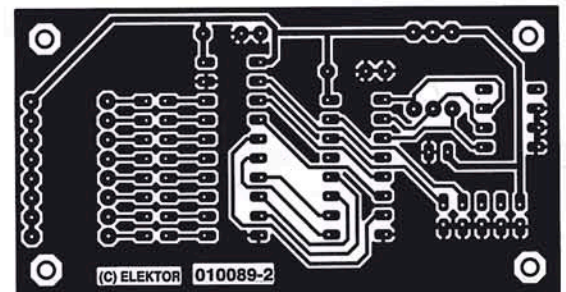
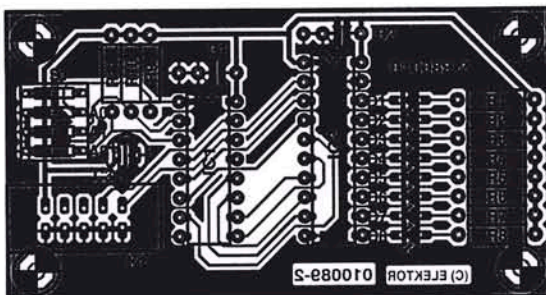


Figure 4. Placa de circuito impreso para el circuito de prueba.

páginas del Servicio de Lectores 010089-11) o visitando la página de Elektor. El programa de prueba incrementa un contador en pasos de uno, hasta alcanzar un valor de 255. Cada nuevo valor se envía a través del bus a la PCB de prueba. Encontraremos las siguientes líneas de código:

```
i2c_init();
i2c_start();
i2c_data = PCF8574_0;
i2c_send();
i2c_data = test;
i2c_send();
i2c_stop();
```

`i2c_start()` inicializa el mensaje I2C. A este le sigue la dirección del dispositivo con el que queremos hablar, es decir, el puerto de expansión. Después de esto, el dato se envía a través de la variable `i2c_data`. Y, por último la sesión de comunicación se cierra.

En el programa ejemplo, a esto le sigue un incremento del contador y la ejecución del bucle una vez más. Éste trabaja bien, pero en principio no es necesario detener la comunicación cuando se direcciona el mismo dispositivo. La inicialización, arranque, envío de la dirección y parada se pueden sacar fuera del bucle en este caso. Por ello, éste es mucho más rápido.

En el segundo programa de ejemplo `i2c_test_rd.nqc`, el puerto de expansión se lee de nuevo. Antes de que suceda esto, la comunicación se ha inicializado de forma normal:

```
i2c_init();
i2c_start();
i2c_data = PCF8574_0;
i2c_send();
```

En lo que se refiere a la función como una entrada, todas las conexiones tienen que estar a nivel alto. Para llegar a esto, se ha enviado un valor de `0xFF` (255) al dispositivo:

```
i2c_data = 0xFF;
i2c_send();
```

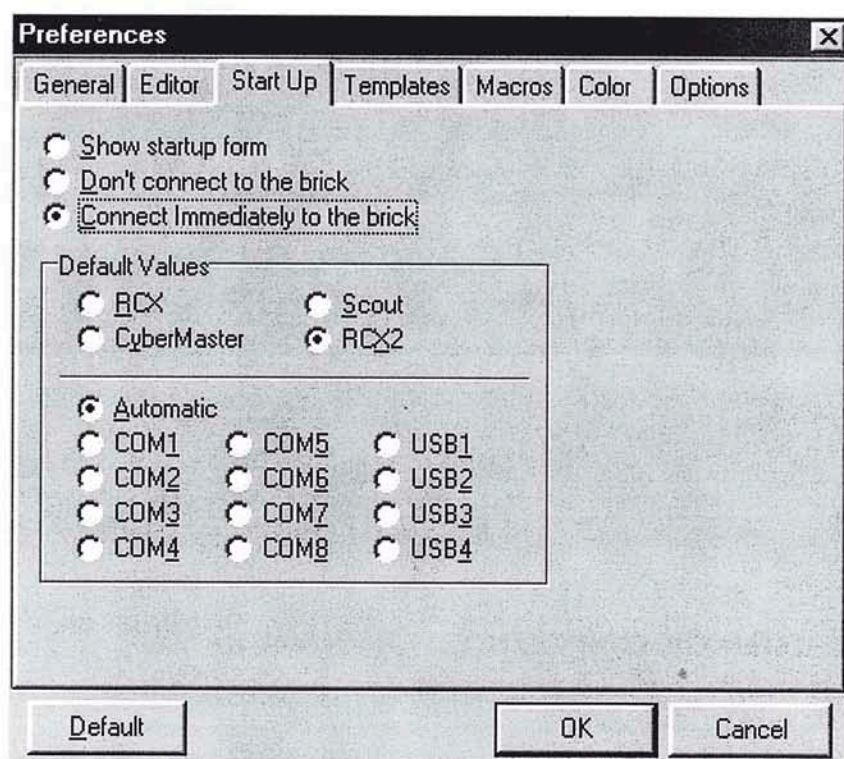


Figura 5. Después de ejecutar BricxCC es importante seleccionar RCX2.

Ahora la operación de lectura del puerto será:

```
i2c_recv();
test = i2c_data;
```

Por último, la comunicación se detiene:

```
i2c_stop();
```

El programa de ejemplo es sensiblemente diferente porque también contiene un bucle. Asimismo ambos programas se diferencian en el bit de reconocimiento `i2c_ackn`. El master puede recibir este bit desde un dispositivo esclavo, una vez enviados todos los datos del esclavo. El inverso también es cierto, se puede enviar un bit de reconocimiento al esclavo una vez que todos los datos se han leído. Esto se puede utilizar en algunas aplicaciones, pero aquí no es necesario.

Parece que con este software el módulo RCX y la interface reciben dos mensajes I2C completos por segundo. Esto no es particularmente rápido, y se tendrá que tener en cuenta para ciertas aplicaciones.

La implementación real de las funciones de usuario se lleva a cabo en

el fichero `i2c_master.nqc`. Cuando esas funciones se llaman desde dentro del programa, este fichero tiene que incluirse con la directiva del compilador `#INCLUDE`.

Hemos evitado conscientemente una discusión de las funciones de bajo nivel, si alguien está interesado puede examinar el código fuente por su cuenta. La única cosa que podríamos mencionar es que la operación de inversión de la interface ha sido compensada para este nivel. En este fichero también podemos indicar la señal del bus que representa el sensor de entrada. Por defecto es SCL para el sensor 1 y SDA para el sensor 2. También hay diferentes direcciones posibles para el PCF8574 (el puerto de expansión en el resto del circuito), las cuales se suministran como `PCF8574_0` a `PCF8574_7` (`0x40h` a `0x40E`).

Esto en definitiva no significa que el software esté limitado a este dispositivo. También pueden utilizarse otros dispositivos I2C, tales como conversores A/D o D/A (por ejemplo PCF8591), los drivers de LED y relé (por ejemplo SAA1064) o incluso un reloj de tiempo real con el módulo RCX.

Páginas web útiles

Bricx Command Center 3.3:

<http://hometown.aol.com/johnbinder/bricxcc.htm>

Not Quite C:

www.cs.uu.nl/people/markov/lego

Sitio de descarga oficial de NQC:

www.enteract.com/~dbaum/nqc/doc

RCX microprogramado con versión 2.0:

<http://mindstorms.lego.com/sdk2/>

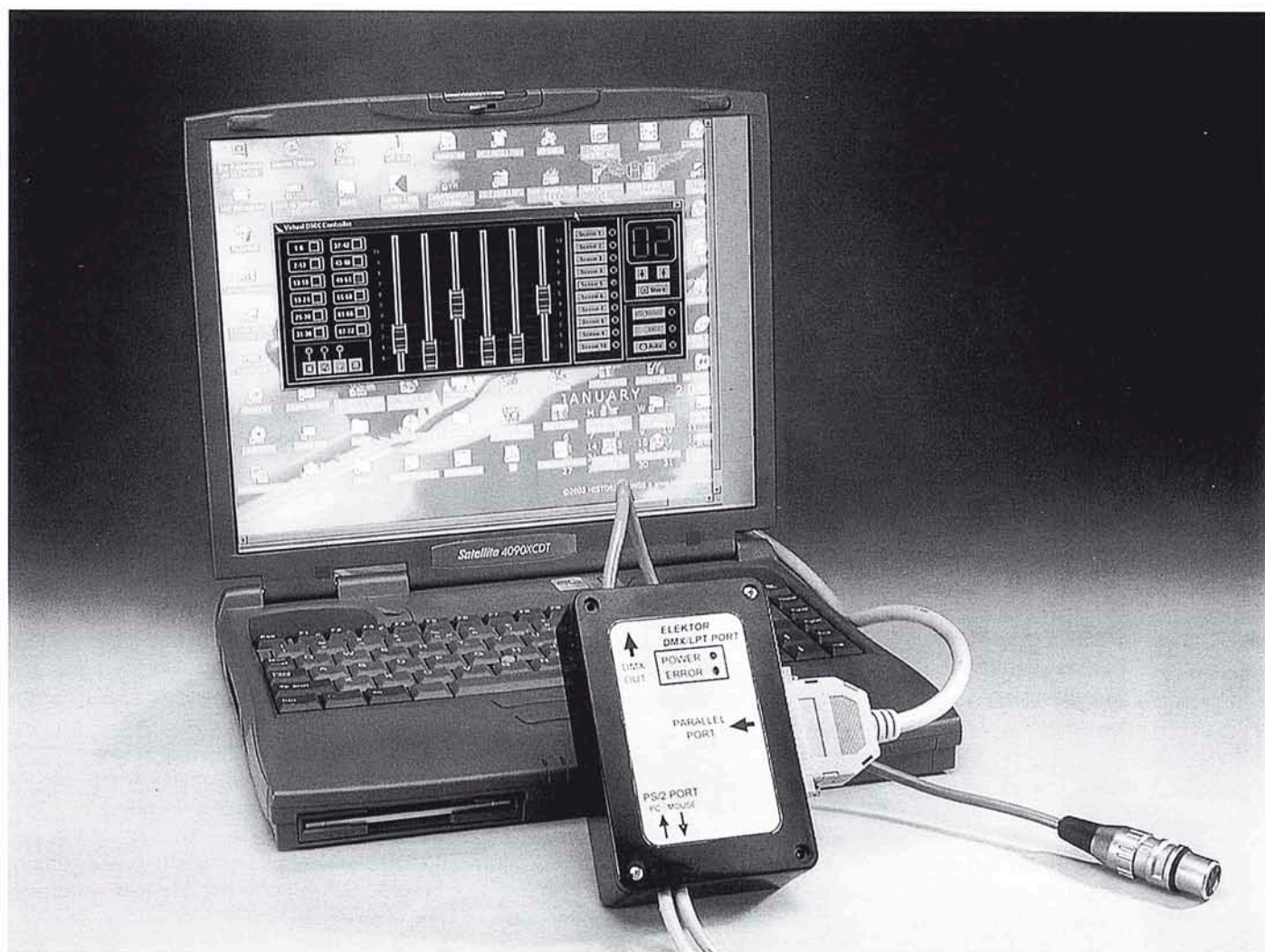
Interface LPT/DMX

480 canales DMX a través del puerto paralelo

Diseñado: B. Bouchez

bbouchez@netcourrier.com

La interface descrita en un número anterior es extremadamente flexible y adecuada para casi todas las aplicaciones, pero es un circuito complejo. Por el contrario, la interface descrita aquí brilla por su sencillez, tanto en construcción como en programación, incluso bajo Windows.



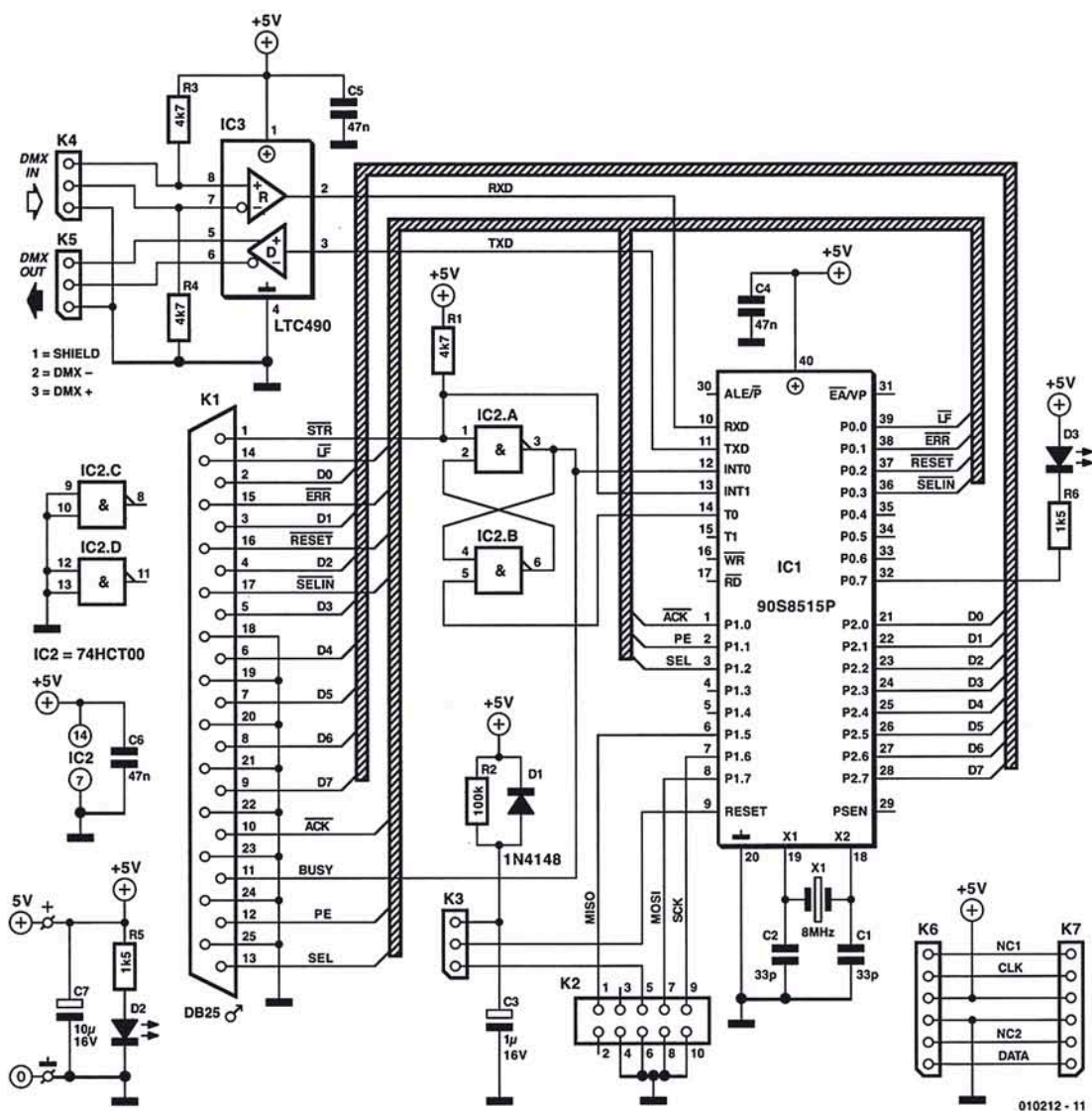


Figura 1. La electrónica de la interface LPT/DMX se concentra en el Atmel AT90S8515 IC.

El protocolo DMX ya ha sido objeto de varios artículos en Elektor. Por ello, aquí describiremos sólo los puntos esenciales para refrescar nuestra memoria. Los lectores que estén interesados en los detalles técnicos del protocolo pueden consultar el artículo 'Secretos del DMX512' en el número de Junio y varios diseños que han aparecido en Elektor ('Interface MIDI/DMX' y 'Demultiplexor DMX').

El protocolo DMX512

El protocolo fue definido en 1986 por la USITT. Esta organización americana es la responsable de la estandarización de todos los aspectos técnicos del área de comunicación, tales como intercambio de información digital entre

paneles de control y dispositivos tales como equipos reguladores y control automático de iluminación.

El protocolo DMX está basado en un enlace serie unidireccional con las especificaciones del estándar RS-485. Los comandos son recibidos por los 'esclavos' en forma de bytes en serie, cada uno de los cuales controla uno de los canales DMX. El enlace trabaja a una velocidad de 250 Kb con un formato de '8 bits', sin paridad y 2 bits de stop. En lo que se refiere al comienzo de cada trama (el cual puede ser tan grande como 512 bytes más el byte de comienzo) para ser determinado, las comunicaciones son interrumpidas ('Break') durante una duración de al menos 2 bits, los cuales permiten a los receptores sincronizarse.

¿Por qué esta interface?

La señal DMX se puede generar a través de un puerto serie simple (ésta es la forma en que se desarrolló este estándar). Sin embargo, el puerto serie de un PC no es totalmente adecuado para esto por dos razones. La primera, los puertos del PC están diseñados para funcionar con el estándar RS-232 (aunque aquí un convertidor RS-232/RS-485 puede ser una solución), la segunda, la velocidad del reloj del puerto serie de un PC hace lo imposible para generar una señal de 250 Kb.

Resumiendo, si queremos utilizar DMX en combinación con un PC, es necesario usar una interface separada. La disposición que se ha tomado aquí es muy sencilla, sólo usaremos el puerto paralelo, el cual está soportado por la mayoría de los sistemas operativos (DOS, Windows 3.x/95/98/ME y LINUX). Más

aún, el PC está lejos de ser el único tipo de ordenador con un puerto paralelo. También es posible utilizar este tipo de interface con máquinas tales como las famosas Atari, Amiga y alguna otra, siempre que tengamos el software adecuado disponible.

En lo que se refiere a mantener esta interface tan universal como sea posible y no limitar su uso en los modernos PCs, sólo se utiliza protocolos Centronics en lugar de protocolos específico para PC (Nibble, Inverso, EPP y ECP).

Esta disposición también tiene una importante ventaja cuando la interface se utiliza con Windows. Normalmente con este sistema operativo es necesario desarrollar drivers para utilizar interfaces específicas, lo cual es un poco complicado. Debido a que esta interface utiliza protocolo Centronics, Windows simplemente lo considera como una impresora e instala los drivers estándar.

Esto significa que incluso bajo entorno Windows, la única programación de la interface será para comandos de control para imprimir. Esta interface se puede controlar utilizando cualquier lenguaje de programación (Delphi, Visual Basic, Visual C, C++ Builder, etc.) que soporte impresión bajo entorno Windows.

Más aún, el diseño de la interface tiene un módulo especial para Windows que nos permite controlarlo sin tener que saber cómo programar el puerto paralelo bajo Windows. También hay un programa especial para la comprobación de la interface. Por último, deberíamos notar que la interface está soportada por el Controlador Soft de programas (más adelante lo veremos).

El hardware

Tal y como podemos ver en la **Figura 1**, el esquema de la interface es bastante sencillo, porque IC1 (un μ P Atmel AT90S8515 RISC) hace prácticamente todo. Contiene todo lo que necesitamos: RAM, un puerto serie y una EEPROM para el programa. Para llegar a este circuito se han necesitado alrededor de 10 chips.

Los únicos semiconductores que necesitamos son IC2 (el cual se utiliza para implementar un flip-flop) e IC3 (un transceptor RS-485 de Linear Technology LTC490CN8). A propósito, el LTC490 se puede sustituir por su equivalente, el SN75179 (que puede encontrarse con facilidad).

En nuestros primeros prototipos la línea de Strobe del puerto paralelo se conectó directamente a una de las entradas de interrupción del microprocesador. La anchura del pulso de esta línea varía tanto que algunas veces no trabaja de forma adecuada. Esto es porque se añadió el flip-flop formado por IC2a/b. Con este cambio se registran hasta los pulsos cortos.

La asignación de pines de K1 coincide con la del conector del puerto paralelo del PC. La interface puede conectarse al Centronics

LISTADO DE COMPONENTES

Resistencias:

R1, R3, R4 = 4k7
R2 = 100 k Ω
R5, R6 = 1k5

Condensadores:

C1, C2 = 33 pF
C3 = 1 mF, 16 V, radial
C4, C5, C6 = 47 nF
C7 = 10 mF, 16 V, radial

Semiconductores:

D1 = 1N4148
D2 = LED, rojo, alta eficiencia
D3 = LED, amarillo, alta eficiencia
IC1 = AT90S8515-8PC, programado, código de pedido 010212-41
IC2 = 74HCT00
IC3 = LTC490 CN8 (Linear Technology)

Varios:

K1 = DB25 enchufe (macho), montaje PCB, pines acodados
K2 = conector de 10 vías
K3, K4, K5 = tira SIL de 3 pines
K6 = conector mini-DIN 6 contactos para montaje PCB
K7 = tira SIL de 6 pines
X1 = cristal de cuarzo 8MHz
PCB, código de pedido **010212-1**
(Ver página del Servicio de Lectores)
Disco, programas y código fuente, código de pedido **010212-11**

El proyecto de software y el dibujo de la PCB también están disponibles para descargarlos gratuitamente desde la página de Elektor www.elektor-electronics.co.uk

mediante un cable estándar. IC3 es un transceptor RS-485 que convierte las señales TTL del puerto serie del microprocesador en las señales RS-485 requeridas por el estándar DMX. Para conseguir esto que parece tan sencillo no debemos de dar aislamiento entre el microprocesador y el transceptor. Si nos preocupamos sobre los valores de tensión excesivos debidos a los bucles de tierra, deberíamos colocar Transiles de protección entre las líneas DMX+ y DMX- en K4 y K5 (pines 2 y 3).

Puede que ya hayamos notado que la interface DMX tiene una entrada DMX IN, pero esta entrada no se utiliza. Ha sido incluida para futuras ampliaciones.

El LED D2 nos indica si la interface tiene alimentación. La misión del LED D3, el cual se excita por el propio microprocesador, es reportar cualquier error que pueda ocurrir en los comandos enviados por el ordenador.

Debido a que esta interface se puede alimentar desde el PC al que está conectada, los valores de R4 y R5 se eligen para usar con LEDs de alta luminosidad y bajo consumo. Si decidimos utilizar LEDs estándar para esas resistencias colocaremos valores de 220 ó 330 W.

Gracias a su consumo de baja corriente (menos de 10 mA), la interface se puede alimentar directamente desde el PC al que está conectado. No es necesario un regulador de tensión. Para obtener la alimentación desde el PC haremos uso de la conexión desde

el PC al teclado, el cual se alimenta desde la alimentación de 5 V. Los conectores K6 y K7 se conectan a un simple cable de teclado (PS/2 o AT), y la tensión necesaria se puede obtener de este cable.

El conector K2 sólo se utiliza para programar el microcontrolador. El cable de programación Atmel se puede colocar directamente a este conector (ver la página de Atmel www.atmel.com para el esquema del circuito). Si pedimos el microcontrolador pre-programado en el Servicio de Lectores (código de pedido 010212-41), K2 se puede omitir.

Por último, el triplete R2/C3/D1 genera un pulso de reset para el microprocesador. Este pulso pasa a través de K3, donde podemos seleccionar la fuente para el reset (conector K2 o el circuito de reset). El jumper K3 no se colocará si queremos programar el microprocesador nosotros mismos. Durante el funcionamiento normal, el jumper debería estar presente en K3 para proporcionar un enlace a R2/C3. Si compramos un microprocesador pre-programado, simplemente podemos reemplazar K3 con un puente de alambre encajado entre las conexiones de los pines 2 y 3.

Aquí viene el hierro soldado

Gracias al pequeño número de componentes, el ensamblaje de la interface es relativamente fácil. La **Figura**

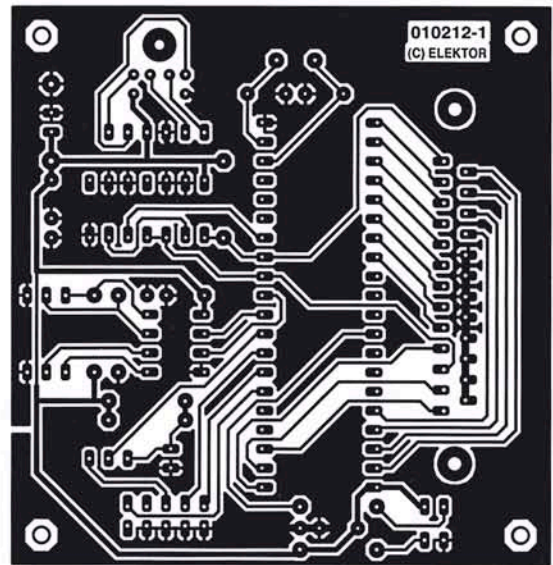
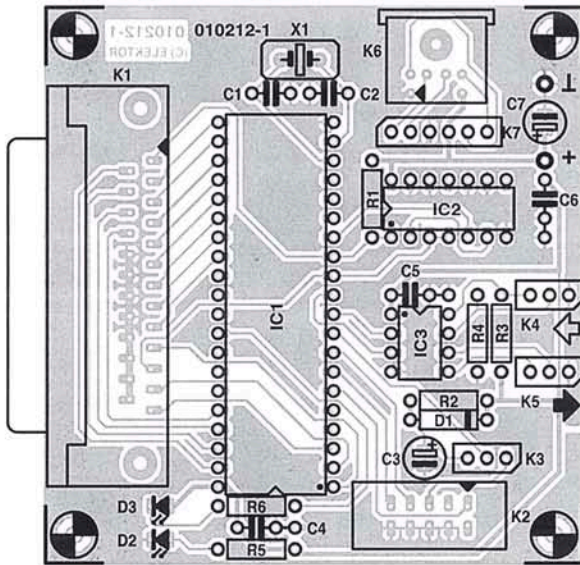


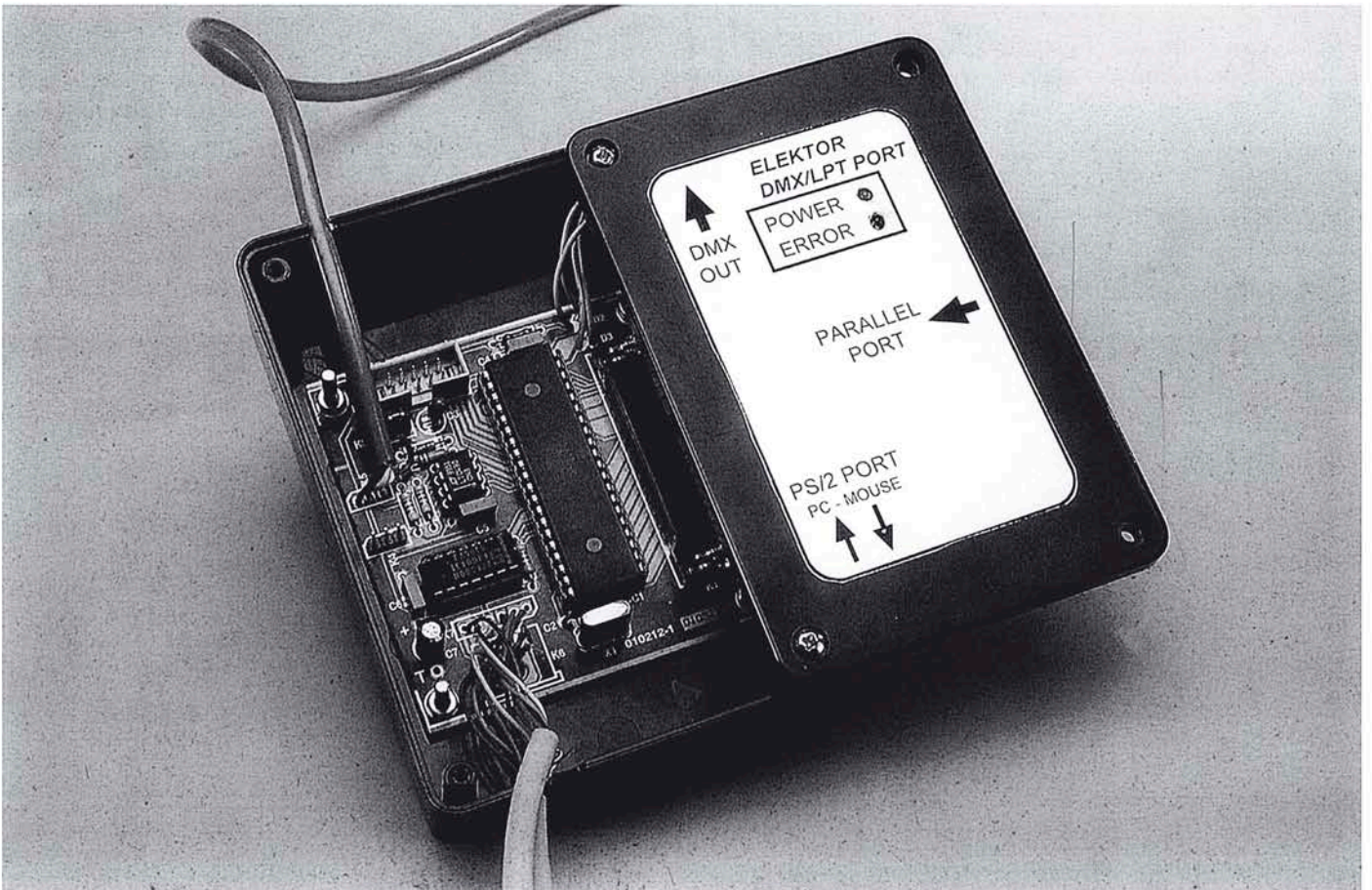
Figura 2. El diseño de una placa de circuito impreso a una cara es bastante sencillo, no pueden cometerse muchos errores durante el ensamblaje.

2 muestra la pista y cara de componentes de la placa del circuito. En lo que se refiere a las pequeñas dimensiones, ésta es una placa de circuito impreso a simple cara (el autor ha hecho una placa que sólo es la mitad

de la versión PLCC del microprocesador de Atmel). Usaremos zócalos de buena calidad para los tres circuitos integrados. Si queremos conectar K1 de la interface directamente al conector DB25 de un PC, nos aseguraremos

de utilizar un conector sin tuercas de seguridad integradas para K1, porque de otra forma el conector no podría enchufarse en el PC.

En lo que a la tensión de alimentación se refiere, hay varias opciones. La solución más simple es tomar la alimentación desde el cable



del ratón o teclado presentes en todo PC. También podemos alimentar la interface desde una alimentación externa con un regulador de 5 V, pero esto es más caro.

Si la interface se va a alimentar desde el cable del ratón o teclado, la solución más sencilla es tomar una prolongación de cable DIN/DIN y cortarlo a la mitad. Entonces las dos mitades de cable se pueden soldar a los pines adecuados de los conectores K6 y K7. La **Figura 3** muestra la asignación de pines del cable PS/2 y DMX y los conectores del chasis. En la placa del circuito se ha implementado K6 como un conector de bus

PS/2. Si colocamos tal conector a la placa del circuito, todo lo que tenemos que hacer es soldar un trozo de cable con un conector PS/2 o DIN a K7. El conector original de ratón o teclado se puede insertar en el conector de bus PS/2 de la placa del circuito.

La caja

El objetivo final es colocar el circuito en una caja. Debido a las pequeñas dimensiones de la placa,

esto no debería presentar ninguna dificultad.

Podemos instalar la placa del circuito en una caja separada y conectada al puerto paralelo del PC mediante un cable adecuado (ver la fotografía de cabecera del artículo), o situar el circuito en una pequeña caja a la que se puede colocar directamente el conector del puerto paralelo del PC.

Tenga en cuenta que la última solución no es siempre la más conveniente, debido al tamaño de la placa del circuito. En un ordenador de oficina normalmente hay bastante sitio en la parte posterior, pero en un portátil, los otros conectores están normalmente bastante cerca del conector de impresora que tiene la pequeña caja DMX y hará más o menos imposible el uso de otros conectores.

La placa del circuito no es particularmente sensible al ruido externo, pero es recomendable usar una caja de plástico apantallada (por ejemplo, una con una capa de grafito conectada a masa).

La salida DMX está implementada de una manera especialmente sencilla, normalmente usando un pequeño trozo de cable colocado en un conector de 3 ó 5 pines XLR. Este cable se suelda a K5.

Lenguaje Centronics

Tan pronto como el circuito se conecte podemos comenzar la prueba y la programación. Gracias al sencillo diseño, es bastante más fácil trabajar de forma adecuada desde el inicio. Sin embargo, se ha escrito un programa especial para hacer más fácil probar la placa del circuito. Este programa es muy fácil de usar, pero no obstante nosotros hemos resumido las distintas órdenes en un archivo separado en disquete flexible.

Además de proporcionar funciones especiales para comprobar la interface (tal como el Error de encendido en el LED D3), este programa también es adecuado para probar focos.

Obviamente nuestra intención es que esta interface se pueda usar con otros programas de prueba. Si queremos hacer las cosas fácilmente, podemos usar los programas Soft Controller I y II desarrollados por el mismo autor. Estos ya se han discutido de forma breve en el artículo que describe la interface MIDI/DMX, con

Tabla I: Interface de comandos.

Comando 'D'	Cambia un grupo de valores para una serie de canales DMX.
sintaxis:	ESC D hi lo nn dd dd dd...
parámetros:	hi: byte alto del primer canal DMX que vamos a cambiar lo: byte bajo del primer canal DMX que vamos a cambiar nn: número de canales DMX a cambiar (1–255) dd: DMX valor para el primer canal a cambiar dd: DMX valor del segundo canal a cambiar dd: DMX value del tercer canal a cambiar etc. ... El número de valores (dd) se debe incluir en el parámetro nn.
Comando 'd'	Cambia el valor de un canal DMX (número 1–256).
sintaxis:	ESC d nn dd
parámetros:	nn: número de canal DMX (0–255 para canales 1–256) dd: valor a enviar
Comando 'e'	Cambia el valor de un canal DMX (número 257–480)
sintaxis:	ESC e nn dd
parámetros:	nn: número de canal DMX (0–224 para canales 257–480) dd: valor a enviar
Comando 'F'	Configura los parámetros estándar (Start Code = 0 and Break Time = 100 ms)
sintaxis:	ESC F
Comando 'I'	Inicializa la interface (también ocurre cuando la alimentación se enciende)
sintaxis:	ESC I
Comando 'S'	Cambia el código de inicio
sintaxis:	ESC S ss
Comando 'Z'	Reseteo completo de la memoria DMX (todos los canales puestos a 0)
sintaxis:	ESC Z
Comando 'T'	Cambia la configuración del Time Break.
sintaxis:	ESC T tt tt: Configuración del Time Break (en pasos de unos 50 ms; el valor estándar es '2' para 100 ms)

el cual también es compatible. Aún más, tal y como ya mencionamos al principio de este artículo, la principal ventaja de este circuito es que puede programarse fácilmente, incluso bajo Windows.

Mientras que la mayoría de las interfaces DMX disponibles comercialmente para PC no se pueden utilizar sin uno de los programas de control y drivers, nuestra interface utiliza un protocolo que está disponible en todos los PCs: el protocolo Centronics.

La interface actúa como una impresora emulando los comandos adecuados. Esto significa que, en lo que se refiere a la ejecución de un comando, todo lo que tenemos que hacer es instruir al PC para que imprima una serie de caracteres. La impresión está funcionalmente incluida en varios lenguajes de programación, tales como Delphi, C++ Builder, Visual Basic, lo cual significa que el mayor problema ya está resuelto.

Si programamos bajo DOS, podremos usar la instrucción LPRINT de Basic para comunicarnos con la interface.

Como también notaremos, esta interface no está limitada para usarse con PCs. Cualquier dispositivo con un puerto Centronics es, por definición, compatible con la interface.

En lo que a programación se refiere nos limitaremos a describir el método que se debe usar para enviar comandos a la interface. Para aquellos lectores que programen bajo entorno Windows hemos desarrollado una pequeña librería DLL que se puede encontrar en el disco (o bien descargar de la página de Internet de forma gratuita). Esto significa que podemos tener problemas al aprender como usar la API Windows (interface para programar la aplicación).

La programación de la interface es realmente muy simple. La interface reconoce un cierto número de comandos, cada uno de los cuales consta de unas letras a la que sigue uno o más parámetros, dependiendo de un comando particular. Para permitir a la interface reconocer el comienzo de un comando, se utiliza el código ESC ('27' en ASCII o '1B' en hexadecimal) como código de identificación.

Como ejemplo, veamos qué bytes son enviados a la 'impresora' para configurar el canal 18 DMX a un valor

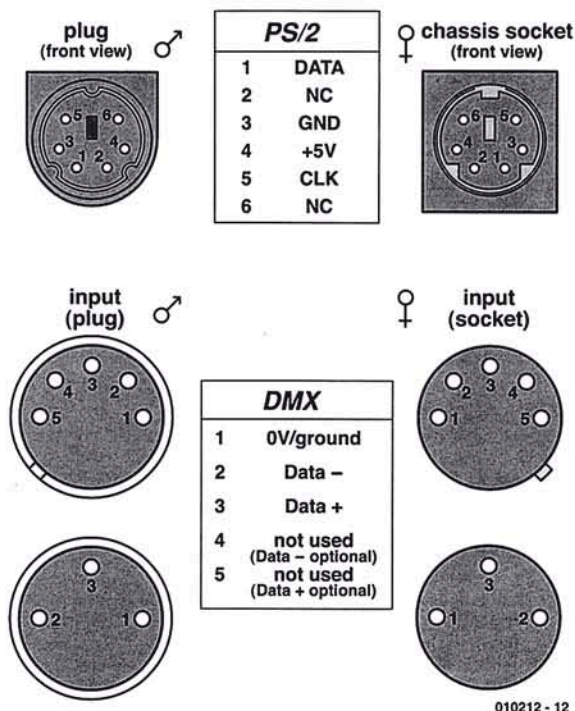


Figura 3. Usando la información de este conector podemos asegurarnos de que el cable de interconexión se coloca adecuadamente sobre la placa.

de 155. Para este propósito usaremos el comando 'd', seguido de un número de canal (el cual puede valer de 1 a 256, pero nosotros aquí usaremos de 0 a 255 para colocarlo en un simple byte), y después seguirlo por el valor DMX que queremos enviar.

Los bytes enviados al puerto paralelo aparecerán de la siguiente manera (los valores hexadecimales se muestran entre paréntesis):

```
ESC (1Bh)
d (64h)
17 (11h)
155 (9Bh)
```

Otro ejemplo: para reinicializar la interface todo lo que tenemos que hacer es enviar el comando 'Z'.

La secuencia de byte es:

```
ESC (1Bh)
Z (5Ah)
```

Sabiendo esto, todo lo que tenemos que hacer para crear nuestro propio 'programa del siglo' es escribir una serie de bytes y enviarlos a la interface.

La Tabla 1 resume todos los comandos reconocidos por la interface.

(010202-1)

Contenido del disco (010212-II)

8515def.inc
defio.inc
Lptdmx.asm
Lptdmx.hex
LPT_DMX_LIB

LPTDMX_TESTER.EXE
LPTDMX_TESTER_E.DOC
LPTDMX_TESTER_F.DOC

fichero que contiene las definiciones de los registros del 8515
fichero que contiene las descripciones de los pines de entrada y salida
fichero en ensamblador para la interface LPT/DMX (en francés)
fichero hexadecimal para la interface LPT/DMX
carpeta que contiene la librería DLL para el control de la interface bajo Windows, junto con el fichero 'readme' (inglés)
programa de prueba para la interface
guía de usuario en inglés para el programa de test
guía de usuario en francés para el programa de test

Curso básico de microcontrolador (IV)

parte 4: el compilador READS51 C

Si alguien pretende trabajar seriamente con microcontroladores, tarde o temprano debe utilizar el lenguaje de programación C. En esta última entrega del curso básico de microcontrolador, usaremos el compilador READS51 C de Rigel.

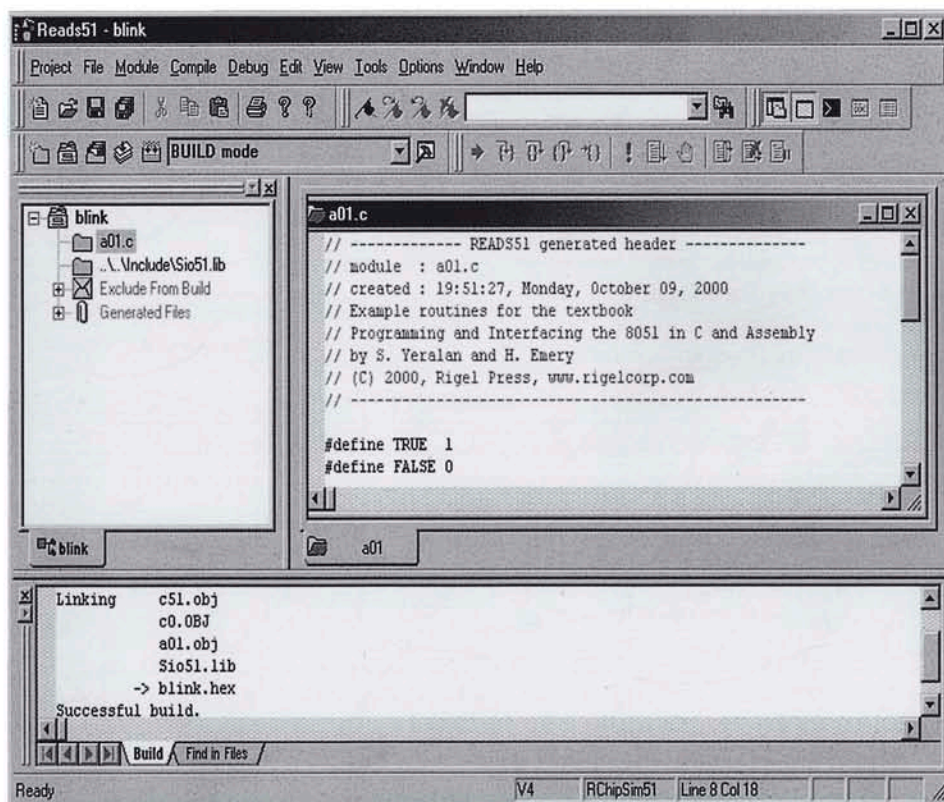


Figura 1. READS51 en acción.

Hasta ahora hemos utilizado el ensamblador BASIC-52 como lenguaje de programación en el Curso Básico de Microcontrolador. Ahora debemos comenzar a trabajar en C, usando un compilador que se puede seleccionar y descargar desde la dirección de Internet:

<http://www.rigelcorp.com/8051soft.htm>

Los ficheros más importantes son:

[SetupReads51.exe y Reads51.pdf](#)

Un compilador C traduce un código fuente en un lenguaje de máquina puro, al contrario que un intérprete Basic, el cual sólo genera código intermedio que se debe inter-

pretar y ejecutar en tiempo de ejecución. El C es mucho más rápido que el Basic.

El lenguaje C lleva en el mercado bastante tiempo y está disponible para muchos sistemas diferentes. Su ventaja más importante es que los programas en C son muy independientes del hardware utilizado. El resultado del gran esfuerzo realizado se puede llevar con facilidad a otros sistemas. El ANSI C se definió al principio de 1988 (ANSI viene de 'Instituto de Estándares Nacionales Americano') para crear un estándar común. Se desarrolló una pequeña versión llamada 'Small C' para sistemas de microcontroladores. Aunque tiene ciertas limitaciones comparado con el ANSI C, tales como la ausencia de variables reales, tiene la ventaja de que se puede utilizar con sistemas muy pequeños. En Internet podemos encontrar varios compiladores de Small C gratuitos. En este curso hemos elegido el READS51, porque es particularmente adecuado y tiene una interfaz de usuario que no está nada mal.

El READS51 fue desarrollado por Rigel para el mercado educativo y está enfocado al uso con sus placas de microcontrolador. La compañía tiene este producto disponible únicamente para fines privados o educativos. Rigel ha dado permiso amablemente a Elektor para usar el

Listado 1. El primer programa ejemplo.

```
// ----- READS51 generated header -----
// module : a01.c
// created : 19:51:27, Monday, October 09, 2000
// Example routines for the textbook
// Programming and Interfacing the 8051 in C and Assembly
// by S. Yeralan and H. Emery
// (C) 2000, Rigel Press, www.rigelcorp.com
// -----

#define TRUE 1
#define FALSE 0

#include <sfr51.h> // P1_0 is defined here
// prototypes
#include <Sio51.h>

main(){
int n;

// --- initialize serial port (9600 Baud) ---
InitSerialPort0(DEF_SIO_MODE);
//DEF_SIO_MODE is defined in <Sio51.h>
putc('\n');

// endless loop
while(TRUE)
{
P1_0=0; // LED on
putc('+');
for(n=0; n<10000; n++); // waste some cycles
P1_0=1; // LED off
putc('0');
for(n=0; n<10000; n++); // waste some cycles
}
}
```

compilador en el Curso Básico de Microcontroladores. Todos los lectores interesados deberían descargarlo de la página de Rigel en Internet e instalarlo en su sistema. También podemos encontrar aquí muchas otras cosas interesantes que nos pueden ser de gran ayuda. Todos los ejemplos para el Curso Básico de Microcontroladores tienen etiquetas y comentarios en inglés. Hemos decidido hacerlo así, para dar un enfoque internacional al curso.

La mejor forma de comenzar con el READS51 es usar uno de los ejemplos que lo acompañan. El proyecto se puede cargar desde 'Project/Open Project'. Si hacemos un doble clic

sobre el fichero fuente en el módulo principal, a01.c, aparecerá el texto fuente en la ventana del Editor (ver Listado 1).

Un programa en C siempre tiene una función principal llamada main() que se activa cuando el programa se ejecuta. A primera vista, el programa parece que sólo tiene esta función, pero en realidad hay alguna otra función relacionada con la interface serie del microcontrolador. Esas funciones están localizadas en el módulo Sio51.h. Éstas abren la interface serie a 9.600 baudios (con una frecuencia de cristal de 11.0592 MHz), la cual es justo la que necesita la placa 'Flash Board' de Elektor. Esta placa, que lleva el

89S8252, se publicó en Diciembre del 2001 en esta revista.

Para los principiantes de C, la notación del programa puede al principio parecer un poco difícil, por lo que daremos algunas explicaciones:

#define TRUE 1

Define una constante (TRUE se sustituirá por '1' allí donde aparezca)

#include <sfr51.h>

Enlace con un fichero cabecera que contiene las definiciones.

main()

```
{
...
}
```

Forma la función principal 'main'. La forman todas las instrucciones que estén contenidas dentro de las llaves que la siguen.

int n;

Declara una variable n de tipo entero, con unos valores permitidos de -32768 a +32767. Un punto y coma (;) termina la línea.

InitSerialPort0

(DEF_SIO_MODE);

Llama a una función con un paso de parámetro, en este caso la función está en el módulo Sio51.h que inicializa el puerto serie.

// endless loop

Un comentario que hace más legible el programa y que no tiene traducción ejecutable de ningún tipo.

while(TRUE)

```
{
...
}
```

Forma un bucle. En lugar de TRUE para un final de bucle deberíamos usar una condición diferente para definir la condición bajo la cual se cumple el bucle. Todas las instrucciones que se ejecutan en el bucle están dentro de las llaves.

P1_0=0;

Una instrucción. Aquí el bit variable P1_0 tiene asignado el valor '0'.

putc('+');

Saca texto a través del puerto serie. La función putc está definida en Sio51.h., un carácter de texto, el cual es una variable o constante de tipo char (carácter = carácter de texto, siempre un byte); se transfiere.

ATMELISP, una nueva herramienta de descarga

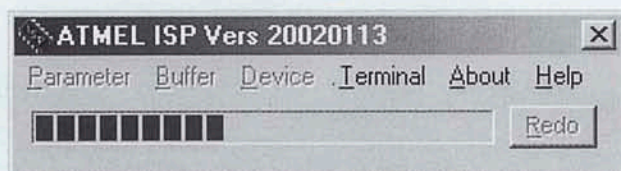
El programa MicroFlash.exe para descargar los programas sobre la placa del 89S8252 sólo trabaja con COM1 y COM2 y no nos informa del resultado de la operación, lo cual puede ser un problema, ya que no sabemos si la operación de descarga se ha hecho o no correctamente. Sin embargo, los lectores de Elektor no nos debemos de preocupar de esto porque se ha desarrollado un programa nuevo y más extenso llamado ATMELISP, el cual permite programar la memoria Flash utilizando varios tipos de sistemas. Junto con el Atmet Starter Kit y una placa, el programa también soporta el sistema Elektor y el sistema ModuleBus (EX52-Flash). El nuevo software se puede descargar de la página de Elektor.

Cuando el archivo zip esté descomprimido, tendremos un programa .exe y un fichero de ayuda comprensivo. La pantalla de inicio (Figura A) es pequeña y se puede colocar fácilmente en el monitor siguiendo a las otras aplicaciones. Sólo aparecerán ventanas mayores cuando se ejecuten funciones de programa. La primera cosa que debemos hacer es seleccionar la interface serie, el dispositivo conectado y otros parámetros críticos. Un clic sobre el botón marcado con 'DK7JD' configura una asignación adecuada a las líneas de programa para las señales RS-232 usadas en la placa de circuito de Elektor. Aquí también podemos ver que es bastante fácil usar ARMELISP para programar cualquier placa de circuito que hayamos desarrollado nosotros mismos y que utilice el mismo procesador, porque simplemente son seleccionadas y asignadas convenientemente tres líneas. En algunos casos, puede ser necesario ajustar los tiempos de retraso. Nuestra experiencia nos muestra que con un PC relativamente lento, el valor del Retardo del Reloj se debe incrementar desde 0 a 0.01 ms. La Figura B nos muestra la ventana para la selección de los parámetros de configuración.

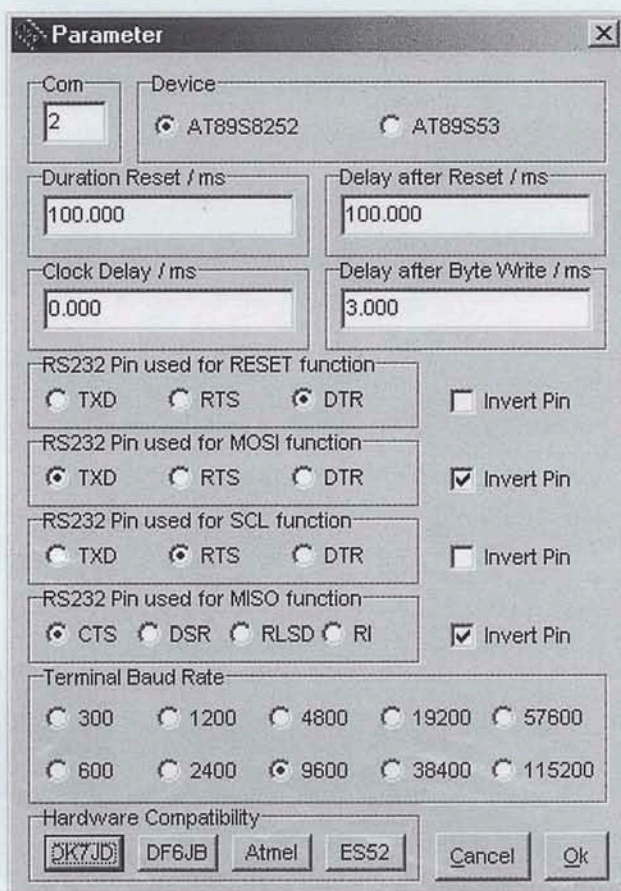
El resto de los procedimientos se pueden ilustrar usando un ejemplo concreto. La Flash ROM del microcontrolador se va a cargar con el primer programa ejemplo desde el compilador C. Esto requiere el código para leer primero el buffer. ATMELISP puede leer ficheros en formatos binario y hexadecimal de Intel. Aquí se carga el fichero Blink.hex. También es importante que echemos un vistazo rápido al editor hexadecimal, después de cargar el fichero (Figura C), para ver el contenido y tamaño del fichero.

Para programar el microcontrolador, seleccionaremos Device/Write Buffer para Código de Memoria. Aquí debemos tener cuidado de no confundir el Código de memoria con la Memoria de Datos, la cual es una EEPROM de 2 Kbytes donde se almacenan los datos del microcontrolador. Los dos tipos de memoria se pueden leer y programar. Además de esto, es posible cargar 'bits cerrojo' en el microcontrolador para evitar cargar el software que se va a leer (Figura D). Con

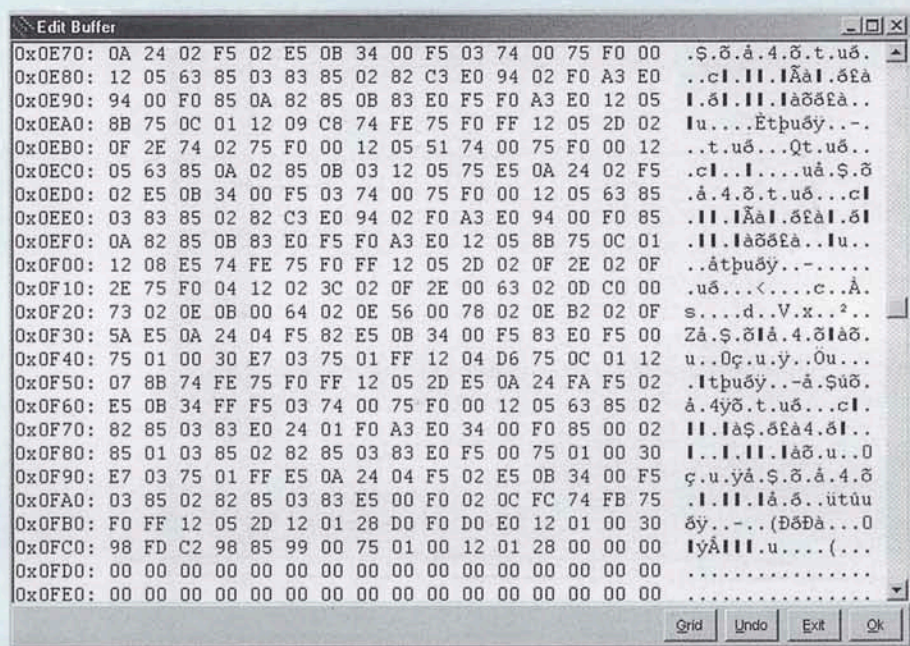
A



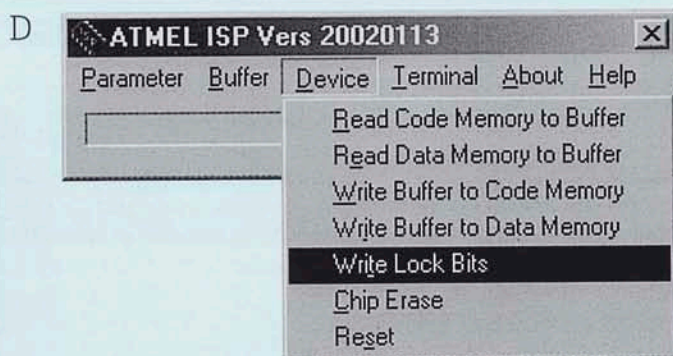
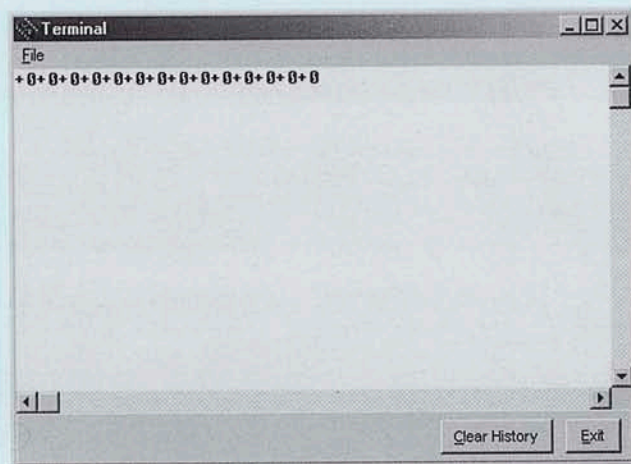
B



C



estos bits debemos de tener mucho cuidado: si todos los bits están a uno, cualquier programación serie del chip está bloqueada. En este caso, también es fácil borrar todo el chip al utilizar el programa. Sólo una programación para-



lela del dispositivo puede sacarnos de este truco.

ATMELISP también incluye una función de terminal (**Figura E**) que se puede utilizar para visualizar las salidas del primer programa de ejemplo. Éste requiere el cable de interface para conectarse al otro conector de interface de la placa.

```
for(n=0; n<10000; n++);
```

Forma un bucle de cuenta que se puede escribir en Basic, como 'For n=1 to 10000: Next n'. Aquí el bucle no contiene ninguna instrucción y como puede verse termina con el punto y coma. Aquí podríamos colocar un grupo de instrucciones entre llaves.

Incluso si no hemos captado completamente todo el programa en C, interesa ver si este programa corre en la placa Flash Board. Antes de hacerlo, compilaremos el programa, lo cual significa traducirlo al lenguaje máquina. El programa se puede traducir utilizando 'Compiler/build' o usando la tecla F9. El

proceso es relativamente complicado, porque los módulos objeto individuales se deben traducir primero y después se linkan todos juntos para formar el programa completo. El resultado final es un fichero en formato Hexadecimal de Intel llamado `Blink.hex`. Éste está colocado en el directorio de proyecto `\work\blink` y se puede ver bajo 'Generated files' como `.hex`.

Ahora el fichero de formato Intel del proyecto se puede transferir a la Flash Board utilizando el programa MicroFlash. Cuando descargamos el programa en la memoria Flash del microcontrolador, notaremos el porqué de la sencillez del programa fuente. La versión traducida es relativamente grande (4 Kb), debido principalmente al módulo C51.obj, con el cual está unido. Este módulo contiene todas las funciones que proporciona READS51, incluyendo una que normalmente no es necesaria aquí.

Después de que el programa se haya descargado con éxito, es el momento de la prueba. Conectaremos un LED y una resistencia en serie entre P1.0 y Vcc, ¡parpadea! Por supuesto, y seguro que la gente que está acostumbrada a trabajar con electrónica convencional dirá que es el mismo resultado que obtendría usando dos simples transistores, pero el programa hace más cosas. También inicializa una interface serie, la cual ahora podemos utilizar. En lo que se refiere a comprobar qué transferencias son posibles, necesitamos un emulador de programa terminal.

Lo más sencillo es usar el programa terminal Basic del curso, pero los parámetros de comunicaciones debemos corregirlos. El

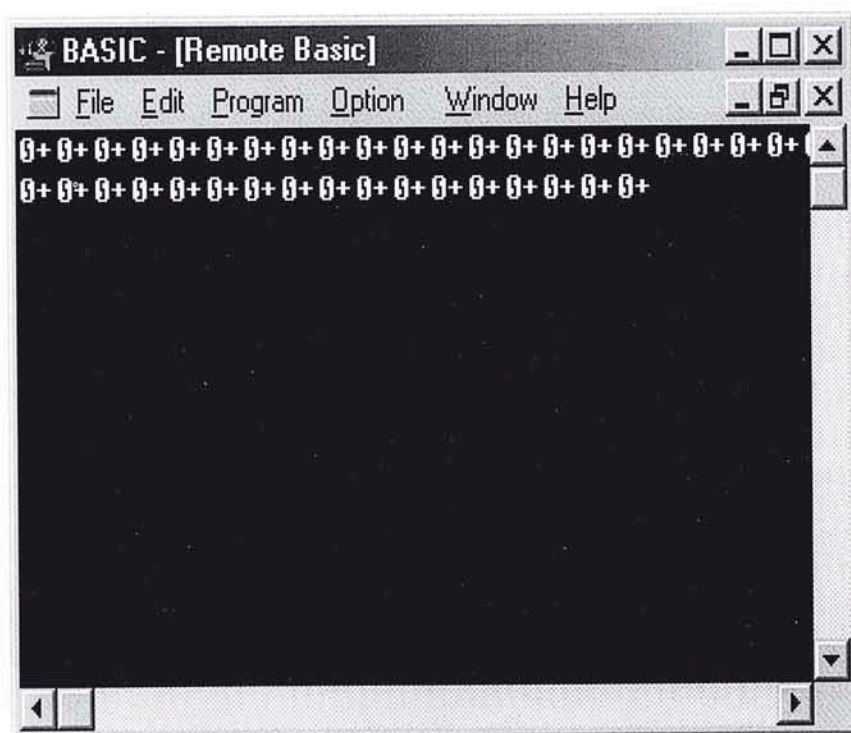


Figura 2. salida de texto en la ventana del Editor.

programa C usa 9.600 baudios, mientras que Basic.exe usa normalmente 19.200 baudios. Sin embargo, es fácil cambiar la velocidad de transferencia usando este programa: abriremos el fichero Basic.ini con un editor de texto y añadiremos la línea 'Baud=9600' (ver Listado 2).

Listado 2. Contenido del fichero

modificado.ini para el terminal Basic.

```
[AHBASIC]
COM=2
Baud=9600
```

El programa terminal nos mostrará ahora lo que envía el programa en C, que no es sino una serie de caracteres '0' y '+' que se alternan con cada cambio de estado de la salida P1.0 (ver **Figura 2**). Esto es porque el programa llama a la función putc para sacar un carácter individual.

Salidas de puerto rápidas

Ahora que hemos visto este primer ejemplo, es hora de escribir nuestro propio programa en C. Comenzaremos con un programa que simplemente genera salidas de puerto rápidas, por lo que podemos hacer alguna comparación con los lenguajes de programación que hayamos utilizado antes.

Lo primero que tenemos que hacer es crear un nuevo proyecto, cuyo nombre debemos introducir en 'Project/New Project'. Aquí escogeremos el nombre Output. Entonces el programa READS51 crea un nuevo directorio con el nombre Work\Output. En el Listado 3 se muestra la salida del puerto. El compilador tiene que saber qué proyecto es el que va a traducir. Esto se consigue ejecutando una vez 'Project/Set Project Active'. Si olvidamos hacer esto, será traducido el último proyecto que hayamos procesado. El nuevo programa genera el código de salida en el fichero Output.hex que ya se puede transferir a la memoria del programa del procesador utilizando MicroFlash. Todo lo que necesitamos para verificar esta función es un osciloscopio o unos altavoces. La señal de alta frecuencia podemos encontrarla en P1.0. Tenemos una frecuencia de 4 KHz: el periodo es de 250 ms. El programa necesita 125 ms para cada nuevo puerto de salida.

Al contrario que Basic, C permite diferentes tipos de variables. El primer programa ejemplo utiliza una variable *n* de tipo entero (int), lo cual significa una variable entera con un valor dentro del rango -32768

Listado 3. Un programa para puertos de salida rápidos.

```
// ----- READS51 generated header -----
// module : C:\Rigel\Reads51\Work\Output\Output.c
// created : 12:33:17, Friday, November 09, 2001
// -----

#define TRUE 1
#define FALSE 0

#include <sfr51.h> // P1 is defined here

main(){
  unsigned char n;
  // endless loop
  while(TRUE)
  {
    for(n=0; n<256; n++)
    {
      P1=n;
    }
  }
}
```

Listado 4. Divisor de frecuencia por 20 .

```
// ----- READS51 generated header -----
// module : C:\Rigel\Reads51\Work\Count\count.c
// created : 18:26:23, Monday, November 12, 2001
// -----

#include <sfr51.h>

void pulse(void){
  while(P1_0);
  while(!(P1_0));
}

main(){
  int n;
  n=0;
  while(1)
  {
    while (n<10)
    {
      pulse();
      n=n+1;
    }
    P1_1=1;
    while (n<20)
    {
      pulse();
      n=n+1;
    }
    P1_1=0;
    n=0;
  }
}
```


Tabla 1. Comparación de los tres lenguajes de programación

Lenguaje	Memoria	Tiempo de bucle	Auto-arranque
Basic-52	8 K ROM, RAM	2.500 μ s	Sólo con EEPROM
READS51 C	>4 K ROM, RAM	125 μ s	Sí
Ensamblador	< 1 K ROM	3 μ s	Sí

a +32767, mientras que en el segundo ejemplo, utiliza una variable de tipo carácter sin signo (unsigned char), lo que equivale a un byte. Sin embargo, los experimentos han mostrado que esto no tiene ninguna consecuencia en la velocidad de ejecución.

La efectividad de los lenguajes de programación se puede comparar fácilmente (ver Tabla 1). El criterio más importante es la cantidad de memoria que toma el programa, su velocidad y la habilidad para implementar un programa para el microcontrolador Flash. Aunque los programas en C utilizan el sistema RAM, el programa completo está colocado solamente en la Flash ROM. Esto es porque un programa en C se arranca sólo cuando se conecta la alimentación, al contrario que los programas en BASIC-52.

Un divisor de frecuencia en C

Buscando algo más complejo recomendamos la realización de un divisor de frecuencia por 20, que ya escribimos en BASIC-52. Una com-

paración directa de los dos programas nos puede ayudar a reconocer las diferencias en la estructura y notación. A este programa se le ha dado el nombre de Cuenta (Count) (ver Listado 4).

El listado muestra una de las ventajas decisivas del C como lenguaje de programación: el programador está forzado a usar un estilo estructurado. Esto hace el programa más fácil de leer. Aquí tenemos la función pulso, la cual suspende la ejecución del programa mientras espera el siguiente flanco positivo en P1.0. Cuando usamos una función, lo normal es pasar un valor y recibir otro de vuelta. Sin embargo, la función pulso no retorna ningún parámetro (void=empty vacío), y no recibe ninguno. El C no hace ninguna distinción entre funciones y procedimientos, como es costumbre en Pascal y Delphi: sólo tiene funciones.

La variable bit P1_0 puede ser '1' ó '0'. Mientras que la condición siguiente del 'while' sea verdadera (=1), se ejecuta un bucle. El segundo bucle contiene la condición actual de forma negada, lo cual se expresa con el símbolo de exclamación '!' (!P1_0)). Entonces el

segundo bucle se acaba cuando el nivel de entrada cambia de '0' a '1', lo cual significa que se ha detectado un flanco positivo. La rutina principal llama la función 'pulse' dos veces: una vez para $n = 0, 1 \dots 9$ y de nuevo para $n = 10, 11 \dots 19$.

Aquí de nuevo la cuestión crítica es, ¿cuál es la mayor frecuencia de entrada que se puede aplicar sin que haya errores? Para esta prueba usaremos un generador de funciones conectado a P1.0 y un osciloscopio conectado a P1.1. Podemos medir que la mayor frecuencia alcanzada es de 3 KHz. Para refrescarnos un poco la memoria, con BASIC-52 sólo alcanzábamos 50 Hz, mientras que usando ensamblador se puede llegar a los 100 KHz.

(010208-5)

Literatura:

Sencer Yeralan/Helen Emery
*Programming and Interfacing The 8051
 Microcontroller in C and Assembly*
 Rigel Press 2000



**ELECTRONICA
ALVARADO**

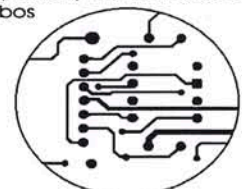
Calle Jaén, 8
 (Metro Alvarado)
 28039 Madrid

915 330 827



ABELLÓ
 Circuitos Impresos

- Simple y doble cara
- Metalizados
- Prototipos
- Pequeñas y medianas series
- Gobos



C/ Escipión 34 bajos
 08023 Barcelona
 Teléfono (93) 212 06 85
 Fax-modem (93) 211 28 65
 e-Mail: abelloci@lix.intercom.es

CONDICIONES GENERALES

Los circuitos impresos, carátulas autoadhesivas, ROMs, PALs, GALs, microcontroladores y disquetes que aparecen en las páginas de ELEKTOR se encuentran a disposición de los lectores que lo requieran. Para solicitarlos es necesario utilizar el cupón de pedido que se encuentra en las páginas anexas.

Este mismo cupón también puede utilizarse para efectuar pedidos de los libros de la colección de ELEKTOR (en versión original inglesa).

- Los ítems marcados con un asterisco (*) tienen una vigencia limitada y su disponibilidad solo puede garantizarse durante un cierto periodo de tiempo.

- Los ítems que no se encuentran en esta lista no están disponibles.

- Los diseños de circuitos impresos se encuentran en las páginas centrales de la Revista. En ocasiones y por limitación de espacio no se garantiza la publicación de todos los circuitos. En estos casos los lectores interesados pueden solicitar los diseños, utilizando el mismo cupón de pedido y les serán enviados a su domicilio contra reembolso de 500 pts. (incluidos gastos de envío).

- Los EPROMs, GALs, PALs, (E)PLDs, PICs y otros microcontroladores se suministrarán ya programados.

Los precios y las descripciones de los diferentes productos están sujetos a cambios. La editorial se reserva el derecho de modificar los precios sin necesidad de notificación previa. Los precios y las descripciones incluidas en la presente edición anulan los publicados en los anteriores números de la Revista.

FORMA DE ENVÍO

Los pedidos serán enviados por correo a la dirección indicada en el cupón de las páginas anexas. Además los lectores pueden formular pedidos por teléfono llamando al número 91 327 37 97 de lunes a viernes en horario de 9,30 a 14 h y de 16 a 19 h. Fuera de este horario existe un contestador telefónico preparado para recoger las demandas. Los gastos de envío serán abonados por el comprador, tal como se indica en el cupón.

FORMA DE PAGO

Todos los pedidos deberán venir acompañados por el pago, que incluirá los gastos de envío, tal como se indicó anteriormente.

El pago puede realizarse mediante cheque conformado de cualquier banco residente en territorio español, giro postal anticipado, tarjeta VISA (en este caso debe indicarse la fecha de caducidad, domicilio del propietario de la tarjeta y firma del mismo).

Nunca se deberá enviar dinero en metálico con el pedido. Los cheques y los giros postales deben ser nominativos a la orden de VIDELEC S.L.

SUSCRIPCIONES A LA REVISTA Y EJEMPLARES ATRASADOS

Las suscripciones o pedido de números atrasados, si se encuentran disponibles, se realizarán a LARPRESS, C/ Medea, 4 5ª planta (Edificio ECU) 28037 Madrid.

Los precios de ejemplares atrasados son de 600 pts más gastos de envío.

COMPONENTES UTILIZADOS EN LOS PROYECTOS

Todos los componentes utilizados en los proyectos ofrecidos en las páginas de la Revista se encuentran generalmente disponibles en cualquier establecimiento especializado o a través de los anunciantes de este ejemplar. Si existiera alguna dificultad especial con la obtención de alguna de las partes, se indicará la fuente de suministro en el mismo artículo. Lógicamente los proveedores indicados no son exclusivos y cualquier lector podrá optar por su suministrador habitual.

CONDICIONES GENERALES DE VENTA

Plazo de entrega: El plazo normal será de 2-3 semanas desde la recepción del pedido. No obstante no podemos garantizar el cumplimiento de este periodo para la totalidad de los pedidos.

Devoluciones: Aquellos envíos que se encuentren defectuosos o con la falta de alguno de los componentes podrán ser devueltos para su reposición, solicitando previamente nuestro consentimiento mediante llamada telefónica al número (91) 3273797 en horario de oficina. En este caso la persona que llame recibirá un número de devolución que deberá hacer constar al devolver el material en un lugar bien visible. En este caso correrá por nuestra cuenta el gasto de envío de la devolución, debiéndolo hacer así constar el remitente en su oficina postal. A continuación se le enviará nuevamente el pedido solicitado sin ningún gasto para el solicitante.

En el caso de que la devolución se realice por otra causa ajena a la revista, sólo se admitirá si el material devuelto se encuentra en perfectas condiciones para ser vendido de nuevo. En este caso al remitente le será devuelto el importe previamente enviado, reteniendo un 10 % del precio para cubrir los gastos de manipulación y embalaje.

En cualquiera de los casos anteriores, sólo se admitirán las devoluciones en un plazo de tiempo de 14 días contados a partir de la fecha de envío del pedido.

Patentes: Algunos de los circuitos o proyectos publicados pueden estar protegidos mediante patente, tanto en la Revista como en los libros técnicos. La editorial LARPRESS no aceptará ninguna responsabilidad derivada de la utilización inadecuada de tales proyectos o circuitos para fines distintos de los meramente personales.

Copyright: Todos los dibujos, fotografías, artículos, circuitos impresos, circuitos integrados programados, disquetes y cualquier otro tipo de software publicados en libros y revistas están protegidos por un Copyright y no pueden ser reproducidos o transmitidos, en parte o en su totalidad, en ninguna forma ni por ningún medio, incluyendo fotocopiado o grabación de datos, sin el permiso previo por escrito de Editorial LARPRESS.

No obstante, los diseños de circuitos impresos si pueden ser utilizados para uso personal y privado, sin necesidad de obtener un permiso previo.

Limitación de responsabilidad: Todos los materiales suministrados a los lectores cumplen la Normativa Internacional en cuanto a seguridad de componentes electrónicos y deberán ser utilizados y manipulados según las reglas universalmente aceptadas para este tipo de productos. Por tanto ni la editorial LARPRESS, ni la empresa suministradora de los materiales a los lectores se hacen responsables de ningún daño producido por la inadecuada manipulación de los materiales enviados.

CONSULTORIO TÉCNICO

Existe un Consultorio técnico telefónico gratuito a disposición de todos los lectores. Este servicio se presta todos los lunes y martes laborables en horario de 17 a 19 h.

El número de teléfono para consultas es el 91 375 02 70.

Código
Precio
(€)

E264 MAYO 2002

Sistema de Medida de Velocidad:

- PCB	010206-1	25,74
- Disk, source and hex files	010206-11	11,38
- 87LPC762, programmed	010206-41	24,34

Control Remoto de Procesos utilizando un Teléfono Móvil (2):

- PCB	010087-1	30,81
- Disk, project software	010087-11	11,38
- GAL16V8, programmed	010087-31	11,33

Sencillo Programador para Micros AVR:

- PCB	010055-1	30,14
- Disk, project software	010055-11	11,13
- Set: PCB + 010055-11	010055-C	30,08

Receptor de Banda VHF:

- PCB	010064-1	30,54
-------	----------	-------

CI multi-propósito para modelismo (II):

- PCB, speed controller	010008-1	11,00
- PCB, hot glow/go-slow	010008-3	11,00
- Disk, source code files	010008-11	14,00
- 87LPC762BN, programmed	010008-41	23,47

E263 ABRIL 2002

Panel Mezclador de Luces:

- PCB	0000162-1	78,00
-------	-----------	-------

Circuito integrado multipropósito para modelismo (I):

- PCB, servo reserve	010008-2	10,58
- PCB, 2-channel switch	010008-4	10,58
- Disk, source code files	010008-11	13,44
- 87LPC762BN programmed	010008-41	23,00

Sistema de Desarrollo PICee:

- PCB	010062-1	38,39
- Disk, example programs	010062-11	11,00
- Set: PCB + 010062-11	010062-C	44,00

Amplificador Final Versátil:

- PCB, amplifier	010049-1	20,00
- PCB, power supply	010049-2	33,00

E262 MARZO 2002

Interfaz de disco duro para puerto de impresora:

- PCB	010047-1	25,59
- Disk, project software	010047-11	10,84
- 7064LC84-15, programmed	010047-31	73,21

Iluminación y caja de cambios:

- Disk, project software	010204-11	10,86
- PIC16C57, programmed	010204-41	25,40

Interrogador maestro:

- PCB, transmitter and receiver	010030-1	39,00
- Disk, project software	010030-11	11,00
- PIC17C44-16/P, programmed	010030-41	59,30

E261 FEBRERO 2002

Placa microcontroladora flash para 89S8252:

- PCB	010208-1	32,00
- Disk, project software	010208-11	11,00

Medidor de descarga/capacidad de batería:

- PCB set	010201-1	34,03
- Disk set, project software	010201-11	19,00
- ST62T65B6, programmed	010201-41	40,00

Cerradura electrónica codificada:

- PCB	004003-1	22,54
- Disk, project software	006001-1	11,00
- PIC16F84-04/P, programmed	006501-1	31,28

Fuente de alimentación digital para laboratorio:

- PCB	000166-1	25,00
- Disk set, project software	000166-11	13,44
- PIC16F84A-04P, programmed 1A version	000166-41	43,00
- PIC16F84A-04P, programmed 2.5 version	000166-42	43,00

Control remoto RC5:

- Disk, project software	000189-11	11,00
- Attiny22L-8PC, programmed	000189-41	20,00

UART USB:

- PCB	010207-1	37,93
- Disk, project software	010207-11	18,00
- CY7C63001A, programmed	010207-41	63,02
- Set: PCB + 010207-11 + 010207-41	010207-C	86,00

E260 ENERO 2002

Control remoto PCM en miniatura (2):

- Transmitter PCB	010205-1	23,52
- Receiver PCB	010205-2	19,84
- 87LPC768FN, programmed	010205-41	37,36
- 87LPC762BN, programmed	010205-42	23,20
- Disk, project software	010205-11	11,01

Medidor de capacidad y descarga de batería:

- PCB, includes discharger PCB	010201-1	34,53
- ST62T65, programmed	010201-41	49,16
- Disk, project software	010201-11	19,24

Demultiplexor DMX de 8 canales:

- PCB	010002-1	41,05
- EPROM 27C256 (programmed)	010002-21	18,91
- Disk, project software	010002-11	13,64

61

E253 JUNIO 2001

Convertor de velocidad de muestreo a 96kHz:

- PCB 010014-1 43,62

Crescendo Edición Millenium:

- PCB, amplifier (mono block) 010001-1 26,47
- PCB, power-on delay 974078-1 16,56

MIDI en el puerto RS232:

- PCB 000139-1 31,49
- EPROM 27C256, programmed 000139-21 18,26
- Disk, driver, source code, hex file 000139-11 11,08
- Set: PCB + 000139-21 + 000139-11 000139-C 53,53

E252 MAYO 2001

Luces MIDI y control de diapositivas:

- PCB 000179-1 76,76
- EPROM 27C256, programmed 000179-12 38,70
- disk, source code & binary 000179-11 28,38

ADC 2001 para audio:

- PCB, converter 010017-1 39,67
- PCB, power supply 010017-2 21,68

Generador de pulsos programable:

- PCB 000200-1 21,87
- Disk set, project software 000200-11a/b 13,54
- PCB + disk set 000200-C 32,18

E251 ABRIL 2001

Tarjeta prototipo para Bus PCI (I):

- PCB 010009-1 112,95
- disk, Windows software 010009-11 12,69
- GAL22V10, programmed 010009-31 20,94
- disk, DOS software 010009-12 12,69
- PCB, 010009-31 + disk 010009-C 146,57

MCS BASIC-52 V1.3:

- Disk, project software 000121-11 29,82
- EPROM, programmed 000121-21 39,97

Controlador de velocidad doble (2):

- PCB, SpeedControl + speedPower2 000070-4 26,65
- PCB, SpeedControl + speedPower1 000070-5 28,55
- ST62R60B86, programmed 000070-41 48,23
- Disk, ST6 source code 000070-11 20,94

Receptor de AM:

- PCB 000176-1 34,90

E250 MARZO 2001

Decodificadores de control remoto RC5:

- PCB 000081-1 17,77
- Disk, project software 000081-11 12,69
- AT90S2343, programmed 000081-41 31,09

Emulador para la memoria EPROM 27C256:

- PCB 000153-1 46,95
- AT89C2051, programmed 000153-41 24,81
- Disk, project software 000153-11 12,69
- PCB + AT89C2051 + disk 000153-C 76,14

GBPB - Placa de prototipo para Gameboy:

- PCB 000151-1 49,5

Sistema de identificación de llamada vía radio:

- PCB, caller unit 000108-1 20,31
- PCB, central receiver 000108-2 20,31
- 3 disk, project software 000108-11a/b/c 24,75
- 1 caller PCB + 1 receiver PCB + disk set 000108-C 56,47

Modulador de anchura de pulsos:

- Disk, GAL listing 000123-11 12,69

E249 FEBRERO 2001

Convertor de sonido a luz PLUS:

- PCB 000107-1 51,39
- Project disk 000107-11 12,69
- PIC16F84, programmed 000107-41 31,09

E248 ENERO 2001

CAN Adapter for ISA Bus:

- PCB 000071-1 64,92
- Project disk 000071-11 13,25
- PCB + project disk 000071-C 73,53

USB Audio-DAC:

- PCB 000169-1 23,18

E247 DICIEMBRE 2000

e-KEY: Sistema de acceso seguro:

- PCB 000089-1 26,38
- disk, source code files 000089-11 17,58
- AT90S1200, programmed 000089-41 28,41

Cámara sobre Tren de Modelismo:

- PCB 000129-1 16,91

(GBDSO) Osciloscopio de muestreo digital en pantalla de consola Gameboy:

- PCB 990082-1 22,32
- disk, DSO Grab and Mathcad demo appl. 996035-1 23,00
- EPROM AT27S256 (PLCC44), programmed 996528-1 37,88
- Set: PCB + 996035-1 + 996528-1 990082-C 74,40

TV PAL Generador de imagen patrón:

- EPM7064, programmed 000084-31 68,32

Receptor de Onda Corta (OC) Regenerativo:

- PCB 000112-1 25,70

Diseño de periféricos (I):

- Set: PCB + 000074-11 000074-C 27,06
- PCB 000074-1 17,59
- Project software 000074-11 13,53

E246 NOVIEMBRE 2000

Salida S/PDIF:

- PCB 000131-1 26,23

E245 OCTUBRE 2000

Modelo digital Märklin para control remoto de trenes:

- Set: PCB + 996016-1 000066-C 46,19
- PCB 000066-1 26,77
- Project disk 996016-1 23,43

Interfaz USB:

- Project disk 000079-11 13,39
- PCB 000079-1 14,73
- Set: PCB + 000079-11 + 000079-41 000079-C 48,87
- CY7C63001ACP (programmed) 000079-41 18,74

E244 SEPTIEMBRE 2000

Tensión de alimentación simétrica:

- PCB 004064-1 11,90

Lámpara de LED blanco:

- PCB 004024-1 8,54

E243 AGOSTO 2000

Puerto de I/O de 8 bits:

- PCB 994077-1 9,76

Adaptador para SB Live! Player 1024:

- PCB 004085-1 8,89

Poleando curvas con HP-GL/2:

- Disk, project software 006005-1 10,68

Implementación del bus I² C:

- Disk, project software 006006-1 6,10
- BASIC interpreter in EPROM 006505-1 8,54

E242 JULIO 2000

Cerradura inteligente para puertas:

- AT89C52-12PC, programmed 000051-41 12,58
- Disk, AT89C52 source code file 000051-11 6,45
- PCB 000051-1 12,79

Lector de tarjetas magnéticas:

- PCB 000054-1 8,06
- AT89C2051-12PC, programmed 000054-41 12,58
- Disk, all project software 000054-11 6,45
- Set: PCB + 000054-11 + 000054-41 000054-C 24,18

Espía de un hilo:

- PIC16F84 (programmed) 000048-1 17,74
- PIC16C54 (programmed) 000048-42 14,19
- Disk, all project software 000048-11 6,45

Interfaz del PC para el Bus CAN:

- PCB 000039-1 15,48
- Disk, all project software 006004-1 9,73

E241 JUNIO 2000

Teclado de funciones especiales:

- PCB 002006-1 25,29
- ST62T60(programmed) 002006-41 49,28
- PCB y 002006-41 002006-C 70,03

Sistema de invención robótico de Lego (2):

- PCB 000040-1 12,97

Medidas mediante Word y Excel:

- Disk, Word template and .DLL 000053-11 12,97

Mezclador MIDI:

- PCB 000021-1 24,00
- Disk, AT90S source code files 996038-1 21,40
- 2 x AT90S2313 (a+b), programmed 996531-1 78,46

Temporizador de reposo RC5:

- Disk, PIC source code files 000026-11 12,97
- PIC16F84, programmed 000026-41 31,77

Pantalla táctil:

- Disk, PIC source code & executable 000055-11 12,97

E240 MAYO 2000

Estimulador de músculos de bajo impacto:

- Disk: source and hex code 000041-11 13,52
- AT89C2051, programmed 000041-41 31,55
- PCB 000041-1 22,45

Puerto paralelo universal de entrada/salida para PCs:

- Set: PCB + 002011-11 002011-C 43,07
- Disk: all project software 002011-11 12,13
- PCB 002011-1 35,18

E239 ABRIL 2000

Control de volumen digital:

- disk, source code listing 990080-11 11,19
- PCB 990080-1 30,58
- EPROM 27C256 (programmed) 006506-1 16,79

Receptor de onda media miniatura:

- PCB 000034-1 17,99

Regulador de carga solar:

- PCB 000019-1 17,99

Medidas de temperatura con un DS1621:

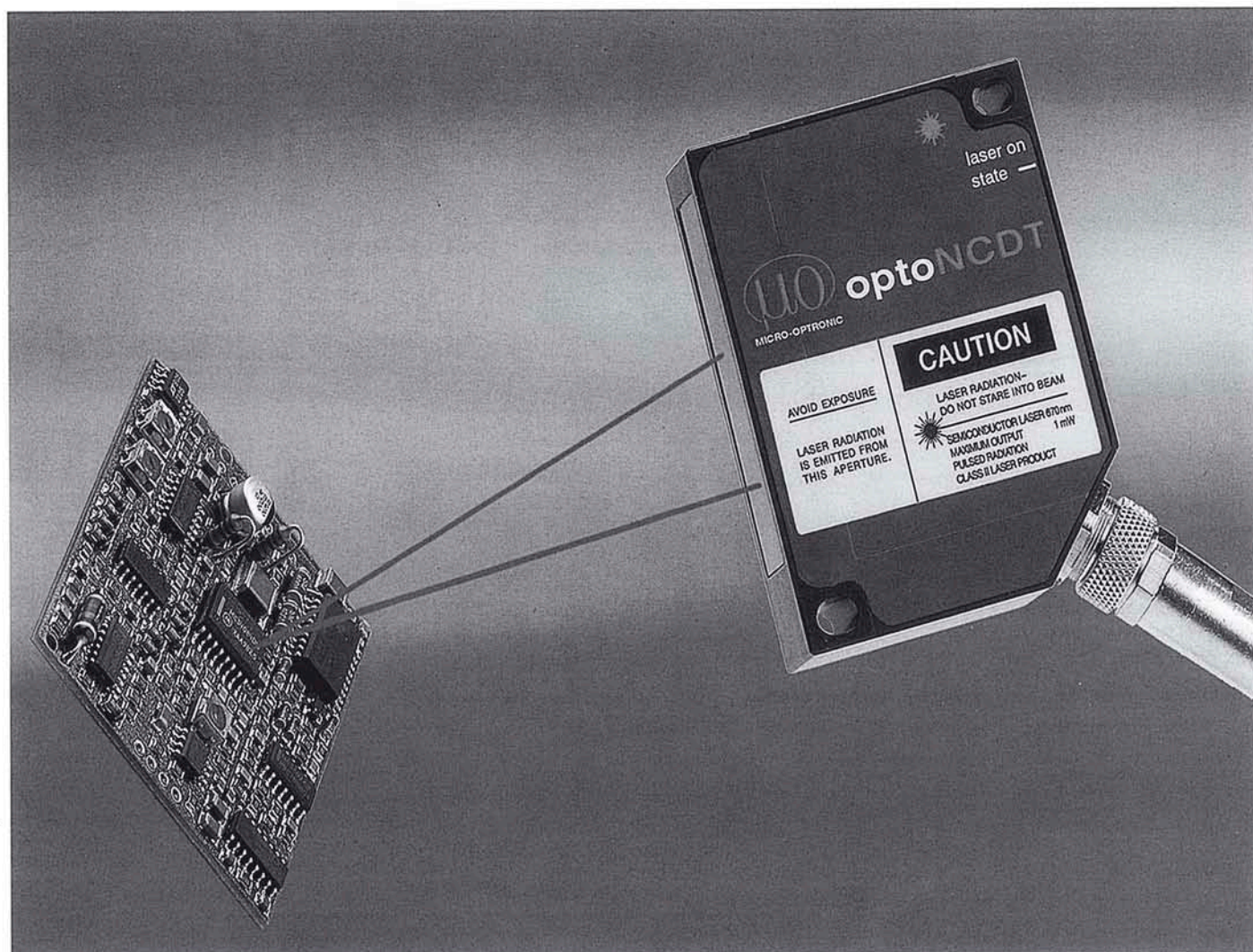
- Disk, project software 996027-1 11,99

Medida a distancia sin contacto

Sensores y principios

Por J. Häuser

La medida de distancias sin contactos es cada vez más importante en el análisis de vibraciones, alineación, posición o curvado. Este artículo explica los métodos de medida y discute su aplicación.



La medida de distancias sin contactos tiene aplicaciones muy prácticas cuando el movimiento de un objeto hace que se deshaga o se vea afectado por acoplamiento de masas o la fuerza ejercida por un dispositivo de medida; o si las superficies son sensibles, en cuyo caso debemos procurar no dañarlas o cuando el movimiento es rápido y se debe mantener.

Estas cuestiones de comprobación y medida son muy comunes en

investigación y desarrollo (I+D), automatización, control de calidad y aplicaciones de máquinas de control. Para todas esas cuestiones hay un gran número de empresas que ofrecen modelos de sensor utilizando varios principios de medida. En los últimos años, han destacado tres métodos de medida de distancias sin contacto: el principio de la corriente de Eddy, el principio capacitivo y el principio de triangulación óptica.

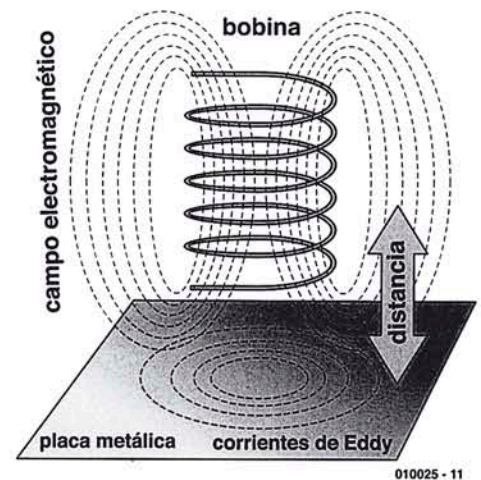


Figura 1. Principio de corriente de Eddy.

Tabla I: Comparación de principios de medida sin contacto

Principios de corrientes de Eddy

Ventajas

- Trabaja con todo tipo de metales conductores de electricidad, tanto si son ferromagnéticos como si no lo son.
- Sensor pequeño.
- Insensible a suciedad, polvo, humedad, aceite, sustancias dieléctricas en medida de agujeros.
- Utilizable en aplicaciones electromagnéticas sensibles.
- Amplio rango de temperatura de operación.
- Elevada precisión.

Limitaciones

- Señal de salida y linealidad dependiente de las propiedades eléctricas y magnéticas de los materiales usados.
- Linealización y calibración individual.
- El oscilador de alta frecuencia limita la longitud del cable de 12 a 18 m.
- Diámetro del sensor (y medida de diámetro de la parte de medida) incrementa como máximo rango incrementado.

Principio capacitivo

Ventajas

- Independientemente del metal utilizado en la medida: la sensibilidad y linealidad son las mismas.
- A temperatura elevada es muy estable, los cambios de conductividad debidos a la temperatura no tienen efecto.
- También es útil con objetos medidos no conductivos.

Limitaciones

- Sensible a materiales dieléctricos en los agujeros medidos y sólo útil en dispositivos limpios y secos.
- El cable del sensor debe ser corto, ya que la capacidad del propio cable afecta a la sintonía del circuito resonante.
- El diámetro del sensor se incrementa tanto como se incrementa el máximo rango.

Principio de triangulación óptica

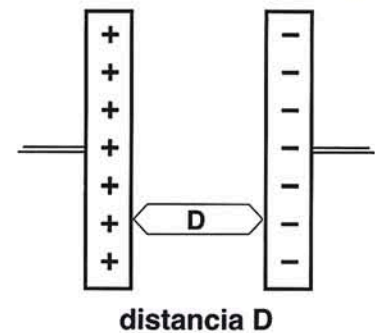
Ventajas

- Medida de pequeños diámetros.
- El sensor puede estar lejos del objeto medido.
- Tiene un gran rango de medida.
- Independiente de los materiales usados.

Limitaciones

- Utilización limitada con superficies lisas (espejos, cristales, CDs, metales pulidos) o superficies con baja reflectividad (las superficies negras mates).
- Utilización limitada con superficies transparentes o parcialmente transparentes (cristal, cerámicas, plásticos tales como el Teflón).
- Los haces pasan por superficies libres de polvo y no obstruidas.

condensador de placas



$$X_C = \frac{1}{j\omega C}; C = \epsilon \cdot \epsilon_0 \cdot \frac{F}{d}$$

$$X_C = \text{constante} \cdot \text{distancia}$$

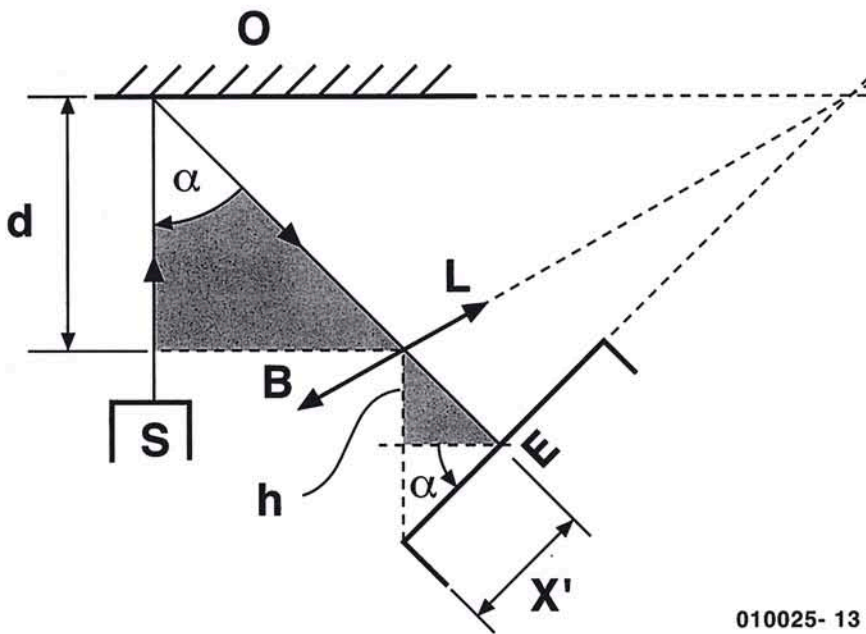
010025 - 12

Figura 2. Principio capacitivo.

El principio de corriente de Eddy

El principio de corriente de Eddy es un método especial de medida inductiva. El efecto está basado en la disipación de energía almacenada en un circuito resonante cuando las corrientes de Eddy son inducidas en un objeto metálico conductivo situado cerca.

Si, como se muestra en la **Figura 1**, se coloca una placa metálica cerca de una bobina por la que circula una corriente alterna, se inducirá un campo electromagnético que inducirá las corrientes de Eddy en el plato. Por las leyes de Lenz, el campo



010025- 13

Figura 3. Principio de triangulación óptica

debido a la corriente de Eddy se opone al campo inducido. La consecuencia es una pérdida de energía que cambia la inductancia efectiva de la bobina del sensor. Por ello, la amplitud de las oscilaciones de la bobina del sensor cambia como una función de la distancia del objeto medido (una placa metálica). Este principio se conoce como el principio de 'Pérdidas de Eddy', y requiere un oscilador con una amplitud y frecuencia estable, que funciona normalmente en un rango de 1 a 2 MHz. Se utiliza una bobina con núcleo de aire (en lugar de ferrita).

El principio de la capacitancia

La medida a distancia sin contacto, por capacitancia, está basada en la teoría del condensador de placas ideal (Figura 2). Un cambio en la distancia de las placas da lugar a un cambio de su capacidad. Se hace circular una corriente alterna de frecuencia constante: la amplitud de la tensión alterna en las placas es proporcional a la distancia entre el sensor y el objeto medido. Al mismo tiempo, se genera una tensión de offset ajustable en la electrónica de control. Después de la desmodulación, las dos tensiones pasan a un amplificador diferencial que genera la salida como una señal analógica. Para medir la reactancia X_c de la placa del condensador, sin ninguna linealización, se emplea una relación directamente proporcional. En la práctica, el sen-

sor se construye como un condensador en anillo y la linealización es casi perfecta, independientemente de la conductividad del metal en el objeto medido.

También los sensores capacitivos se pueden utilizar con materiales aislantes. Se requieren circuitos extra para obtener una señal de salida lineal con tales objetos, y una constante dieléctrica estable para lograr una característica fiable.

El principio de triangulación óptica

Este principio refleja un haz de láser pulsante sobre la superficie del objeto medido, y se basa en triángulos idénticos (Figura 3). El primer triángulo es el 'triángulo objeto', entre el objeto y las lentes, y el segundo es el 'triángulo imagen', entre las lentes y el detector. El detector tiene un diodo de efecto lateral o columna de sensores CCD. Se requiere una lente en el camino del rayo reflejado del objeto para permitir que los ángulos de incidencia y reflexión sean diferentes. Ésta es la única forma de resolver el tema de la profundidad. Más aún, el detector se debe configurar con un ángulo conocido. Aquí deberíamos tener en cuenta

la regla de Scheimpflug, la cual estima que el enfoque óptico se obtiene cuando el objeto, lentes y detector interceptan el plano en un simple punto. Esto nos da la siguiente relación:

$$\Delta d = \frac{B \cdot h}{\cos(\alpha)} \cdot \frac{1}{\Delta x}$$

Los fabricantes convierten la distancia física medida por el sensor en un rango de tensión estandarizada (por ejemplo 0 V a 10 V o 0 (4) mA a 20 mA). Alternativamente, el valor es digitalizado y transmitido sobre una interface RS-232 o RS485 para un PC. Con la ayuda de una tarjeta adecuada para PC (ver 'Tarjetas de medida PCI', Elektor Octubre 2000), y un software adecuado, se pueden solucionar un gran número de problemas.

(010025-1)

Literatura

Nota técnica T01d producida por:

Micro-Epsilon Messtechnik GmbH & Co. KG,
Postfach 1254
D-94493 Ortenburg
Alemania

Further information está disponible desde:

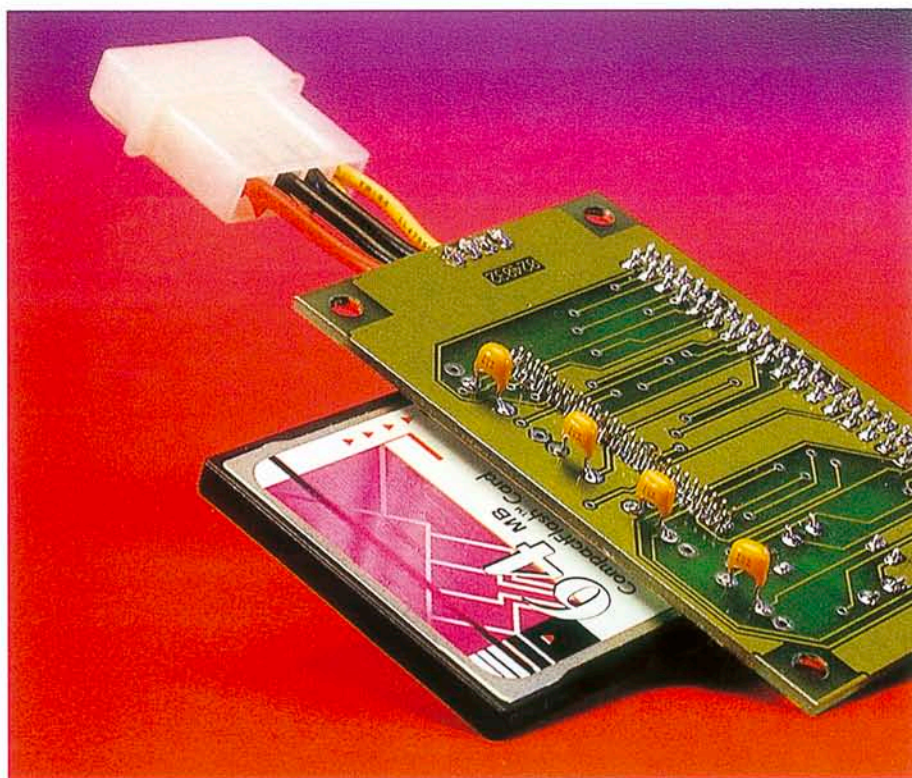
MJH Software
Mr. J. Häuser
Am Stadion 28
D-07629 Hermsdorf
Alemania
Tel./fax: +49 36601 82313
E-mail: jhaeuser@tridelta-hermsdorf.de

Controlador de CompactFlash para Bus IDE

memoria de estado sólido para el PC

Diseñado por Paul Goossens

Las tarjetas CompactFlash son un dispositivo de memoria que retienen su contenido sin necesidad de que esté presente una tensión de alimentación. Se usan, sobre todo, en cámaras digitales, entre otros equipos. Gracias a la “inteligencia” presente en este tipo de tarjetas, también pueden conectarse, fácilmente, a un ordenador, por ejemplo, para usarlas como “disco de estado sólido”. El adaptador que se presenta en este proyecto facilita la conexión de todo tipo de tarjetas CompactFlash a un ordenador.



Una tarjeta CompactFlash es una pequeña tarjeta de memoria, desarrollada originalmente por la casa Sandisk, con unas dimensiones aproximadas de 4 x 4 cm. Puesto que esta tarjeta utiliza células de memoria no volátil, el contenido de la memoria se mantiene durante años, incluso sin tener conectada una tensión de alimentación. Este tipo de tarjetas se emplean muy a menudo en las cámaras digitales de fotos.

En la actualidad, la capacidad de memoria que se puede almacenar en estas tarjetas llega hasta un total de 1

Gbyte. Así, incluso la casa IBM suministra pequeños discos duros en formato CompactFlash, con una capacidad de hasta 1 Gbyte.

La **Figura 1** nos muestra las dimensiones y las designaciones de los terminales de una tarjeta CompactFlash (que a partir de ahora denominaremos abreviadamente “tarjeta CF”). Hay dos espesores diferentes

especificados para estas tarjetas: las tarjetas Tipo I, con un grosor de 3,3 mm y las tarjetas Tipo II, con un grosor de 5 mm. La mayoría de las tarjetas de estado sólido son del Tipo I, mientras que los discos duros con tarjetas CF son del Tipo II.

El precio de las tarjetas CompactFlash ha caído vertiginosamente en el último año, de manera que podemos comprar una tarjeta con una capacidad de 64 Mbytes por menos de 100 euros.

Internamente una tarjeta CF está formada por un determinado número de módulos de memoria gestionados por su propio controlador. Los datos de entrada y salida se transfieren en paralelo, justo como sucede con un disco duro. Gracias a su diseño "inteligente", este tipo de tarjetas puede trabajar fácilmente junto con sistemas microcontroladores u ordenadores.

Los adaptadores para tarjetas PCMCIA están preparados para permitir utilizar las tarjetas CompactFlash en ordenadores portátiles. Actualmente, estos adaptadores sólo consisten en extensores para la conexión del bus.

Existe una amplia variedad de lectores de tarjetas disponibles en el mercado, montadas con un puerto USB, para el puerto serie, o para el puerto paralelo, de manera que las tarjetas CF puedan emplearse fácilmente con ordenadores.

Gracias a su precio relativamente asequible y a su característica de memoria no volátil, las tarjetas CompactFlash tienen un interés incuestionable para su uso como tarjetas de almacenamiento de capacidad media para ordenadores, por ejemplo, de manera que pueden sustituir fácilmente a las disqueteras tradicionales. Para ello no necesitamos comprar un lector independiente, ya que el circuito que describimos en este artículo proporciona una solución mucho más sencilla.

Circuito pasivo

La inteligencia presente en la tarjeta permite el uso de tres modos diferentes de trabajo. Uno de estos modos es el llamado "modo IDE verdadero", que se obtiene conectando el terminal 9 a masa. En este modo el comportamiento externo de la tarjeta es el mismo que si se tratase de un disco duro como una interfaz IDE. Lo único que necesitamos es un pequeño adaptador con dos conectores que permitan conectar la tarjeta CompactFlash al bus IDE del ordenador.

El esquema eléctrico de este circuito adaptador se muestra en la **Figura 2**. Este adaptador consiste, principalmente, en dos conectores, aumentado por algunos componentes pasivos añadidos. Así, el conector K1 es el adaptador para el bus IDE, mientras que K2 proporciona la conexión para la tarjeta CF.

El resto de componentes puede describirse rápidamente. De este modo, el diodo LED D1 hace que la actividad de lectura y escritura de

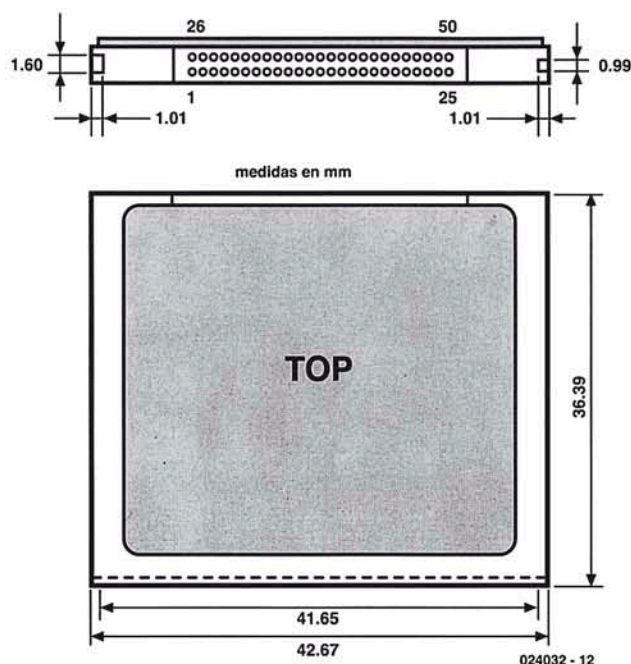


Figura 1. Dimensiones de la tarjeta CompactFlash y asignación de terminales en su conector (ver Tabla 1).

Tabla 1: Designación de terminales

Función	Terminal	Terminal	Función
GND	1	26	CDI
D03	2	27	D11
D04	3	28	D12
D05	4	29	D13
D06	5	30	D14
D07	6	31	D15
CE1; CE1; CS0	7	32	CE2; CE2; CS1
A10	8	33	VS1
OE	9	34	IORD
A09	10	35	IOWR
A08	11	36	WE
A07	12	37	RDY; BSY; IREQ; INTRQ
VCC	13	38	VCC
A06	14	39	CSEL
A05	15	40	VS2
A04	16	41	RESET; RESET; RESET
A03	17	42	WAIT; WAIT; IORDY
A02	18	43	INPACK
A01	19	44	REG
A00	20	45	BVD2; SPKR; DSAP
D00	21	46	BVD1; STSCHG; PDIAG
D01	22	47	D08

Notas:

XX = Señal invertida

Para terminales con tres designaciones (aa; bb; cc):

aa: en modo Memoria

bb: en modo E/S

cc: en modo IDE verdadero

la tarjeta CF sea visible, donde la resistencia R1 determina la cantidad de corriente que pasa a través del diodo LED. El terminal 39 del conector de la tarjeta CF está unido al punto de + 5 V por medio de la resistencia R2. Esto hace que la tarjeta CF actúe como disco maestro en el bus IDE. Si el terminal 39 está conectado a masa por medio del puente JP1, la tarjeta CF actúa como disco esclavo en el bus IDE. Los dos condensadores, C1 y C2, realizan funciones de desacoplo en la tensión de alimentación.

Existe además otro conector (K3) en la placa de circuito que se usa para conectar la tensión de alimentación. A pesar del hecho de que la tarjeta CompactFlash sólo necesita las líneas de + 5 V y masa, se ha utilizado un conector de cuatro hilos en el montaje para mantener compatible este conector de alimentación con los conectores de alimentación internos del ordenador. Los terminales de + 12 V de este conector no están siendo utilizados.

Montaje

La placa de circuito impreso de doble cara que se muestra en la Figura 3 es bastante fácil de trabajar para montar todo el circuito. Aunque el montaje se compone de dos conectores y de unos pocos componentes auxiliares, el conector de la tarjeta CF de 50 terminales es un poco difícil de montar.

La separación entre terminales de un conector normal es de 0,1 pulgadas, pero en el caso del conector de la tarjeta CF es de tan sólo 0,05 pulgadas (ligera-mente superior a 1 mm). Por lo tanto, es prácticamente imposible unir los dos conectores utilizando pequeñas longi-tudes de hilo. Incluso trabajando con una placa de circuito impreso es esen-cial tener mucho cuidado y utilizar una punta de soldadura muy fina.

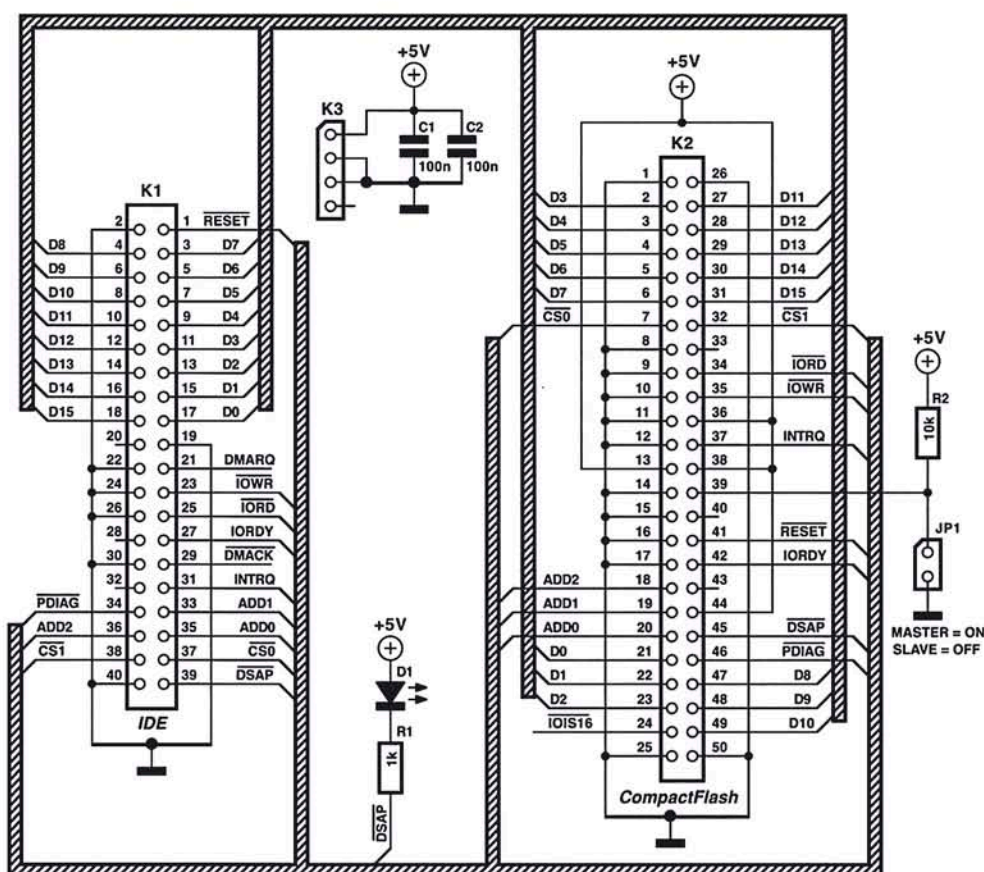
Si su entusiasmo ha llegado a ser tal que ha decidido construir este circuito, nos gustaría proporcionarle algún con-sejo: el primero de ellos es conseguir fácilmente el conector de la tarjeta CF. Este conector está disponible a través del suministrador Farnell, entre otros, pero la mayoría de las tiendas de com-ponentes electrónicos no tienen este tipo de conector en stock. Cuando ten-gamos el conector podremos construir la placa del circuito y montar el resto de los componentes.

Los conectores, las resistencias y el diodo LED se montan todos en la cara superior de la placa, sin embargo, los dos condensadores de desacoplo se instalan sobre la cara inferior, lo que significa que debemos cortar bastante

sus terminales por la parte superior de la placa, de manera que no toquen la tarjeta CompactFlash cuando esté insertada.

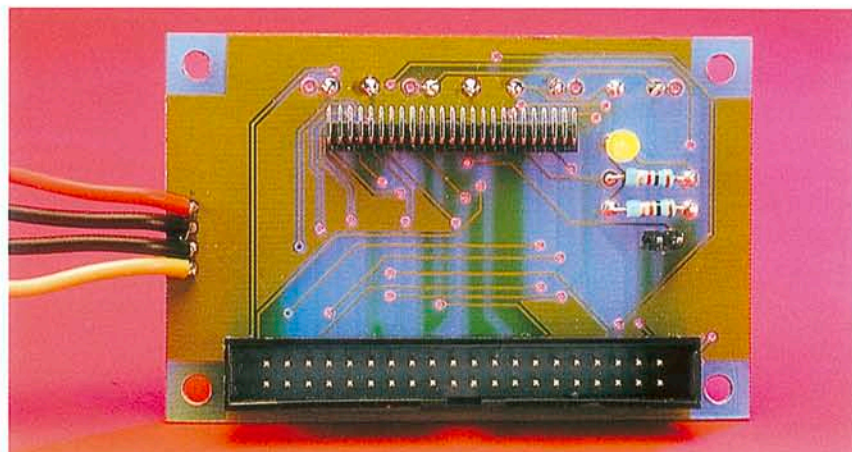
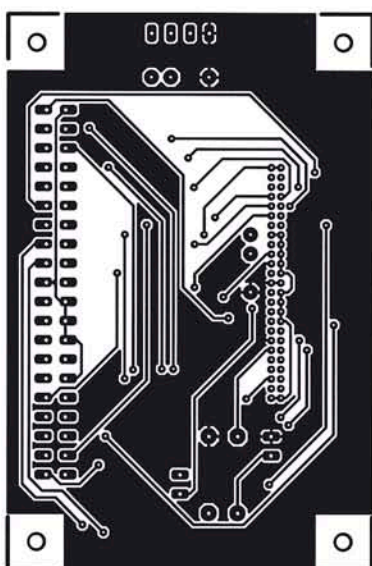
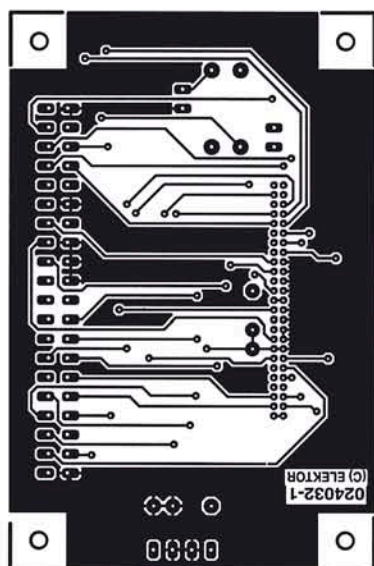
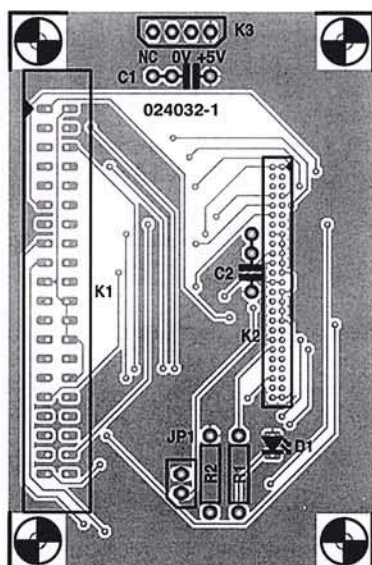
Para el conector de la tensión de ali-mentación podemos usar un cable de extensión como los de las fuentes de ali-mentación de un ordenador (es un com-ponente estándar que se obtiene fácil-mente en cualquier establecimiento de ordenadores), cortándole el conector hembra del extremo correspondiente. Seguidamente soldaremos los hilos uni-dos al conector macho sobre la posición K3 de la placa de nuestro montaje. El hilo rojo es la línea de + 5 V, los dos hilos negros son los hilos de masa y el cable amarillo es la línea de + 12 V.

Una vez que hemos realizado todo este trabajo, podemos montar el puente JP1 según nuestras necesida-des, es decir, en función de si quere-mos que el ordenador vea nuestra tar-jeta CF como disco esclavo o como disco duro o un lector CD-ROM en el bus IDE en cuestión y ya está configu-rado como maestro). Sin el puente montado, la tarjeta se reconoce auto-máticamente como disco maestro.



024032 - 11

Figura 2. Esquema eléctrico del adaptador sin componentes activos.



Siguiendo con el montaje, la tarjeta adaptadora puede instalarse dentro de nuestro ordenador y conectada al mismo. Una opción es utilizar una bahía de 5,25 pulgadas libre del panel frontal de nuestro ordenador y sujetar la placa en la parte trasera de dicho frontal *con la cara superior hacia abajo*. El conector K2 debe estar situado justo detrás de la apertura. La placa del circuito se puede conectar a uno de los buses IDE de la placa madre utilizando el conector K1 y un cable IDE estándar. De igual manera, conectaremos K3 a uno de los conectores complementarios libres de la fuente de alimentación del ordenador.

En este momento todo está listo para comenzar a trabajar. Para estar seguros de que todo está perfecto y utilizar el montaje de forma segura, recomenda-

mos insertar y retirar la tarjeta CF sólo cuando el ordenador esté desconectado.

Una vez que hayamos conectado en la tarjeta CF por primera vez y arrancado el ordenador, el sistema operativo Windows nos informará de la presencia de un nuevo disco añadiendo el controlador correspondiente para el nuevo dispositivo instalado. Normalmente el controlador ya estará presente en nuestro ordenador y este procedimiento se realizará de forma automática, pero en algunos casos es posible que se nos pida insertar el CD-ROM de instalación de Windows en el lector de CD-ROMs.

Una vez instalado el controlador, el sistema operativo Windows reconocerá automáticamente la tarjeta CF como una especie de disco duro donde podremos leer y escribir ficheros de manera tradicional en la letra de unidad asignada ("E:", "F:"...).

Como conclusión de este artículo, nos gustaría proporcionarles un *consejo importante*: *conecte siempre la tarjeta CF en el conector de la placa de nuestro montaje en la posición correcta, es decir, la cara superior de la tarjeta CF debe estar enfrentada con la cara de componentes de la placa de circuito impreso de nuestro montaje*. Éste es también el motivo por el que la tarjeta se monta en el ordenador con los componentes hacia abajo. La fotografía del principio de este artículo nos muestra claramente lo que intentamos decirle. Si queremos trabajar de forma segura podemos colocar una pequeña punta de obstrucción en la ranura de inserción para la tarjeta CF. Como podemos ver en la **Figura 1**, las ranuras guía en ambos laterales de la tarjeta CF tienen diferentes anchos, hecho que podemos aprovechar para crear un sistema que evite una inserción incorrecta de la tarjeta.

LISTA DE MATERIALES

Resistencias

R1 = 1 KΩ
R2 = 10 KΩ

Condensadores

C1, C2 = 100 nF

Semiconductores

D1 = Diodo LED amarillo de baja corriente

Varios

K1 = Conector "boxheader" SIL de 40 terminales para montaje en Placa de Circuito Impreso

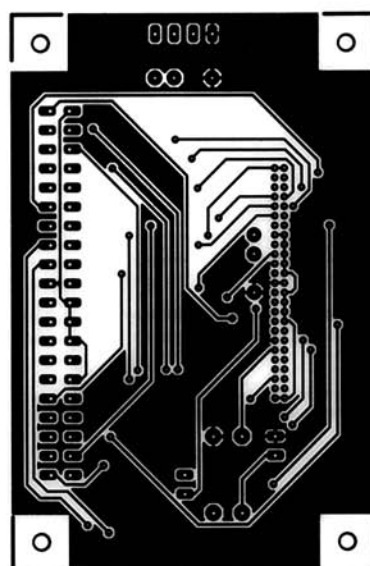
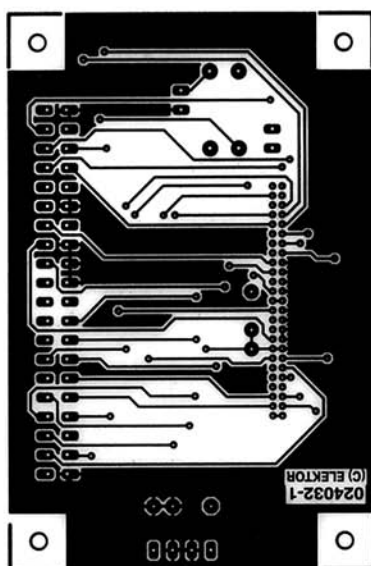
K2 = Conector "pinheader" SIL de 50 terminales en ángulo recto para montaje en placa de circuito impreso, con 0,05" de separación entre terminales (Farnell # 3078127)

JP1 = Conector "pinheader" de 2 terminales con puente de configuración

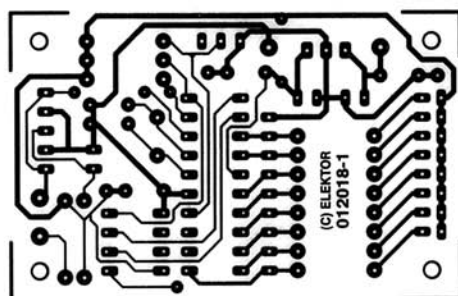
Cable alargador de tensión de alimentación de PC para PCB, con código de pedido **Nº: 024032-1** (ver página del servicio de lectores)

Figura 3. La placa de circuito impreso de doble cara facilita el montaje del circuito, especialmente en lo que se refiere al montaje del conector K2.

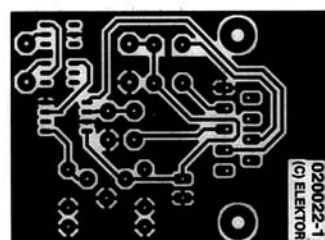
(024032-1)



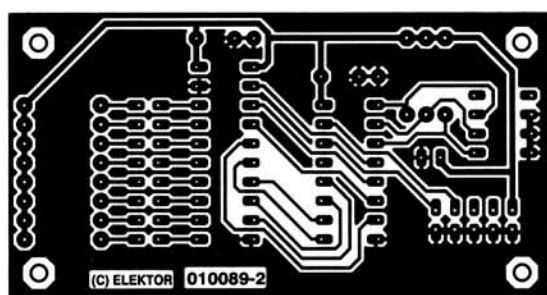
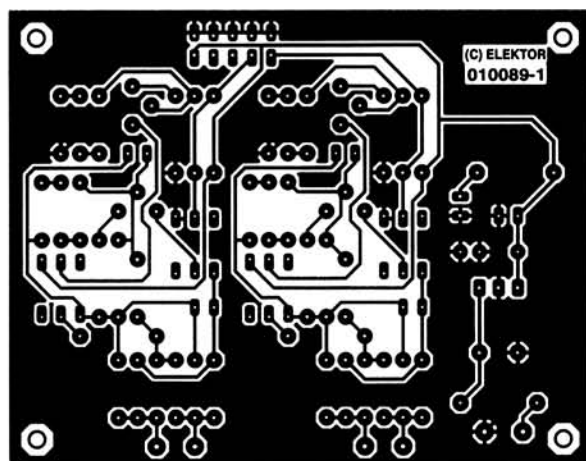
EPS024032-1
Controlador de CompactFlash para Bus IDE



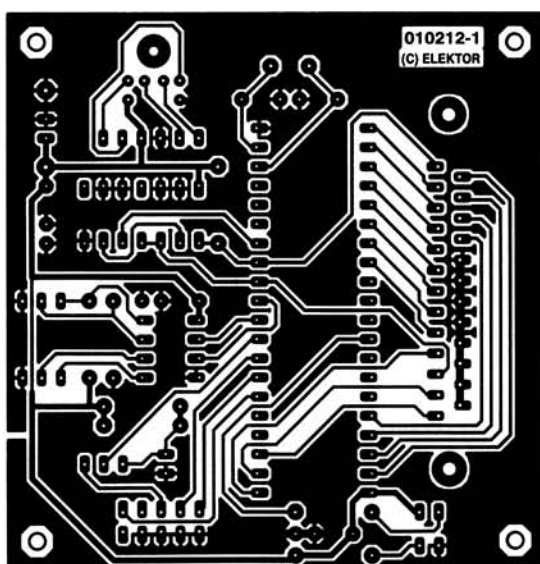
EPS012018-1
Receptor de Infrarrojos
Multi-estándar



EPS020022-1
Interfaz Serie para el
Bus 1-Wire de Dallas



EPS010089-1
Interface I2C para Bloque Lego RCX



EPS020202-1
Interface LPT/DMX