PROGRAMADOR

Con una intervención mínima sobre la estructura del hardware y la utilización de un código estándar base se pueden desarrollar un enorme número de dispositivos electrónicos utilizando lógica programable. La utilización del soldador y de los componentes tradicionales para la realización de circuitos digitales puede pasar a la historia utilizando el dispositivo que presentamos en estas páginas.

PLD: CPLD y FPGA

El acrónimo **PLD** (**P**rogrammable **L**ogic **D**evice) hace referencia a circuitos integrados digitales programables. La principal característica de este tipo de circuitos es que no se han diseñado para realizar una determinada función lógica sino que pueden ser **programados** para implementar **cualquier función**, más o menos compleja.

Puesto que se trata de productos de **propósito general** no tienen una aplicación única y específica, son **muy flexibles**. Eso sí, suelen algo **más lentos** y disipan **más potencia** que un producto diseñado para desarrollar una **función específica**.

A pesar de esto, por cuestiones principalmente económicas y para enfrentarse a un mercado tecnológicamente cada vez más exigente, esta categoría de productos ha ido poco a poco creciendo hasta alcanzar una **gran cuota de mercado**, en algunos casos incluso **superior** a la de los **microcontroladores**.

En efecto, en los últimos años se ha hecho cada vez más necesaria la posibilidad de poder **reprogramar un circuito** o un **componente individual** para desarrollar distintas funciones **sin modificar** físicamente el **hardware**.

Para responder a esta exigencia se desarrollaron los **microcontroladores**, como las familias **ST6**, **ST7** y **PIC**. Estos dispositivos son reprogramados para ejecutar programas de aplicaciones específicas.

De forma paralela también se están desarrollando muchos **dispositivos de lógica programable**. La diferencia fundamental es que en estos integrados **no se ejecuta** un **código** sobre un hardware predefinido, como en un microcontrolador, sino que se **reconfigura el hardware** interno del integrado en función de las necesidades (mediante un **código**).

Por esta razón en entornos **PLD** son casi equivalentes las palabras **código** y **circuito**, ya que se utiliza código "para dar forma al circuito". La escritura del código provoca que se **conecten** físicamente **cables** y **componentes lógicos** dentro del circuito integrado **PLD**.

Por ejemplo, si se quiere construir un **contador de 12 bits**, con 3 biestables, 12 puertas AND y decodificadores para los displays de 7 segmentos, habría que diseñar y realizar un circuito impreso específico, adquirir los componentes y montarlos en el impreso. Si una vez realizado el circuito se precisara en su lugar un **contador de 16 bits** habría que **tirar** casi todo, **rediseñar** el circuito, **comprar** componentes y **montarlos** de nuevo.

Utilizando **PLDs** este tipo de problemas no aparecerán. En efecto, basta con implementar el circuito deseado con un **software de desarrollo** instalado en un **PC** corriente y **programar** el dispositivo. Si la tipología, el número o la disposición de los componentes cambiara en el futuro bastará con **reprogramar** el dispositivo.

Los **PLDs** tienen, en general, una **elevada capacidad** y un considerable número de **terminales**. Para tener una idea preliminar, en el circuito que proponemos la programación utilizada en el ejemplo utiliza en torno a un **10%** de su capacidad lógica.

Otra característica común a todos los PLD es que **absorben poca corriente** y **cuestan** relativamente **poco**.

Un aspecto fundamental son los **lenguajes de programación** utilizados. Como veremos en próximos artículos se pueden utilizar **métodos gráficos** o **esquemáticos**, incorporando funciones lógicas predefinidas interconectadas a gusto de cada uno. Los lenguajes de programación hardware de alto nivel como **VHDL** (**VHSIC H**ardware **D**escription **L**anguage) permiten una completa **portabilidad del código** generado y una **independencia completa** del **hardware**.

Estas cuestiones implican los siguientes aspectos:

1) El código escrito para el PLD que proponemos puede programar **cualquier circuito PLD** de cualquier fabricante, obviamente siempre que tenga la suficiente lógica y que utilice código estándar.

para dispositivos CPLD

2) Utilizando PLDs **nadie podrá copiar** completamente vuestros circuitos puesto que es necesario el código que programe el microchip.

3) Una vez aprendido el lenguaje y la filosofía base del código se pueden **realizar todo tipo de aplicaciones**. En muchas de ellas no es necesario utilizar microcontroladores y aprender Assembler, utilizando PLDs es más que suficiente.

Fig.1 El programa Quartus II, de la compañía Altera, es el sistema integrado de desarrollo para CPLD que proponemos.

GLOSARIO de TÉRMINOS COMÚNMENTE UTILIZADOS

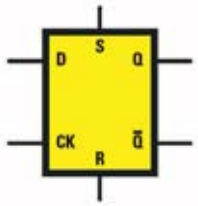
Para evitar que el artículo contenga excesivas indicaciones hemos recogido en este **mini glosario** los términos comúnmente utilizados en relación con el diseño asistido por ordenador y con la electrónica digital. De esta forma el artículo contiene toda la información necesaria para su completa compresión y se hace más sencilla su lectura.

CAD (**C**omputer **A**ided **D**esign): Este acrónimo referencia a las técnicas y a los programas que permiten a los usuarios realizar proyectos, normalmente relacionados con la ingeniería. En nuestro caso proyectos de circuitos electrónicos.

Clock: Señal de onda cuadrada con una frecuencia precisa utilizada para sincronizar los circuitos.

En nuestro caso la señal reloj (clock) es una señal de onda cuadrada con un duty cycle del 50%. Es indispensable para la correcta sincronización entre el programador y el microchip.

Flip-flop o **Biestable**: Elemento de **memoria** que puede mantener indefinidamente en el tiempo un valor, siempre y cuando esté alimentado. El valor almacenado se actualiza con el valor presente a su entrada en correspondencia con una señal de control (normalmente se utiliza un frente de la señal de clock).



Biestable tipo D. Cuando a su terminal CK llega un frente del reloj (clock) los datos presentes en sus salidas (Q y Q̄) se actualizan con el valor presente en su terminal de entrada D. La salida Q̄ tiene un nivel lógico opuesto a la salida Q.

La actualización del valor de las salidas se suele denominar **muestreo**, por tanto la señal de salida es la señal de entrada muestreada.

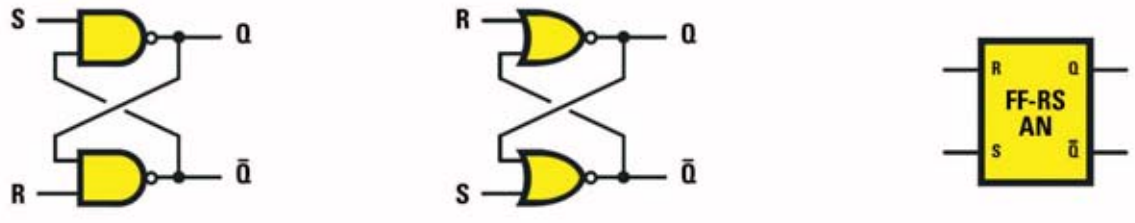
En general un biestable puede ser inicializado, presentar un valor predefinido en la salida antes del primer muestreo, reseteado, presentar en la salida un valor forzado de la entrada principal, habilitado, deshabilitado o sencillamente puede muestrear. Cuando está deshabilitado la salida presenta el último valor muestreado.

A menudo también dispone de una **salida negada** que se identifica con la misma referencia que la **salida normal** pero **suprarayado**.

Existen muchos tipos de biestables, si bien el más común para **almacenamiento** de información es el biestable tipo **D**.

I/O (**Input/Output**): Señales que comunican un bloque con el mundo exterior a éste.

Lógica: Se suele sobrentender que lleva implícito el adjetivo **booleana**, es decir sistema basado en dos posibles valores (**verdadero** o **falso**). A estos valores se les suelen asociar términos numéricos para manipularlos con ex-

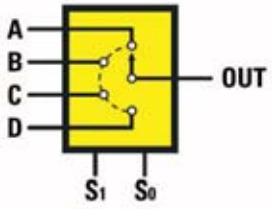


El circuito de la izquierda es un biestale Set-Reset realizado con puertas NAND. El circuito central también es un biestable Set-Reset, pero realizado con puertas NOR. A igualdad de niveles lógicos en las entradas, la puerta lógica NAND y la puerta lógica NOR presentan en la salida, respectivamente, niveles lógicos opuestos a los niveles de las puertas lógicas AND y OR.

presiones matemáticas, comúnmente se asocia **1** a verdadero y **0** a falso.

Multiplexor: Componente que tiene dos o más entradas principales, una única salida y un número de entradas de control dependiente del número de entradas principales. Con las señales de **control** se **selecciona** una única **entrada**, cuyo valor se lleva a la salida.

Si, por ejemplo, se tienen **4 entradas principales** son necesarias **2 señales de control** para **seleccionar** una de las entradas. Cuando las señales de control toman el valor "00" se lleva a la salida el valor presente en la primera entrada, con "01" el valor presente en la segunda entrada, con "10" el valor presente en la tercera entrada y con "11" el valor presente en la cuarta entrada.



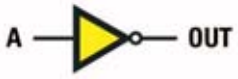
Esquema simplificado de un multiplexor de 4 entradas. En función de los niveles lógicos presentes en las líneas de control (S1-S0) la salida queda cortocircuitada con una de las entradas (A-B-C-D).

Puerta AND: Puerta lógica que tiene una única salida y dos o más entradas. La salida solo toma el valor "1" cuando **TODAS** las entradas tienen un nivel "1", si no es así la salida toma valor "0".

Puerta NOT: Puerta lógica que tiene una única salida y una única entrada. El valor de la salida es el **inverso** del valor de la entrada, es decir la salida

La salida de una puerta lógica AND solo tiene valor 1 cuando todas las entradas están a nivel lógico 1.

toma valor "1" cuando la entrada tiene valor "0" y toma valor "0" cuando la entrada tiene valor "1".



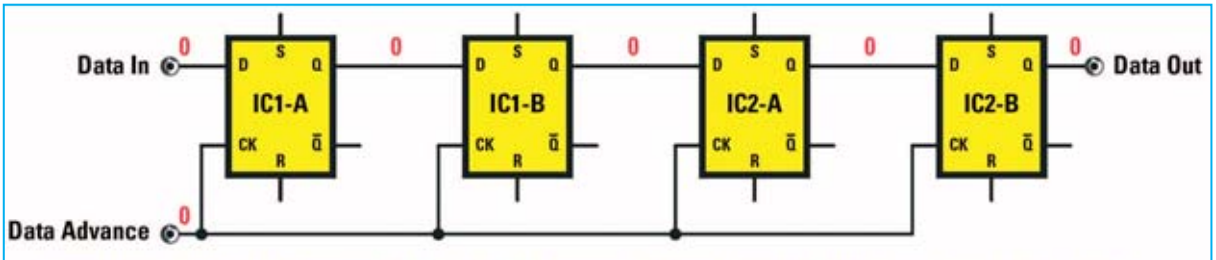
A la puerta lógica NOT también se la denomina INVERSOR, ya que el nivel lógico de la salida es el inverso del nivel lógico de la entrada.

Puerta OR: Puerta lógica que tiene una única salida y dos o más entradas. La salida solo toma el valor "1" cuando **AL MENOS UNA** las entradas tiene un nivel "1", si no es así la salida toma valor "0".

La salida de una puerta lógica OR solo tiene valor 1 cuando al menos una de las entradas está a nivel lógico 1.

Señal: Una variación de tensión o de corriente en un conductor. Si no indicamos lo contrario nos referiremos a **señales digitales binarias**, es decir a señales que solo pueden tomar dos valores (**0 voltios** o **Vcc**).

Los cambios de valor (de 0 a Vcc o de Vcc a 0) se denominan **frentes** o **flancos**. Los dos valores de tensión se asocian a los valores lógicos "**0**" y "**1**", que representan un **bit** (**b**inary digi**t**).



Conectando la salida de un biestable a la entrada del siguiente a cada impulso de reloj (clock) el dato se lleva al biestable siguiente. El registro de desplazamiento aquí representado está formado por 4 biestables conectados en serie, por lo que el dato de entrada (Data In) se presenta a la salida (Data Out) después de cuatro impulsos de reloj (Data Advance).

Se denomina frente o flanco de **subida** al paso de "**0**" a "**1**" y frente o flanco de **bajada** al paso de "**1**" a "**0**".

Shift-register (**Registro de desplazamiento**): Grupo de **registros** conectados en **serie**, con la salida del anterior conectada a la entrada del siguiente. A cada pulso de reloj (clock) el dato almacenado en un flip-flop pasa al siguiente de la cadena.

Se puede utilizar, por ejemplo, para **retardar** un dato un cierto número de pulsos de reloj (clock), tantos como el número de registros conectados en serie.

En el ejemplo mostrado en esta misma página "**Data in**" aparecerá en la salida "**Data out**" después de cuatro impulsos de reloj "**Data Advance**".

Vcc: En general, tensión de alimentación. En nuestra tarjeta tiene un valor de **3,3 voltios**.

CPLD y FPGA

Hay muchos y diferentes **PLDs** disponibles comercialmente, desde los más sencillos y antiguos, como las **PAL** (**P**rogrammable **A**rray **L**ogic) y las **GAL** (**G**eneric **A**rray **L**ogic) a los más complejos y de última generación, como los **CPLD** (**C**omplex **P**rogrammable **L**ogic **D**evice) y las **FPGA** (**F**ield **P**rogrammable **G**ate **A**rray). Sobre estos últimos basamos este y próximos artículos.

Las **PAL** contienen **filas** de puertas **AND** y **OR**, elementos fundamentales de cualquier función lógica, con **interconexiones programables** para realizar circuitos lógicos combinacionales, **sin** disponibilidad de **biestables**, ni por lo tanto de realización de **circuitos secuenciales**. Las **GAL** son sencillamente **PAL** borrables y **reprogramables**.

Los diferentes fabricantes han ido desarrollando arquitecturas cada vez más potentes y funcionales, que frecuentemente unen características de diversas tecnologías, lo que hace difícil trazar la frontera entre un chip **CPLD** y **FPGA**.

Los **CPLD** retoman la arquitectura de las **PAL**, con múltiples **filas** de **OR** y **AND**, añadiendo **registros** que pueden ser excluidos, **modos avanzados** de **interconexión** y, a menudo, una **memoria interna no volátil** para el almacenamiento del código o simplemente a disposición del usuario.

Disponen de potentes **bloques** de **lógica combinatoria LE** (**L**ogic **E**lements) integrados en una red de **canales predefinidos de interconexión** con **pequeños retardos controlados** para permitir la implementación de complejos esquemas lógicos **secuenciales** de **alta velocidad**.

Por este motivo la densidad de integración de estos componentes no logra alcanzar niveles muy elevados. Los chips **CPLD** comerciales integran hasta unos **2.000 LE**, para capacidades superiores hay que pasar a las **FPGA**.

Las **FPGA** están basadas en **bloques lógicos LC** (**L**ogic **C**ells), elementos con menor potencialidad que los **LE** de los **CPLD**, inmersos en una compleja r**ed de interconexión** más o menos **segmentada**.

El bloque lógico **varía** bastante entre diversos **fabricantes**, pudiendo ser implementado con arquitecturas radicalmente diferentes.

Entre estas arquitecturas está la basada en bloques compuestos por una **LUT** (**L**ook-**U**p **T**able) de **4 entradas** y **1 salida**, elemento que tiene el aspecto de una **tabla de la verdad** booleana con **4 bits de entrada** y **1 bit de salida**, más un **registro** eventualmente excluible.

Estos bloques son **más pequeños** que los **LE** de los **CPLD**, por lo que se puede integrar en un mismo chip un número bastante **mayor** de bloques.

Es bastante frecuente que los chips **FPGA** dispongan de una **memoria volátil** y de una **memoria no volátil**, incluso en los más avanzados se puede encontrar integrado un **microcontrolador**.

Las capacidades de integración pueden llegar a más de **100.000 LC**. De hecho con esta tecnología se pueden implementar **microprocesadores** para **ordenadores personales**.

Como confirmación del hecho que supone la poco visible frontera entre los chips **CPLD** y **FPGA**, el chip que presentamos aquí pertenece la categoría **CPLD**, pero además dispone de funciones **LUT** de **cuatro entradas**, una característica normalmente relacionada con las **FPGA**.

ALTERA MAX II EPM240T100C5N

El dispositivo presente en la **Tarjeta de prueba LX.1686** pertenece a la categoría de los **CPLD**. Se trata del **MAX II EPM240T100C5N**, fabricado por **Altera**, empresa líder en el sector de los circuitos de lógica programable.

CARACTERÍSTICAS PRINCIPALES

El microchip está construido con tecnología de **0,18 µm**, **6 niveles** de **metalización interna** y tiene una capacidad de **240 LE**. Aunque es el más pequeño de su categoría dispone de un elevado número de **terminales** (**100**), de los que **80** están a disposición del usuario, divididos en **seis bancos**.

Entre los terminales hay algunos que pueden utilizarse como terminales normales o bien explotando una **función específica**. Hay cuatro con una distribución interna "en árbol" para alcanzar todos los puntos del microchip con pequeños **retardos**, muy útiles, por ejemplo, para transportar varias señales de reloj. Hay un terminal de reinicio (**reset**) que **borra** todos los registros internos y un terminal que pone todas las salidas en estado de **alta impedancia** (**output-enable**). La disposición de todos los terminales se muestra en la Fig.2.

Se trata de un microchip **muy rápido**, no es difícil superar los **200 MHz**. Para **nuestros propósitos** hemos utilizado un cuarzo de **20 MHz**, valor más que suficiente para una introducción a este nuevo mundo y muy lejos de su límite de funcionamiento. De esta forma incluso un circuito no muy bien optimizado podrá funcionar sin problemas.

Posee **dos memorias internas no volátiles** que mantienen los datos en ausencia de alimentación: La **CFM** (**C**onfiguration **F**lash **M**emory) y la **UFM** (**U**ser **F**lash **M**emory).

La primera se utiliza para almacenar la **programación** de la tarjeta mientras que la segunda, con una capacidad de **8.197 bits**, se utiliza como **memoria de usuario** no volátil con un **ancho** de palabra **programable** hasta **16 bits**.

Fig.2 Disposición de terminales del microchip MAX II EPM240T100C5N producido por Altera. Se trata de un chip con lógica programable que dispone de 100 terminales de los cuales 80 están disponibles para I/O. Los terminales dedicados a la programación son 4: Terminal 22 (línea de control TMS), terminal 24 (clock TCK) y terminales 23-25 (programación y control según el estándar JTAG).

Gracias a la **CFM** no se necesita una memoria externa para la programación. El **MAX II** es **autosuficiente**, está disponible inmediatamente y en cualquier momento, **manteniendo** los **datos** en **ausencia** de **alimentación**.

Permite **programación ISP** (**I**n **S**ystem **P**rogrammability), es decir se puede programar **sin modificar el hardware** y **mientras está funcionando** una programación anterior diferente a la nueva.

El **interior** del microchip funciona con una tensión de **1,8 voltios**, ahora bien puesto que dispone de reguladores internos ha de **alimentarse externamente** con una tensión no mayor de **3,3 voltios**, justo la tensión a la que opera nuestra tarjeta.

Los **terminales de salida** pueden proporcionar una **corriente máxima** de **25 mA**, permitiendo, por ejemplo, el encendido de un **diodo LED** sin necesidad de amplificación externa.

Es importante tener presente que no se han de superar **130 mA por banco**. Si, por ejemplo, quisiéramos **15 mA** por terminal, en un **banco** solo podríamos utilizar **8 terminales** para no poner en riesgo el microchip.

El aparato está programado en modo **JTAG** (**J**oint **T**est **A**ction **G**roup) por nuestro **programador LX.1685** para conectar el **puerto paralelo** de un PC. Utiliza **4 terminales** para la programación de este modo (terminales **22-23-24-25**).

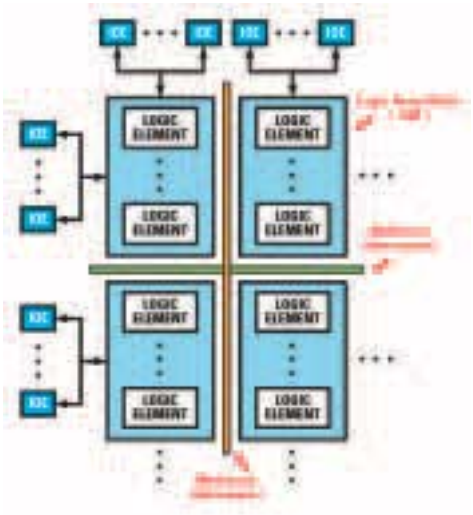


Fig.3 Esquema de bloques interno del MAX II. Nuestro dispositivo tiene una estructura de matriz basada en el direccionamiento de 4 líneas y 6 columnas que implementan un total de 240 LE.

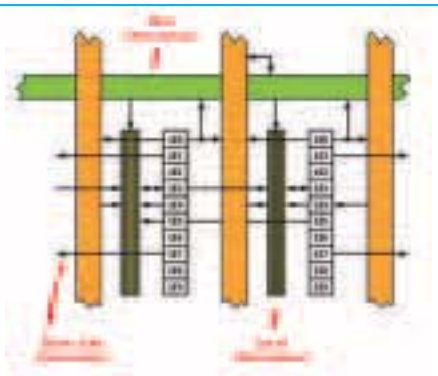


Fig.4 Cada LAB contiene 10 LE que se interconectan gracias a una red local (Local Interconnect) más rápida que los grandes buses utilizados para realizar conexiones entre distintos LAB situados a gran distancia entre sí (Row Interconnect).

ESTRUCTURA INTERNA

El **esquema de bloques interno** se muestra en la Fig.3. Aunque para programarlo correctamente no es necesario conocer su estructura interna, al menos hasta que se realicen circuitos de cierta complejidad, es de mucha utilidad para entender como funcionan en general las lógicas programables tomando como base nuestro dispositivo.

Para implementar la función deseada el **MAX II** dispone de una estructura de **matrices** basada en el **direccionamiento** de **líneas** y **columnas** (**MultiTrack Interconnect**) para conectar los **L**ogic **A**rray **B**locks (**LAB**) que contienen cada uno **diez** **L**ogic **E**lements (**LE**).

En nuestro dispositivo se utilizan **6 columnas** y **4 líneas** para un total de **24 LAB**, esto es **240 LE**. En el exterior se encuentran los **IOE** (**I/O E**lements), **buffers bidireccionales** para los terminales de entrada/salida que ofrecen varios modos de funcionamiento (**trigger Schmitt**, niveles lógicos **TTL**, niveles lógicos **CMOS**, etc.).

En la Fig.4 se muestra la estructura de un **LAB**. Se pueden observar los **diez LE** interconectados entre sí gracias a una **red local** muy veloz en la que se encuentran las señales de **control** para las conexiones en **cascada**.

Por ejemplo, si una función lógica necesita utilizar **4 LE** en cascada para ser implementada, en vez de utilizar las **grandes líneas globales** que pueden conectar los **LAB** a gran distancia (representadas en **verde** y en **naranja** en la Fig.4) se pueden utilizar las conexiones locales más rápidas.

Además dispone de **conexiones locales** entre **LAB adyacentes** (**DirectLink**), y también para los **terminales exteriores**.

Cada **LE** puede controlar otros **30 LE** situados en **LAB adyacentes** más los **10** contenidos en el propio **LAB**.

El **programa** de **compilación** y **programación** explotará automáticamente las características de la arquitectura implementando en bloques adyacentes las funciones más grandes y que necesitan mayor lógica, **optimizando** así los retardos de propagación de la señal y aumentando de esta forma la frecuencia a la que puede funcionar el circuito.

Cada **LAB** puede tener a lo sumo **26 posibles entradas** más otras **10** procedentes de la salida de cada **LE**. Además, hay toda una serie de señales de control global dedicado (**clock**, **reset**, **enable**, etc.).

LE: LOGIC ELEMENTS

Profundizando más en el nivel de abstracción nos encontramos la más pequeña **unidad lógica** de la arquitectura del **MAX II**, la **base** en la que se implementan las **funciones lógicas** descritas a alto nivel con **VHDL** o con el **esquemático**.

Analizando esta estructura se puede entender como es posible la **transformación** del **código** escrito en un **PC** en **hardware**.

El punto de partida es el esquema mostrado en la Fig.5. El bloque principal es la **L**ook-**U**p **T**able (**LUT**) de **cuatro entradas**, capaz de implementar cualquier función de cuatro variables de entrada y una de salida simplemente realizando su **tabla de la verdad**.

Para entender el funcionamiento tomemos, por ejemplo, una **LUT** de **dos entradas** programadas para desarrollar la sencilla función **AND** con **dos variables**. La tabla de la verdad de una AND es la siguiente:

Entrada B	Entrada A	Salida (OUT)
0	0	0
0	1	0
1	0	0
1	1	1

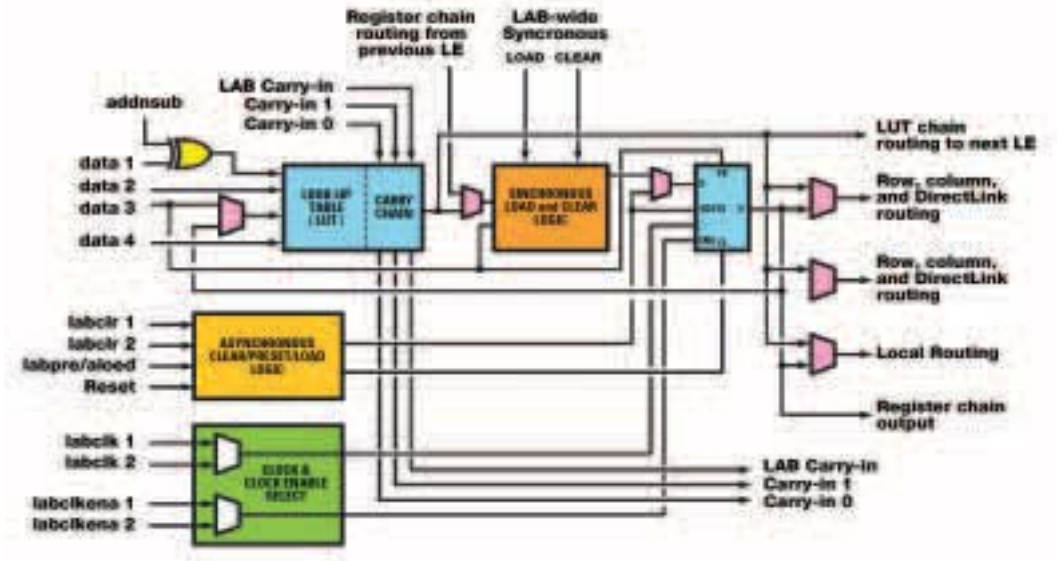


Fig.5 Esquema de bloques de un LE del MAX II. El bloque principal es una LUT de cuatro entradas. Los multiplexores y las interconexiones están controlados por la memoria de configuración interna, que se programa desde un PC. El usuario solo tiene que describir el funcionamiento del circuito mediante código, es el compilador el encargado de seleccionar el modo más adecuado para implementarlo.

Esto es exactamente lo que hay que programar en la **LUT** para desarrollar la función.

Extendiendo el concepto se puede entender como una **LUT** de **cuatro entradas** puede expresar cualquier función de **cuatro variables** y una salida según su **tabla de la verdad** que será extrapolada por el **compilador** a partir del **código escrito**.

En el ejemplo este podría ser el resultado de la compilación de una **línea de código**:

OUT = A and B

Gracias a las señales **carry-in**, **carry-out** y a las **funciones disponibles** para cada **LUT** es posible ampliar la complejidad del circuito a funciones de **más** de **cuatro variables** y **más** de **una salida**.

En la **LUT** hay un completo **registro programable** que incluye la lógica necesaria para las funciones de **reset**, **preset** y **enable**, gracias a las cuales, como veremos, es posible crear **lógica secuencial** y **máquinas de estados**.

Todos los bloques restantes son **multiplexores**, necesarios para realizar la **programación** del **LE** y para realizar las **conexiones** con el exterior a través de la **red local** o **global**.

Por ejemplo, el multiplexor situado **bajo el registro** sirve bien para **excluirlo**, y, por lo tanto, para hacer pasar el dato directamente por la **LUT** a otro bloque, o bien para **incluirlo**.

Todos estos multiplexores e interconexiones están controlados por la **memoria de configuración** interna del **MAX II**, es decir la memoria que programaremos con el PC.

Es importante tener presente que el modo en el que se programan los controles es prácticamente **invisible al usuario**, ya que solo se tiene que preocupar de describir correctamente el funcionamiento de su circuito mediante el **código**. El **compilador** interpretará el código y a **elegirá** el **modo más adecuado** para implementarlo.

No obstante también es posible **intervenir manualmente**, por ejemplo eligiendo la disposición de los **LE** en el circuito. Este tipo de actuaciones solo se hacen para **optimizar en profundidad** el proyecto y una vez que se tiene un gran dominio del entorno.

Para terminar indicamos que cada **LE** tiene **dos modos de funcionamiento** (**normal** y **aritmética dinámica**). El modo para una aplicación concreta **es elegido** por el **compilador** en base al código a implementar.

El **modo normal** es el más común, implementa funciones generales. El **modo aritmética dinámica** es útil cuando se trata de implementar funciones aritméticas como sumas, diferencias, contadores, acumuladores, comparadores, con un **grado de paralelismo superior a uno**.

ESQUEMA ELÉCTRICO del PROGRAMADOR

Como se puede observar en el esquema eléctrico de la Fig.7 el **Programador para CPLD** utiliza el **puerto paralelo** para comunicarse con el ordenador (ver CONN.1).

NOTA: Quienes deseen profundizar en el funcionamiento del **puerto paralelo** del **PC** pue-

Fig.6 Conexiones, vistas desde arriba, de los integrados TTL utilizados en el Programador para CPLD LX.1685. También se muestran las conexiones de un diodo LED.

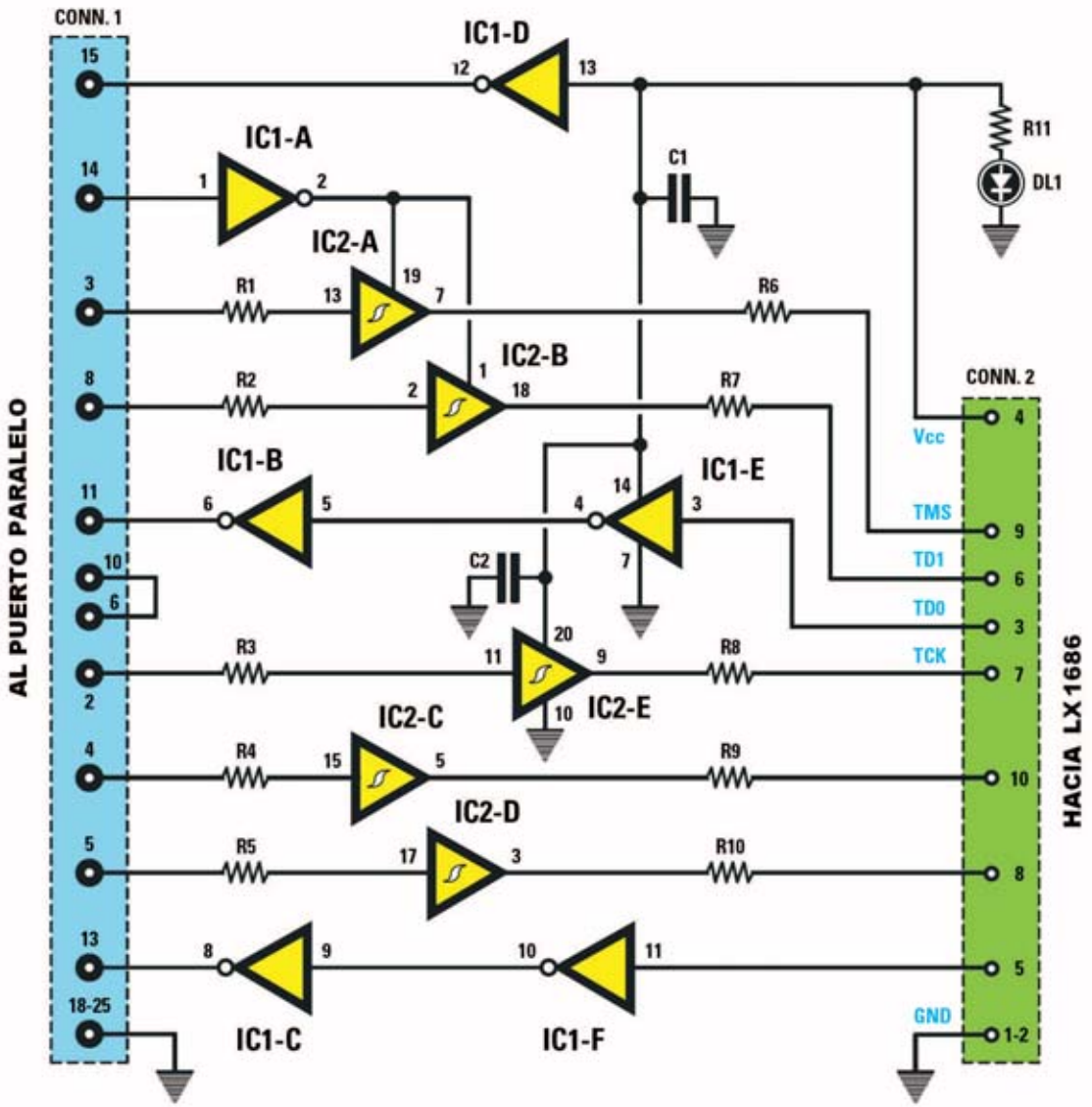


Fig.7 Esquema eléctrico del Programador LX.1685. Explotando las prestaciones del puerto paralelo de un PC, que puede utilizarse como multicanal serie, es posible utilizar el protocolo de comunicación estándar JTAG. Esta placa recibe la tensión de alimentación (3,3 voltios) de la Tarjeta de prueba LX.1686 a través del terminal 4 del conector CONN.2.

LISTA DE COMPONENTES LX.1685

R1 = 100 ohmios
R2 = 100 ohmios
R3 = 100 ohmios
R4 = 100 ohmios
R5 = 100 ohmios
R6 = 33 ohmios
R7 = 33 ohmios
R8 = 33 ohmios
R9 = 33 ohmios
R10 = 33 ohmios
R11 = 470 ohmios
C1 = 100.000 pF poliéster
C2 = 100.000 pF poliéster
DL1 = Diodo LED
IC1 = Integrado TTL 74HC04
IC2 = Integrado TTL 74HC244
CONN.1 = Conector 25 terminales
CONN.2 = Conector 10 terminales

NOTA: Todas las resistencias utilizadas en este circuito son de 1/4 vatio.

den leer el artículo asociado al **Téster para puerto paralelo** publicado en la **revista Nº241**.

Con el software adecuado es posible transformar **cada línea** de datos del puerto (**D0**-**D7**) en una **conexión serie individual**. Un ejemplo de esta aplicación es el software de gestión para el **Excitador FM 88-108 MHz KM.1619** presentado en la **revista Nº247**.

La **alimentación** de los dos integrados del programador se realiza a través de la **Tarjeta de prueba**, cuyo funcionamiento se describe posteriormente en este mismo artículo, a través del **terminal 4** (**Vcc**) del conector **CONN.2**. Desde el mismo punto se alimenta el diodo LED **DL1**, cuya función es indicar, mediante su encendido, que el programador está alimentado correctamente.

El **protocolo de comunicación** entre el **programa** proporcionado por **Altana** y el **CPLD** está definido por el **estándar JTAG** que, sin profundizar en detalles, prevé la **programa-**

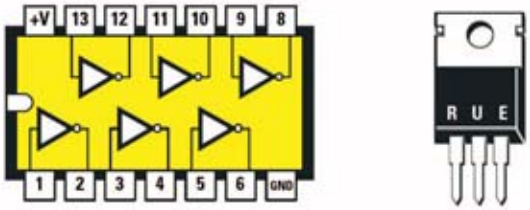


Fig.8 Conexiones, vistas desde arriba, del integrado TTL 74HC04, que, junto al cuarzo, constituyen el oscilador para el reloj interno del CPLD. Las conexiones del integrado LM.317, que junto a las resistencias R1-R2-R3 estabilizan la tensión a 3,3 voltios, se muestran vistas frontalmente.

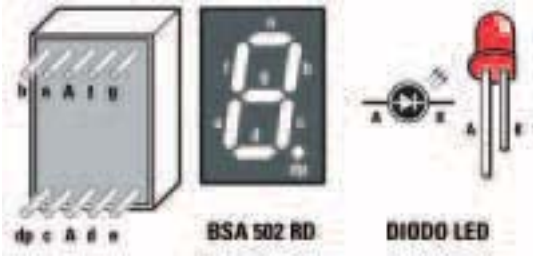


Fig.9 Conexiones, vistas por detrás, del display de 7 segmentos BSA 502 RD. El terminal más corto de los diodos LED (cátodo) se conecta al CPLD mediante una resistencia limitadora.

LISTA DE COMPONENTES LX.1686

R1 = 220 ohmios
R2 = 180 ohmios
R3 = 180 ohmios
R4 = 100 ohmios
R5 = 220 ohmios
R6 = 220 ohmios
R7 = 220 ohmios
R8 = 220 ohmios
R9 = 220 ohmios
R10 = 220 ohmios
R11 = 220 ohmios
R12 = 220 ohmios
R13 = 220 ohmios
R14 = 220 ohmios
R15 = 220 ohmios
R16 = 220 ohmios
R17 = 220 ohmios
R18 = 220 ohmios
R19 = 220 ohmios
R20 = 220 ohmios
R21 = 220 ohmios
R22 = 220 ohmios
R23 = 220 ohmios
R24 = 220 ohmios
R25 = 10.000 ohmios
R26 = 10.000 ohmios
R27 = 10.000 ohmios
R28 = 1 Megaohmio
R29 = 1.000 ohmios
R30 = 10.000 ohmios
R31 = 10.000 ohmios
R32 = 10.000 ohmios
R33 = 10.000 ohmios
R34 = 10.000 ohmios
C1 = 10 microF. electrolítico
C2 = 10 microF. electrolítico
C3 = 10 microF. electrolítico
C4 = 100.000 pF poliéster
C5 = 100.000 pF poliéster
C6 = 100.000 pF poliéster
C7 = 22 pF cerámico
C8 = 22 pF cerámico
C9 = 10.000 pF poliéster
C10 = 10.000 pF poliéster
C11 = 10.000 pF poliéster
C12 = 10.000 pF poliéster
C13 = 10.000 pF poliéster
DS1 = Diodo 1N.4007
DL1-DL4 = Diodos LED
DISPLAY1-2 = BSA 502 RD (ánodo común)
IC1 = Integrado LM.317
IC2 = Integrado TTL 74HC04
IC3 = CPLD MAX II EPM240T100C5N
XTAL1 = Cuarzo 20 MHz
P1-P5 = Pulsadores
BUZZER = Cápsula piezoeléctrica 12V
CONN.1 = Conector 10 terminales

NOTA: Todas las resistencias utilizadas en este circuito son de 1/4 vatio.

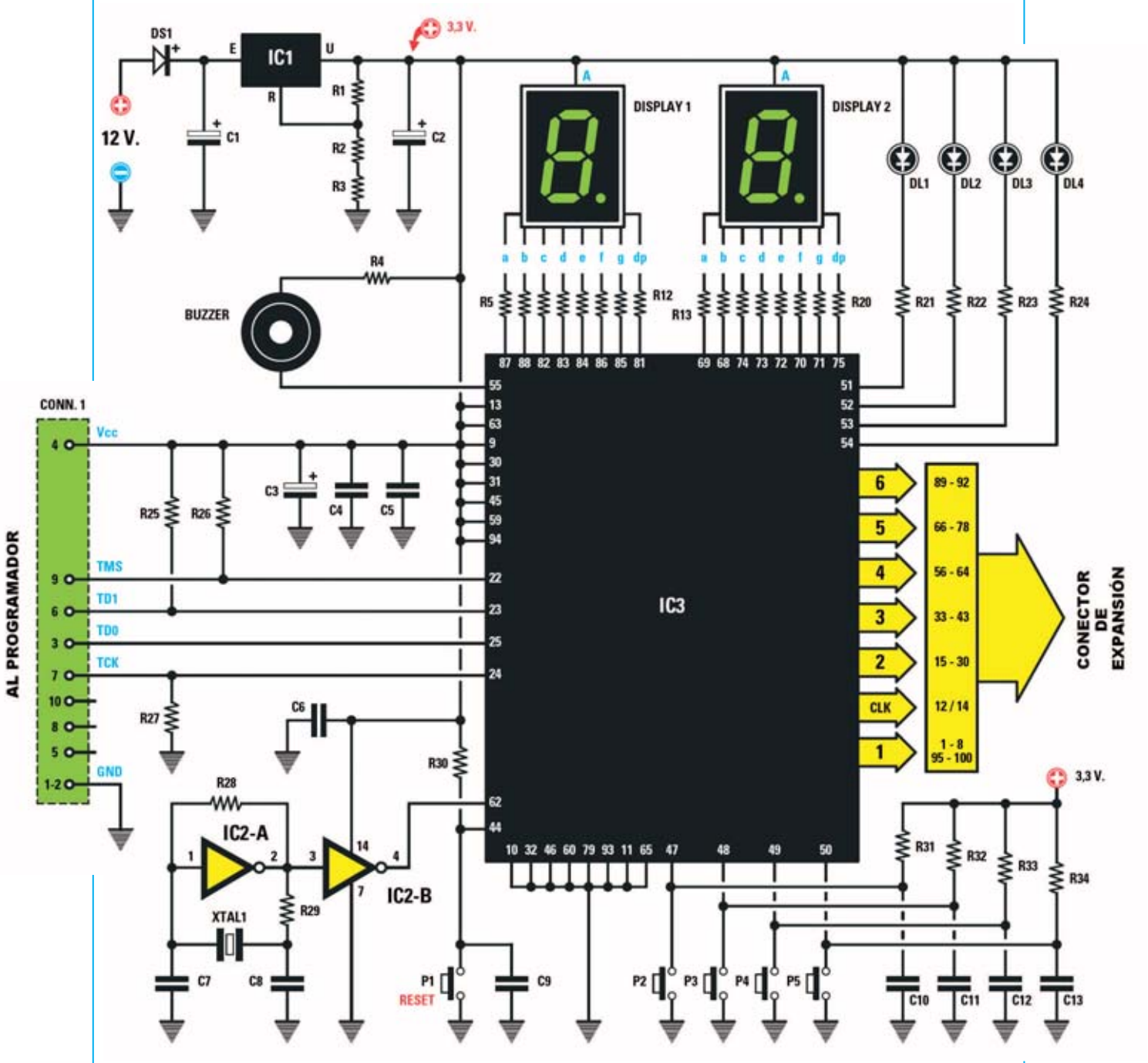


Fig.10 Esquema eléctrico de la Tarjeta de prueba LX.1686. El programa test.pof, incluido en el CD-ROM CDR1685, permite verificar la correcta programación del CPLD. Los pulsadores P2-P4 activan el buzzer, controlan los displays y los diodos LED. El pulsador P1, conectado al terminal 44 del CPLD, reinicia el microchip. Hemos previsto la instalación de un conector de expansión para vuestras futuras tarjetas.

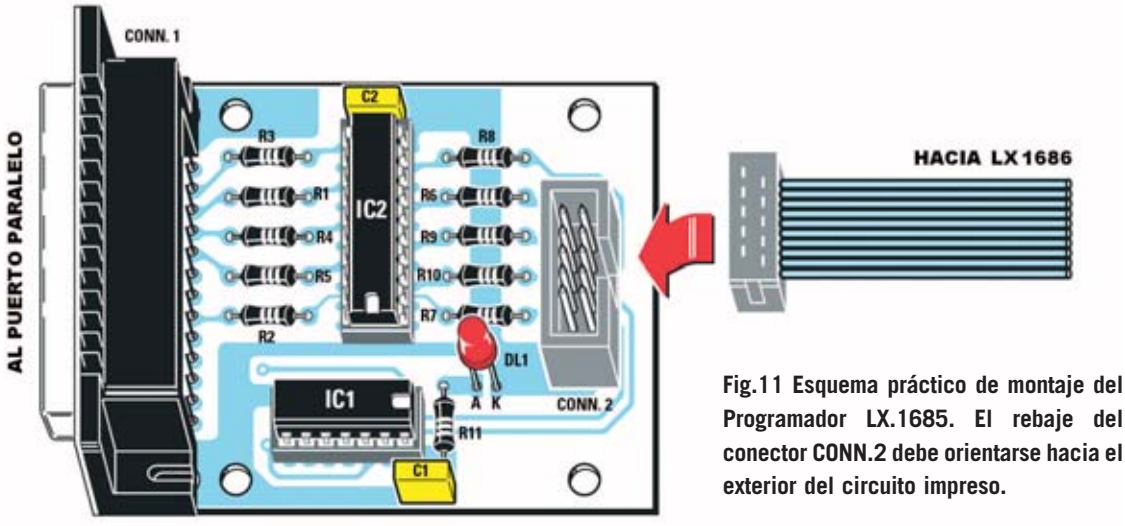


Fig.11 Esquema práctico de montaje del Programador LX.1685. El rebaje del conector CONN.2 debe orientarse hacia el exterior del circuito impreso.

ción serie de todos los registros del microchip por la línea **TDI**, con posibilidad de **verificar** el correcto funcionamiento interno del dispositivo gracias a la lectura de los mismos registros a través de la línea serie **TDO**.

El **handshaking**, es decir la correcta **sincronización** del programador y del microchip, se realiza a través de la línea **TCK** (**clock**) y la de línea **TMS** (**control**).

ESQUEMA ELÉCTRICO de la TARJETA de PRUEBA

El esquema eléctrico de esta tarjeta (ver Fig.10) está diseñado para soportar el **chip CPLD** y para proporcionar **entradas** y **salidas**, esto es, pulsadores, diodos LED (DL1-DL4), dos displays y un zumbador.

Para la **alimentación** del **CPLD**, y del **programador**, hemos utilizado un integrado **LM.317** (**IC1**) que estabiliza los **12 voltios** de entrada a **3,3 voltios** mediante el divisor formado por **R1**, **R2** y **R3**.

Los inversores **IC2/A** e **IC2/B** (incluidos en un integrado TTL **74HC04**) junto al cuarzo **XTAL1** forman el **oscilador** para el **reloj interno** del **CPLD**.

En el circuito impreso hemos previsto los taladros para instalar un **conector de expansión** que permita conectar vuestros **futuros proyectos**.

REALIZACIÓN PRÁCTICA LX.1685

Son realmente pocos los componentes a montar en el circuito impreso de doble cara **LX.1685** que soporta el **Programador** para **CPLD** (ver Fig.11).

Aconsejamos comenzar el montaje con la instalación de los **zócalos** para los circuitos integrados **IC1**-**IC2**, teniendo cuidado para no provocar cortocircuitos entre pistas adyacentes y orientando sus muescas de referencia en forma de **U** tal como se indica en el esquema de montaje práctico.

Acto seguido se pueden montar los **dos conectores**, **CONN.1**, un conector de **25 terminales** utilizado para la conexión al puerto paralelo del ordenador, y **CONN.2**, un conector de **10 terminales** utilizado para conectar la Tarjeta de prueba LX.1686.

Los terminales del conector **CONN.1** solo permiten su instalación en una **única posición**, en cambio **CONN.2** es **simétrico**. Para realizar adecuadamente su montaje hay que orientar su pequeño **rebaje** de referencia en forma de **U** hacia el **exterior** del circuito impreso (ver Fig.11).

Las **resistencias** incluidas en el kit son todas de **1/4 vatio** y tienen solo tres valores diferentes. Hay **5 resistencias** de **100 ohmios** (marrón-negro-marrón) que han de montarse en correspondencia con las referencias **R1** a **R5**, otras **5 resistencias** de **33 ohmios** (naranja-naranja-negro) que han de montarse en correspondencia con las

Fig.12 Fotografía del prototipo del Programador LX.1685 una vez montados todos sus componentes.

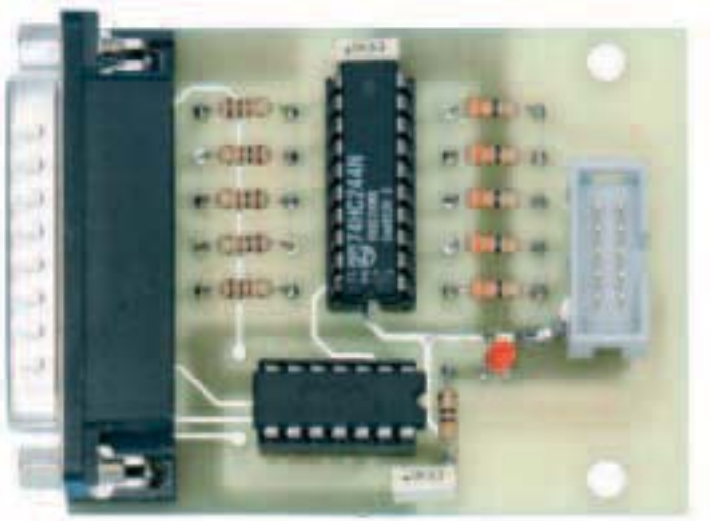
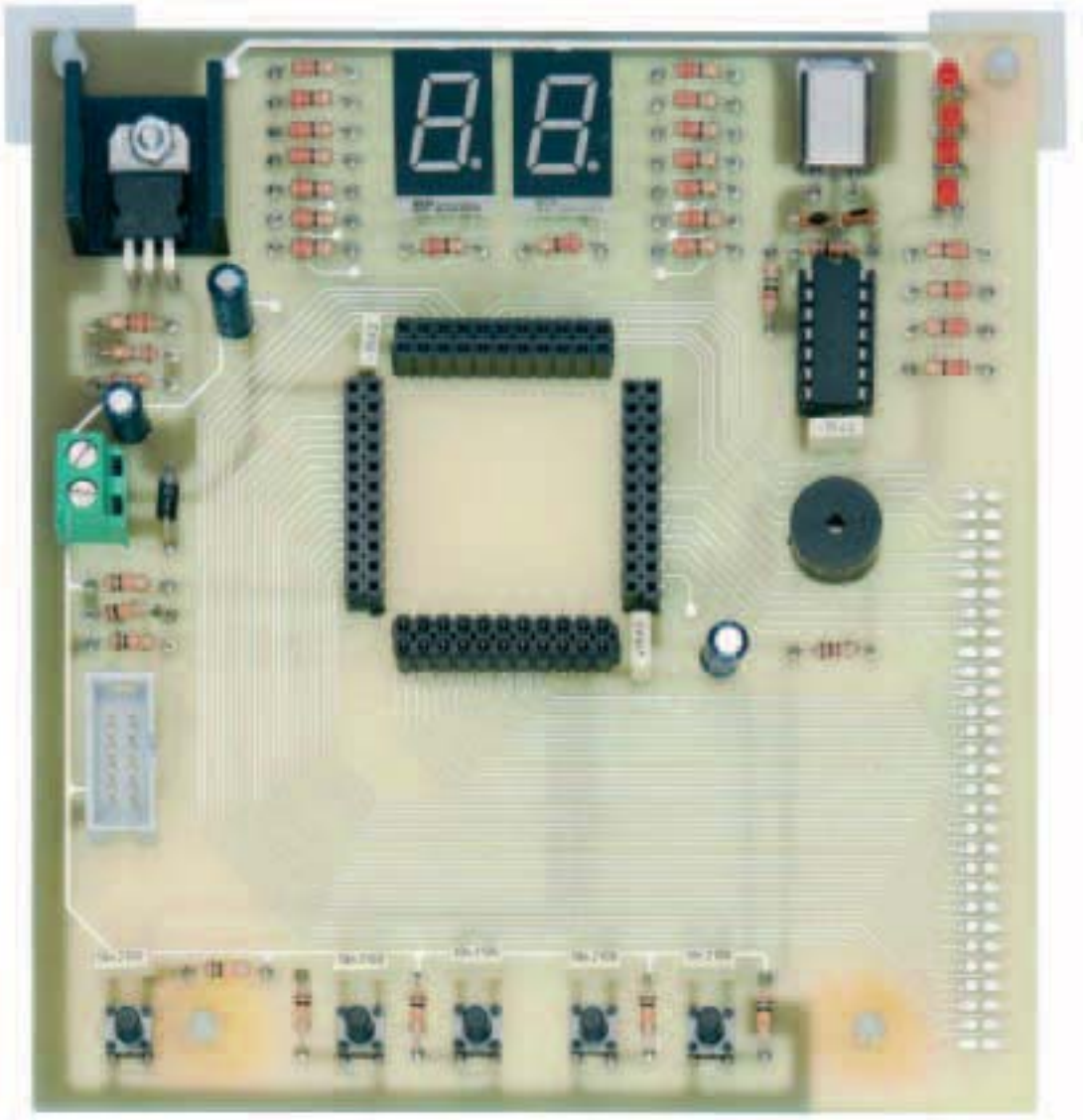


Fig.13 Fotografía del prototipo de la Tarjeta de prueba LX.1686 una vez montados todos sus componentes. Sobre esta tarjeta se conecta la placa premontada en SMD KM.1686.



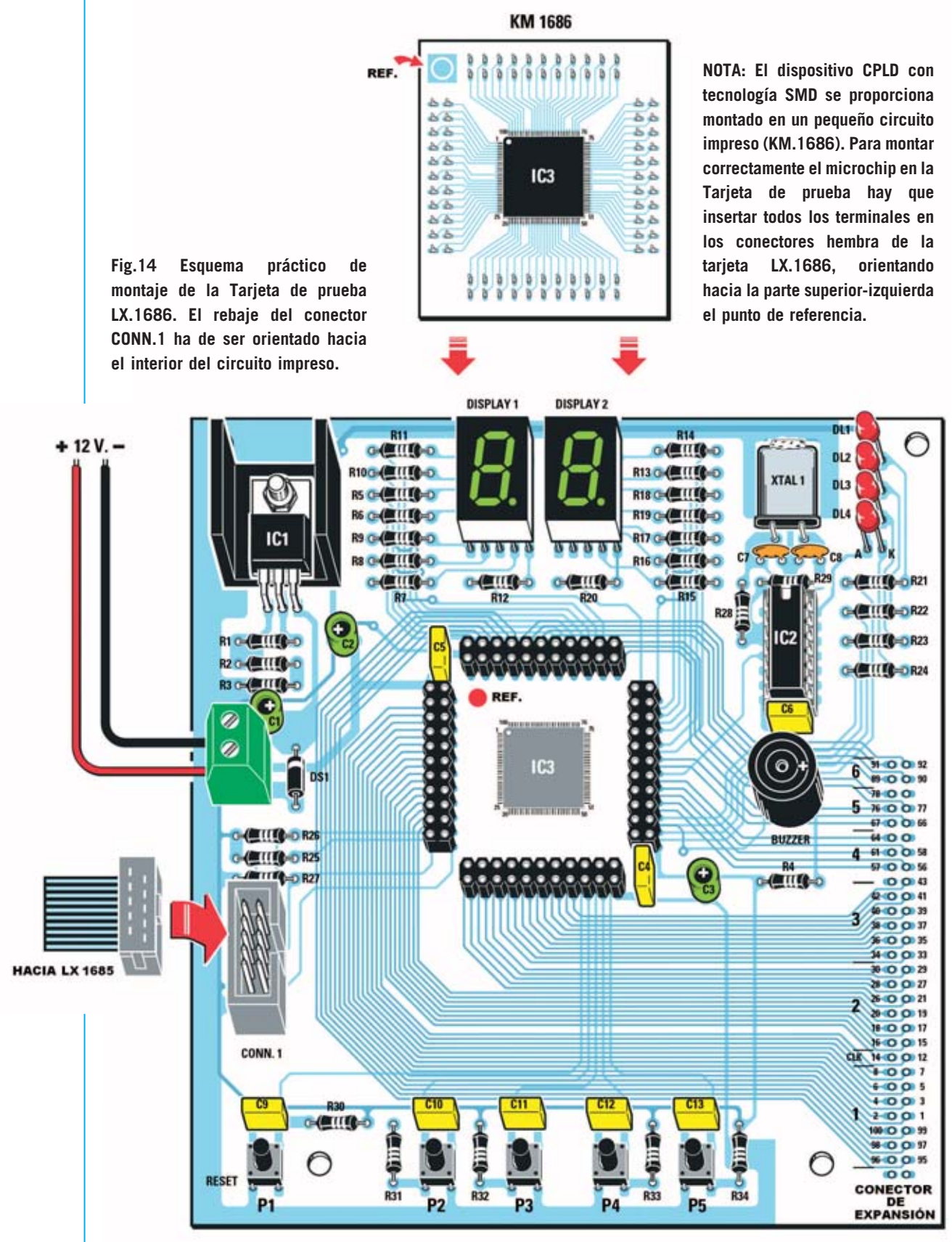


Fig.14 Esquema práctico de montaje de la Tarjeta de prueba LX.1686. El rebaje del conector CONN.1 ha de ser orientado hacia el interior del circuito impreso.

NOTA: El dispositivo CPLD con tecnología SMD se proporciona montado en un pequeño circuito impreso (KM.1686). Para montar correctamente el microchip en la Tarjeta de prueba hay que insertar todos los terminales en los conectores hembra de la tarjeta LX.1686, orientando hacia la parte superior-izquierda el punto de referencia.

referencias **R6** a **R10** y **una resistencia** de **470 ohmios** (amarillo-violeta-marrón) que ha de montarse en correspondencia con la referencia **R11**.

Ahora se puede montar el **diodo LED DL1**, respetando la **polaridad** de sus terminales (el terminal **más largo**, el ánodo, ha de insertarse en el agujero identificado con la letra **A**).

Una vez soldadas las resistencias se puede proceder al montaje de los **dos condensadores** de **poliéster**, ambos de **100.000 picofaradios**.

Para concluir el montaje solo queda introducir los **integrados IC1**-**IC2** en sus correspondientes zócalos, orientando sus muescas de referencia en coincidencia con las muescas de referencia de los zócalos.

REALIZACIÓN PRÁCTICA LX.1686

La parte más importante de esta tarjeta es el microchip CPLD **MAX II EPM240T100C5N**. Puesto que se distribuye en formato **SMD** lo proporcionamos **montado** en un **pequeño impreso** que incluye conectores (**KM.1686**).

De esta forma **no** es necesario disponer de **herramientas SMD**, con el soldador y las herramientas tradicionales se podrá montar el circuito impreso de la Tarjeta de prueba para CPLD.

La tarjeta ha sido pensada para su utilización **sin** necesidad de instalarla en un **mueble contenedor**. No obstante, una vez concluido el montaje, es aconsejable instalar los **separadores de plástico** para sustentarla con **firmeza**. Además, de esta forma, las soldaduras no harán contacto con la superficie de apoyo **evitando** posibles **cortocircuitos**.

El montaje puede comenzar con la instalación del **zócalo** que soporta a **IC2**, teniendo cuidado para no provocar cortocircuitos entre pistas adyacentes. Ha de montarse orientando hacia **arriba** su muesca de referencia en forma de **U**.

A continuación se pueden montar las **resistencias**, los **condensadores** de **poliéster**, los **condensadores cerámicos** y los tres **condensadores electrolíticos**, respetando en estos últimos la **polaridad** de sus terminales.

Es el momento de montar el diodo **DS1**, orientando hacia **arriba** su **franja blanca** de referencia.

El montaje puede continuar con la instalación de los **4 conectores hembra dobles** utilizados como zócalo de conexión para el circuito impreso **KM.1686** que contiene el **CPLD**. Al soldar los terminales hay que prestar mucha atención para no provocar cortocircuitos entre pistas adyacentes.

Llegado este punto se pueden instalar los **4 diodos LED rojos** utilizados para visualizar los efectos del programa de prueba y vuestros programas futuros. En correspondencia con el agujero que tiene serigrafiada la letra **A** hay que introducir el **terminal más largo** (**ánodo**).

Los **displays** de **7 segmentos** se montan directamente en el circuito impreso **sin** utilizar **zócalos**, tal como muestra el esquema de montaje práctico.

Ahora hay que montar, en posición **horizontal**, el cuarzo **XTAL1**, fijando su **encapsulado** al **circuito impreso** con una pequeña **soldadura**.

La **cápsula piezoeléctrica resonadora** (**buzzer**) ha de montarse respetando la polaridad de sus terminales (ver Fig.14). Después ya se pueden montar los **5 pulsadores**.

Ha llegado el momento de instalar la **clema** de dos polos utilizada para la **alimentación** y el conector de **10 terminales** (**CONN.1**), orientando su **rebaje** de referencia hacia la **derecha**.

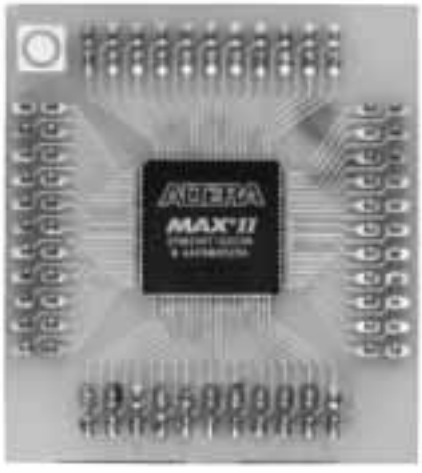


Fig.15 Fotografía del prototipo de la tarjeta SMD KM.1686 que contiene el microchip CPLD de 100 terminales.

Seguidamente hay que montar el integrado estabilizador **LM.317** (**IC1**). Para disipar el calor producido durante su funcionamiento este integrado precisa una **aleta de refrigeración**. El integrado se monta en posición **horizontal** doblando previamente sus terminales en forma de **L**. El integrado y la aleta se fijan al impreso utilizando un **tornillo metálico** y su correspondiente **tuerca**.

Para terminar el montaje hay que instalar el **integrado IC2** en su zócalo correspondiente, orientando su **muesca** de referencia hacia **arriba**, e instalar la tarjeta SMD **KM.1686** en los conectores que hacen la función de zócalo, orientando el **punto serigrafiado** sobre el impreso hacia la parte **superior-izquierda**.

PRUEBA de la TARJETA

Como suministro de energía del sistema hay que utilizar un **alimentador** de **12 voltios** capaz de proporcionar al menos **500 mA**, como nuestro **LX.92**.

En primer lugar hay que realizar una **verificación visual** de los componentes y alimentar el circuito **sin** conectar el **Programador LX.1685** al **PC**.

Si todo va bien se puede pasar al paso siguiente, la **programación** del **CPLD**.

Hay que seguir detalladamente las instrucciones correspondientes a la **instalación** del programa **Quartus II** y cargar, desde el CD-ROM que proporcionamos, el **programa** que **verifica** los pulsadores, displays, diodos LED y el zumbador de la tarjeta **LX.1686**.

Nombre en código ... QUARTUS II

El nombre parece evocar personajes de películas de ciencia-ficción, nada más lejos de la realidad. **Quartus II** es un **entorno de desarrollo completo** para realizar el **código** de programación, para efectuar el **ensamblado** y para **programar** los dispositivos **CPLD**.

En este artículo mostramos el procedimiento de **instalación del programa** y como utilizarlo con el **Programador LX.1685**. Junto al Programador LX.1685 proporcionamos un **CD-ROM** que contiene todo lo necesario para aprender los fundamentos de la programación de estos componentes realmente extraordinarios.

Para utilizar el programa es necesario solicitar la **licencia** a **ALTERA**, empresa desarrolladora de **Quartus**, que la concede de forma **totalmente gratuita**. La única "pega" es que la licencia caduca tras algunos meses, aunque se puede volver a **renovar gratuitamente** tantas veces como se desee.

El programa debe instalarse sobre una máquina que disponga del sistema operativo **Microsoft Windows XP**.

Fig.16 Para instalar el programa Quartus II hay que hacer doble click sobre MI PC y seleccionar la unidad en la que se ha introducido el CD-ROM CDR1685.

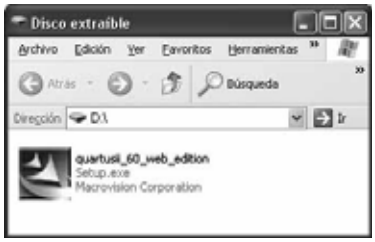


Fig.17 Para lanzar el proceso de instalación del programa Quartus II hay que hacer doble click sobre el icono con el nombre del programa.

Fig.18 Para continuar hay que hacer click en el botón NEXT. De esta forma comienza la instalación.

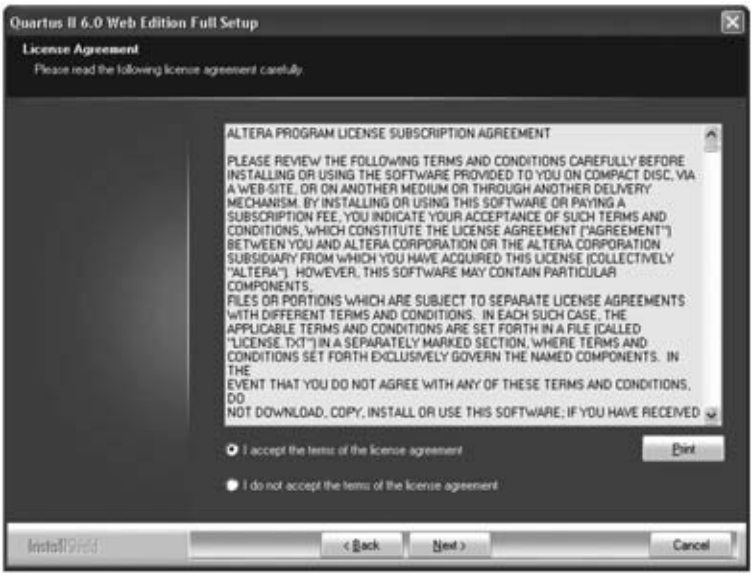


Fig.19 Cuando aparezca esta ventana se han de aceptar los términos de la licencia y hacer click en el botón NEXT.

Fig.20 Para continuar hay que introducir vuestro nombre, después hay que hacer click en el botón NEXT.

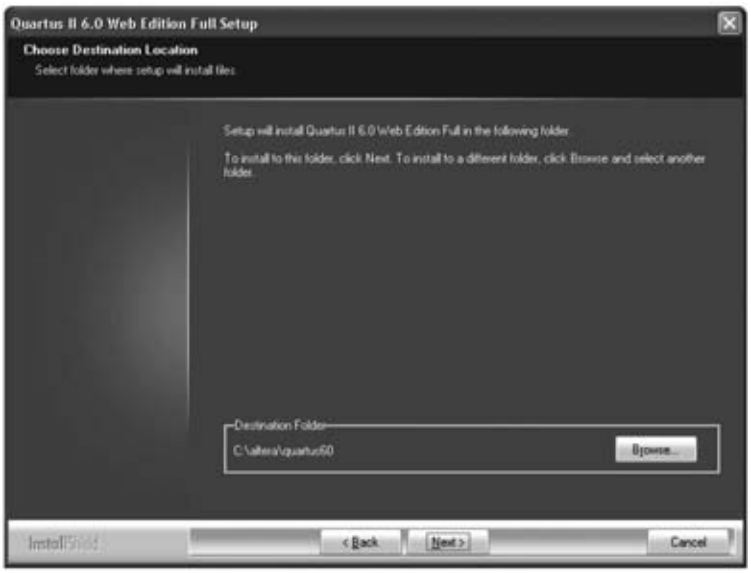


Fig.21 Ahora se ha de seleccionar el directorio destino, es aconsejable no cambiarlo. De forma predeterminada es C:\altera\quartus60.

Fig.22 Acto seguido hay que seleccionar el tipo de instalación, en nuestro caso completa (COMPLETE). Para continuar hay que hacer click en el botón NEXT.

Fig.23 En esta pantalla se insta a la selección de la entrada al programa a través del menú INICIO del escritorio de Windows (el valor predeterminado es ALTERA). Como de costumbre para continuar hay que hacer click en el botón NEXT.

Fig.24 Una barra de progreso verde indica el estado de copia de los archivos en el disco duro del ordenador.

Fig.25 si se desea crear automáticamente un icono de acceso directo a Quartus II en el escritorio hay que responder SI (ACEPTAR) a esta pregunta.

Fig.26 La instalación del programa ha terminado. Solo queda hacer click en el botón FINISH.

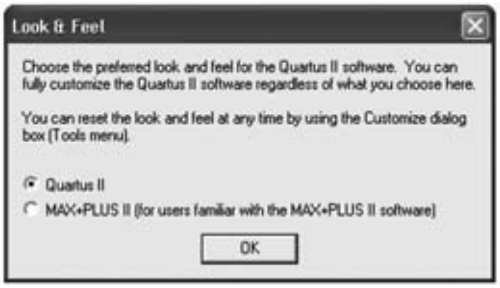


Fig.27 El programa permite dos modos operativos. Sin modificar nada hay que hacer click en OK.

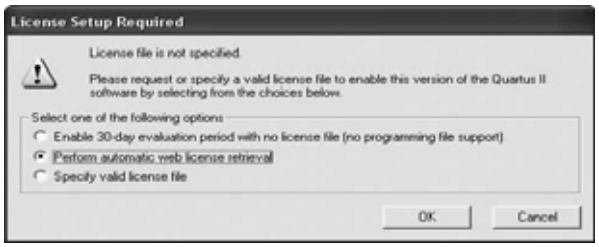


Fig.28 Para realizar el registro en línea hay que seleccionar la segunda opción (PERFORM AUTOMATIC WEB LICENSE RETRIEVAL) y hacer click en OK.

Fig.29 Aspecto de la Web de Altera.

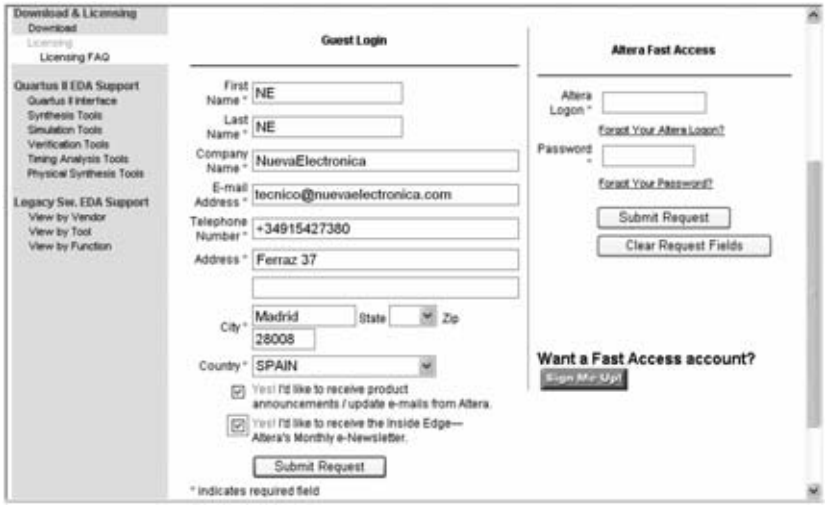


Fig.30 Los campos marcados con un asterisco rojo adjunto son obligatorios.

NOTA: Si en el ordenador corre **Windows XP** no son precisos más requisitos ya que la instalación de Windows XP en un ordenador lleva **implícitos** los **requisitos hardware** que precisa el programa **Quartus II**.

Es necesaria una conexión a **Internet** y **deshabilitar** temporalmente el **firewall** y el **antivirus** de vuestro ordenador, ya que **estos programas pueden bloquear** los **puertos de comunicaciones** utilizados para conectar el **Programador**.

Para realizar la **instalación** solo hay que seguir el procedimiento mostrado en las **figuras 16** a **26**.

Una vez instalado el programa hay que conectarse a la **Web** de **ALTERA** para realizar el **proceso de registro** que proporciona la **licencia temporal** para su utilización de forma gratuita.

Durante el proceso de registro, además del nombre, es importante poner un **correo electrónico operativo**. La contraseña es opcional (ver Fig.30).

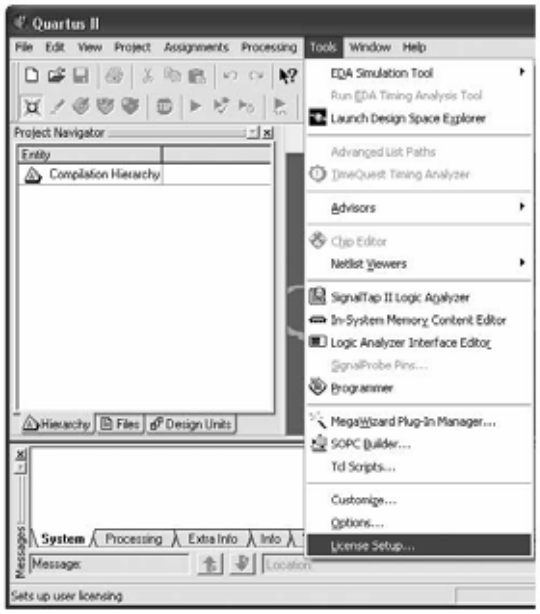
Todos los **campos** que tienen un **asterisco rojo adjunto** han de **rellenarse obligatoriamente**.

Si todo ha ido bien recibiréis por **correo electrónico** la **contraseña** y el **Login** en un documento adjunto en forma de **archivo DAT**.

Hay que **guardar** este archivo en el **directorio** en el que se ha instalado **QUARTUS**. Si no se ha modificado el directorio predeterminado durante la instalación es **C:\altera**.

Para **habilitar la licencia** es necesario abrir el programa y seleccionar en el menú **Tools** la opción **License setup**. En la ventana mostrada en la Fig.32

Fig.31 Altera confirma de esta forma la solicitud de licencia.



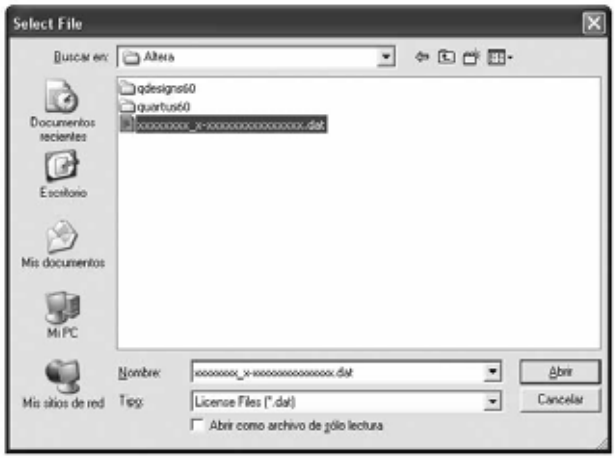


Fig.32 Para habilitar la licencia hay que seleccionar del menú TOOLS la opción LICENSE SETUP. A continuación hay que abrir el archivo .DAT que contiene vuestras credenciales.

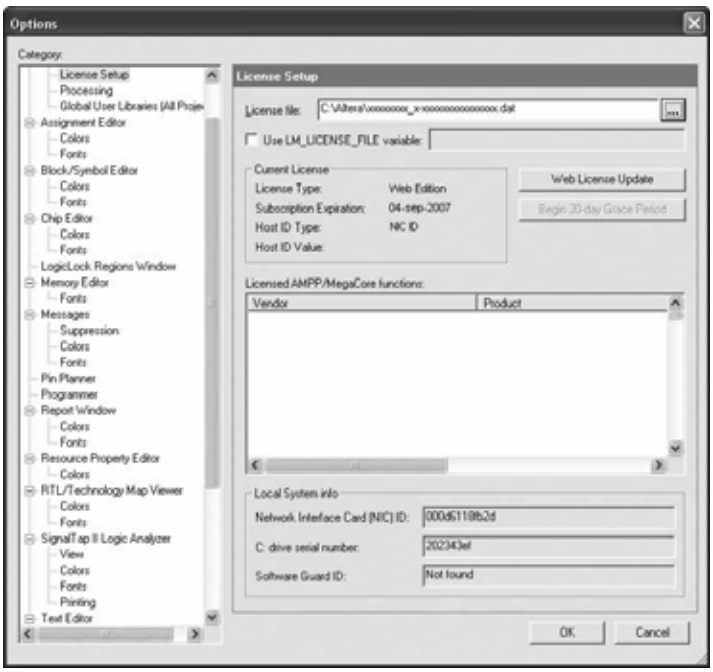


Fig.33 Tras verificar que la fecha de caducidad de la licencia (SUBCRIPTION EXPIRATION) es posterior a la fecha de vuestro ordenador hay que hacer click en OK.

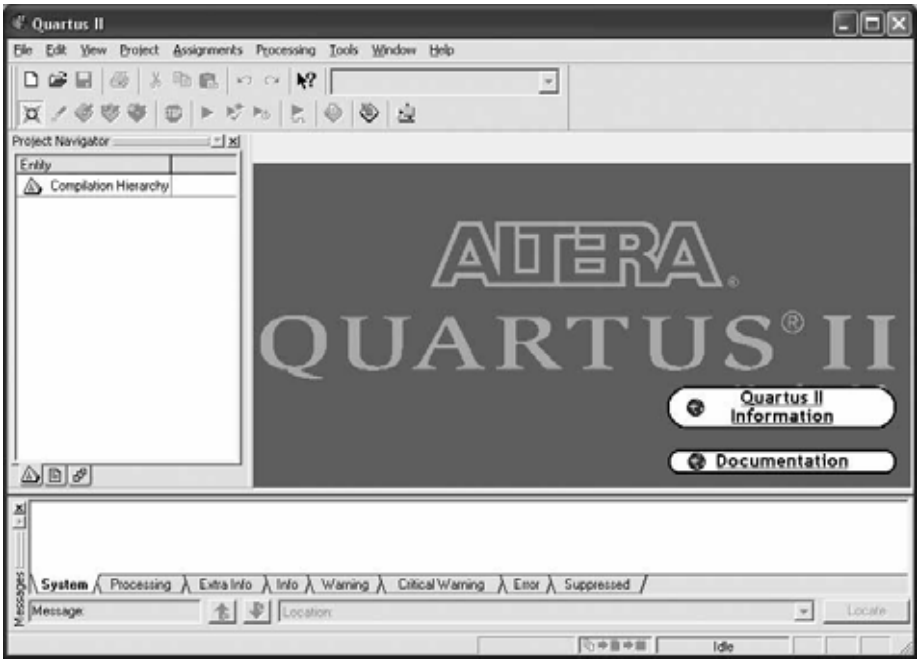


Fig.34 Aspecto de la pantalla principal del programa Quartus II versión 6.0. Para estar seguros de que no se interrumpe el flujo de información en los puertos de comunicación es conveniente deshabilitar los cortafuegos (firewalls) y los antivirus antes de realizar la programación.

hay que seleccionar el directorio en el que está instalado el programa y el **archivo** con extensión **.DAT** que contiene la contraseña temporal.

Hay que asegurarse de que la **fecha de caducidad** (ver **Subcription Expiration** en la Fig.33) sea **posterior** a la **fecha del sistema**.

Haciendo click en **OK** el programa está listo para trabajar con los microchips **CPLD**.

Puesto que la **licencia** es **provisional** hay que **renovarla** de vez en cuando, solicitándola **gratuitamente** a **Altera** como la primera vez.

No obstante se puede adquirir una **licencia profesional** que **no** precise **renovación**. Para los **profesionales** es aconsejable.

PRIMERA PROGRAMACIÓN Y PRUEBA

Una vez más creemos oportuno mencionar que antes de abrir **Quartus II** hay que deshabilitar los **firewalls** y los **antivirus**, ya que algunos de estos programas, como Norton o Kaspersky, **detienen** temporalmente los **puertos de comunicación**.

El programa está listo para escribir el código, montarlo y cargarlo en el CPLD instalado. A continuación exponemos, paso a paso, el procedimiento necesario para realizar una **verificación preliminar** del **hardware**, es decir realizar la comprobación del funcionamiento correcto del **Programador LX.1685** y de la **Tarjeta de prueba LX.1686**.

Como parte de esta verificación mostramos como se **carga** un **código ejemplo** para comprobar nuestras tarjetas.

1) Conectar el **Programador LX.1685** al puerto paralelo del ordenador y la **Tarjeta de prueba LX.1686** al Programador.

2) Alimentar el circuito.

3) Abrir **Quartus II** (ver Fig.34).

4) Hacer click en el menú **Tools** y seleccionar la función **Programmer** (ver Fig.35).

5) Hacer click en el botón **Hardware Setup**. Se abrirá la ventana mostrada en la Fig.36.

6) En el campo **Currently selected hardware** hay que seleccionar **ByteBlasterII**. Por último hay que cerrar la ventana haciendo click en el botón **Close**.

7) Si todo ha ido bien se abrirá una ventana similar a la mostrada en la parte superior de la Fig.37.

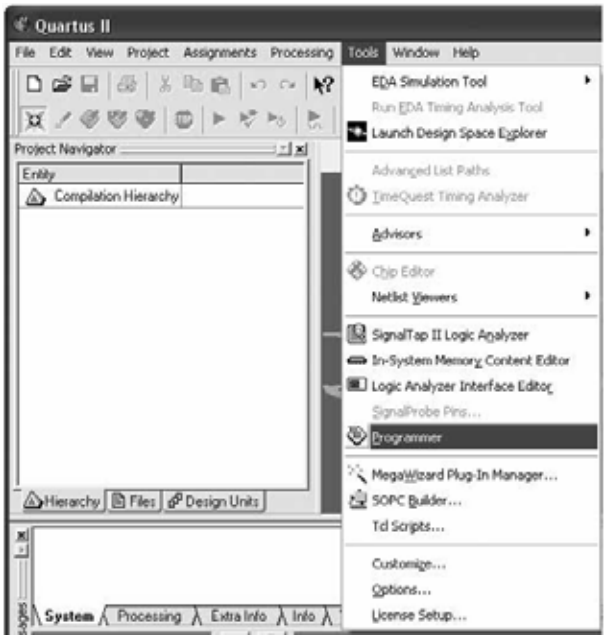


Fig.35 Una vez conectado el programador al ordenador y a la tarjeta de prueba para programar el dispositivo CPLD hay que seleccionar la función PROGRAMMER del menú TOOLS.

Fig.36 Hay que hacer click en HARDWARE SETUP y, en la ventana que aparece (aquí reproducida), seleccionar BYTEBLASTERII como hardware seleccionado. Una vez realizada esta operación hay que hacer click en CLOSE.

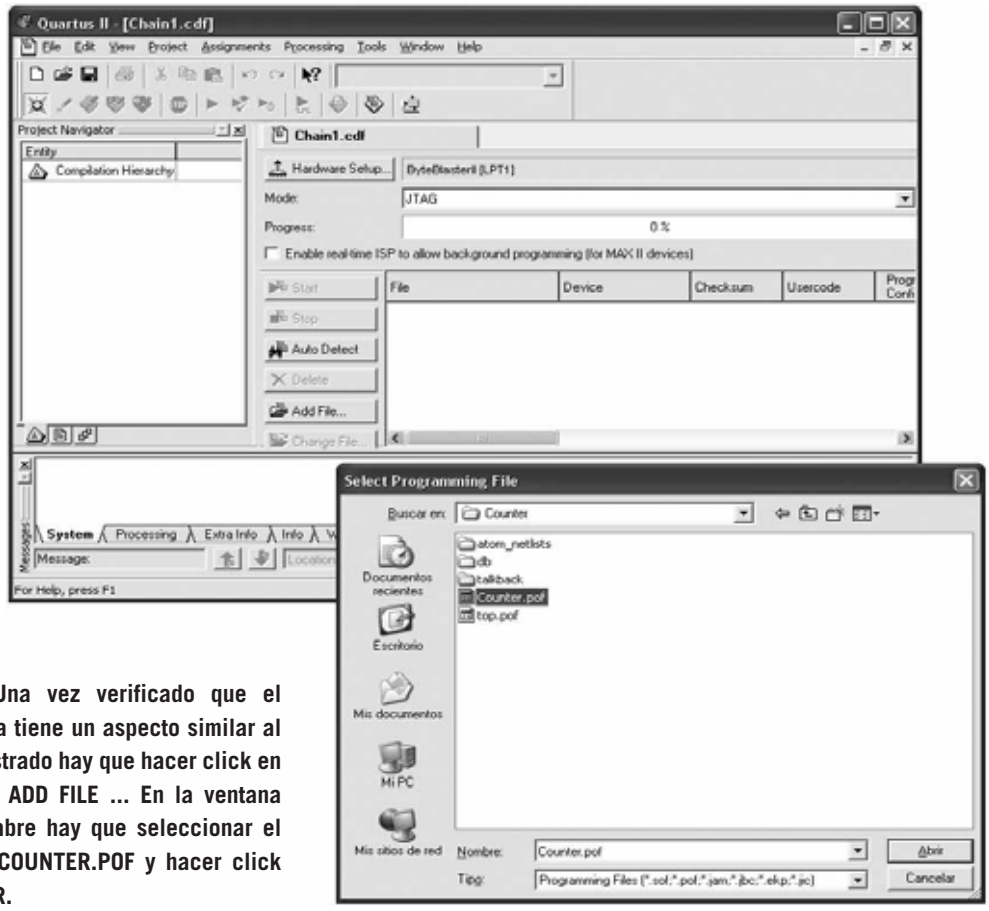


Fig.37 Una vez verificado que el programa tiene un aspecto similar al aquí mostrado hay que hacer click en el botón ADD FILE ... En la ventana que se abre hay que seleccionar el archivo COUNTER.POF y hacer click en ABRIR.

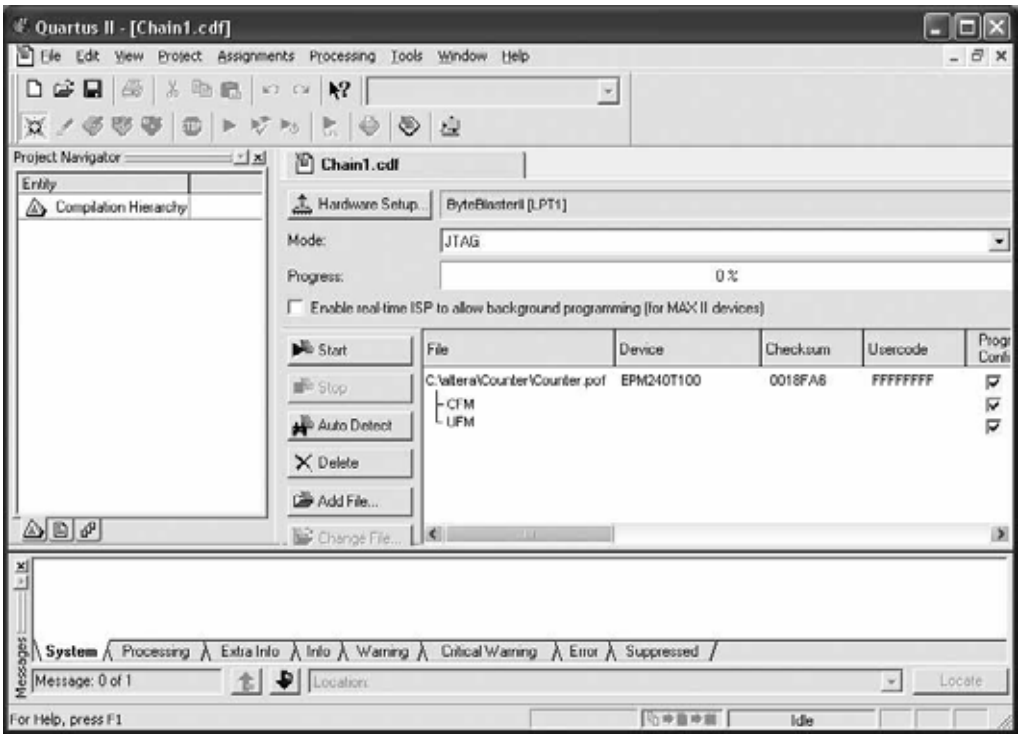


Fig.38 En la columna PROGRAM/CONFIGURE (la columna situada más a la derecha en la imagen) hay que marcar las 3 casillas de verificación. Para iniciar la programación del CPLD MAX II EPM240T100C5N hay que hacer click en START.

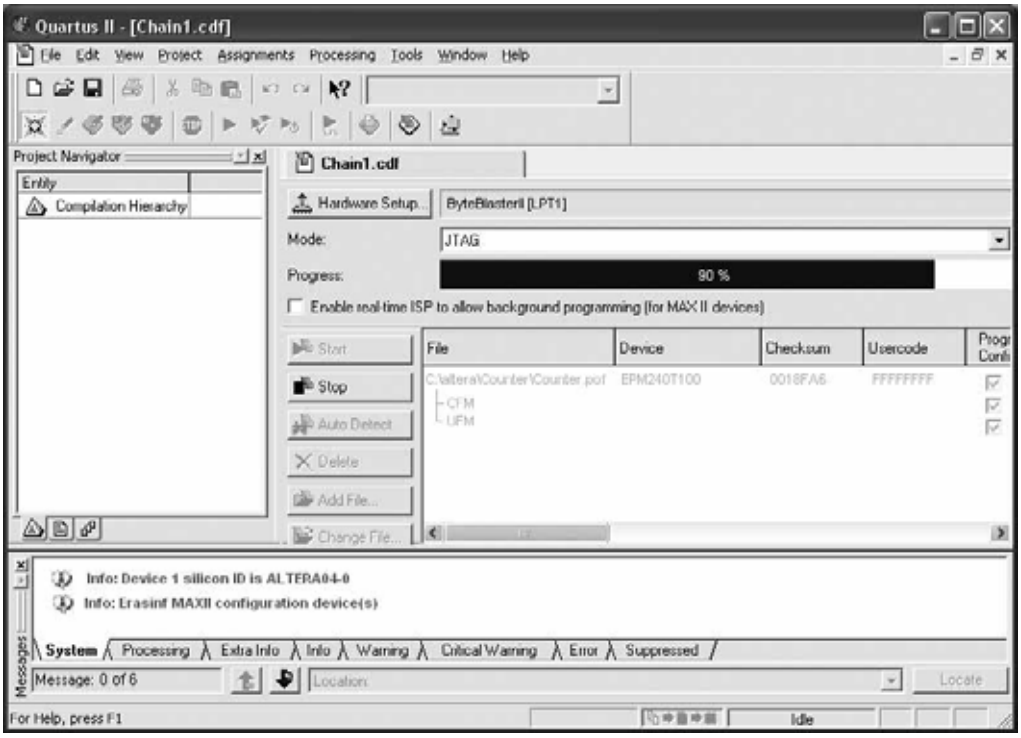


Fig.39 Durante la programación del microchip Quartus II tiene un aspecto similar al aquí mostrado. El estado de la programación se indica mediante una barra de progreso con fondo azul, al llegar al 100% la programación ha terminado.

8) Ahora hay que hacer click en el botón **Add File**.

9) Hay que posicionarse en el directorio donde se encuentra el archivo **Counter.pof** (la extensión **pof** corresponde a **p**rogramming **o**bject **f**ile). Una vez localizado el archivo hay que hacer click en **Abrir** (ver Fig.37).

10) El programa ofrecerá un aspecto similar al mostrado en la imagen de la Fig.38.

11) En la columna **Program/Configure** hay que marcar las **tres casillas de verificación**.

12) Ya solo queda hacer click en el botón **Start** para lanzar la **programación** del **CPLD**. El programa tomará un aspecto similar al mostrado en la Fig.39.

Si todo funciona correctamente el circuito realizará las siguientes operaciones de **verificación**:

- Presionando el pulsador **P2** se enciende el diodo LED **DL1**.

- Presionando de nuevo el pulsador **P2** el diodo LED **DL1** se apaga.

- Accionando el pulsador **P3** se encienden secuencialmente los diodos LED **DL2**-**DL3**-**DL4**.

- Cada vez que se accione el pulsador **P5** avanzará el contador de los **displays** (cuenta de **0** a **99**).

- Cada vez que se accione el pulsador **P4** retrocederá el contador de los **displays**.

- El pulsador **P1** reinicia el **CPLD** (función **Reset**).

NOTA: A veces al accionar los pulsadores **P4**-**P5** se producen **saltos** en el contador. Es normal en esta tarjeta ya que **no** dispone de una **rutina anti-rebotes**. La presión del pulsador produce **picos de tensión** que son interpretados por el dispositivo **CPLD** como **presiones múltiples** del pulsador. Al tratarse de una tarjeta de prueba no hemos filtrado estas señales.

Adicionalmente en el **CD-ROM** incluido con el **Programador LX.1685** se encuentra el archivo **test.pof**. Este código está desarrollado para probar las conexiones de los componentes en la tarjeta **LX.1686**.

Con este código se pueden verificar **funciones adicionales**:

- **Intermitencia** del display.

- Encendido de tres diodos LED en **cuenta binaria**.

- **Lógica AND** asociada con dos pulsadores. Mediante el encendido del cuarto LED se indica si los dos pulsadores se han accionado simultáneamente.

- Emisión sonora de un tono a través del **zumbador** cuando se acciona un pulsador. Se emite un tono con una octava menor cuando se acciona el otro pulsador.

- **Reinicio** del CPLD cuando se acciona el pulsador Reset (**P1**).

PRECIOS de REALIZACIÓN

LX.1685: Todos los componentes necesarios para realizar el **Programador CPLD** (ver Fig.11 y Fig.12), **incluyendo** circuito impreso, integrados, conector para el puerto paralelo, manguera de conexión de 10 hilos para conectar el programador a la Tarjeta de prueba LX.1686 y el **CD-ROM CDR1685** .....41,88 €
NOTA: El **CD-ROM CDR1685** contiene el programa **Quartus II**, es decir el paquete completo para la escritura del código de programación, para ensamblar y para programar los dispositivos **CPLD**. Además contiene dos programaciones de prueba (**counter.pof** y **test.pof**) .....13,10 €
LX.1686: Todos los componentes necesarios para realizar la **Tarjeta de prueba CPLD** (ver Fig.13 y Fig.14) **incluyendo** circuito impreso, integrados, cuarzo, dos displays, diodos LED, zumbador, pulsadores y la tarjeta SMD **KM.1686** (ver Fig.15) con el chip CPLD **MAX II EPM240T100C5N** .....86,86 €
LX.1685: Circuito impreso .....7,32 €
LX.1686: Circuito impreso .....33,68 €
CA05.2: Cable con dos conectores de 25 terminales para la conexión del **Programador CPLD** al puerto paralelo del **ordenador**.....7,80 €

ESTOS PRECIOS NO INCLUYEN I.V.A.