ENTORNO DE DESARROLLO PARA MICROPROCESADORES 80X86

J.C. Pineda y L. Closas
Departament d'Enginyeria Electrònica
Universitat Politècnica de Catalunya
c/Gran Capitán s/n, módulo C4 08034 Barcelona
Tlf: (93) 401 74 78

RESUMEN.- A la hora de diseñar sistemas basados en microprocesadores actuales, surgen determinados problemas al implementar la realización física. Las principales causas de estos problemas, suelen ser la gran cantidad de líneas (buses) que posee el sistema y las elevadas frecuencias de trabajo. En base a esta problemática, se hace patente la necesidad de disponer de un método, capaz de simplificar el desarrollo de estos sistemas.
El entorno aquí presentado responde a esta necesidad, facilitando al diseñador, el desarrollo de sistemas basados en los μP 80X86 de Intel.

1.- DESCRIPCION DEL ENTORNO

A nivel de hardware el entorno de desarrollo está formado por:

- Un ordenador **maestro**, que es desde donde se ejecutan todos los comandos y se depuran los programas. El ordenador puede ser cualquier modelo de PC compatible (no importa antigüedad) y debe poseer como mínimo, una tarjeta de comunicaciones serie.
- Una placa base de PC compatible, que actuará de terminal **esclavo**. El modelo de placa, dependerá del μP para el cual se quiera desarrollar la aplicación. Además, deberá incorporar una tarjeta de comunicaciones serie y una fuente de alimentación, por supuesto.
  Un detalle importante referente a la placa, es que las memorias ROM deben estar conectadas mediante zócalos (lo cual ocurre casi siempre), o de lo contrario no servirá para nuestros propósitos.

En la Figura 1 puede observarse un pequeño croquis, en el cual se diferencian las partes que constituyen el entorno.
La comunicación maestro-esclavo, se realiza a través del puerto serie a una velocidad de 9600 baudios, 8 bits de datos, paridad impar y 1 bit de stop.
Para llevar a cabo las comunicaciones, se emplean tramas cuya estructura puede observarse en la Figura 2.

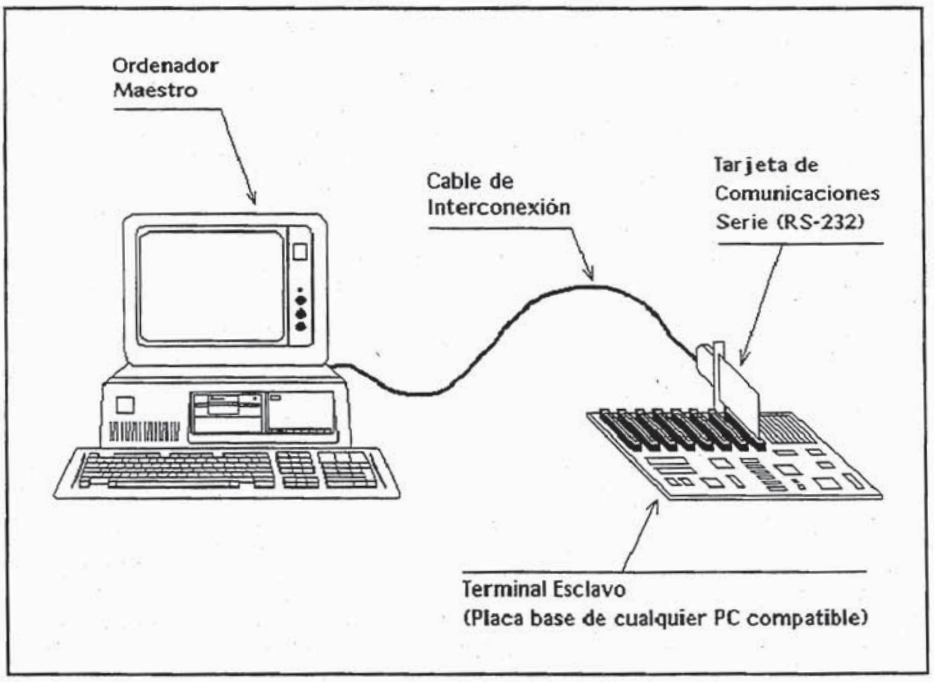

Figura 1.- Conexión maestro-esclavo

Todas las tramas tienen una longitud fija de 135 bytes.
El primer byte ("COMANDO") identifica la orden que ha de ejecutar el esclavo, por ejemplo "Ejecutar instrucción", "Leer registros", etc. Los próximos cuatro bytes, se utilizan para especificar direcciones de memoria o ports E/S. El siguiente byte, señala cuantos bytes de datos se están utilizando, ya que el campo de datos tiene una cantidad fija de 128. A continuación se transmiten los datos en cuestión (bytes de información). Para acabar, el último es un byte de redundancia, utilizado para controlar la integridad de la trama.
En ambos terminales, maestro y esclavo, la recepción de datos se lleva a cabo, efectuando un polling continuo del puerto serie.

Comando	Desplazamiento		Segmento		N° de Bytes Válidos	Bytes de Datos	Checksum
1 byte	byte bajo	byte alto	byte bajo	byte alto	1 byte	128 bytes	1 byte

Figura 2.- Formato de las tramas

En lo referente al software, el entorno consta de dos partes bien diferenciadas:

- Programación del maestro.
- " " esclavo.

El programa que se ejecuta en el maestro, es la cara visible del entorno de desarrollo, ya que desde él se controlan todas las operaciones.
En el esclavo, el programa de comunicación con el maestro está grabado en una memoria EPROM, y se ejecuta al conectar la alimentación de la placa. Sus principales misiones son:

- Inicializar correctamente la placa.

- Ejecutar los comandos que recibe del maestro.

Para inicializar correctamente la placa, es necesario analizar la ROM-BIOS que ésta incorpora de fábrica. De esta forma se hallará la inicialización concreta de la placa estudiada. En caso de trabajar con otra placa base diferente, el proceso se ha de repetir, ya que cada modelo posee una inicialización particular.

Hasta este punto, hemos visto en líneas generales las partes que constituyen el entorno. A continuación, se describen las principales opciones disponibles al ejecutar el maestro:

- Edición de ficheros de texto.
- Ensamblado de ficheros fuente para 80X86.
- Linkado de ficheros objeto para 80X86.
- Depuración de programas.
- Tareas básicas sobre ficheros del directorio de disco actual.

La depuración de programas, permite realizar ciertas operaciones sobre el software diseñado por el usuario. En la Tabla I, se muestra una breve descripción de los comandos disponibles.

OPERACION	DESCRIPCION
Cargar programa	Carga cualquier fichero .COM en la placa esclava, en el segmento de memoria especificado
Almacenar memoria	Almacena el contenido de la zona de memoria especificada, en ficheros de diversos formatos: - Binarios - Desensamblado de instrucciones - Prototipos
Ejecutar	Lanza la ejecución a la dirección de memoria especificada
Ejecutar paso a paso	Ejecuta una instrucción y visualiza los registros del microprocesador
Imponer condiciones de ruptura	Ejecuta un programa hasta que se cumplan ciertas condiciones (valores de registros, direcciones de memoria, etc)
Colocar puntos de ruptura	Sitúa un punto de ruptura dentro del programa de usuario y a continuación lanza su ejecución
Ver y modificar registros	Permite visualizar o modificar el contenido de los registros
Ver y modificar memoria	Permite visualizar o modificar el contenido de la memoria

Tabla I.- Comandos de depuración

Una de las opciones más interesantes, es la de poder construir ficheros "prototipo". Estos ficheros, se utilizan para grabar las memorias EPROM de una placa prototipo, y contienen entre otras cosas, el programa de aplicación diseñado por el usuario. Una vez grabada la EPROM, solo tenemos que sustituirla por la que tiene la placa esclava. El resultado será un sistema 80X86 programado a gusto del diseñador. Cabe decir que la placa prototipo es la

placa esclava, pero con distintas memorias ROM.

2.- POSIBILIDADES DEL EQUIPO

El software de control del sistema prototipo (programa de usuario), puede escribirse en cualquier lenguaje de programación, siempre y cuando el código ejecutable resultante, se encuentre en un fichero .COM. También es indispensable que el programa en cuestión, no utilice ninguno de los servicios ni de la ROM-BIOS ni del DOS, a excepción de las funciones de "Terminación de programa". Teniendo en cuenta estas consideraciones, no hay ninguna otra restricción a la hora de desarrollar programas en C, Pascal, Basic, Ensamblador, etc. Al margen de estas posibilidades, el entorno de desarrollo está pensado para facilitar la creación de programas en lenguaje ensamblador, haciendo posible la compilación y posterior reubicado desde el propio entorno.
Una vez el diseñador ha obtenido un fichero .COM, cargará este sobre la memoria de la placa esclava, a partir de la dirección especificada. A continuación es posible depurar el programa, ejecutándolo paso a paso, por puntos de ruptura o ininterrumpidamente. En cualquier caso, desde el terminal maestro es posible seguir la evolución de los registros, de la memoria y de los ports E/S.
Cuando el programa está totalmente depurado y funciona a la perfección, se elige la opción de construir un fichero prototipo y a continuación, se graban las EPROM de la placa con ese fichero. De esta forma, sustituyendo dichas memorias, se obtiene una placa prototipo en la cual se ejecuta permanentemente, el software creado por el diseñador.
A modo orientativo, en la Figura 3 puede observarse un pequeño esquema que enumera los pasos a seguir, hasta llegar al objetivo final: Construcción de una placa prototipo de propósito particular.

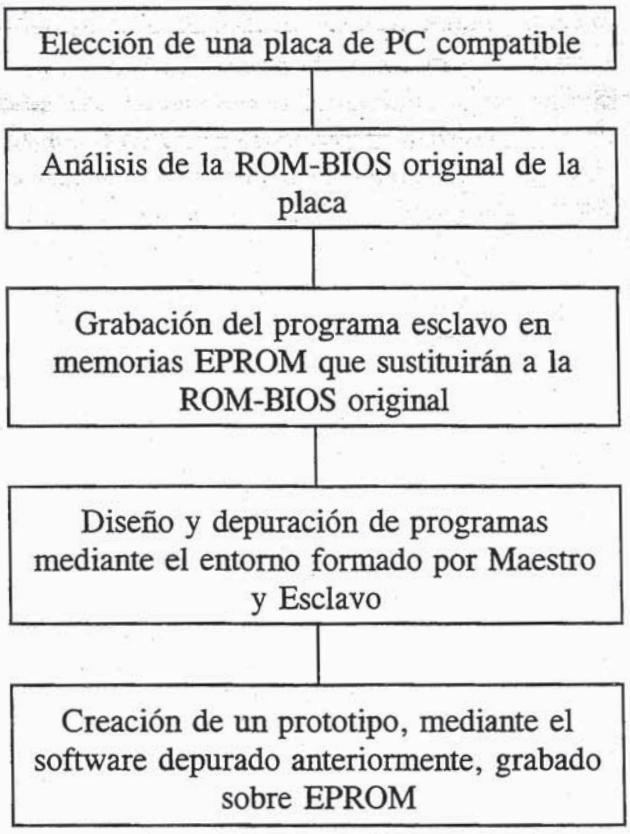

Figura 3.- Proceso para desarrollar un sistema 80X86

Por último destacaremos como característica relevante, el hecho de que el entorno permita depurar código para microprocesadores de hasta 32 bits, como son el 80386 o el 80486, manteniendo la compatibilidad con toda la familia 80X86. Esto quiere decir que se pueden depurar programas de 16 o de 32 bits, indistintamente, sobre los μP 8088, 8086, 80188, 80186, 80286, 80386, 80486, etc. Internamente, el entorno, está diseñado para trabajar con estructuras de registros de 32 bits y con un desensamblado de instrucciones también de 32 bits.

3.- MAPA DE MEMORIA DE LA PLACA ESCLAVA

A la hora de cargar programas en la RAM de la placa esclava, será de vital importancia conocer la ubicación exacta de este tipo de memoria. Para ello habrá que tener en cuenta, el mapa de memoria de la placa esclava. En la Figura 4 puede observarse el mapa de memoria correspondiente a una placa esclava del tipo PC-AT, basado en las notaciones de C.Eggebrecht [1].

FFFFF F0000	Programa Esclavo
EFFFF E0000	Ampliaciones de la ROM-BIOS (Video ROM)
DFFFF C0000	Sin utilizar
BFFFF BC000	Reservado al Buffer de Video
BBFFF B8000	Buffer del adaptador Color/Gráficos
B7FFF B1000	Reservado al Buffer de Video
B0FFF B0000	Buffer del adaptador Monocromo
AFFFF A0000	Buffer temporal de video avanzado
9FFFF 00800	Zona de memoria RAM disponible para ser utilizada por el usuario
007FF 00000	Zona de memoria RAM reservada para el funcionamiento interno del programa esclavo

Figura 4.- Mapa de memoria de un esclavo PC-AT

Tal y como se puede comprobar en la Figura 4, el espacio de memoria accesible es de 1 MByte. Esto es debido a que tanto el programa esclavo como el software de usuario, deben trabajar en el "Modo Real" de la familia 80X86 [2]. En este modo cualquier microprocesador 80X86, se comporta básicamente como un 8086 con el juego de instrucciones y registros ampliado. El resto de modos disponibles, aumentan el direccionamiento de memoria y utilizan características de protección. Por cuestiones de compatibilidad, el software de depuración (programa esclavo) ha sido escrito para trabajar en modo real, dado que es el único común a toda la familia. Si se utiliza un modo diferente en los programas de usuario, es muy probable perder el control sobre la placa esclava.

4.- CONCLUSIONES

Por un lado, el entorno de desarrollo posee un gran valor didáctico, puesto que es sencillo de manejar y ayuda a comprender el funcionamiento de los μP 80X86. Por otro, es una gran herramienta de cara al diseño industrial con estos microprocesadores.
La idea básica de este proyecto, ha sido proporcionar al diseñador un método fácil y económico, para desarrollar sistemas basados en microprocesadores 80X86:

- Fácil, porque no es necesario construir el hardware del sistema, simplemente hay que diseñar un software y grabar unas EPROM.
- Económico, debido a que todo el hardware utilizado es de uso común y está muy popularizado.

Puede dar la sensación de que los sistemas diseñados de esta forma, están muy limitados, pero realmente no es así. El hardware de la placa base de un PC compatible, es una estructura rica en recursos y que además, es fácilmente ampliable. En base a esto, es posible diseñar sistemas más complejos, tomando como eje central la placa base y añadiendo a continuación, las ampliaciones necesarias (slots de expansión). De esta forma se pueden potenciar características como:

- Mayor tamaño del programa de control, ya que una parte de él podría estar almacenada en ampliaciones de ROM.
- Ampliación del sistema, mediante la conexión de todo tipo de dispositivos al bus de expansión.

Al margen de estas peculiaridades, también es importante resaltar la versatilidad de esta técnica, permitiéndonos efectuar diseños que utilicen desde un 8088, hasta un 80486 o incluso superiores, independientemente de su frecuencia de trabajo. Gracias esto, el campo de aplicación es muy grande y proporciona un amplio abanico de posibilidades al diseñador.

5.- REFERENCIAS

[1] Lewis C.Eggebrecht. "Interfacing to the IBM Personal Computer".
Ed. Sams, USA (1990).

[2] Intel Corporation. "80386. Guía del programador y Manual de referencia".
Ed. Anaya Multimedia . Intel (1989).