Desarrollo de proyectos electrónicos

Francisco Aguayo Gonzalez[1]

Después de una descripción de las diversas etapas de elaboración de un proyecto electrónico implantado sobre microprocesadores y periféricos, el autor detalla las diversas fases de desarrollo del «software» y «hardware», haciendo un especial énfasis en el interconexionado entre ambos —la repartición de tareas «hard/soft» es una fase crítica del proyecto—, pasando luego a describir el perfil del equipo necesario en función del «hardware» y «software» por los que se haya optado. Al final del artículo se resumen las ventajas (disminución de costes de fabricación y tiempos de desarrollo así como, mayor fiabilidad) e inconvenientes (inversión inicial y mayor cualificación del personal) del empleo de microsistemas en los proyectos electrónicos.

El desarrollo de un proyecto electrónico implementado sobre microprocesadores y periféricos integrados es un proceso complejo que puede descomponerse en etapas, según se muestra en el diagrama de flujo de la fig. 1.

Definición del proyecto: Son las especificaciones de la tarea a realizar, en términos de los requisitos que ésta fija en el microsistema; es decir, en ella se diseña una posible solución al problema planteado. Este diseño engloba la selección o montaje de un sistema a microprocesador completo y el diseño de las funciones que deben incorporarse a él por programa. La consideración técnica fundamental en este punto es determinar si las prestaciones del sistema «hard/soft» resultantes serán suficientes, es decir, si cumplirán las especificaciones planteadas. Es necesario en esta fase determinar la forma y relación de las entradas y salidas del microsistema, la estructura y la cantidad de información a procesar, la velocidad de procesamiento necesaria, etc...

Comprobación de hipótesis de diseño: La comprobación de la hipótesis de diseño y técnicas usadas puede llevarse a cabo como se ve en la fig. 1, por tres métodos, ya sea por separado o conjuntamente.

— La comprobación sobre el papel: Se procede ejecutando el programa manualmente, anotando en una tabla los diferentes contenidos de cada registro y los estados del micro a cada paso. De esta forma se puede comprobar que el programa da los resultados deseados. Es un método tedioso. Se utiliza a nivel de diagrama de flujo para comprobar si el diseño está bien.

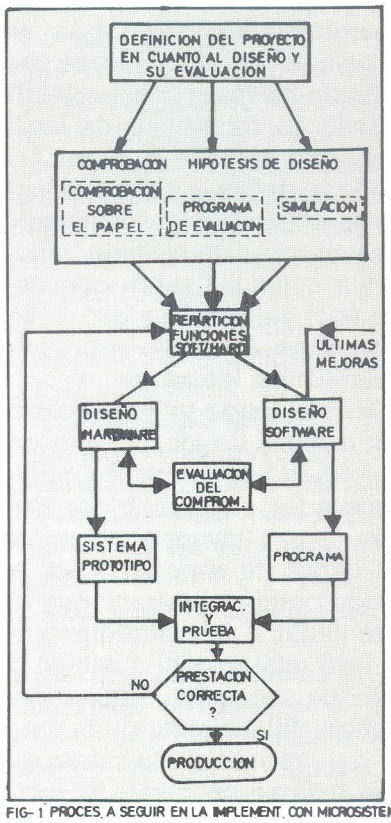

FIG- 1 PROCES. A SEGUIR EN LA IMPLEMENT. CON MICROSISTEMA

— Los programas de evaluación o programas patrón (beuchmark): Son los que confecciona el usuario a fin de comprobar y medir de algún modo la aptitud de un determinado microprocesador para la aplicación a que se destina. A menudo los fabricantes suministran programas que denominan «de evaluación». Un programa de evaluación auténtico ha de ser escrito por el usuario.

— El tercer método es la simulación, que puede hacerse de di-

(1) Profesor encargado de curso de la EUITI de Sevilla.

ferentes formas. La simulación de programas puede hacerse escribiendo estos programas en lenguaje de alto nivel y pasándolos en simulación por un computador potente. Es un método costoso y largo, pero se obtienen resultados muy precisos. Es, sin duda, el método más preciso.

Finalmente, como siempre, el principal criterio para decidir si un diseño determinado cumplirá o no las especificaciones solicitadas es la experiencia. Los tres métodos citados pueden servir como herramienta de trabajo.

Repartición «hard/soft»: Una vez especificado el proyecto, se pasa a una de las fases críticas del desarrollo: La repartición de las tareas «hard/soft». Será en esta fase donde se defina la distribución de tareas entre el microprocesador y los circuitos de entrada/salida, lo que condicionará la complejidad final del diseño y, por lo tanto, su coste. Debido a la gran flexibilidad del «software», el criterio más utilizado es la simplificación máxima de los circuitos que conforman la periferia del microprocesador, realizando por «software» tantas funciones como éste sea capaz de soportar, pero teniendo siempre presente que deberá llegar a un compromiso entre este criterio y el aumento de memoria necesaria, como consecuencia del aumento de la longitud del programa. Así mismo, deberá tenerse en cuenta el incremento en la duración de los programas, en algunas aplicaciones. El compromiso a este nivel es de tipo económico principalmente y se debe plantear en base a factores como la cantidad estimada del producto a fabricar, la preparación del personal y naturalmente la capacidad operativa del microprocesador seleccionado.

Diseño «hard». Diseño «soft»: Una vez esbozada la repartición de tareas entre «software» y «hardware», se pueden desarrollar ambas por separado paralelamente. El desarrollo de la parte hardware de un microsistema puede ser relativamente sencillo en un sistema estándar, pudiendo complicarse, si fuera necesario, con interfases especiales. Por lo general, el problema de diseño se centra en la programación, aunque posteriormente nos referiremos no sólo a esta sino también al «hardware».

Evaluación del compromiso: Durante el periodo de diseño hay que reconsiderar continuamente el reparto de funciones establecido anteriormente entre «hardware» y «software», pues puede verse la necesidad de hacer un nuevo reparto de tareas.

Sistema prototipo y programas: Una vez concluidos estos diseños, se tendrá respectivamente un prototipo de «hardware» y su programa que cabe esperar que sea adecuado al problema que se está tratando.

Integración y comprobación: La integración consiste simplemente en situar el programa en el prototipo del microsistema y realizar la puesta a punto del conjunto. Esta suele ser la parte más compleja y larga del desarrollo.

Evaluación de prestaciones: Una vez montado el sistema completo con un «hardware» y un «software» debidamente depurados, el sistema debe pasar por la evaluación de las prestaciones del mismo.

Producción: Una vez superada la fase anterior, el trabajo ha terminado y puede pasarse a la producción del mismo. Si no cumple las funciones, se ha de volver a la fase de repartición de funciones entre el «hardware» y el «software». Hemos de destacar que, si el diseño es correcto, no tendrán que efectuarse más cambios en el «hardware» y todas las mejoras o modificaciones posteriores han de poder realizarse por «software». En los apartados siguientes se exponen más profundamente los aspectos concernientes al desarrollo del «soft» y «hard».

Desarrollo del software

El desarrollo del software lo podemos considerar dividido en las siguientes etapas:

- Definición de las funciones software.
- Estimación del timing de software.
- Desarrollo de los programas:
  - Diseño del programa.
  - Codificación.
  - Depuración.
  - Prueba.
  - Mantenimiento.

Algunas de las etapas expuestas anteriormente están en íntima relación simbiótica con algunas descritas en el apartado de «hardware», en cuanto a la consecución de objetivos previstos, ya que como ya se ha dicho la solución al proyecto será un compromiso entre «soft» y «hard».

Definición de las funciones del «software»: Como norma general y siempre que las disponibilidades de memoria y velocidad de procesamiento lo permitan, se tiende a realizar el mayor número de funciones por «software».

Estimación del tiempo de «software»: Antes de proceder al desarrollo de los programas, es necesario efectuar una estimación del desarrollo del «software», en cuanto al tiempo, con objeto de estar seguros de que con el microprocesador disponible —o elegido—, el juego de instrucciones y el sistema de implementación pensado, se cumplen los requerimientos del «software», necesarios para el proyecto con el «hardware» disponible.

Desarrollo de programas: Bajo este apartado, hemos creido oportuno agrupar una serie de aspectos que, si no transcurren en orden cronológico al expresado, sí son los que son necesarios dar en la elaboración de los programas y puesta a punto; éstos son:

- Diseño del programa
- Codificación
- Depuración
- Prueba

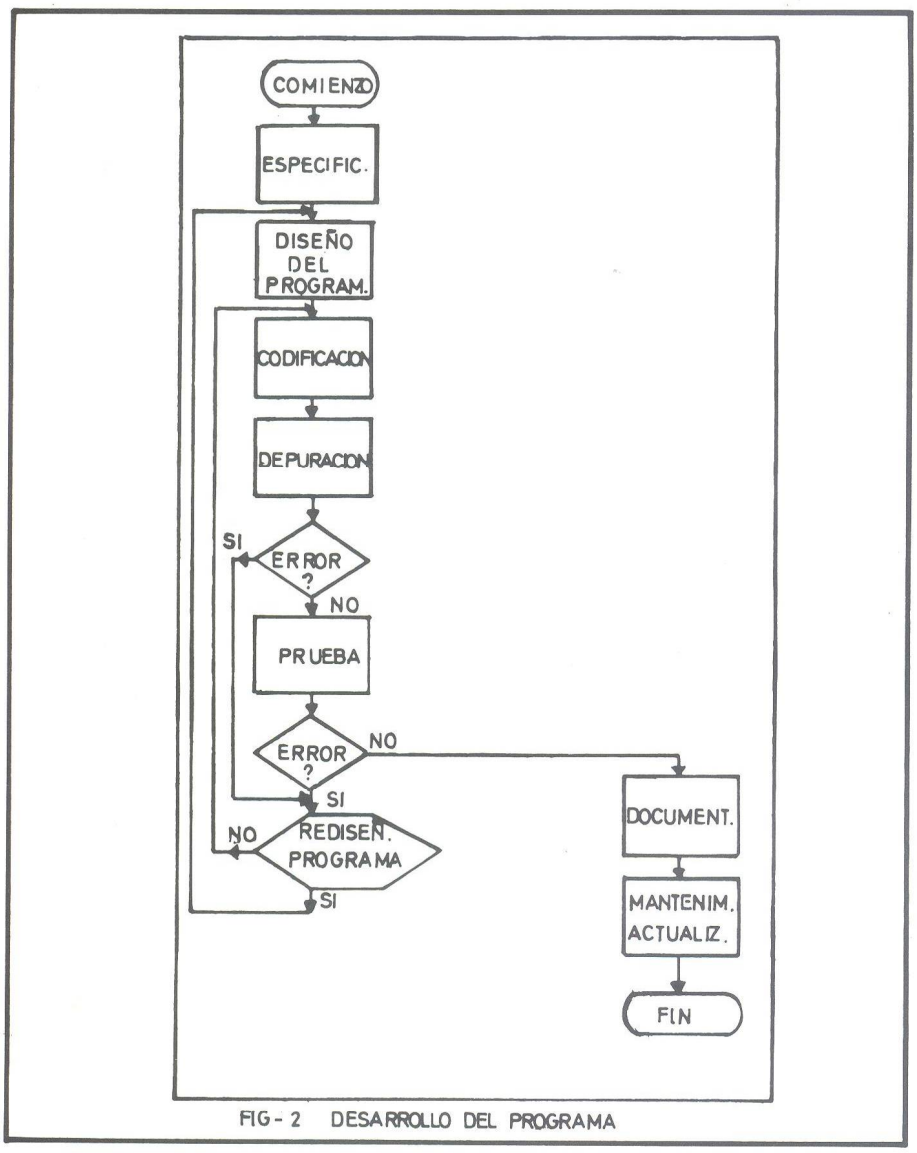

FIG-2 DESARROLLO DEL PROGRAMA

— Documentación
— Mantenimiento

Diseño del programa: En esta etapa se trata de perfilar el programa del microsistema, para la realización de tareas definidas, de forma tal que posteriormente pueda ser convertido fácilmente en el programa propiamente dicho. Entre las técnicas usadas en esta fase, tenemos:

— Utilización de ordinogramas
— Programación estructurada
— Programación modulada
— Diseño top-down

Todas ellas tienen principios comunes; muchos de los cuales son los mismos que se aplican a cualquier clase de diseño, tales como:

— Proceder en etapas pequeñas. No tratar de hacer demasiadas cosas al mismo tiempo.
— Dividir los trabajos largos en pequeñas tareas, lógicamente separadas. Hacer estas tareas independientes unas de otras, siempre que sea posible, lo que permitirá probarlas separadamente y, de esta forma, siempre que sea necesario realizar cambio en una de ellas no afectará a las demás
— Realizar el flujo de control tan sencillo como sea posible, lo que facilitará la detección de errores
— Hacer descripciones gráficas siempre que sea posible
— Enfatizar el principio de la claridad y simplicidad
— Proceder de manera minuciosa y sistemática, utilizando procedimientos estándares.
— No utilizar métodos de los que no se esté seguro o usarlos cuidadosamente
— Tener siempre presente que el sistema debe ser depurado, probado y mantenido
— Tener todo el diseño terminado antes de proceder a su codificación
— Prestar atención especial a los factoes que pueden ser modificados

Organigramas: La utilización de los organigramas es el más cómodo de los métodos de diseño de programas. La ventaja principal es una representación gráfica y, como tal, mucho más intuitiva que una descripción por medio de palabras. Asímismo, a través de él, el proyecto puede ser visualizado como un conjunto, al mismo tiempo que permite ver las relaciones entre sus diversos componentes; en la figura 3 damos los símbolos usados en la confección de ordinogramas. De forma general y resumida, podemos decir que la confección de ordinogramas resulta de utilidad:

— Como documentación del programa, puesto que tiene una simbología estandar y son comprensibles para personas sin una formación en la programación
— Como una herramienta de diseño

Sin embargo, la utilización de organigramas no establece más que un esbozo del comienzo del proyecto, ya que su depuración es difícil y muchas veces es más complicado su diseño que el programa mismo.

Programación modular: En proyectos complejos la utilización de organigramas como herramienta de trabajo no es demasiado satisfactoria. Sin embargo, la especificación del problema y el uso de organigramas puede aportar las ideas necesarias para dividir el proyecto en tareas parciales o módulos; esto es conocido como «Programación modular». Evidentemente, el problema a resolver es cómo dividir el programa en

FIG- 3 SIMBOLOGIA DE LOS ORGANIGRAMAS

Programación estructurada: Es una técnica en la que cada parte del programa consta de elementos de un limitado grupo de estructura y cada una de estas estructuras tiene solamente una entrada y una salida, lo que establece una clara secuencia de operaciones y al mismo tiempo permite aislar los errores. Ejemplo de una de éstas en fig. 4.

Este método, al ser muy sistemático, podemos decir que es útil de forma general en las siguientes situaciones:

— En programas largos
— En aplicaciones en las que la disponibilidad de memoria no sea crítica
— En aplicaciones de manipulación de cadenas de caracteres, control de procesos

Diseño «Top-down»: Una vez dividido el proyecto en tareas parciales y realizadas éstas, se debe proceder a probar cada módulo o estructura independiente de las demás y enlazar unas con otras. El procedimiento estándar conocido como «Bottom-up» requiere un trabajo extra en la prueba y depuración, dejando la tarea de módulos y cómo enlazar todos ellos.

Como resúmen, podemos decir que la programación modular es útil si se siguen las siguientes reglas:

— Usar módulos ni excesivamente cortos ni excesivamente largos
— Tratar de hacer módulos razonablemente generales
— Emplear el tiempo necesario en módulos que sean útiles en otros programas o en muchas partes del programa a desarrollar
— Aislar los módulos diferentes y separarlos lógicamente, siempre que ello sea posible
— Restringir el flujo de información entre módulos e implantar cada decisión de diseño en un simple módulo
— No modularizar las tareas sencillas

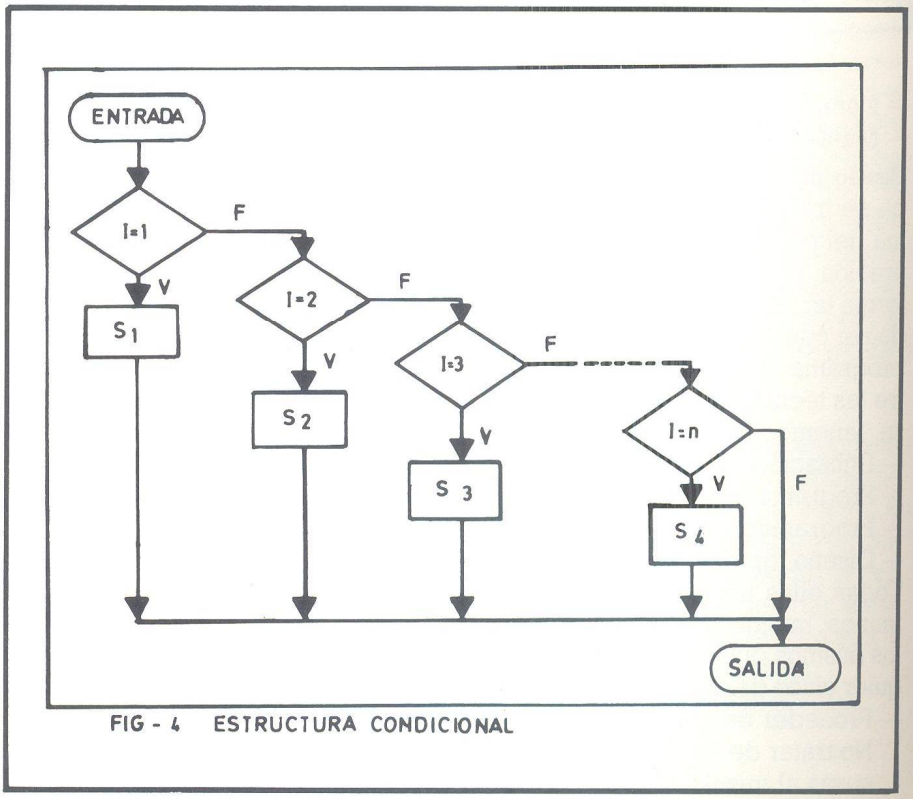

FIG - 4 ESTRUCTURA CONDICIONAL

integración para el final. Es decir, que el método «bottom-up» primero presenta una estructura de solución al problema en términos de diferentes módulos; luego construye la solución final por encadenamiento de submódulos, bien definidos, para construir módulos de nivel superior. El diseño «top-down», por el contrario, intenta, a partir del tratamiento general del programa, la obtención de tratamientos más elementales, subordinados cada uno de ellos al tratamiento general de superior jerarquía en el contexto del problema. Cada uno de estos tratamientos, a su vez, puede descomponerse en otras funciones más elementales, subordinadas a ellos, dando lugar al gráfico arborescente de la figura 5.

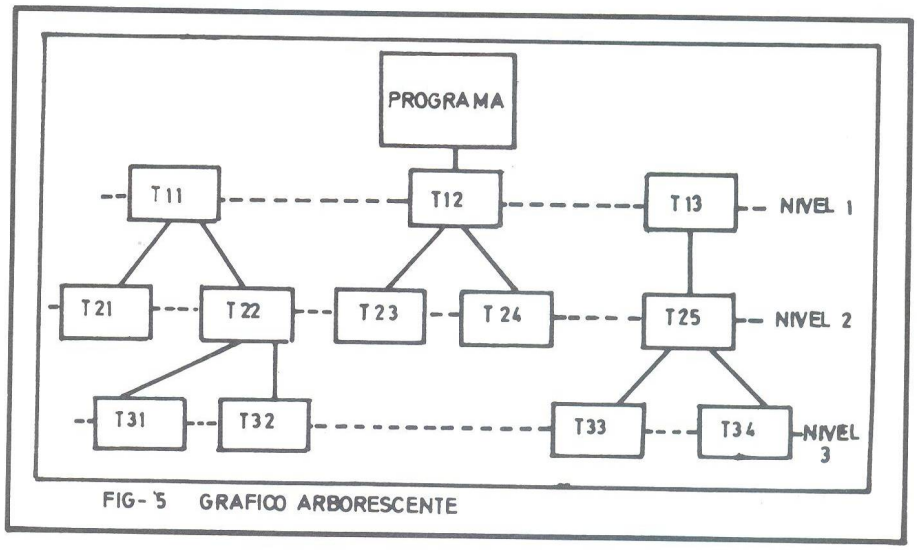

FIG- 5 GRAFICO ARBORESCENTE

Codificación: Una vez elaborada la estructura del «software» del microsistema, se debe elegir el lenguaje de programación en el que se desarrollarán los programas del proyecto. Debido a que el microprocesador sólo entiende el lenguaje máquina (binario) y éste es muy oneroso en su manejo, existen dos niveles de lenguaje: ensamblador y lenguaje evolucionado de alto nivel.

Lenguaje ensamblador: Este tipo de lenguaje permite escribir cada instrucción de lenguaje máquina en forma nemónica o simbólica. La programación en ensamblador requiere la manipulación de registros de microprocesador y circuitos de E/S, bits de estados, etc., lo que obliga a un buen conocimiento de la estructura «hardware» del sistema, aunque, gracias a ello, se logra el aprovechamiento máximo de las posibilidades del microprocesador. La rigidez de las reglas de sintaxis del lenguaje ensamblador obliga a documentar el programa, pues el texto será poco informativo. La principal ventaja de los ensambladores respecto de los lenguajes de alto nivel, radica en la eficacia, tanto en el tiempo de ejecución como en ocupación de memoria.

Lenguaje de alto nivel: Son lenguajes de tipo Basic, PL/M, etc., que permiten al usuario especificar el algoritmo mediante instrucciones más complejas y potentes. Los lenguajes de alto nivel emplean instrucciones que normalmente se asemejan mucho al vocabulario usado en la aplicación para definir el algoritmo. Las ventajas que ofrece frente al lenguaje ensamblador, en cuanto a programación, son:

- Mejor control del software, por empleo de técnica de programación estructurada y modular
- Gran sencillez de escritura.
- Su estructura autodocumentada.
- Independencia frente al microprocesador, lo que permite definir y construir programas sin considerar el microprocesador a implementar físicamente.

Es posible en ocasiones la adopción de soluciones mixtas, utilizando el lenguaje ensamblador para las tareas de precisión y el lenguaje de alto nivel para los puntos de desarrollo donde no existen problemas de tiempo y/o memoria, siempre que se mantenga dentro de los límites de eficacia aceptables.

Depuración y prueba: Realizada la función de traducción, se dispone ya de un programa objeto, que puede ser ejecutado por el microprocesador, sobre el cual debe efectuarse la serie de pruebas previstas para asegurar que el programa funcione correctamente. Esta fase de desarrollo, conocida como depuración y prueba, es, sin duda, la que ocupará un mayor tiempo en el proceso de desarrollo «software». Para la realización de las pruebas es posible utilizar programas de apoyo para la depuración del programa. Entre las herramientas de depuración utilizadas, podemos citar:

- Ejecución paso a paso
- Establecimiento de puntos de prueba (Break points)
- Programa de acceso a los registros del microprocesador
- Programa de acceso a memoria

Reglas prácticas para el diseño «top-down»

- Llevar la descomposición hasta sus últimas consecuencias.
- Diferir decisiones referidas a detalles concretos, tanto como sea posible.
- Considerar las posibles alternativas a una decisión determinada.
- Escoger en primer lugar la decisión más fácil.
- Tomar el menor número de decisiones en cada paso del diseño.

Un paquete de documentación «software» de un microsistema ha de incluir, al menos:

— Organigramas generales
— Descripción escrita del programa
— Lista de parámetros y definiciones
— Mapa de memoria
— Listado de documento del programa
— Plan de pruebas y resultados

Así mismo:
— Organigrama de datos
— Programas estructurados

Todo ello conforma el paquete de documentación mínima aceptable. Sin embargo, existen proyectos que exigen un mayor esfuerzo en la documentación «software», por lo que exige la elaboración de tres documentos esenciales:
— Manual de lógica programada.
— Guía del usuario.
— Manual de mantenimiento.

— Establecimiento de «Trazas»; es un registro de los pasos previos efectuados por el programa

Documentación y actualización: El desarrollo del «software» exige no solamente la elaboración de un determinado programa, sino que éste debe ser suficientemente documentado, ya que ésta es una parte importante del producto final. La documentación no sólo es una ayuda para las fases de prueba de depuración, sino que puede ser esencial para el posterior uso y actualización del programa. Las soluciones a articular para conseguir estos objetivos son:
— Programas autodocumentados
— Inclusión de comentarios
— La utilización de organigramas como documentación
— Programación estructurada como documentación
— Mapas de memoria
— Lista de parámetros de definiciones
— Librería de subrutina

Desarrollo de hardware:

Como hemos visto, la realización de un proyecto electrónico basado en microsistemas es un proceso simbiótico entre elementos «software» y «hardware»; así pues, en base a esto y dado que los elementos «software» los hemos expuesto en los apartados anteriores, pasamos a indicar de una forma somera los elementos a considerar en el desarrollo del «hardware», que son:
— Especificación funcional del sistema
— Diagrama de bloques
— Análisis de los requerimientos del sistema
— Selección de microprocesador
— Definición de los interfases «hardware» y de las funciones «software»
— Prueba del «hardware» con rutinas de test y programas OFF-LINE
— Prueba conjuna de «software y hardware»

Especificación funcional del sistema: En esta primera fase se trata de especificar el sistema de entradas y salidas (E/S) y establecer, de una forma grosera, las funciones que el mismo debe realizar. Se puede pensar en algunas limitaciones primarias que el sistema debe poseer, como pueden ser: tamaño, consumo, autonomía, peso, etc. Esta fase se podría identificar con el establecimiento de un pliego de condiciones que el sistema debe cumplir.

Diagramas de bloques: Esta fase puede ser una continuación de la anterior, en la cual se desciende ya a más detalles en cuanto a estructura y funcionamiento del sistema. Se especifica de una manera clara el sistema de entradas y salidas. Coincide esta segunda fase con el tradicional diagrama de bloques, utilizado en el diseño convencional.

Análisis de los requerimientos del sistema: En esta tercera fase se desciende ya a nivel de detalle. Se especifican tiempo, niveles de

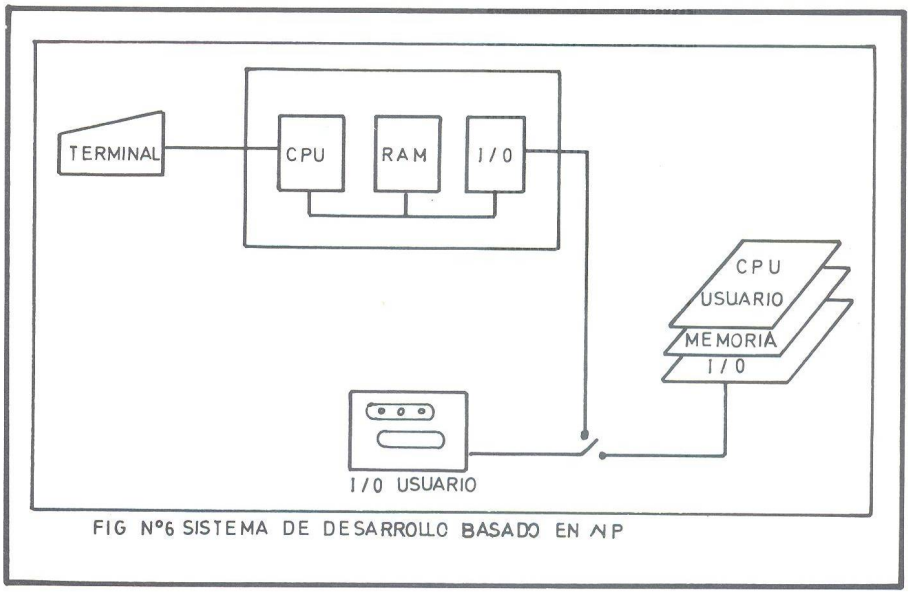

FIG Nº6 SISTEMA DE DESARROLLO BASADO EN NP

interrupción, requerimientos estimados de memoria, etc. Un factor que hay que cuidar mucho, pues debe ser definitivo a la hora de la elección del microprocesador, es la exigencia del tiempo del sistema para su trabajo en tiempo real. Dicho de otro modo, cual debe ser la duración del ciclo de instrucción del microprocesador que elijamos, para que sea capaz de realizar las funciones en tiempo real.

Selección del microprocesador: El microprocesador que elijamos debe reunir las siguientes tres condiciones fundamentales:

— Debe ser capaz de ralizar la función para la cual lo hemos elegido
— Debe tener disponibilidad presente y futura
— Los costos por unidad de producción, más los costos de amortización del desarrollo, han de ser mínimos

Existen diferentes consideraciones o factores que influyen favorablemente o de modo desfavorable en las tres exigencias anteriores, que son:

a) Longitud de la palabra:
4 bits
8 bits
16 bits
32 bits

La elección del número de bits es fundamental, pues influye en la velocidad, número de instrucciones necesarias, etc.

b) Juego de instrucciones y modos de direccionamiento:
Interesa que ambos sean lo más completos posible, puesto que:

— Con pocas instrucciones se puede implantar una función
— Se dispone de un acceso eficiente a los datos
— Las operaciones resultan más rápidas
— Se reduce la cantidad de programa almacenado

c) Arquitectura de la CPU:
Debe estar configurada de forma que tenga:

— El mayor número de registros
— Buen sistema de E/S
— Stock interno o externo

d) Tiempo de ciclo de instrucción:
Este es un concepto que es relativo a las necesidades de tiempo de diseño. Conviene examinar el ciclo de instrucción de todas las instrucciones.

e) Capacidad de interrupción:
Debemos observar las siguientes consideraciones:

— ¿Un sólo «pin» o varios «pins» de interrupción?
— ¿Facilidad de vectorización?
— ¿Se guardan automáticamente los registros?
— ¿Cúal es el tiempo de respuesta a las peticiones de interrupción?

f) Familia de chips.— Soporte «hardware» del microprocesador:
En muchos casos es esencial disponer de este apoyo. Debemos preocuparnos de si la familia dispone de:

— Interfases para los «buses»
— Interfases de E/S. (Paralelo y serie)
— Clock
— «Timer»
— etc.

g) Elección de la memoria:
La memoria suele ser el costo mayor del sistema; por lo tanto, la elección del microprocesador debe estar supeditada a la misma. Debemos investigar:

— Necesidad de memoria especial o estandar
— Memoria rápida o lenta
— Si se dispone de mecanismos que permitan la inclusión de memorias lentas
— Si las memorias que son necesarias tienen segundas fuentes.

h) Requerimientos de tensión:
— Número de tensiones (1, 2, 3...).
— Consumo de cada uno de ellos.

i) Ayudas que posee, tanto para el software como para el hardware en el desarrollo:

— Manual de programación
— Literatura de aplicaciones
— Soporte de vendedores
— Assembler, simulator
— Compilador, sistema operativo en disco
— Sistema de desarrollo

j) Condiciones de funcionamiento:
— Rango de temperatura
— Inmunidad al ruído
— Vibraciones, choques, márgen de humedad

k) Segunda fuente:
Este dato es fundamental por razones obvias. Teniendo en cuenta la influencia de cada uno de los factores en las exigencias de éste, es como debemos hacer la elección del microprocesador.
El precio del microprocesador en sí no debe ser un condicionante, pues es despreciable frente al precio de interfases y memorias.

Definición de las interfases «hardware» y de las funciones «software»: En realidad estas dos fases son complementarias, pues hay que tener en cuenta que lo que no se haga por «hardware», se deberá hacer por «software» y viceversa. Siempre que la velocidad del procesador nos lo permita, debemos procurar hacer el mayor número de funciones por «software». Esto influirá fundamentalmente en el costo final del sistema y permitirá hacer un mayor número de modificaciones debido a la gran flexibilidad del «software».
Se puede comprender fácilmente la íntima relación que existe entre «soft» y «hard» en los sistemas diseñados con microprocesador. Esto va a implicar la necesidad de que las personas responsables de estos diseños reunan una serie de características especiales, «soft-hard».

Prueba del «hardware» con rutinas de test y programas «Off-Line»: En esta fase se prueba por separado el «software» y el «hardware». Los sistemas de desarrollos para microprocesadores actuales disponen de las facilidades convenientes para que esto pueda realizarse.
Ello servirá para observar, por separado, las deficiencias de ambos y corregirlas. Si no se hiciera así, serían más difíciles de descubrir estas deficiencias, pues no podríamos estar seguros de que el error fuera atribuible al «software» o al «hardware»; al menos, no en todos los casos.
Esta fase servirá también para conocer las peculiaridades del «hard» y del «soft» y poder así pasar a la fase definitiva de la prueba conjunta.

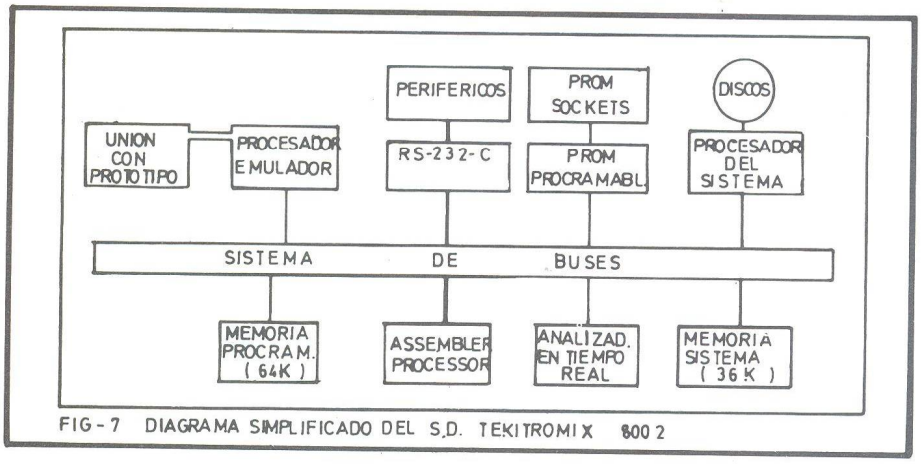

FIG-7 DIAGRAMA SIMPLIFICADO DEL S.D. TEKITROMIX 8002

Prueba conjunta de «software» y «hardware»: Esta es la etapa definitiva y no deben surgir grandes problemas si se han respetado convenientemente todas las fases anteriores. Los equipos de desarrollo de los sistemas de microprocesadores actuales disponen de unos sistemas llamados ICE (In Circuit Emulator), que ayuda de una forma notable a esta fase final; permite una forma de operación híbrida entre el sistema de desarrollo y nuestro prototipo y así puede detectarse cómodamente la existencia de posibles deficiencias u olvidos.

Perfil de equipo para el desarrollo de estos proyectos

Describiremos, en forma somera, las características «soft-hard» de las personas dedicadas a la implementación de proyectos basados en microprocesadores.
A título orientativo, vamos a enumerar las características que se deberían considerar como más adecuadas para la constitución de un equipo:

- La persona encargada del software debe tener suficiente capacidad para comprender el hardware de los elementos externos que van a ser controlados
- La persona encargada del hardware deberá poseer los suficientes conocimientos del funcionamiento de ordenadores, programa, almacenado, tiempo real, etc. De esta forma logrará formar un tandem con la persona encargada del software y será capaz de buscar la solución más adecuada al problema
- A veces y en sistemas sin excesiva complicación, puede ser la misma persona la que se encargue del hardware y del software. Se debe tener en cuenta aquí que es normalmente más fácil para un hombre hardware aprender software que para un especialista en software aprender hardware
- Para aplicaciones en las cuales haya una gran complejidad de software y los sistemas de E/S estén muy standarizados, es mejor que el personal esté más familiarizado con la programación en tiempo real y el lenguaje ensamblador
- La formación de un equipo para proyectos de sistemas basados en microprocesadores, que sólo tenga experiencia en grandes ordenadores y en lenguaje de alto nivel, puede resultar peligroso. La capacidad y paciencia para «pensar en detalles» son importantes atributos que deben poseer las personas que se dediquen a este trabajo
- En la primera fase del proyecto el número de personas ha de ser pequeño

Aspectos a considerar en el empleo de microsistemas en proyectos electrónicos

Pasamos a relacionar una serie de razones que pueden ayudar a la hora de tomar la decisión sobre la conveniencia o no de implementación de proyectos electrónicos basados en microprocesadores.
Ventajas:

- Los costes de fabricación de los productos pueden descender significativamente
- Los tiempos de desarrollo se pueden reducir considerablemente
- La capacidad de reacción es mucho mayor, es decir que los productos pueden ser lanzados al mercado antes que otros competidores, que hagan sus diseños de la forma tradicional
- Aumento en la fiabilidad al poder sustituir parte del hardware y su interconexionado por software
- El aumento de la fiabilidad permite aumentar el plazo de garantía de los productos
- Reducción de los gastos de documentación

Desventajas:

- Necesidad de hacer una inversión inicial para obtener el equipo de desarrollo
- Las características formativas que ha de reunir el personal
- Menor facilidad para detectar fallos y su consiguiente problemática en el mantenimiento

Bibliografía:

1.— Allocca J.A.: «Electronnic Instrumentation»; Prentice Hall, 1983.
2.— Angulo J.Mª: «Microprocesadores.— Diseño Práctico de Sistemas»; Paraninfo, 1983.
3.— Mandado E.: «Procesadores Programables»; Marcombo, 1979.
4.— Muñoz E.: «Microprocesadores», Tomos I, II, III. Departamento de Publicaciones E.T.S.I.T. Madrid.
5.— Roonay Z.: «Microprocesadores; Marcombo, 1980.