Conciencia Tecnológica No. 37, Enero - Junio 2009

Diseño de un Modelo de Sistema Digital para Desarrollo Didáctico, sin Memoria no Volátil

Investigación

Ing. José de Jesús Soto Parra
Libertad No. 8, Cd. Cuauhtémoc, Zac.
jjsoto_99@yahoo.com, jjsoto_99@hotmail.com

Resumen

El artículo presenta los fundamentos para la construcción de un sistema de apoyo didáctico para equipos electrónicos digitales basados en procesador (microprocesador/microcontrolador) cuya característica principal es la no utilización de memoria ROM en cualquiera de sus presentaciones. Por sus siglas en inglés, un SDK (System Design Kit), está diseñado como una herramienta didáctica de alta prestación, en la cual es posible depurar aplicaciones, probar prototipos ó bien, como una herramienta que permite probar las capacidades de un procesador específico. Las principales ventajas de éste modelo son sistema multiprocesador que puede operar bajo arquitectura Harvard ó Von Newmann, sin hacer uso de los recursos tanto de hardware como de software del procesador; no requiere de sistema operativo (programa monitor), la memoria tanto de código como de datos es totalmente disponible para el usuario.

Palabras clave: microprocesador, microcontrolador, multiprocesador, SDK, arquitectura Harvard, Arquitectura Von Newmann, SDK sin ROM, SDK sin monitor.

Abstract

This paper presents the foundations for the construction of a didactic support system for digital electronic systems based on processor (microprocessor or microcontroller) whose main characteristic is ROMLESS in anyone of its presentations. For their initials in English, a SDK (System Design Kit), it is designed as a didactic tool of high benefit, in which is possible to debug applications, to test prototypes or well, like a tool that allows to test/know the capacities of a specific processor. The main advantages of this model are multiprocessor system that can operate in Harvard or Von Newmann architecture, without making use of the so much resources of hardware like of software of the processor; it does not require of operating system (program "monitor"), all the memory is available to the user completely.

Keywords: Microprocessor, Microcontroller, Multiprocessor, SDK, Harvard architecture, Von Newmann Architecture, SDK ROMLESS, SDK without monitor.

Introducción

Actualmente existe una gran variedad de microprocesadores/microcontroladores –en lo sucesivo los llamaremos procesadores- orientados a satisfacer necesidades específicas del mercado, para lo cual es necesario contar con personal capacitado para cubrir las necesidades de desarrollo y/o producción de los sectores involucrados.

Para cada procesador específico debe construirse un sistema de desarrollo (SDK) único, conformado por una tablilla de circuito impreso que contiene el procesador en cuestión, un programa monitor grabado en algún tipo de memoria ROM, memoria RAM para aplicaciones del usuario, circuitos digitales/analógicos adicionales para operar el sistema y un software de soporte.

También se cuenta con programas (software) capaces de emular el funcionamiento de varios procesadores cuya característica principal es que no pueden ejecutar programas en tiempo real y no se pueden interconectar a otros dispositivos.

Desarrollo

En el caso de un SDK tradicional, el monitor consume recursos del sistema, redirecciona el vector de interrupciones del procesador y esto ocasiona que pueda ser necesario que redireccionemos nuestro programa de prueba tal como se muestra en la figura 1.

Adicionalmente, el usuario debe conocer la estructura y funciones del monitor para evitar alguna interferencia del programa a probar con los elementos del monitor.

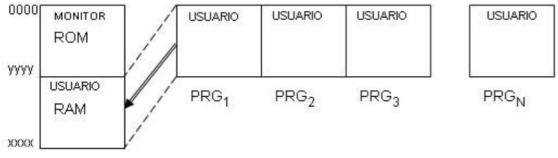

Figura 1. Redireccionamiento de programas del usuario

Cabe mencionar que el monitor consume recursos tanto de **hardware** como de **software** del sistema. Por otra parte, es necesario tener un conocimiento claro de la ubicación y funciones del MONITOR, lo cual, en otras palabras, implica que antes de aprender acerca del **procesador**, tenemos que aprender acerca del **SDK**.

Una vez que hemos depurado nuestro programa, hacemos las correcciones necesarias en el direccionamiento de memoria (ver figura 2) y mediante un programador de memorias (conectado a una PC), programamos nuestra ROM y construimos nuestro sistema particular.

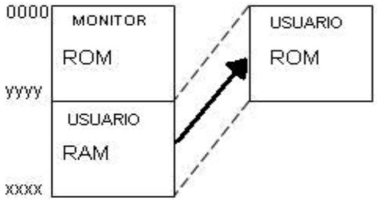

Figura 2. Redireccionamiento de aplicación del usuario

Si deseamos realizar una modificación a nuestro programa, tenemos que programar otra ROM, o bien, dependiendo del tipo de ROM, podemos borrar su contenido y volverla a programar, como se muestra en la figura 3. Un tipo muy común de memoria que puede reprogramarse, requiere de un borrador de luz ultravioleta para borrar el contenido de la misma en un periodo de 20 minutos aproximadamente.

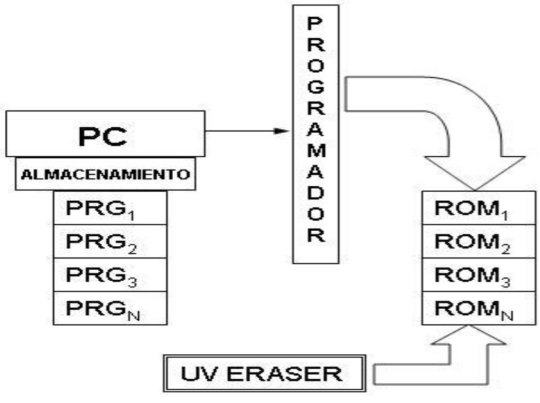

Figura 3. Programación de variantes de programa

Para cada variante de nuestro programa, necesitamos programar una memoria ó bien reprogramar la que tengamos. Este modelo requiere de un programador y un borrador de luz ultravioleta. Para el caso de utilizar memoria flash, se requiere el software/hardware de soporte acorde al dispositivo a utilizar.

Por otra parte, se tiene una gran dependencia del exterior en cuanto al suministro de herramientas de desarrollo para éste tipo de elementos.

Para superar las limitantes que impone el sistema de desarrollo tradicional, se ha diseñado y construido un modelo de SDK multiprocesador (incluido el software –programas- de soporte necesario), cuya característica principal es que no cuenta con memoria ROM, con todas las ventajas que ello implica. Todo el espacio de memoria del procesador está disponible para el usuario.

La figura 4 muestra los elementos básicos de un SDK multiprocesador sin memoria ROM, la cual ha sido sustituida por una memoria RAM (para una mejor comprensión de los términos se recomienda una lectura del apéndice).

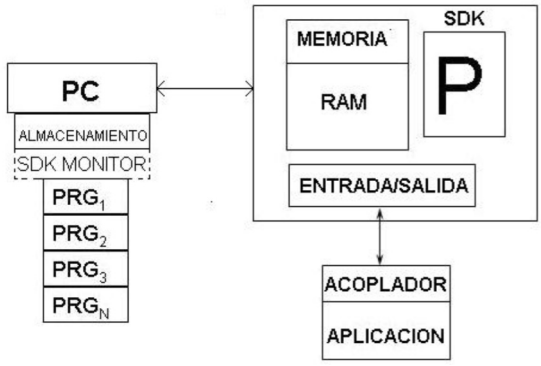

Figura 4. Elementos de un SDK sin memoria ROM

Al no tener ROM, por definición carece de sistema operativo (MONITOR), y en cierta forma, se puede decir que el sistema operativo del SDK reside en el exterior (en este caso, en la PC), tal como se muestra en la figura 5.

Por su operación, a este modelo se le ha denominado **RLEOS** (por sus siglas en inglés: **R**om **L**ess with **E**xternal **O**perative **S**ystem).

El software –programas- de soporte (PC) maneja una imagen de la memoria del SDK, la cual puede transferirse desde y hacia el SDK.

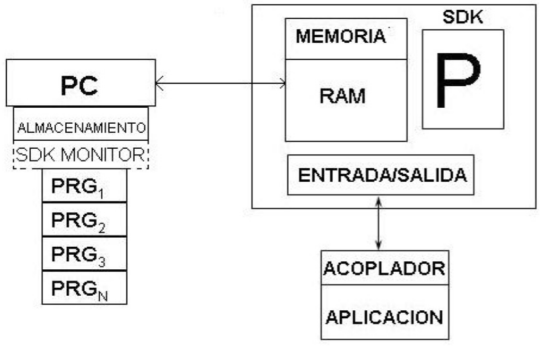

Figura 5. Ubicación del monitor del SDK sin ROM

Mediante la PC accesamos el SDK e intercambiamos instrucciones y datos, tal como se muestra en la figura 6. Posteriormente el SDK ejecuta el programa.

Se pueden crear rutinas que permitan registrar el estado interno del procesador, similares a las de un SDK clásico e incluirlas en programas de aplicación, con fines de depuración.

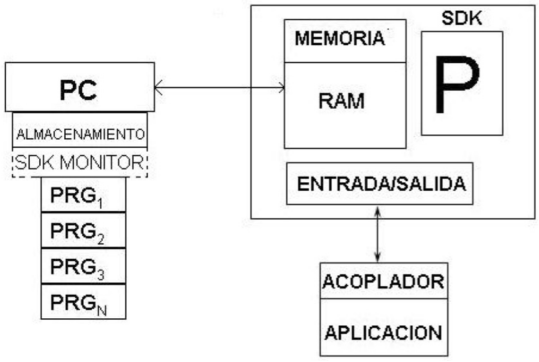

Figura 6. Acceso a la memoria del SDK sin ROM

En cuanto a la sección de memoria, podemos tener tantos programas como variantes del mismo deseemos, tal como se muestra en la figura 7.

Figura 7. Mapa de memoria del SDK sin ROM (RLEOS)

No tenemos que hacer ningún redireccionamiento, ya que el **modelo RLEOS** no consume recursos del sistema, tanto de hardware como de software.

De ésta forma, podemos crear, depurar y ejecutar programas sin necesidad de un programador de memorias ni de un borrador de luz ultravioleta (ver figura 8).

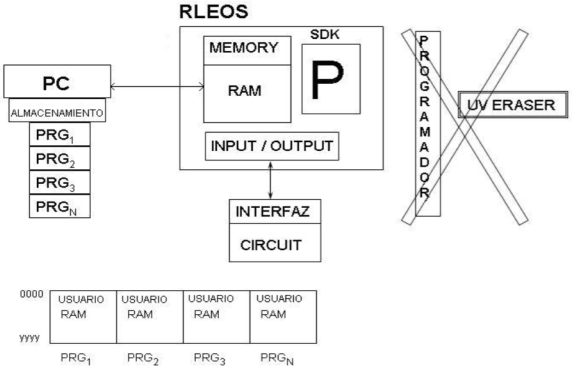

Figura 8. Requisitos de hardware/software del RLEOS

Otra gran ventaja que tiene el modelo **RLEOS,** es que el procesador puede ***sustituirse*** por otro que tenga la misma distribución de terminales (pin-out, en inglés) o bien mediante un acoplador para el caso de una distribución de terminales diferentes, con la misma versatilidad, (ver figura 9); es decir, el modelo RLEOS es multiprocesador, ya que cada variante del procesador requiere, en el esquema clásico del SDK, un monitor específico, no así en el RLEOS, en virtud de carecer de ROM.

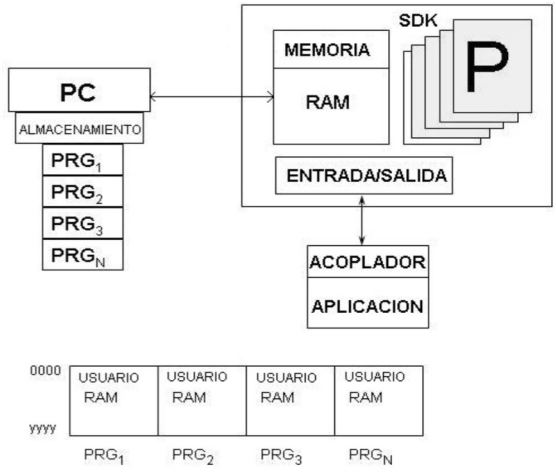

Figura 9. Característica multiprocesador del RLEOS

Otra gran ventaja del modelo RLEOS, es que el software de soporte tiene la capacidad de acceder directamente (y por lo tanto, manipular) cualquier dispositivo ligado a el/los buses de memoria del sistema.

Hasta éste momento solo se ha planteado un SDK bajo el modelo **RLEOS**, sin hacer referencia a un procesador en especial. Para efectos de demostración, se construyó un SDK basado en el µC **8031** (familia MCS´51) y se diseñó un programa que permite operar el dispositivo como un SDK clásico y otro programa que no utiliza las funciones del SDK clásico, lo cual permitió experimentar las ventajas del modelo **RLEOS.**

El programa que opera como SDK clásico tiene integrada una rutina de registro/recuperación del estado del procesador, una rutina de retardo de tiempo y una rutina de ejecución paso-a-paso (Single Step en inglés), dependiente de un temporizador interno, haciendo uso del vector de interrupciones.

En este SDK, se puede reemplazar el 8031 por un 8032 (el cual tiene 128 bytes más de RAM interna, un temporizador con su respectiva interrupción) y explotar plenamente sus capacidades sin necesidad de programar algún tipo de memoria ROM. Cabe mencionar que un 8031/32 tiene 32 terminales de entrada-salida, de las cuales 16 se utilizan en el manejo de memoria externa (sin hacer uso del área de datos, usando solamente el área de código); pues bien, de las 16 restantes el

modelo **RLEOS** no utiliza ninguna para su operación, quedando totalmente disponibles para el usuario.

El software –programas- que se requiere para operar el RLEOS, es bastante funcional (tiene presentación en texto normal y presentación en simulación de 7 segmentos), de tamaño reducido (ocupa menos espacio del proporcionado en un disket de 5-¼ pulgadas –obsoletos-), generando archivos en formato binario y formato Intel, maneja 2 bloques de memoria de 64KB (64KB code y 64KB data), incluye funciones para examinar memoria byte por byte y por bloques de 256 bytes, transferencia a y desde el SDK, salvaguarda de información en formato binario y formato Intel y para el caso de manejo de imagen de registros del procesador, una sección que permite una visualización comprensible de éstos (se cuenta con software para crear un archivo descriptor de registros, el cual es utilizado por el software principal).

Adicionalmente, el **RLEOS** opera bajo la arquitectura Harvard ó Von Newmann; en el caso de los procesadores de la familia MCS-51, se cuenta con diseños que pueden operar, en un mismo modelo, exclusivamente –a elección del usuario- en ambas arquitecturas; bajo la arquitectura Harvard, se tienen disponibles los 16 pines de los puertos 1 y 3, bajo la arquitectura Von Newmann, se disponen solo de 14 pines (8 en P1 y 6 en P3),. La aplicación más compacta del RLEOS requiere solamente de 5 circuitos integrados (chips): el procesador, la memoria RAM, el latch para direcciones y 2 chips de soporte, de la familia 74xxx. Bajo ningún modelo, en este procesador, se contemplan chips más complejos (PAL´s u otros) que la familia 74xxx. Cabe aclarar que la memoria RAM utilizada NO es de doble puerto o algún tipo especial de memoria RAM, es una memoria RAM ordinaria (5116, por ejemplo).

Conclusiones

El modelo RLEOS rompe un paradigma acerca de la construcción de un SDK, en cuanto a la sección de memoria. La computadora ligada al RLEOS no requiere de gran sofisticación; se ha probado plenamente con una PC basada en el 8086 (de las primeras que aparecieron en el mercado), un desempeño óptimo se consigue con una PC basada al menos en el 80486. Lo anterior hace del RLEOS una opción bastante atractiva para la construcción de un SDK multiprocesador que proporcione al usuario una disposición total del espacio de memoria, así como el control total del vector de interrupciones y cualquier otra prestación que ofrezca el procesador elegido.

El RLEOS es un modelo de SDK que no usa ROM en cualquier presentación; puede operar bajo la arquitectura Harvard ó Von Newmann, utilizando memoria estándar. Permite un acceso total tanto a la memoria de código como a la memoria de datos, sin utilizar el vector de interrupciones ni sistemas de comunicación del SDK.

Apéndice

En ésta sección se describen los fundamentos de construcción y operación de un SDK (System Design Kit).

Para efectos de éste trabajo, utilizaremos la siguiente terminología:

- **Procesador** (P): Nos referimos a un microprocesador (µP) o a un micro-controlador (µC). Un µC consta de un µP y algunos periféricos integrados en un solo chip el cual puede incluir memoria ROM.
- **ROM:** Nos referimos a cualquier tipo de memoria no volátil (ROM, PROM, EPROM, EEPROM, FLASH, etc.). La ROM no pierde su contenido al retirarle la alimentación; habitualmente es de solo lectura.
- **RAM:** Nos referimos a cualquier tipo de memoria volátil, que se utiliza para almacenamiento temporal. La RAM es de escritura y lectura y pierde su contenido al retirarle la alimentación.

Sistema basado en procesador (SBP): Un SBP se conforma, esencialmente, por los siguientes elementos (ver figura 10):

- Sistema de memoria (incluye la ROM y la RAM). Aquí residen las instrucciones y datos que hacen que el sistema funcione coherentemente para lograr un fin específico. Estas instrucciones y datos forman el programa.
- Procesador. Se encarga de ejecutar el programa que reside en la memoria.
- Periféricos de entrada y salida, mediante los cuales el procesador se intercomunica con el exterior.

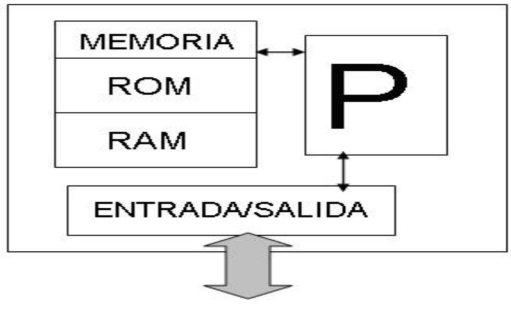

Figura 10. Elementos de un sistema basado en procesador

Cuando el sistema tiene como fin servir como herramienta para entrenamiento, prueba de prototipos y/o depuración de programas, estamos hablando de un **S**ystem **D**esign **K**it (SDK).

Un SDK requiere, además de los elementos básicos de un sistema basado en procesador, de (ver figura 11):

- Un **teclado** para alimentar el programa
- Un **visualizador** para conocer el estado del sistema y
- Una **interfaz** (acoplador) acorde a los requerimientos propios de la aplicación diseñada.

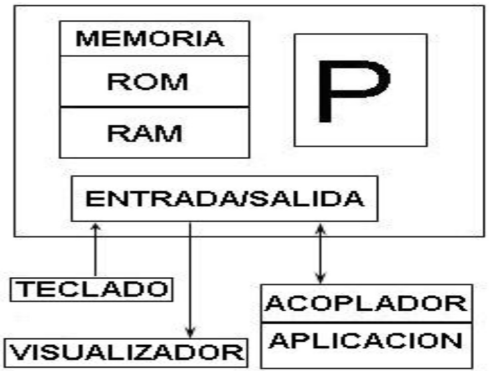

Figura 11. Elementos de un SDK

Para no vernos en la necesidad de teclear a cada vez nuestros programas, podemos utilizar un medio de almacenamiento ligado a una computadora personal **(PC);** aquí podemos tener varios programas almacenados, listos para ser transferidos al SDK, tal como se muestra en la figura 12.

Dado que la PC tiene visualizador y teclado, estos elementos pueden ser opcionales en el SDK cuando esta ligado a una PC.

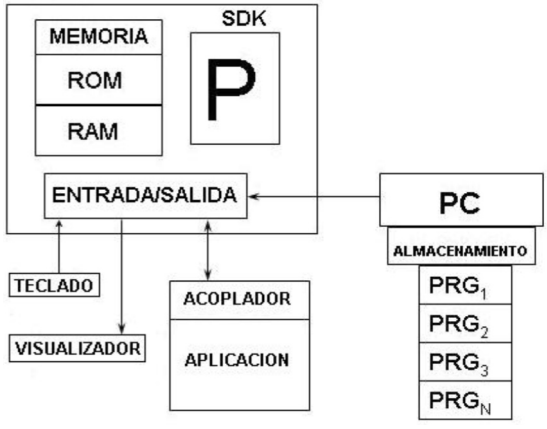

Figura 12. Transferencia de programas desde la PC al SDK

En la sección de memoria tenemos el siguiente arreglo (ver figura 13):

- El Sistema Operativo ó programa que da servicio al SDK **(MONITOR)**, está programado en una **ROM** y ocupa una sección específica de la memoria
- Área disponible para el usuario, en **RAM**. En ésta área se depositan los programas a probar. Debe ser externa al procesador.

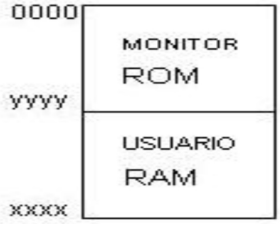

Figura 13. Mapa de memoria de un SDK

Referencias

[1] MCS ® MICROCONTROLLER FAMILY USER´S MANUAL. Order no. 272389-002. 1994.

Artículo recibido: 17 de octubre de 2007

Aceptado para publicación: 20 de mayo 2009