Conciencia Tecnológica No. 32, Julio-Diciembre 2006

Boot Loader para Microcontroladores PIC serie 18

Reporte de Proyecto

Francisco Javier Villalobos Piña, Héctor Ulises Rodríguez Marmolejo, J. Rafael Molina Contreras y Rodolfo Trejo Vázquez
Departamento de Ingeniería Eléctrica Electrónica
Instituto Tecnológico de Aguascalientes Av. A. López Mateos 1801 Ote. Fracc. Ojocaliente FOVISSSTE
Aguascalientes, Ags. Tel:01(449)9105002 ext. 104.

fvillalobospia@yahoo.com

Resumen

El presente trabajo expone el diseño de un boot loader, programa que permanece residente en un microcontrolador PIC18F452. Dicho programa permite grabar cualquier aplicación en el PIC sin el uso de un programador. Esta ventaja, no solo facilita el diseño de aplicaciones de control digital mediante la familia 18 de Microchip, sino que evita que el procesador se monte y desmonte continuamente, debido a lo cual, disminuyen los daños que por mal manejo pudiera sufrir el procesador. Se reporta igualmente el programa en visual Delphi, que permite leer, grabar y ejecutar los archivos hexadecimales del procesador, y probar la integridad del dispositivo

Palabras Clave

Boot Loader, PIC18F452

Introducción

Debido a la flexibilidad y la gran cantidad de recursos de hardware y de velocidad que ofrecen los diferentes modelos de microcontroladores PIC, el diseño de aplicaciones en todos los ámbitos de la ingeniería, se hace cada vez más popular aplicando dichos dispositivos. Mención aparte merece la familia 18, la cual ha sido diseñada para ser programada en lenguaje C, lo que permite diseñar un programa en mucho menor tiempo, pero se tiene la necesidad también de contar con una herramienta que permita descargar la aplicación a la memoria del programa Flash del microcontrolador de manera fácil. Tradicionalmente, la metodología para el diseño, pasa por desconectar el procesador de la tablilla de experimentos o del circuito impreso y llevarlo a algún programador de micros conectado a un computador personal repitiendo el proceso hasta que se completa la aplicación. Este ir y venir del dispositivo en el proceso de programación, con el paso del tiempo, provoca generalmente el deterioro del procesador básicamente por desgaste mecánico, a lo que debe añadirse, la posibilidad de que el procesador se conecte en forma incorrecta debido a la fatiga o el cansancio del diseñador, lo que redundaría en un daño irreversible para el mismo. Buscando dar solución a la necesidad mencionada en párrafos anteriores, Microchip ha diseñado ya herramientas para descargar la aplicación a la memoria del programa Flash del microcontrolador, dichas herramientas presentan sin embargo, el inconveniente de que es necesario reacomodar los vectores del programa principal y de las interrupciones. Con la herramienta que se reporta en este trabajo, se evita dicho reacomodo de los vectores del programa principal y de las interrupciones.

Desarrollo experimental

Para llevar a cabo los trabajos que se reportan en este documento, se utilizó una computadora personal, un microcontrolador PIC18F452 de la familia 18 que permite borrar y grabar la memoria Flash del procesador en tiempo de ejecución**,** y los lenguajes ensamblador y el visual Delphi. Para comunicar el procesador y la computadora personal se utilizó un enlace serial asíncrono RS232 en modo *null-modem* y un convertidor de niveles TTL a RS232.

Resultados

La Fig. 1, muestra el mapa de memoria del PIC18F452 [1], en la que queda residente el programa denominado boot loader. Dicho programa, desarrollado en lenguaje ensamblador para la optimización máxima de código, permanece en la parte alta del procesador y utiliza 7% de la capacidad de memoria disponible para instrucciones. Debido a esto, el usuario pierde una zona de memoria de magnitud 0x7FFF-0X6FC0=0x103F, que traducida en bytes, supone una pérdida de 4159. El hecho de que el programa esté escrito en lenguaje ensamblador, permite además aprovechar la característica de borrar y grabar la memoria flash del procesador en tiempo de ejecución de la familia 18. Dado que este procesador tiene una memoria total de 32K bytes y un core que consume 2 bytes por instrucción, se cuenta en realidad con 16k

instrucciones, menos el 7% de la capacidad de memoria que ocupa el boot loader.

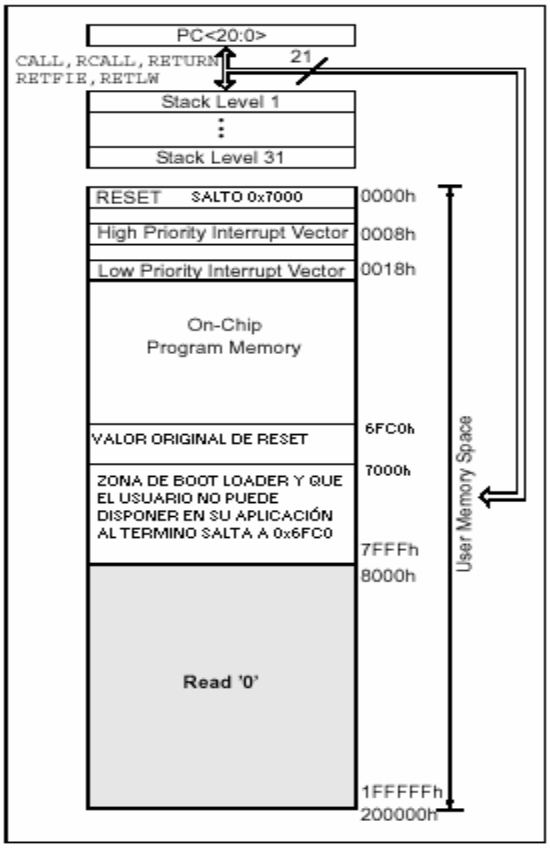

Fig. 1 Organización de la memoria de programa en PIC18F452

La operación del programa básicamente obedece la siguiente rutina: cuando el cpu enciende, salta al vector de *reset*, donde existe una instrucción *goto* a la dirección 0x7000, en la que se encuentra todo el programa que dialoga con la computadora personal para la carga, la lectura, la ejecución y la verificación de la memoria. Una vez que se concluyen estas actividades se hace un salto a la dirección 0x6FC0, donde se guardó el vector del *reset* original de la aplicación de usuario. En dicha dirección se cuenta con un *goto* a una de las zonas de memoria que fueron remplazadas por las del *boot loader* con el objeto de otorgar la dirección al programa de usuario nuevamente. Cuando se tiene el sistema en una plataforma en la que no hay computadora personal, el sistema arranca y trata de comunicarse con la computadora personal, pero si transcurren más de 100mseg sin respuesta, el sistema inmediatamente ejecuta el programa que se tenga almacenado.

La Fig. 2 muestra el diagrama esquemático que se utilizó para comunicar el procesador y la computadora personal. Tal como puede verse, el sistema de comunicación es básicamente un enlace serial asíncrono en modo *null-modem* usando el estándar RS232. El sistema comunicación muestra como se puede apreciar, un procesador serie 18 con su interconexión a las terminales RC6/TX y RC7/RX del puerto serial y un convertidor de niveles TTL a RS232. El convertidor de niveles TTL a RS232 que se utilizó en el caso que nos ocupa fue el MAX232. Cabe hacer notar que también se utilizó un circuito denominado USB to RS232 converter con el objeto de resolver el problema de los equipos que ya no cuentan con el puerto serial y que se tuvieron excelentes resultados de comunicación igualmente. En otros casos, es factible que se use otro cable convertidor, que contará internamente con un circuito similar.

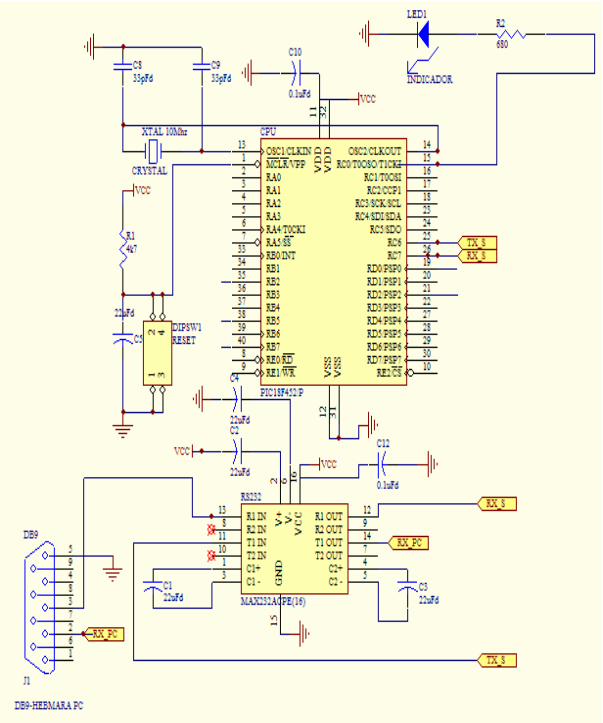

Fig. 2. Comunicación del Boot Loader

El programa que permite leer, grabar y ejecutar los archivos hexadecimales del procesador y probar la integridad del dispositivo fue desarrollado en un lenguaje de programación visual DELPHI [2] tomando en consideración las características de los diferentes sistemas operativos en donde se ejecuta normalmente. Con este objeto, y buscando facilitar su uso, se creó también en un lenguaje especial denominado INNOSTEUP [3], el instalador automático que permite la comunicación entre la PC y el procesador. Esto evita los problemas de instalación que se tendrían normalmente, si solamente se contara con el programa que permite leer, grabar y ejecutar los archivos hexadecimales del procesador, y probar la integridad

del dispositivo. La Fig. 3, muestra la ventana característica del programa desarrollado en visual Delphi.

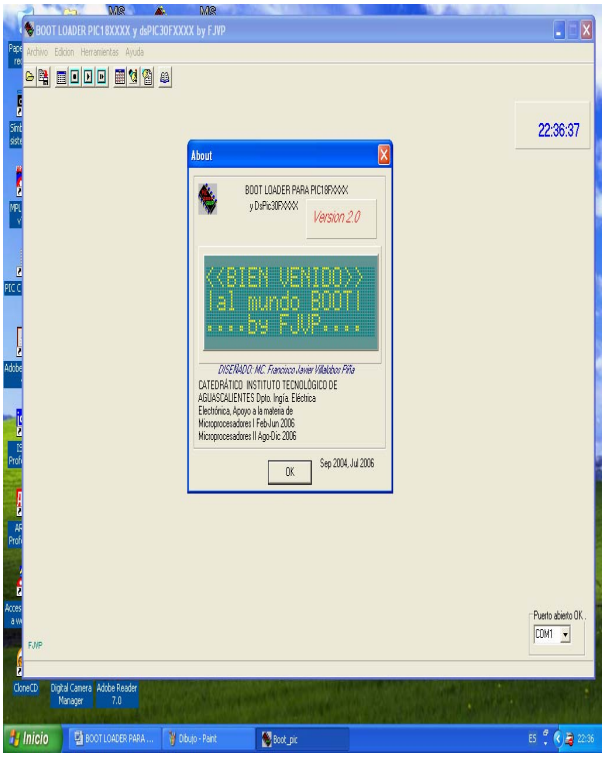

Fig. 3 Ventana principal del programa en Delphi

Este programa cuenta con una ventana que permite visualizar la memoria de programa del procesador según se muestra en la Fig. 4. Además, respeta la característica manejada por el medio integrado de desarrollo de microchip MPLAB IDE [4] en el que se presentan la dirección de memoria, la dirección de cada palabra, y como un recurso adicional que no tiene la ventana de microchip, el sistema que se reporta en este trabajo, presenta la numeración de cada bloque y un indicativo de qué línea se grabará en el CPU. En todo esto se ha tomado en cuenta que el proceso de carga de programas se le facilitaría al diseñador, si contara con la información relacionada con el progreso de grabado, con el porcentaje de la memoria usada, con la información relacionada con línea de inicio y término de grabado, así como con el nombre y la ruta del archivo que se presenta, lo mismo que el modelo del procesador y una ventana que muestre las características más generales del procesador, como la cantidad de memorias FLASH, EEPROM, y RAM.

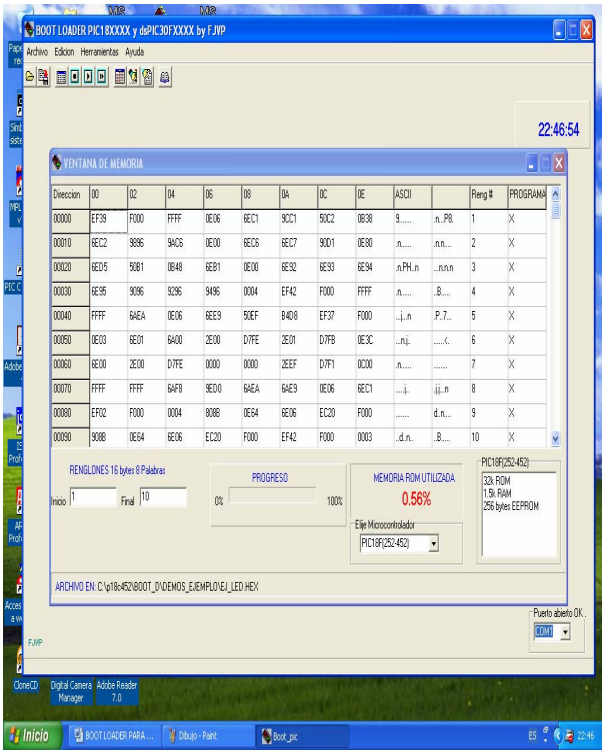

Fig. 4 Ventana de Memoria del programa

La Fig. 5, es una ventana que le muestra al usuario la información relacionada con las claves de acceso, parámetros de configuración del sistema, constantes, etc. Dicha información, pregrabada en la memoria EEPROM, que es una memoria no volátil, viene muchas veces en el archivo máquina y normalmente se asigna mediante la herramienta de diseño de origen, entre las que pueden mencionarse el lenguaje ensamblador y alguna otra herramienta de alto nivel como PICC.

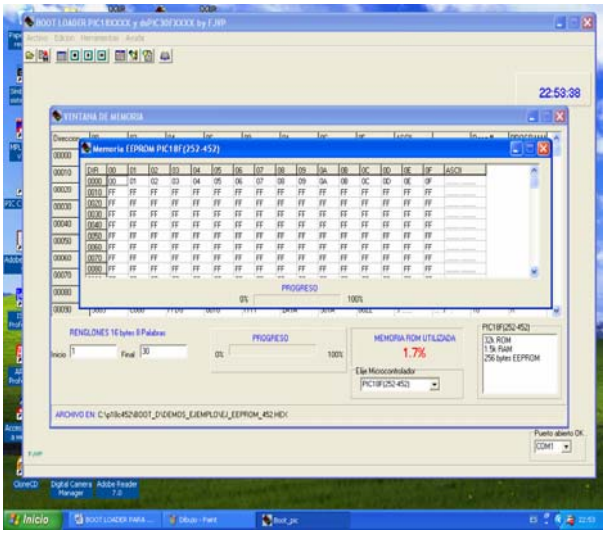

Fig. 5 Ventana de memoria EEPROM del CPU

Además de proporcionar la información relacionada con la memoria del programa y de la memoria EEPROM del cpu, el programa en visual Delphi, le brinda al usuario los botones clásicos de acceso rápido de cualquier aplicación estándar de Windows. La Fig. 6, muestra la ventana que le permite al diseñador Abrir y Guardar archivos. La Fig. 7, muestra la ventana que le permitirá copiar o modificar manualmente alguna dirección. Esto supone una ventaja cuando se quieren probar los efectos de dañar arbitrariamente algún mnemónico, cosa que suele hacerse con cierta frecuencia en las materias de ingeniería de microprocesadores.

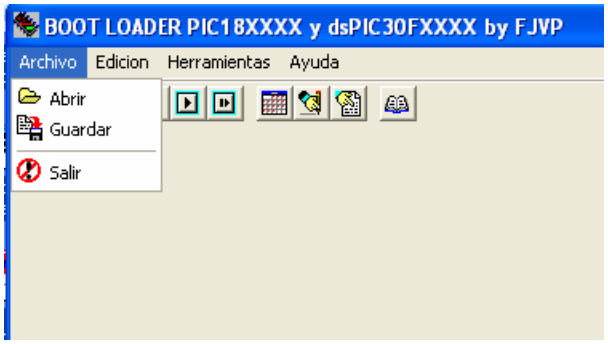

Figura 6 Submenú Archivo

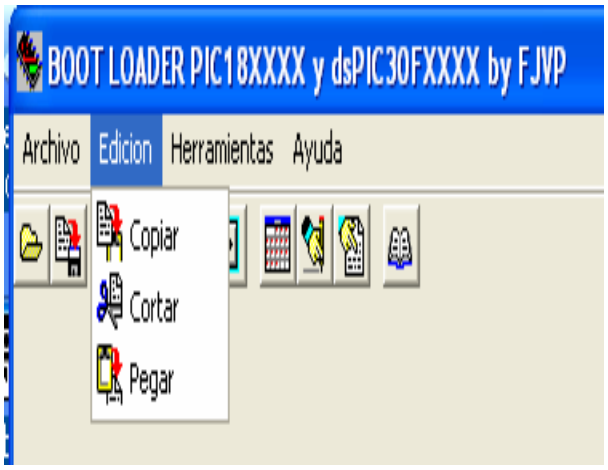

Figura 7 Submenú Edición

La Fig. 8, por su parte, muestra el menú de herramientas del programa con los botones de acceso rápido, mismo que permite efectuar acciones escritura y lectura con los dos tipos de memoria mencionados previamente, la de programa, que es una memoria de tecnología flash y la de datos no volátiles, que es de tecnología EEPROM. En el caso de la memoria de programa, el trabajo que se reporta en este artículo, ofrece además un submenú único que permite ejecutar el programa almacenado.

Figura 8 Submenú Herramientas

Finalmente la Fig. 9, muestra el submenú de ayuda, en el que por ahora, sólo se muestra la información referente a la creación de la herramienta.

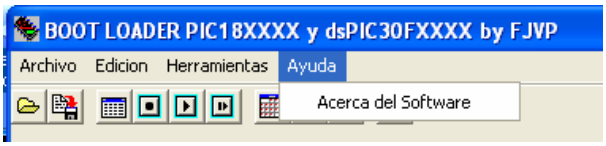

Figura 9 Submenú Ayuda

Conclusiones

Se diseñó un programa en un microcontrolador PIC18F452 y se extendió posteriormente a otros modelos (PIC18F252, PIC18F458, PIC18F4520, PIC18F6720). Las velocidades de operación con las que se ha experimentado, van desde 10 hasta 40 Mhz, pasando por 12 y 20 Mhz. El sistema que se reporta, empezó a utilizarse para el diseño de un prototipo para el cobro automático del transporte urbano en Aguascalientes, y posteriormente se utilizó para el diseño en las aulas en el Instituto Tecnológico de Aguascalientes apoyando las materias de Microprocesadores I y II. De acuerdo con los resultados, puede decirse que este trabajo ha tenido un gran impacto en el proceso enseñanza aprendizaje, dado que los alumnos han mostrado un gran interés por el diseño usando esta herramienta, misma que pueden tener en su casa y no sólo en el laboratorio.

Referencias

[1] PIC18FXXX FAMILY DATA SHEETS, MICOCHIP, U.S.A., 2006

[2] Reisdorph Kent, (1998) *Borland Delphi 4*, Sams Borland Press, U.S.A.

[3] Jordan Russell, "*Inno Setup Compiler*", Portions Copyright, USA. 1997-2003

[4] MPLAB IDE V 7.40, MICROCHIP, USA, 2006