Volumen 5 (2019)
XXV Verano de la Ciencia
ISSN 2395-9797

EDISEÑO DE SISTEMA DE DESARROLLO PARA PROGRAMAR MICROCONTROLADORES EMPLEANDO UN PUERTO USB.

Lemus Najera, Alejandro Daniel (1), Paniagua Medina, Juan José (2), Vargas-Rodríguez, Everardo* (3), Guzman-Chavez Ana Dinora (3)

1 [Ingeniería Electrónica, Universidad de San Carlos de Guatemala, Guatemala] | Dirección de correo electrónico: [alejandrolemus1996@gmail.com]
2 [Ingeniería en Comunicaciones y Electrónica, División de Ingenierías, Campus Irapuato-Salamanca, Universidad de Guanajuato] | Dirección de correo electrónico: [juanjosepaniaguam@hotmail.com]
3 [Departamento de Estudios Multidisciplinarios, División de Ingenierías, Campus Irapuato-Salamanca, Universidad de Guanajuato] | Dirección de correo electrónico: [evr@ugto.mx*; ad.guzman@ugto.mx]

RESUMEN

En este trabajo se presenta la propuesta del diseño un programador de microcontroladores, el cual está basado en LabVIEW de National Instruments. La programación del microcontrolador se realizó usando el modo de programación en sistema (ISP: In System Programming). El programador diseñado considera tanto el hardware como el software necesario para realizar las funciones de programación de los microcontroladores. En el hardware implementado consiste principalmente de un puente de conversión de comunicación serial USB al formato Interfaz Periférica Serial (SPI), que es el requerido por una gran cantidad de familias de microcontroladores para implementar la programación ISP. Además el programador proporciona las señales de control de Reset y de alimentación del microcontrolador objetivo durante un proceso de programación ISP. El software desarrollado controla el hardware y realiza las rutinas de activar el modo ISP, escritura y lectura de la memoria FLASH, identificación de la microcontrolador. Para probar el principio de operación el programador se implementó para los microcontroladores AT89LP51 y AT89LP52 del fabricante Atmel, pero se tiene la flexibilidad para agregar diferentes modelos de diferentes fabricantes agregando rutinas al programa de LabVIEW.

ABSTRACT

In this work the design of a microcontroller programmer, based on LabVIEW of National Instruments, is presented. In this design the microcontroller is programmed by the In System Programming (ISP) mode. Therefore, for the programmer implementation both software and hardware were built. Moreover, the hardware integrates a USB-SPI bridge, since the SPI protocol is required by a large amount of microcontroller families for implementing the ISP. Furthermore, the programmer hardware provides and control the Reset and power supply signals of the target microcontroller during an ISP programming session. Additionally it is presented that the software developed control the overall programmer hardware and performs some functions such as: activate the ISP mode, write and read the FLASH memory, identify the target microcontroller. Finally the programmer was implemented to program the AT89LP51 and AT89LP52 of Atmel, as a proof of principle. However, due to the flexibility of

Palabras Clave
Microcontroladores; Programador; LabVIEW; Comunicación Serial.

the programmer it is possible to add models of different manufacturers by just adding the corresponding functions to our main LabVIEW program.

Introducción

El uso de microcontroladores en las áreas de ingeniería es de uso masivo, debido a que son dispositivos altamente flexibles ya que se pueden implementar soluciones de una manera sencilla y a muy bajo costo. Por lo que existen en el mercado una gran cantidad de diferentes modelos de microcontroladores producidos por diferentes fabricantes. Cada microcontrolador tiene sus características propias, como son el número de puertos, número de entradas/salidas, convertidores analógicos-digitales o digitales-analógicos, entre otros. Por lo que los usuarios muchas veces deben seleccionar el que sea más apropiado para su aplicación y a su costo-beneficio.

Existen en el mercado programadores universales de microcontroladores, los cuales pueden programar algunos modelos de algunos fabricantes, sin embargo su costo es relativamente alto, por lo que para algunos usuarios puede representar una limitante, por ejemplo en el sector educativo. Así mismo existen programadores para algunas familias de microcontroladores los cuales tienen un costo mucho más bajo, el problema es que si requerimos programar un microcontrolador de un fabricante diferente o de otra familia debemos de adquirir otro programador. Por otra parte existe luego el problema de la disponibilidad comercial de encontrar un programador para algunos modelos, por lo que se complica el poder cambiar de fabricante o de familia de microcontrolador. Por lo que en este trabajo mostramos que podemos desarrollar un programador de microcontroladores, empleando una interfaz serial SPI e implementando un programa en lenguaje C o LabVIEW. Aquí presentamos los principios básicos que debemos seguir para programarlos empleando comunicación serial SPI, la cual es empleada en términos generales por una gran diversidad de fabricantes. La diferencia es que cada fabricante/familia usa diferentes comandos o datos para programar los dispositivos, por lo que se tiene potencial de ir agregando dispositivos al programador realizando algunas funciones de software específicas para cada dispositivo a programar, sin hacer cambios en el hardware. En este trabajo y con fines de demostración implementamos un programador para los microcontroladores AT89LP51 y AT89LP52 de Atmel (Atmel, Corporation, 2011).

Figura 1- Diagrama a bloques del programador.

Principio de operación del Programador de Microcontroladores

El microcontrolador que desarrollamos consiste de 2 etapas, una de hardware y otra de software. Él diagrama a bloques general del equipo lo podemos representar como se muestra en la en la Figura 1. La descripción de estos bloques es la siguiente:

a) Bloque de PC-software, esta etapa consiste en un programa desarrollado en LabVIEW en cual tiene como objetivo leer el archivo que se va a programar en el microcontrolador, organizar los datos en las páginas de memoria correspondientes y controlar la operación del circuito puente a través de un puerto USB de la PC.

b) El circuito puente, es un dispositivo que se comunica con la PC a través de un puerto serial USB, y que además cuenta con un puerto con Interfaz Periférica Serial (SPI), por el cual puede enviar y recibir datos en modo full-duplex. Además, este circuito cuenta con una interfaz de entrada/salida (I/O) de 4 bits. Ambas interfaces se controlan con el software de la PC.

c) El tercer bloque está compuesto por algunos dispositivos electrónicos básicos que ayudarán a controlar el estado de las señales de la interfaz SPI, el Reset y VDD del microcontrolador.

En la configuración del programador presentado no se requiere programar ningún circuito electrónico en particular. El único software que se desarrollo es el de LabVIEW y es el que controla las señales (SPI y IO) del circuito puente. En las siguientes secciones detallaremos cada uno de los bloques mencionados.

In System Programming (ISP)

El modo In System Programming (ISP) sirve para poder realizar diferentes funciones como

por ejemplo: identificar el dispositivo, borrar, leer o escribir la memoria de programa, cambiar el estado de los fusibles (fuses) de configuración del microcontrolador, entre otras (Atmel, Corporation, 2006).

Muchas familias de microcontroladores disponen de 2 interfaces básicas para ingresar y realizar funciones en el modo ISP. Una de estas interfaces es de tipo paralelo, mientras que la segunda es de tipo Interfaz Periférica Serial (SPI). En nuestro caso nos enfocamos en esta última interfaz. La interfaz serial SPI es síncrona full-duplex que consta de 4 líneas: reloj (SCK), entrada maestra - salida esclava (MISO), salida maestra-entrada esclavo (MOSI) y selector de esclavo (SS). Una ventaja es que se reducen el número de líneas en comparación con las que se requieren con la interfaz paralela. En este tipo de comunicación se debe de definir quien el dispositivo maestro y quien el esclavo, en nuestro caso el programador será el maestro y el microcontrolador que vamos a programar será el esclavo, de acuerdo a los requerimientos técnicos de la programación vía el modo ISP.

Por lo tanto y debido a que las computadoras personales no cuentan con puertos SPI se requiere de un circuito que sirva de puente para convertir los datos del formato SPI a USB y viceversa. Además el circuito puente debe de realizar el papel de maestro.

Puente de comunicación USB-SPI

El software programador está diseñado para ejecutarse en una computadora personal con puertos USB. Además por el puerto USB deberá de controlar un circuito puente que realice la conversión en el formato de envío/recepción de los datos. En nuestro caso empleamos un puente USB-SPI modelo FT4222H del fabricante FTDI. Este dispositivo nos permite que desde el software de la PC (LabVIEW) podamos configurar uno de sus puertos con el formato SPI en cualquier modo maestro o esclavo con 4 líneas: SCK, MOSI, MISO y SS, como lo requiere el modo In System Programming (ISP) de los microcontroladores. Por otra parte el FT4222H también nos permite poder controlar la velocidad del reloj (SCK) para la transmisión/recepción de los datos (datos SPI (desde 93.75 KHz hasta 40 MHz), así como el formato de disparo del reloj (flaco positivo o negativo) ((FTDI), Future Technology Devices International Limited, 2014). Para controlar y configurar el dispositivo FT4222H empleamos los controladores (Future Technology Devices International Limited (FTDI), 2016) y librerías (Future Technology Devices International Ltd (FTDI), 2011) proporcionadas por el fabricante y las vinculamos con LabVIEW.

Por otra parte, cuando el circuito FT4222H se configura para operar en modo maestro single, se puede disponer de una interfaz paralela de entrada/salida de 4 bits. Las líneas de esta interfaz también pueden ser configuradas y controladas desde el programa principal de LabVIEW. En nuestro caso, estas terminales fueron útiles para enviar algunas señales de control para manipular el Reset del microcontrolador y cuando las señales del puerto serial SPI están en alta impedancia. Esto es indispensable para poder acceder, inicializar, realizar un procedimiento de programación y salir del modo ISP.

Etapa de Hardware del programador

El circuito eléctrico que compone la etapa de hardware se presenta en la Figura 2. En este caso aparte del FT4222H se agregó un controlador de bus 74LS245, el cual nos permite poner en estado de alta impedancia las señales MOSI y CSK que entran al microcontrolador, en nuestro caso lo configuramos para estas señales se activen únicamente cuando el FT42222H ponga en nivel bajo la señal de selección de esclavo (SS).

Es importante mencionar que, para que el microcontrolador pueda ingresar al modo In System Programming (ISP), después realice un proceso específico y salga del modo ISP, es necesario que el circuito maestro tenga control de las terminales Reset y VDD del microcontrolador. Por lo que en nuestro caso agregamos a nuestra etapa de hardware un circuito 74LS04, con el cual negamos la señal SS del FT4222H y posteriormente la conectamos a VDD, de esta manera el microcontrolador solo estará "encendido" mientras SS esté en bajo, lo cual ocurre cuando el maestro está enviando o leyendo un dato al esclavo.

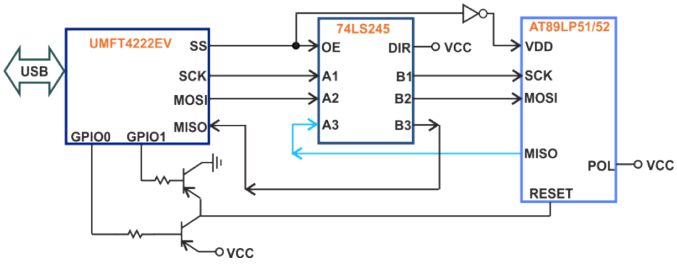

Figura 2- Circuito eléctrico del programador ISP para el microcontrolador AT89LP52

Así mismo empleamos un par de transistores pnp, para controlar el estado del Reset. La base de estos transistores la controlamos con 2 terminales (GPIO0 y GPIO1) de la interfaz paralela del FT4222H (Figura 2). Las formas de onda de las señales GPIO0, GPIO1 y SS que debe enviar el FT4222H y que controlamos desde el programa de LabVIEW se muestran en la Figura 3.

Figura 3- Formas de onda de las señales de control para acceder, programar y salir del modo ISP.

Software del Programador

El software principal de nuestro programador lo implementamos en LabVIEW de National Instruments. Este software está conformado de diferentes funciones, las más importantes son: a) comunicación con el FT4222H, b) identificación del dispositivo, c) lectura de archivos de archivos .HEX y organización en forma de páginas de memoria FLASH, d) programación de la memoria FLASH y e) lectura de la memoria FLASH. A continuación vamos a describir con más detalle cada una de estas funciones.

Memoria de Código FLASH del AT89LP51/52

Normalmente, los programas del microcontrolador se almacenan en una memoria flash interna. Las características de la memoria son diferentes para cada microcontrolador. Por ejemplo, cambian el tamaño de la memoria, la forma en la que está segmentada y cambia el proceso de programación. Por lo que es necesario que el software del programador realice el mapeo de la información que deseamos escribir en la memoria en las localidades de memoria que corresponden.

En particular los microcontroladores Atmel AT89LP51 y AT89LP52 poseen 4K y 8K bytes, respectivamente, de memoria Flash de código, la cual puede ser programada a través de la interfaz SPI. En estos microcontroladores, la memoria de código se divide en páginas de código de 128 bytes. A su vez cada página se divide en dos subpáginas, denominadas baja y alta, cada una de estas subpáginas es de 64 bytes. De esta manera el microcontrolador AT89LP51 tiene 64 sub-paginas y el AT89LP52 tiene 128 sub-paginas de acuerdo con el tamaño de su memoria de código. En estos dispositivos se puede escribir o leer una sub-página de memoria FLASH a la vez.

Lectura de archivos de archivos .HEX

Los archivos con extensión .hex tienen un formato de texto ASCII cuyas líneas obedecen al formato Intel HEX, el cual es utilizado para la programación de microcontroladores, EPROMs y otros circuitos integrados (Arm Limited (or its affiliates))

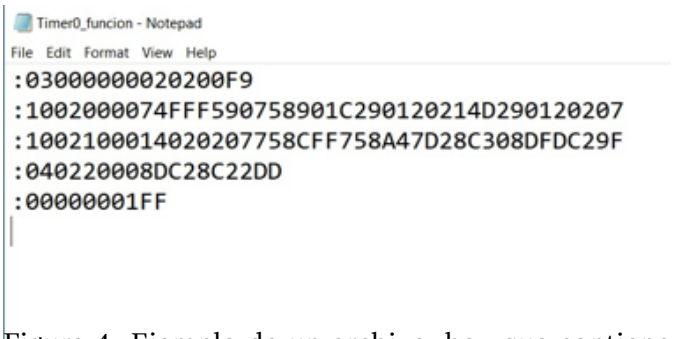

Figura 4- Ejemplo de un archivo .hex que contiene los datos e instrucciones para ser programados en el microcontrolador.

En la Figura 4 se presenta un ejemplo de un archivo .hex. Cada línea se denomina registro HEX, el cual contiene los datos así como las direcciones de memoria codificados con dígitos hexadecimales. El formato de un registro HEX tiene la siguiente estructura:

:LLAAAATT[DD...]CC

en donde cada grupo de letras corresponde a un campo diferente. Cada campo está compuesto de por lo menos de 2 dígitos hexadecimales, los cuales conforman un byte. Dichos campos se describen a continuación:

1-: es el código de inicio de cada registro.

2-LL es la longitud del registro, dicho campo representa el número de bytes de datos que hay en la línea.

3-AAAA es la dirección de memoria inicial para los datos en el registro.

4-TT es el tipo de registro, dicho campo sólo puede contener valores entre 00 y 05, los cuales definen el tipo de campo de datos. Los 6 tipos de registros son:

a) 00: Datos.
b) 01: Fin de archivo.
c) 02: Dirección extendida de segmento.
d) 03: Dirección de comienzo de segmento.
e) 04: Dirección lineal extendida.

f)05: Comienzo de dirección lineal.
5-DD representa un byte de datos, un registro puede contener múltiples bytes, el número de bytes de datos en el registro debe coincidir con el número especificado en el campo LL.
6-CC es el campo de checksum del registro. Dicho campo debe coincidir con el complemento a dos de la suma de todos los campos anteriores salvo el código :.

La función que implementamos en LabVIEW extrae del archivo .hex únicamente los datos e instrucciones que serán programados en la memoria flash del microcontrolador. Además, esta función también organiza la información del archivo .hex en un mapa de memoria específico para nuestro microcontrolador.
Organización de los datos a programar en forma de sub-páginas de memoria FLASH
Como se mencionó anteriormente el microcontrolador AT89LP52 cuenta con 128 sub-páginas de 64 bytes cada una, mientras que el AT89LP51 tiene 64 sub-páginas también de 8 bits. Considerando esto, en nuestro software diseñamos una matriz de 128×64 bytes (para el AT89LP52). De esta matriz cada renglón corresponde a una sub-página de la memoria flash del microcontrolador. Así mismo cada byte (columna) de cada renglón de la matriz corresponderá a una localidad de memoria FLASH específica. La dirección específica la podemos calcular como:

(1.1)

Donde es el número de renglón y es la columna de la matriz, ambos inician en 0.
En la Figura 5 se observa un fragmento de la matriz de bytes generada por nuestra rutina de lectura-interpretación del archivo HEX. Obsérvese que, las casillas que se encuentran vacías contienen el valor FF. Así mismo, la rutina de lectura-interpretación del archivo HEX, realiza la operación de leer las direcciones de cada registro del archivo HEX y ubicarlos en las posiciones de la matriz que correspondan, esta tarea es sumamente importante, para poder realizar adecuadamente la programación de las páginas de código de la memoria flash del microcontrolador.

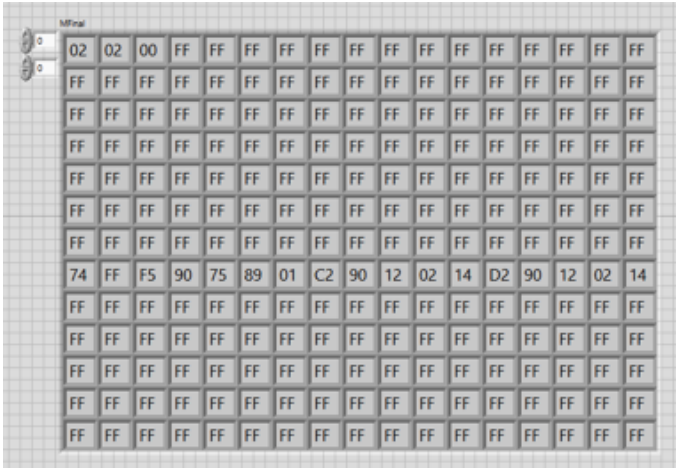

Figura 5- Fragmento de la matriz de bytes generada por el programa en LabVIEW, donde cada reglón corresponde a los datos que se programarán en una subpágina de memoria del microcontrolador.

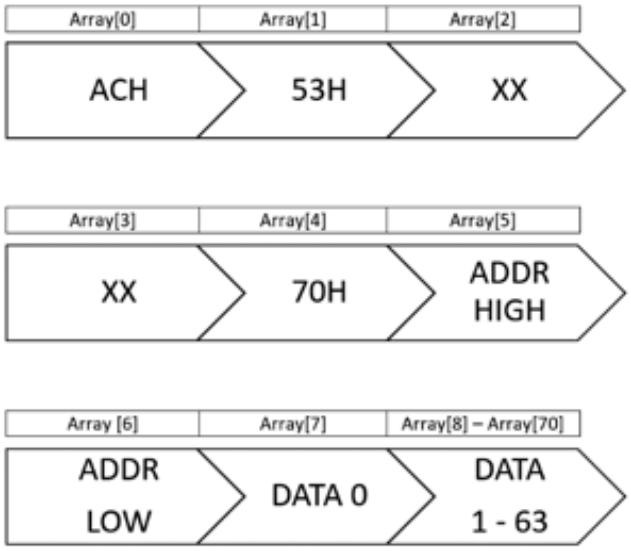

Figura 6- Secuencia de datos que se envían al microcontrolador para poder realizar la escritura de una subpágina de código.

Escritura de una sub-página de la memoria de código
Para programar una sub-página de 64 bytes en la memoria del microcontrolador, se desarrolló una rutina de programa en LabVIEW, encargada de extraer un reglón (sub-página) de la matriz generada en la sección anterior. A este arreglo, la rutina, le concatena otros 7 bytes específicos, con un formato particular atendiendo los requerimientos de la hoja de datos del microcontrolador. De esta manera el programa de LABVIEW debe de hacer que el FTDI4222H envíe la secuencia de datos que se presenta en la Figura 6 por el puerto SPI.

La descripción de los códigos enviados en la secuencia de programación de una subpágina de memoria de programa del AT89LP52 es la siguiente:
1-Program Enable, Opcode 1: Primer código para iniciar el modo de programación (ACh).
2-Program Enable, Opcode 2: Segundo código para iniciar el modo de programación (53h).
3-Address Low: Dirección requerida para iniciar el modo de programación (XXh Don't care).
4-Data 0: Dato requerido para iniciar el modo de programación (XXh Don't care). En este byte el programador debe de recibir, por la terminal MISO, el dato 69H del microcontrolador, el cual indica que el entró exitosamente al modo de programación en sistema (ISP).
5-Opcode: Indica el código de operación de una sub-página de memoria FLASH. Nosotros empleamos el código de borrar-escribir una sub-página (70H).
6-: Indica la parte alta de la dirección de la sub-página de la memoria del microcontrolador que se programará.
7-: Indica la parte baja de la dirección de la sub-página de la memoria del microcontrolador que se programará.
8-Data 0: Es el byte inicial de la cadena de datos.
9-Data 1 - 63: Son los bytes restantes de la conjunto de datos que se escribirán en la memoria interna del microcontrolador.
Aquí es relevante mencionar que de forma general la dirección de cada sub-página está definida por 7 bits de la siguiente forma: . Los bits forman el número de la página y el bit indica si es la sub-página alta (1) o baja (0). Sin embargo, debemos tener presente que en estos microcontroladores la dirección de la sub-página se divide en dos registros de 8 bits, identificados como ADDR HIGH y ADDR LOW, con el siguiente formato:

(1.2)

y

(1.3)

Rutina de identificación del microcontrolador
Otra rutina que desarrollamos es la de lectura de la página de identificación del microcontrolador. Esta rutina debe de enviar un arreglo de bytes semejante al descrito para el de la programación de la sub-página de código. En concreto debemos de enviar un arreglo de 10 bytes de los cuales los primeros 4 son exactamente los mismos que en el caso anterior y los bytes 31|00|00|XX|XX|XX (Figura 7). En esta caso el microcontrolador debe de contestar, por la línea MISO, 10 bytes, numerados de 0 a 9. En estos bytes el numero 3 debe de contener el dato 69H, y los bytes 7 al 9 deben de contener el 1EH, 54H y 5H respectivamente para el AT89LP51, mientras para el microcontrolador AT89LP52 debe de contestar 1EH, 54H y 6H.

De manera general, la misma secuencia para entrar al modo de programación e inicializar el modo es la misma para cualquier operación que deseemos realizar en el modo de programación ISP, como por ejemplo borrar la memoria, escribir una página con auto borrado, cambiar el estado de los fuses, etc. La única diferencia es el comando de operación y el número de bytes enviados/recibidos, para realizar la operación deseada. Para detalles de todos los comandos de programación, consultar el manual de datos del fabricante del microcontrolador Atmel AT89LP51/52 (Atmel, Corporation, 2011).

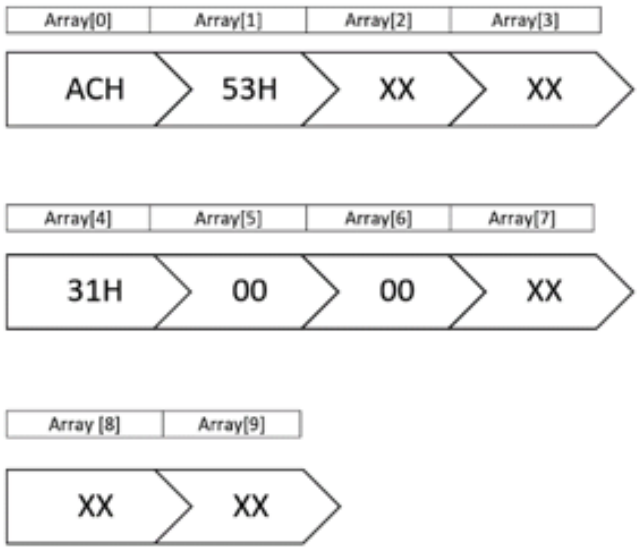

Figura 7- Formato del arreglo de bytes que se envían al microcontrolador para el proceso de identificación.

Programa Principal del Programador
Como ya se mencionó anteriormente, el programa principal se desarrolló en LabVIEW y está compuesto por varias rutinas, que han sido descritas en secciones previas. En la siguientes figuras se muestran algunas capturas de pantalla de los paneles de control de las funciones de identificación del microcontrolador que se va a

programar y de la lectura de un archivo .Hex y como se mapean sus datos en la matriz de sub-páginas de memoria FLASH y finalmente en la Figura 10 se presenta el diagrama de la subrutina del programa que realiza la función de programación de una sub página de código con auto limpiado (auto-erase). Esta función borra una subpágina de 64 bytes y escribe los códigos correspondientes. Debido a que es un programa gráfico se requerirían varias páginas para poder presentar los diagramas y los paneles de control para cada sub-rutina del programa, sin embargo los usuarios puede contactar a los autores para obtener una copia del programa completo.

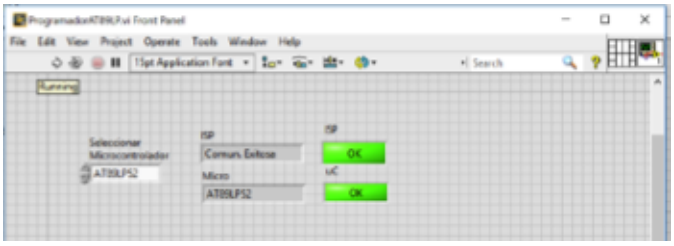

Figura 8- Captura de pantalla del resultado de la función de identificación del microcontrolador.

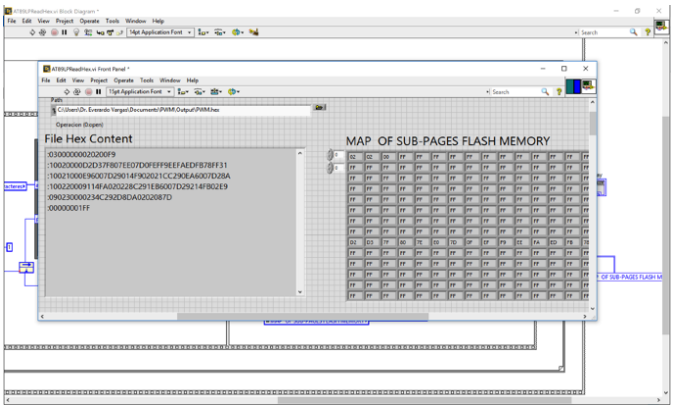

Figura 9- Captura de pantalla del resultado de la función de lectura del archivo .hex y el mapeo correspondiente de los datos extraídos en una matriz donde cada renglón corresponde a una sub-página de la memoria FLASH.

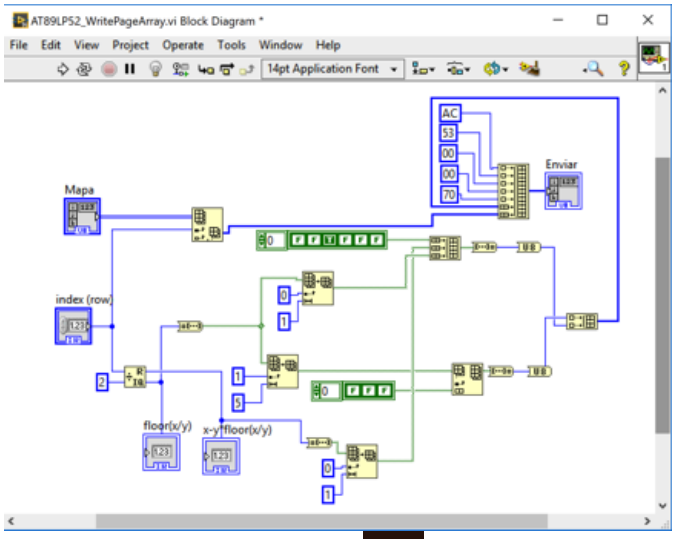

Figura 10- Diagrama de la función de escritura de una sub-página de memoria FLASH, se recibe la matriz de subpáginas, se indica el renglón (subpágina) que se va a escribir con auto-limpiado (auto erase).

Conclusiones

En este artículo se presentó un diseño simple de un programador de microcontroladores por medio del modo In System Programming (ISP), la cual empleada comúnmente por varios fabricantes. Se presentó el circuito electrónico que implementamos y se demostró que solo se requiere desarrollar un programa en la PC para que controle de manera integral el programador. En nuestro caso desarrollamos el programa en LabVIEW un software de instrumentación ampliamente usado a nivel global, pero se puede desarrollar en lenguaje C o C++. Finalmente con fines de demostración elegimos los microcontroladores AT89LP51 y AT89LP52 de Atmel, pero podemos agregar sub-rutinas específicas a nuestro programa principal para poder programar otros microcontroladores.

Agradecimientos

A. D. Lemus Najera, agradece a la universidad de Guanajuato por la oportunidad de llevar a cabo la investigación de verano, junto con el Departamento de Estudios Multidisciplinarios, igualmente agradece al Dr. Everardo Vargas Rodríguez por su asesoría en cada momento durante el transcurso de la investigación.

REFERENCIAS

Arm Limited (or its affiliates). (s.f.). General: INTEL HEX File Format. Obtenido de http://www.keil.com/support/docs/1584/

(FTDI), Future Technology Devices Internatio-

nal Limited. (2014). FT4222H USB2.0 to QuadSPI/I2C Bridge IC.
Atmel, Corporation. (2006). AT89LP In-System Programming Specification.
Atmel, Corporation. (2011). AT89LP51, AT89LP52.
Future Technology Devices International Limited (FTDI). (2016). AN_396 FTDI Drivers Installation Guide for Windows 10.
Future Technology Devices International Ltd (FTDI). (2011). Software Application Development D2XX Programmer's Guide.