PROGRAMACIÓN EN CAMPO DE MICROCONTROLADORES PIC

*Carlos Iván Camargo Bareño**
carlos_ivan_camargo@yahoo.com

Este artículo presenta las principales ventajas de la programación serial de los microcontroladores PIC (ICSP), y describe el paquete de software CAINPIC, desarrollado para realizar la programación de cualquier microcontrolador que permita ICPS desde el puerto paralelo. Además, se realiza una introducción a la adquisición de datos utilizando el computador, y muestra cómo se pueden obtener retardos del orden de microsegundos utilizando el comando RDSTC.

This paper show the 'In Circuit Serial Programming' of Microchip microcontroller´s. And Presents CAINPIC, a software for ICSP Programming using Parallel Port Interface. Beside, make Data Acquisition Introduction and show how to implement nanoseconds delay using the RDSTC command (available in Pentium or K62 processors)

In Circuit Serial Programming (Icsp)

La Programación en Circuito o en Campo es una técnica en la cual un dispositivo es programado después de ser colocado en la placa de circui-to impreso. Conlleva las siguientes ventajas:

- Reducción de costo de actualizaciones: la técnica permite reprogramar el contenido total de la memoria de programas de un dispositivo con memoria Flash. Los dispositivos OTP (One Time Programming) pueden ser programados de tal forma que sólo se utilice un espacio de la memoria, y dejar disponibles otro de sus segmentos para futuras versiones del programa. Con esto no es necesario comprar otro chip para realizar una actualización

- Calibración del sistema durante la fabricación: muchos sistemas requieren ser calibrados para obtener un funcionamiento ideal; ICSP permite realizar estas modificaciones fácilmente

- Calibración del sistema en campo: una vez el circuito se coloque en el sitio de trabajo es posible realizar modificaciones para un desempeño óptimo

- Permite realizar programaciones a distancia: una de las posibles aplicaciones de ICSP es la de la programación remota, es decir, se puede diseñar un sistema que realice la actualización vía telefónica o vía Internet; con esto se reducirían enormemente los costos de mantenimiento.

---

* Ingeniero Electricista Universidad Nacional de Colombia, Magister en Universidad de los Andes, profesor Universidad Distrital F.J.C., adscrito a la Facultad Tecnológica

La técnica ICSP está disponible en las siguientes Familias de microcontroladores de MICROCHIP:

- PIC12C5XX OTP, 8-pines
- PIC16CXXX OTP, Gama Media
- PIC17CXXX OTP Gama Alta
- PIC16F8XX FLASH, Gama Media

Protocolo de Programación

El protocolo a seguir para la programación serial es muy sencillo. Los PIC utilizan las siguientes cuatro señales para este propósito:

VPP : Voltaje de Programación (12 – 14V)
VDD: Voltaje de alimentación (2 – 5.5V)
CLK: Señal de Reloj
DIO : Señal bidireccional para entrada y salida de datos.

Para realizar la programación, lectura y borrado de la memoria de los microcontroladores el Protocolo ICSP utiliza una serie de comandos que pueden o no tener argumentos (un argumento es una palabra de 14 Bits). El siguiente diagrama de tiempos nos muestra la secuencia para enviar un comando con argumento:

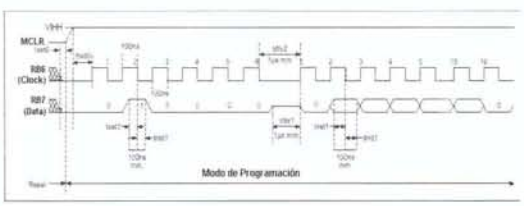

Figura 1. Diagrama de tiempos de un comando con argumento.

Como puede observarse en la Figura 1, inicialmente se debe colocar al microcontrolador en modo de programación; esto se realiza colocando las señales de Clock y Data en un nivel lógico bajo, mientras se eleva el voltaje de la señal MCLR de V_{IL} a V_{IH} (12 a 14V), y la señal de V_{DD} de V_{IL} a un nivel de voltaje adecuado. Una vez que se encuentra en este modo se pueden acceder las memorias de datos y programa.

Las señales de reloj y dato deben ser generadas de tal forma que en los flancos de bajada del «clock» se encuentre el valor correspondiente al comando enviado. Por ejemplo, en la figura se envió el comando 000010, dado que se envían primero los bits menos significativos. Todos los comandos tienen un tamaño de 6 bits. A continuación se debe enviar el argumento del comando (si es aplicable, ya que no todos lo requieren). Debe existir un tiempo mínimo entre el envío del bit más significativo del comando y el bit de «start» del argumento (normalmente de 1ms). El argumento está formado por 14 bits, pero siempre debe enviarse un bit de «start» y uno de «stop» (Nivel lógico bajo).

Los comandos disponibles para el protocolo ICSP son:

Load Configuration (0): al recibir este comando el contador de programa se coloca en la posición 2000H. El argumento corresponde a la palabra de configuración.

Load Data for Programm Memory (2): al recibir este comando el microcontrolador escribirá el argumento en la actual posición de memoria. Si este comando se ejecuta por primera vez la dirección de memoria es 0.

Read Data From Programm Memory (4): al recibir este comando el microcontrolador envía a continuación el contenido de la memoria en la actual posición de memoria. Si este comando es el primero que se ejecuta retornaría el contenido de la primera posición de la memoria de programa (Dirección 0)

Increment Address (6): este comando se utiliza para aumentar la actual posición de memoria. Se debe ejecutar siempre que se realicen los comandos *Load* o *Read Data From Programm Memory;* de lo contrario siempre se leería o escribiría la misma posición de memoria. Este comando no requiere argumento.

Begin Erase Programm Cycle (8)

Load Data for Data Memory (3): similar al *Load Data For Programm Memory*, pero actúa sobre la memoria de datos

Read Data From Data Memory (5): similar al *Read Data From Programm Memory,* pero actúa sobre la memoria de datos

Bulk Erase Programm Memory (11): se utiliza para borrar la memoria de datos. Este comando no requiere argumento.

Los últimos tres comandos no se pueden utilizar en todos los microcontroladores, ya que no todos poseen memoria de datos.

Bulk Erase Programm Memory (9): se utiliza para borrar la memoria de Programa. Este comando no requiere argumento.

Algoritmos de Programación

Los siguientes diagramas de flujo muestran los pasos a seguir para la correcta programación de la memoria de programa (Figura 2), y la memoria de datos (Figura 3).

Obsérvese que siempre se realiza una verificación para comprobar que el dato escrito se almacenó correctamente. Si el dato que se lee del microcontrolador no corresponde al enviado, algunos de ellos permiten realizar 8 o 25 intentos (MAXCP) para que quede correctamente programado.

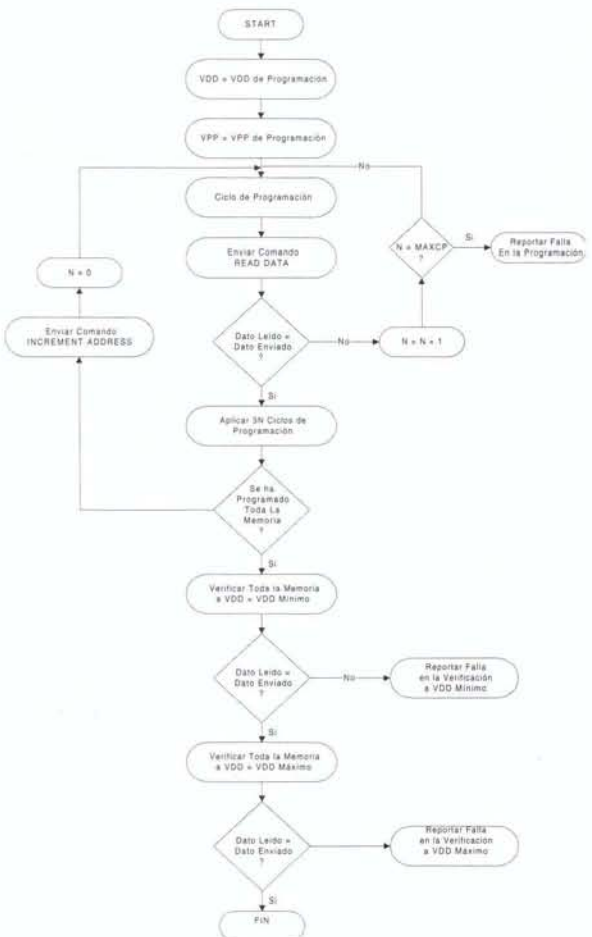

Figura 2. Ciclo de Escritura de la Memoria de Programa

Existen algunos microcontroladores en los que MAXCP = 1, es decir, sólo se puede realizar un intento de programación por posición de memoria.

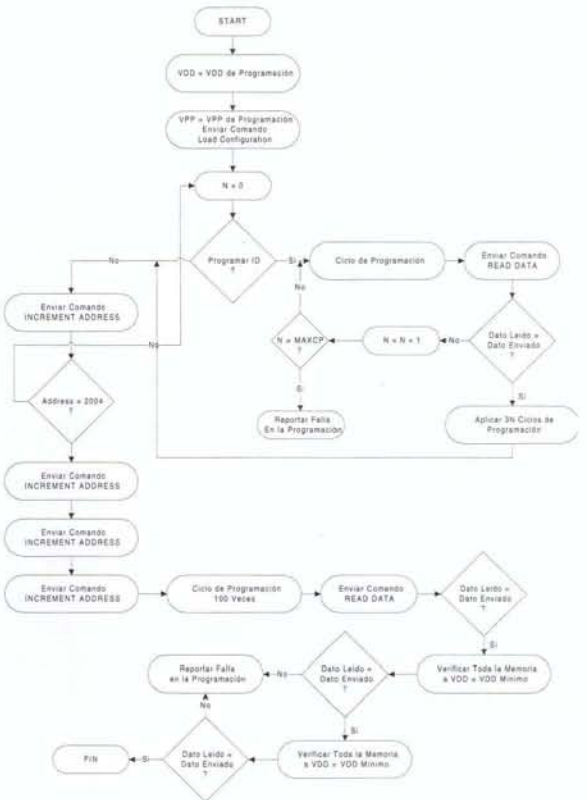

Figura 3. Ciclo de Programación de la Memoria de Datos

El Ciclo de Programación utilizado en los diagramas de flujo anteriores realiza las siguientes operaciones:

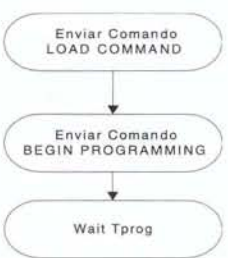

Figura 4. Ciclo de Programación

donde Tprog puede ser de 100 mS, 10 ms o 2 mS.

Como puede observarse en los diagramas de flujo, una vez que se programan todas las posiciones de

memoria se debe realizar una verificación del contenido de la memoria a diferentes niveles de voltaje. Microchip.

Consideraciones Tecnicas para Utilizar ICSP

Dado que el protocolo de programación utiliza cuatro pines del microcontrolador, se debe incluir un circuito que aísle estos pines cuando se realice la reprogramación. Un circuito adecuado para realizar esta operación se muestra en la siguiente figura:

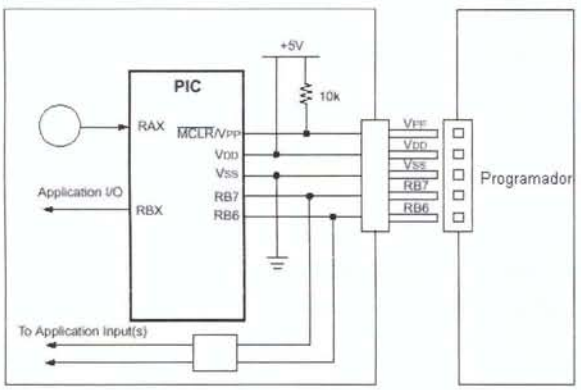

Figura 5. Circuito recomendado para utilizar ICSP

En la Figura 4 se observa que se debe incluir un conector para poder permitir la programación del microcontrolador; además se debe permitir el aislamiento de los pines correspondientes a las señales (clock y data), si se usan en esta aplicación.

Cainpic Software de Programacion ICSP

Debido a los problemas existentes actualmente a la hora de realizar la programación de un microcontrolador (costo de las herramientas comerciales y de sus actualizaciones), se ha diseñado un software que permite la programación serial de los microcontroladores que tengan implementado el protocolo. El software debe ser independiente del dispositivo, y permitir la adición de nuevos microcontroladores con sólo variar un archivo de configuración. Lo anterior evita al usuario las molestias de las actualizaciones.

El principal problema que tienen los programas existentes es que están realizados para un procesador determinado y/o para una determinada velocidad del procesador. Además, no presentan flexibilidad de conexión con el hardware de programación (conexiones fijas y no programadas)

CAINPIC fue desarrollado completamente en Borland DELPHI y pretende convertirse en una herramienta de trabajo útil para los estudiantes de electrónica que no poseen los recursos necesarios para acceder a una herramienta comercial.

La elección de DELPHI como herramienta de programación se debe a que este es un compilador y permite realizar funciones en assembler.

Cainpic: Detalles Técnicos

Uno de los principales problemas en la realización de CAINPIC fue la creación de una base de tiempo confiable y que no dependiera del procesador. Es decir el software debe funcionar en cualquier computador: INTEL o AMD, que son los procesadores más utilizados.

Debido a que el ciclo de programación de los PIC16C6XX, PIC16C7XX, PIC16C9XX y 16C55X requieren un retardo de máximo 100 ms., y las función Sleep de Windows proporciona un retardo de mínimo 1ms, fue necesaria la implementación de una rutina que suministrara retardos del orden de microsegundos.

Los procesadores de INTEL, desde el Pentium Pro, y los de AMD desde el K6II, incorporaron en su set de instrucciones una dedicada a pruebas de velocidad. Esta instrucción es la RDTSC (Read Time-Stamp Counter).

El Time-Stamp Counter es un registro de 64 bits que almacena el conteo de los ciclos de reloj transcurridos desde que se conectó el computador o desde que se aplicó la señal de RESET al procesador. La instrucción RDTSC (310F H) retorna los 32 bits más significativos del registro en EDX y los 32 menos significativos en EAX. El código en Delphi para obtener la lectura del Time-Stamp Counter es el siguiente:

```
Function GetTimeStamp:Comp;
Var
  TimeStamp: Record
   Case Byte Of
    1: (Whole: Comp);
    2: (Lo,Hi: LongInt);
   End;
Begin
 asm
 dw 310Fh // rdtsc
 {$IfDef CPU386}
  mov  [TimeStamp.Lo],eax
  mov  [TimeStamp.Hi],edx
 {$Else}
 DB D32
  mov  word ptr TimeStamp.Lo,AX
  DB D32
  mov  word ptr TimeStamp.Hi,DX
 {$EndIf}
 End;
  Result := TimeStamp.Whole;
End;
```

¿Para qué sirve saber el número de ciclos de reloj de una máquina? Si se sabe la velocidad del procesador puede calcularse la equivalencia de un ciclo de reloj, es decir:

Duración de un ciclo de Reloj = (1/Velocidad del Procesador)

Si se desea tener un retardo de N segundos se necesitarían:

No Ciclos = (No. Segundos) * (Frecuencia)

Surge entonces otro problema: ¿Cómo determinar la velocidad del procesador empleado? En realidad es sencillo. Se sabe que las funciones:

Sleep(1)
GetTickCount

proporcionan un retardo de 1 ms; si se llama la función GetTimeStamp antes y después de ejecutar esta instrucción se tendrá el número de ciclos de reloj que ésta emplea en ejecutarse. Se sabe que este tiempo es 1ms, luego la velocidad del procesador será:

Frecuencia = No. de Ciclos /$1x10^{-3}$.

El código en Delphi para obtener la velocidad del procesador es el siguiente:

```
Procedure GetCPUSpeed;
Var
  T1 : DWord;
Begin
  T1 := GetTickCount;
  While T1 = GetTickCount do
    OldCyc := GetTimeStamp;
  While GetTickCount < (T1 + 400) do;
    NewCyc := GetTimeStamp;
  CPUClock :=  2.5e-6*(NewCyc-OldCyc);
End;
```

En la función «GetCPUSpeed» se esperan 400 ms en lugar de 1ms. «CPUClock» es una variable que almacena la velocidad del procesador. Con estas dos funciones se puede implementar una rutina que proporcione retardos del orden de los microsegundos. La siguiente rutina realiza este retardo.

```
Procedure usDelay(us : LongInt);
Var
  Cycles : Real;
  S      : String;
Begin
  Cycles := (CPUClock*us);
  OldCyc := GetTimeStamp + Cycles;
  While (GetTimeStamp) < OldCyc do;
End;
```

Otro problema encontrado durante la realización del programa fue el control del puerto paralelo. Su solución se encontró una vez más en el lenguaje assembler, ya que las funciones IN y OUT permiten la entrada y salida de datos respectivamente desde cualquier posición de Memoria. Por lo tanto, deben leerse y escribirse datos a la dirección de memoria correspondiente al puerto paralelo (esta dirección puede tomar los siguientes valores: 2BC, 3BC o 378). Las rutinas en Delphi para la entrada y salida de datos son:

```
Procedure OutP(Data:Byte);
Begin
  asm
    mov  AL,Data
    mov  AH,00
    mov  DX,&PPORT
    out  DX,AX
  End;
End;
```

```
Function InP(Address : Word):Byte;
Begin
  asm
    mov  DX,&Address
    In   Al,DX
    mov  @Result,AL
  End;
End;
```

Programador

El circuito programador utiliza el puerto paralelo como entrada y salida de datos. Su presentación se realiza en el anexo 1.

Cainpic: Manual del Usuario

CAINPIC es «freeware», si se utiliza para propósitos educativos y se distribuye con el código fuente. Si se utiliza o modifica parte del código, se sugiere enviar al autor de este artículo el código modificado vía correo electrónico**.

Archivo De Dispositivos Device.INI

Este archivo es el alma del programa; en él se encuentra toda la información sobre los microcontroladores que necesita el programa para su correcto funcionamiento, y además permite el fácil ingreso de nuevos microcontroladores. El formato es el siguiente:

Sección «fuses»: en esta sección se incluye la información de la palabra de configuración para todos los microcontroladores soportados. El formato de cada configuración es:
Fuses 1 - - - - - - - CP0 PWRTE WDTE OSC1 OSC0

La línea debe empezar con la palabra «fuses» y después un número, el cual no puede repetirse; luego se coloca la información correspondiente a cada pin; la primera casilla corresponde al bit 13 (MSB) y la última al bit 0 (LSB). Este tipo de palabra de configuración corresponde a un chip como el PIC16C61.

La siguiente sección corresponde a los dispositivos propiamente dichos y su formato es el siguiente:

DEVICE PRG DAT ALG FUSES WARN PINS MAXPC

Donde:
Device: Nombre del dispositivo (máximo 10 caracteres)

PRG: tamaño de la memoria de programa, en notación hexadecimal

DAT: tamaño de la memoria de datos.

ALG: algoritmo de programación utilizado; las siguientes son las opciones de algoritmos:

1. E2PROM
2. EPROM and OTP
3. None
4. PIC12X5XX 12 Bits
5. EEPROM 24CXX

FUSES: el número de la palabra de configuración correspondiente al chip

WARN: mensajes de warning

PINS: número de pines

MAXPC: número máximo de ciclos de programación

Siguiendo con el ejemplo del PIC16C61 la configuración sería:

PIC16C61 3FF 0 2 01 0 18 25 2.5 6.0

Lo que indica que el dispositivo es el PIC16C61 tiene una memoria de programa de 3FF Bytes, no tiene memoria de datos, algoritmo de Programación 2 (EPROM & OTP), utiliza la configuración 1 de Fuses, no utiliza señales de «warning», tiene 18 pines, y permite máximo 25 ciclos de programación.

Esas dos líneas son suficientes para definir cualquier chip.
Adicionalmente, en el archivo HelpDevice.txt se define la ayuda para la configuración el formato es el siguiente:

** Nota del autor: el software fue desarrollado para suplir las necesidades de los estudiantes de microprocesadores en la Universidad Distrital, Facultad Tecnológica, y está siendo utilizado actualmente en los cursos: Microprocesadores I y II y Control Digital

```
Fuses 1  Begin
  CP0, Code Protection Configuration Bit
  1 = Code Protection Off
  0 = Code Protection On

  PWRTE, Power Up Timer Enabled Configuration Bit
  1 = Power Up Timer Enabled
  0 = Power Up Timer Disabled

  WDTE, WDT Enable Configuration Bits
  1 = WDT Enabled
  0 = WDT Disable

  OSC[1:0], Oscillator Selection Configuration Bits
  11 = RC
  10 = HS
  01 = XT
  00 = LP
End
```

El menú de ayuda aparecerá en la ventana el programa para facilitar al usuario la configuración del microcontrolador.

El formulario principal de CAINPIC es el siguiente:

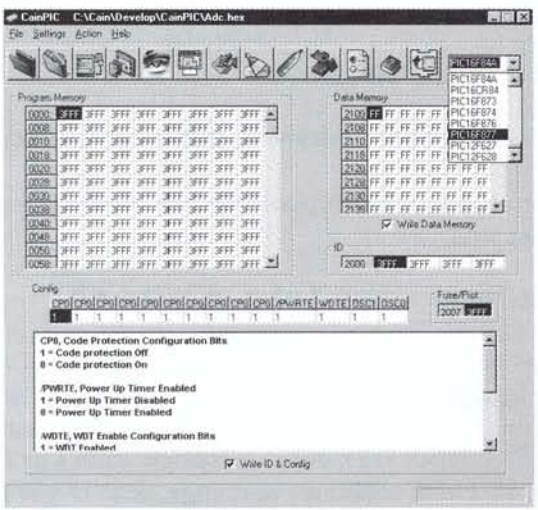

Figura 6. Apariencia de CAINPIC

CAINPIC permite realizar las siguientes funciones:


Read Hex File: permite leer archivos generados con MPLAB en el formato INHX8M, y los carga en un buffer de programación, a la vez que los despliega en las tablas: Programm Memory, Data Memory, ID y Config.


Save As Hex: permite guardar el contenido del buffer a un archivo en formato INHX8M


Programm: Programa el chip con los datos del buffer


Verify: verifica que el chip se haya programado correctamente


Read: lee las memorias de programa y datos, los IDS y la palabra de configuración y lo despliega en las tablas: Programm Memory, Data Memory, ID y Config.


Blank Check: verifica si se borró el chip


Write Fuses: programa los ID y la palabra de configuración


Erase Buffer: borra el contenido del buffer


Test Hardware: permite comprobar el correcto funcionamiento del programador


Settings: permite variar la configuración del puerto paralelo

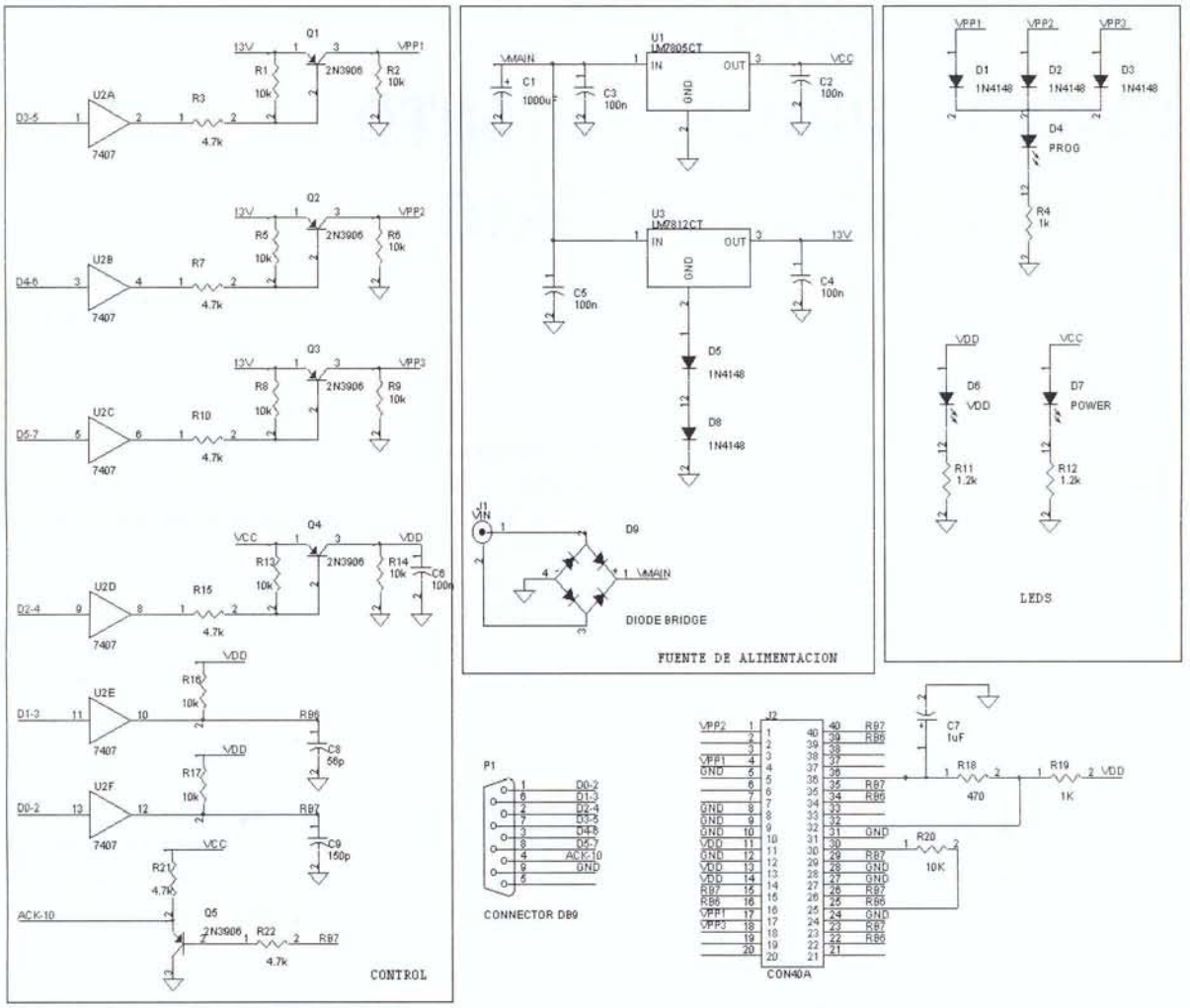

Figura 7. Circuito Programador Propuesto.

Conclusiones

La programación en circuito o en campo es de gran ayuda en los sistemas digitales, ya que permite ahorro de tiempo de desarrollo y agiliza las operaciones de actualización y calibración.

Utilizando las rutinas propuestas se pueden obtener retardos del orden de nanosegundos, lo cual sirve para aumentar la velocidad de respuesta de las tarjetas de adquisición de datos.

CAINPIC es una herramienta que permite realizar la programación serial de microcontroladores PIC. Es flexible, y permite adicionar nuevos dispositivos sólo con modificar un archivo de texto.