Dispositivos Chipbas8, microcontroladores HC08 programables en lenguaje BASIC

A. Salvá, L. Altamirano

Departamento de Control y Robótica, Facultad de Ingeniería, UNAM, México, D. F., salva@dctrl.fi-b.unam.mx, CP 04510 México

***Resumen* — Para desarrollar el software de base requerido por aplicaciones basadas en microcontroladores, se emplean tanto lenguajes de bajo nivel de tipo ensamblador y de alto nivel como podrían ser BASIC y C, entre otros. En este trabajo se presentan microcontroladores de la familia HC08 de Freescale a los que se les ha agregado firmware que valida las bibliotecas asociadas con un compilador cruzado de BASIC denominado MINIBAS8A diseñado acorde con éstas. Esto hace que los dispositivos mencionados puedan ser programables empleando BASIC que es un lenguaje de alto nivel de fácil uso. A los dispositivos con el valor agregado constituido por las bibliotecas de MINIBAS8A se les ha denominado genéricamente Chipbas8_xx, donde la cadena 'xx' denota el miembro de la familia HC08 empleado. En este artículo, se presenta además una descripción concentrada acerca de cómo se desarrolló MINIBAS8A y un ejemplo ilustrativo de programación empleando éste.**

I INTRODUCCIÓN

Hoy en día, los microcontroladores están detrás del funcionamiento de diversos dispositivos de uso doméstico, automotriz o industrial, como podrían ser dispositivos de medición, controladores PID, o bien algo de uso doméstico como un control remoto de TV o el arbitraje del funcionamiento de un horno de microondas entre otras aplicaciones. Lo que da vida a éstas es el software de base cargado en memoria no volátil del microcontrolador, ya que estos son, en esencia, computadoras digitales contenidas en un chip.

Los lenguajes de programación empleados para desarrollar el software de base asociado con las aplicaciones basadas en microcontroladores pueden ser en lo general los siguientes:

Lenguaje de máquina, para el cual cada instrucción elemental ejecutable por el procesador está codificada en binario, de hecho, puede decirse que está es la *lengua materna de la computadora* como podría ser el microcontrolador con el que se trabaje en un momento dado.

Lenguaje de tipo ensamblador**,** para el cual existe una correspondencia uno a uno entre las instrucciones elementales de éste y las propias en lenguaje de máquina del procesador de la computadora donde se ejecute el programa. Las instrucciones básicas aquí mencionadas por lo general implicarán operaciones donde únicamente estarán involucrados uno ó dos operandos.

Lenguajes de alto nivel, como podría ser "C" o bien BASIC, para los cuales el programador puede declarar expresiones que impliquen más de dos operandos y/o tipos de operación, además de contar con estructuras de programación tales como 'if' o bien 'for' entre otras, que facilitan la programación de los algoritmos asociados con la aplicación que se esté desarrollando en un momento dado.

Si la aplicación a desarrollar no hace uso extensivo de aritmética, el programa asociado con ésta puede escribirse cómodamente empleando lenguaje ensamblador; éste podría ser el caso por ejemplo, del control secuencial de los estados asociados con el perfil de encendido/apagado de las lámparas asociadas con un semáforo simple de control de tráfico, o bien, un simple control ON/OFF.

Por otro lado, si la aplicación hace uso extensivo de aritmética, como podría ser un controlador PID, el desarrollo del programa asociado con ésta, se efectúa de una manera ágil empleando para ello un lenguaje de alto nivel; esto no quiere decir que ésta no pueda ser realizada empleando lenguaje ensamblador; sin embargo, el proceso es mucho más complicado para el programador.

II COMPILADORES Y ENSAMBLADORES

Ya sea que la aplicación se desarrolle empleando un lenguaje de alto nivel, o bien, uno de tipo ensamblador, a la serie de sentencias que integran las declaraciones asociadas con el

algoritmo que corresponda se le denomina programa fuente; a partir de éste se debe generar el correspondiente programa en lenguaje de máquina, ejecutable en el procesador de la computadora que realice la aplicación.
Al software que genera el código ejecutable a partir de un programa fuente escrito en lenguaje ensamblador, por lo general se le denomina programa ensamblador. Por otra parte, al programa que genera el código ejecutable a partir de un programa fuente escrito en un lenguaje de alto nivel, se le denomina compilador. Cuando el software de compilación y/o ensamble se ejecuta en una computadora cuyo procesador no es el destino final del código ejecutable, a éste se le denomina ensamblador cruzado (cross assembler), o bien, compilador cruzado (cross compiler); este es el caso de los compiladores y ensambladores disponibles en el mercado para programar microcontroladores; ya que el software que genera el código objeto corre en una computadora de propósito general como puede ser una PC y el código generado se ejecuta en el procesador que contiene el microcontrolador empleado en el desarrollo de la aplicación.

III EL COMPLILADOR MINIBAS8A

Actualmente el software de compilación o ensamble que se desarrolle para un determinado procesador, se escribe empleando un lenguaje de alto nivel disponible para la computadora donde corra éste. Para el compilador MINIBAS8A se empleó VisualBasic de Microsoft. En lo general, de acuerdo con cánones propios de ciencias de la computación, el proceso de compilación de un programa fuente escrito en un lenguaje de alto nivel puede desdoblarse en cuatro etapas [4]. Éstas se describen a continuación, mencionando algunos detalles propios de MINIBAS8A

A. Analizador léxico

En esta etapa se identifican en el programa fuente original cadenas de caracteres denominadas tokens, que denotan entre otras, cosas tales como: nombres permitidos de variables; palabras reservadas como podrían ser ‘for’ o ‘print’; o bien, operadores aritméticos, lógicos o relacionales. A cada tipo diferente de token se le asocia un número al que se le llama la clase del token. Para MINIBAS8A se predefinieron 30 clases de tokens. Cada vez que se encuentra un token no asimilable a una clase valida, esto se reporta como error léxico.

B. Analizador de sintaxis y semántica

En esta etapa se determina si la sintaxis empleada en cada una de las declaraciones que conforman el programa fuente es correcta. Además del análisis sintáctico, en esta etapa se debe verificar la coherencia semántica de las declaraciones presentes en el programa fuente.
A partir del archivo de tokens generado en la etapa A, se efectúa el análisis sintáctico y semántico del programa fuente, generándose dos archivos de salida; uno que contiene los reportes de los diversos errores que podría haber contenido el programa fuente y otro que contiene lo que se denomina como código intermedio (CI). En lo general, éste es generado por MINIBAS8A siguiendo las siguientes consideraciones: Las sentencias donde en el programa fuente original aparece la asignación del resultado de una expresión a una determinada variable del usuario, son convertidas en una serie de expresiones de asignación, donde el miembro derecho de éstas contiene sólo uno o dos operandos; cada uno de ellos podría ser una variable declarada por el usuario, un número explícito, o bien, una variable interna que genera el compilador en el proceso de análisis de expresiones. Por otra parte, los miembros izquierdos de las expresiones de asignación mencionadas serán variables internas o de usuario. Además, la contraparte en el CI que corresponda a estructuras de programación tales como ‘for’ o ‘if’ contendrán en sus argumentos sólo variables internas o de usuario; o bien, números explícitos.

Es importante destacar que el CI es independiente del procesador donde se vaya a ejecutar el programa, esto implica que las etapas A y B del proceso de compilación pueden ser empleadas para la generación de código ejecutable para diversos procesadores, como podrían ser los contenidos en diferentes tipos de microcontrolador.

C. Generación de código en ensamblador

En esta etapa, a partir del código intermedio se genera un programa en el lenguaje ensamblador propio del procesador destino; para esto, se debe contar con un conjunto de rutinas escritas también en el mismo ensamblador, éstas deberán realizar entre otras cosas las siguientes:

- Operaciones aritméticas y lógicas, tanto para enteros como para números reales de punto flotante.
- Validación de las diversas funciones propias del lenguaje como podrían ser entre otras: Raíz cuadrada (sqr(x)), seno (sin(x)) o tangente (tan(x)).
- Realización de intercambios de información entre el programa en ejecución y la consola de interfaz

empleada. Para MINIBAS8A, la consola por defecto es un emulador de terminal.

Al conjunto de rutinas mencionadas se les denomina como las bibliotecas del compilador. Para MINIBAS8A éstas son 140.

D. Generación de código de máquina

En esta etapa, mediante el empleo de un programa ensamblador, se obtiene el código de máquina ejecutable en el procesador destino, para MINIBAS8A se usa el ensamblador ENS08 [3] diseñado previamente por uno de los autores.

IV CODIGO GENERADO POR MINIBAS8A

Para fines ilustrativos, en esta sección se muestran los códigos intermedio y en ensamblador correspondientes a una línea de un programa fuente en lenguaje BASIC, compilado con MINIBAS8A. La línea en cuestión es:

```
z1 = sqr ( a1 + b1 ) / pk
```

Donde z1, a1, b1 y pk son variables reales de precisión sencilla y se aprecia que a la variable z1 se le ha de asignar el resultado de la raíz cuadrada de la suma de a1 y b1 dividido entre el valor de la variable pk.
El tramo de código intermedio asociado es el siguiente:

```
t!1 = a1 + b1
t!3 = sqr ( t!1 )
z1 = t!3 / pk
```

Donde t!1 y t!3 son variables internas de tipo real de precisión sencilla generadas por el compilador al procesar la expresión cuyo valor debe asignarse a la variable z1.

El tramo de código en ensamblador correspondiente es el siguiente:

```
ldhx #a1
jsr vmemfl#acpfps1
ldhx #b1
jsr vmemfl#acpfps2
jsr sum#pfps
ldhx #t!1
jsr acpfps1#vmemfl
ldhx #t!1
jsr vmemfl#acpfps1
jsr raiz#cuad
ldhx #t!3
jsr acpfps1#vmemfl
ldhx #t!3
jsr vmemfl#acpfps1
ldhx #pk
jsr vmemfl#acpfps2
jsr div#pfps
ldhx #z1
jsr acpfps1#vmemfl
```

Donde se pueden apreciar diversos llamados a subrutinas que forman parte de las Bibliotecas de MINIBAS8A.

V EL CONCEPTO CHIPBAS8

Un dispositivo Chipbas8 es un microcontrolador de la familia HC08 de Freescale que contiene en su memoria no volátil firmware denominado NBCP8_BIBAS8, el cual está integrado por las bibliotecas del compilador MINIBAS8A y el núcleo básico de comunicaciones de un programa denominado PUMMA_08+; el cual permite que el chip pueda ser manejado vía puerto serie desde una computadora de tipo PC. El software manejador es ejecutable bajo WINDOWS (M/2000/XP/V). De aquí en adelante al núcleo de comunicaciones mencionado se le denotará NBCP8. El programa manejador contiene al ensamblador ENS08, el compilador cruzado MINIBAS8A y facilidades para la prueba y depuración de aplicaciones entre las que destacan el poder borrar y programar la memoria FLASH del microcontrolador con código objeto generado por ENS08, MINIBAS8A, o bien, algún otro ensamblador o compilador existente en el mercado.
Los miembros de la familia HC08 para los cuales se ha probado el concepto Chipbas8 son el 68HC908GP32, denominado Chipbas8_gp32 y el MC9S08GB60 denominado Chipbas8_gb60. En las figuras 1 y 2 se muestran los mapas de memoria de los Chipbas8 aquí mencionados.

MAPA DE MEMORIA DEL CHIPBAS8_GP32	
Dir Hex	Uso
0000 003F	Zona 1 de registros de control y operación (RCO).
0040 023F	Memoria RAM (512 localidades).
8000 D7DB	Memoria Flash de usuario (22,492 localidades).
D7DC D7FF	Vectores de usuario.
D800 FDFF	Firmware NBCP8_BIBAS8.
FE00 FF7D	Zona 2 de registros de control y operación. ROM Monitor.
FFDC FFFF	Vectores del NBCP8.

Fig. 1. Mapa de memoria del chipbas8_gp32

Para el caso del Chipbas8_gp32 el usuario cuenta con 22492 localidades de memoria FLASH para colocar el programa asociado con la aplicación; en

tanto que para el Chipbas8_gb60 el número de localidades disponibles para este mismo fin es 51104.

MAPA DE MEMORIA DEL CHIPBAS8_GB60	
Dir Hex	Uso
0000 007F	Zona 1 de registros de control y operación (RCO).
0080 10FF	Memoria RAM (4,096 localidades).
1080 17FF	Zona 1 del firmware NBCP8_BIBAS8.
1800 182B	Zona 2 de registros de control y operación.
182C DFCB	Memoria Flash de usuario (51,104 localidades).
DFCC DFFF	Vectores de usuario.
E000 FFFD	Zona 2 del firmware NBCP8_BIBAS8.
FFFE FFFF	Vector de reset del NBCP8.

Fig. 2. Mapa de memoria del chipbas8_gb60.

Para fines de los dispositivos Chipbas8_gp32 se diseñó una tarjeta de evaluación denominada MINICON_08A. En la figura 3 se muestra una foto de ésta.

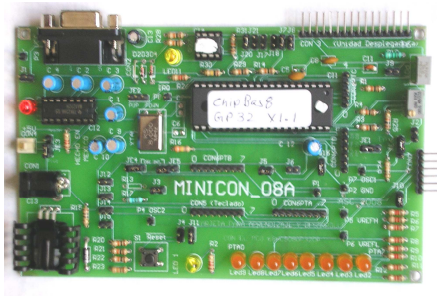

Fig. 3. Tarjeta MINICON_08A

En lo que toca a los dispositivos Chipbas8_gb60, la viabilidad del concepto se probó experimentalmente en la tarjeta DEMO9S08GB60 fabricada por la empresa AXIOM. Para una descripción detallada acerca de la tarjeta MINICON_08A y de las facilidades propias del software manejador PUMMA_08+ relacionadas al compilador MINIBAS8A, pueden verse [2] y [3], no se incluyen aquí por razones de espacio.

VI ASPECTOS DE BASIC PRESENTES EN MINIBAS8A

MINIBAS8A contempla diversas características propias del lenguaje BASIC tales como: Tipos de variables y formato de diversas estructuras de programación. En cuanto a tipos de variables estos pueden ser: Entero signado de uno, dos o cuatro bytes; real de precisión sencilla y string. Además, para cada uno de los tipos se pueden declarar arreglos de dimensión 1, 2 ó 3. Es importante señalar que el código fuente no requiere del clásico número colocado en el extremo izquierdo de cada renglón del programa fuente. Quien haya programado en BASIC empleando QuickBasic de Microsoft se sentirá en casa.

En cuanto a estructuras de programación se manejan las siguientes: ‘if then else endif’, ‘for next’, ‘while wend’, ‘do while loop’, ‘do until loop’ ‘ do loop while’, ‘do loop until’ y ‘select case end select’

Otras propiedades importantes son el manejo de expresiones aritméticas y/o booleanas; o bien, relacionales; el poder usar sentencias ‘input’ y ‘print’ y el contar con las funciones propias del lenguaje (entre otras: sqr(.),sin(.),cos(.), exp(.)).

VII CARACTERÍSTICAS ESPECIALES DE MINIBAS8A

Aunado a las facilidades de programación mencionadas en la sección anterior, MINIBAS8A contempla una serie de propiedades que facilitan la programación de sistemas embebidos basados en microcontrolador, estas se enlistan a continuación:

- Incrustación de código en ensamblador.
- Diseño de rutinas de servicio de interrupción escritas en BASIC.
- Diseño de rutinas de servicio de interrupción escritas en ensamblador.
- Se pueden definir variables especiales ligadas con registros de trabajo del microcontrolador.
- Se pueden colocar en memoria datos numéricos de tipo entero o real de precisión sencilla.

VIII EJEMPLO DE PROGRAMACIÓN

En esta sección se muestra un ejemplo de programación asociado con una aplicación muy simple, su inclusión aquí es únicamente para fines ilustrativos de la funcionalidad de MINIBAS8A como herramienta para desarrollo de aplicaciones usando dispositivos Chipbas8 y empleando BASIC en el diseño del software de base asociado con éstas. El dispositivo empleado en este ejemplo es el Chipbas8_gp32.

La aplicación consiste en un medidor de la velocidad promedio de vehículos que pasan entre dos puntos denominados ‘A’ y ‘B’ separados por una distancia denominada ‘sd’. En cada punto existen sendos sensores denominados ‘SA’ y ‘SB’, los cuales generan cada que pasa un vehículo, sendos pulsos verificados en bajo de un milisegundo. Se supone que solo puede pasar un móvil a la vez y que para cada uno de ellos ha de mostrarse su velocidad promedio; además, por cada grupo de ‘n’ vehículos que hayan pasado se debe desplegar la media de la velocidad promedio de

todos ellos. En la figura 4 se muestra un esquema simplificado de la disposición de los sensores; el conexionado de las señales de los sensores a el dispositivo Chipbas8_gp32 empleado, y el enlace de éste con el emulador de Terminal usado para validar la consola de interfaz con el usuario.
Para medir el intervalo de tiempo asociado con el paso de cada uno de los vehículos se emplea la instancia de interrupción por sobreflujo del temporizador 1 del dispositivo, configurada ésta de modo que el sobreflujo se dé cada 10ms. En la rutina de servicio asociada se incrementa en uno una variable entera que contendrá la cuenta de centésimas de segundo que han transcurrido al pasar cada móvil entre los puntos 'A' y 'B'. Detalles acerca de cómo se configura el temporizador pueden verse en [1], no se incluyen aquí por razones de espacio.

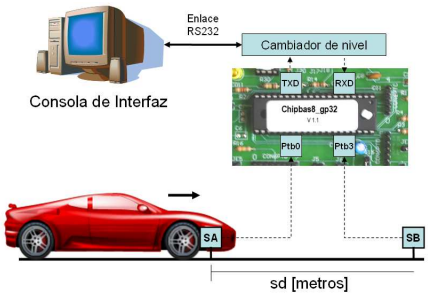

Fig. 4. Esquema simplificado del medidor de velocidad promedio empleando el dispositivo Chipbas8_gp32.

El programa fuente en BASIC se denominó 'medvelpr' y está contenido en el archivo 'medvelpr.b'. La secuencia de ejecución se enlista a continuación:

- Se pide al usuario la distancia 'sd' en metros.
- Se pide al usuario el número 'n' de vehículos a considerar.
- Cada que pasa un vehículo se despliega su velocidad promedio.
- Una vez que han pasado 'n' móviles se muestra la media de sus velocidades.
- Se regresa a pedir 'n' para repetir el proceso o terminar la sesión.

Nota: Si las variable 'n'es cero o negativa termina la sesión.

El programa asociado con la aplicación aquí descrita se diseñó para ejecución autónoma [3] y en éste se emplean varias de las propiedades especiales de MINIBAS8A, tales como el diseño de rutinas de servicio de interrupción escritas en BASIC; el uso de variables especiales de usuario; colocación de datos de tipo *integer* (declaraciones de vectores de usuario) e incrustación de código en ensamblador. Se distingue además el uso de diversas estructuras de programación propias del lenguaje BASIC contempladas por MINIBAS8A.
A continuación se muestra el código fuente completo del programa aquí descrito.

Código fuente del programa 'medvelpr'

```
'Definición de variables
'especiales de usuario tipo byte
  defvarbptr ptb &h1
  defvarbptr config1 &h1f
  defvarbptr t1sc &h20
  defvarbptr t1modh &h23
  defvarbptr t1modl &h24
'Definición de variables
'especiales de usuario tipo integer
  defvariptr t1mod  &h23
'.....
'Deshabilitación del COP
 config1=1
'Se inicializa t1mod para que
'el intervalo entre sobreflujos sea
'10 ms @ Fbus =2 Mhz.
   t1mod=&h4e1f
'.....
'Incrustación de código en ensamblador
'para habilitar interrupciones globalmente.
iniens
 cli
finens
'.....
lecsd: cls
  input "Dar 'sd' en metros,sd=",sd
  if sd<=0 then lecsd

  while 1
  nv%=1
  sumvelpr=0.
  input"n=",ntop%
  if ntop%<=0 then
  print "Fin de la sesión, para reiniciar"
  print "oprimir el botón de RESET"
  end
  endif

while nv%<=ntop%
contcent%=0

 'Espera cruce de punto 'A'
  do
  varch~=ptb and &h1
  loop while varch~=1
 '.....
t1sc=&h40 'pstop<--0,toie<--1
print "El vehículo ";nv%;" cruzó el punto 'A'"
```

```
 'Espera cruce de punto 'B'
 do
 varch~=ptb and &h8
 loop while varch~=8
'.....................
t1sc=&h0 'pstop<---0,toie<--0
print "El vehículo ";nv%;" cruzó el punto 'B'"
 velpr=360.*sd/contcent%
 print"Vp para el vehículo ";nv%;" es";velpr;" Km/h"
 print
 sumvelpr=sumvelpr+velpr
 nv%=nv%+1
 wend

 medv=sumvelpr/ntop%
 print"La media de las velocidades es";medv;" Km/h"
 print
 wend
‘Rutina de servicio de interrupción
‘por sobreflujo del TIM1
servovf:
 glip’ guarda en pila localidades propias de bibliotecas.
varaux~=t1sc
t1sc=t1sc and &h7f'tof<---0
contcent%=contcent%+1
relip ‘recupera de pila localidades propias de bibliotecas
retint

'Declaración de vector de usuario
'para el evento de sobreflujo del TIM1.
dataw &hd7f2 servovf
'Declaración de vector de RESET
'de usuario.
dataw &hd7fe &h8000
```

En la figura 5 se muestra la pantalla de la consola de interfaz cuando se ha ejecutado el programa anterior considerando tres vehículos y que la separación entre los sensores es de diez metros.

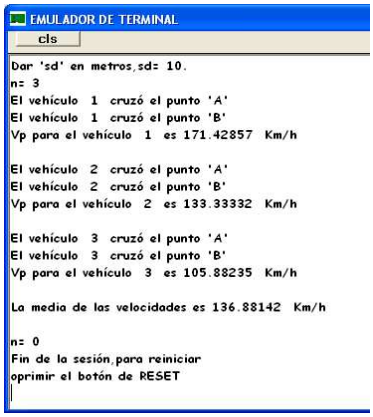

Fig. 5. Despliegue en la consola al ejecutarse el programa ‘medvelpr’ bajo las condiciones mencionadas.

IX CONCLUSIONES

Los dispositivos Chipbas8 en conjunción con el compilador MINIBAS8A, constituyen un medio funcional y de fácil uso para el desarrollo de aplicaciones de instrumentación, control y afición; empleando el lenguaje BASIC para el desarrollo del software de base implicado.

En la industria existen otras herramientas para el diseño de aplicaciones basadas en microcontroladores de la familia HC08 de Freescale; pero éstas están orientadas en cuanto a lenguajes de alto nivel únicamente al lenguaje C, que se ha vuelto un estándar de programación para estos dispositivos, desde luego muy poderosa y transportable, pero no es la única forma de programar una computadora como podría ser un microcontrolador HC08. Para aplicaciones que corran en tiempo real en algunos casos sería más conveniente usar lenguaje C; sin embargo, muchas aplicaciones de este tipo podrán desarrollarse tanto en C como con el BASIC validado por MINIBAS8A. En el caso de aplicaciones que no requieran ejecución en tiempo real definitivamente MINIBAS8A constituye una herramienta de fácil uso y utilidad.

Finalmente se menciona que el desarrollo de tanto el compilador MINIBAS8A como de los dispositivos Chipbas8 está basado en gran medida en la experiencia de campo de casi 20 años de uno de los autores en el diseño de software y hardware para desarrollo y aprendizaje con los microcontroladores HC11, HC12 y HC08 de FREESCALE. Es por esto que no se mencionan muchas referencias en este trabajo además de por razones de espacio.

REFERENCIAS

[1] Freescale, MC68HC908GP32 Microcontrollers. Technical Data, 2002. Archivo mc68hc908gp32.pdf, descargable desde la página de Feescale. www.freescale.com

[2] A. Salvá, L. Altamirano, “SAD_908, SISTEMA PARA APRENDIZAJE Y DESARROLLO CON MICROCONTROLADORES DE LA FAMILIA 68HC908”, Memoria de ELECTRO 2008, Chihuahua México, Octubre de 2008.

[3] A. Salvá, AIDA08, Ambiente Integrado para Aprendizaje y Desarrollo con Microcontroladores de la Familia 68HC908 de Freescale, (Guia de usuario). Noviembre de 2008. Archivo guaida8b.pdf, descargable desde la página; http://dctrl.fi-b.unam.mx/~salva

[4] K. A. Lemone, FUNDAMENTOS DE COMPILADORES, CECSA, México D. F., 1996.