MICROCONTROLADORES SIMULADOR y EMULADOR DE MICROCONTROLADOR ST6225

DR. JOSÉ LUIS GUTIÉRREZ TEMIÑO, DRA. Mª JOSÉ GIL LARREA
Universidad de Deusto
E.S.I.D.E.
Avda de las Universidades 24
48007 BILBAO
Tel. 34-4-4453100 (Ext 2607)
Fax.34-4-4451408
Email: jguti@inf.deusto.es

RESUMEN.- Los cada vez mayores niveles de integración alcanzados en el campo de la microelectrónica han permitido la aparición de auténticos computadores en miniatura: los microcontroladores. Sobre un único circuito integrado se ha logrado integrar, además de los elementos básicos de un ordenador (CPU, RAM y ROM), una serie de circuitos adicionales que simplifican enormemente el diseño de sistemas electrónicos. En este trabajo se explican y analizan dos herramientas desarrolladas para facilitar la depuración de programas diseñados para microcontroladores de la familia ST6 de Thomson. Estas utilidades son el simulador y el emulador de monochip, **MC**SIM y **MC**EM respectivamente.

1.- INTRODUCCION

Un microcontrolador, o monochip, es un dispositivo orientado al control de sistemas. Es un circuito integrado que contiene un programa de control, según el cual se generan determinadas salidas en función de las entradas recibidas. En el mercado se pueden encontrar una amplia gama de estos productos, capaces de ajustarse a cualquiera que sea la complejidad del sistema a controlar.
Es un dispositivo idóneo, por su versatilidad y precio, para aplicaciones industriales. Ejemplos de funciones que podría llevar a cabo uno de estos chips irían desde el control automático de temperatura de una caldera hasta la implementación de una calculadora digital.
El monochip sobre el cual se han desarrollado MCSIM y MCEM ha sido el modelo ST6225, siendo MCSIM, además, compatible con los modelos ST6210, 15 y 20, pertenecientes todos ellos a la familia de microcontroladores ST6 de Thomson. El ST6225 ha sido diseñado para aplicaciones de complejidad media y dispone, a grosso modo, de un *Timer*, un conversor A/D de 8 bits, veinte líneas de E/S y una memoria ROM o EPROM de 4Kb.

2.- MCSIM *(Simulador de monochip)*

El simulador de monochip es una herramienta software para depuración de errores lógicos en los programas de control de monochip.
Veamos cómo utilizar MCSIM. El usuario debe cargar desde MCSIM el programa de control limpio de errores de compilación (esta operación se realiza con el compilador AST6, proporcionado por el fabricante del monochip). En pantalla aparecerá sobreiluminada la primera línea a ejecutar del programa de control y un gráfico del monochip con todas sus líneas. También se muestra información sobre todos y cada uno de los registros: registros de puertos, de *watchdog*, etc...
El usuario puede realizar, entre otras, las siguientes operaciones:

- Ejecutar el programa de forma continua o línea a línea
- Simular interrupciones desde menú (Interrupción del conversor, NMI, Reset)
- Modificar cualquier dirección de la RAM del monochip, registros de datos de puertos incluídos
- Colocar *breakpoints* en cualquier punto del programa.

El simulador de monochip está compuesto por un programa principal y varios subprogramas o unidades. Las unidades más importantes son dos:

- Unidad de desensamblado. Esta unidad traduce el código hexadecimal de una instrucción a su nemónico correspondiente.
- Unidad de ejecución. Su función, como se puede intuir, es la ejecución de instrucciones. Toda instrucción, a excepción de la instrucción 'NOP' (*Ninguna operación*), se puede encasillar dentro de uno de estos dos grupos: instrucciones de modificación de memoria (LD, ADD, AND, etc...) o instrucciones de modificación de secuencia del programa (JP, CALL, JRR, etc...). Por tanto, la ejecución de una instrucción sólo consiste, bien en la actualización de las posiciones de memoria implicadas, bien en la modifificación del PC (*Contador de programa*).

Los pasos que sigue la unidad de desensamblado son:

a). Lectura del código de la instrucción actual. El código de la instrucción actual se encuentra ubicado en el monochip en la dirección de memoria ROM apuntada por el PC. En el simulador, dicha memoria toma la forma de un *array*. El código de la instrucción actual en el simulador se encuentra en una celda de dicho array, concretamente en aquella cuyo índice coincida con el PC. En el ejemplo, suponemos que el PC apunta a la dirección 880h, que contiene 1Fh. O sea, el código de la próxima instrucción a ejecutar es 1Fh.

b). Tratamiento del código. El objetivo de esta fase es generar el nemónico de la instrucción actual. Mediante sentencias condicionales se consigue el tratamiento individual de cada código. En el ejemplo, la acción a llevar a cabo para el código 1Fh es:

' *Nemónico* = "LD A, " + ROMarray(PC+1) '

Donde *Nemónico* es una variable de tipo cadena, ROMarray(PC+1) es el segundo byte de la instrucción (84h) y ' + ' es un operador de concatenación de cadenas. La variable *Nemónico* contendría la cadena ' LD A, 84h ', resultado de desensamblar la instrucción '1F 84'.
La unidad de desensamblado debe reconocer todos los códigos que soporta el lenguaje de programación del monochip (alrededor de sesenta y uno). Su programación es una tarea ardua pero, como se habrá podido comprobar con el ejemplo, muy sencilla.

La unidad de ejecución opera de forma similar:

a). **Lectura del código de la instrucción actual**. Operación idéntica a la descrita para la unidad de desensamblado.

b). **Tratamiento del código.** En esta etapa se llevará a cabo la ejecución de la instrucción. Al igual que en la unidad de desensamblado, cada código requiere un tratamiento individual. El tratamiento de cada instrucción se reduce, en un lenguaje de alto nivel como Pascal, a una media de tres sentencias de programa fuente. Retomemos el ejemplo. El código de la instrucción actual es 1Fh. Este código representa la instrucción ' LD A, *rr* '. Es una instrucción de transferencia, que carga en el acumulador (A) el contenido del registro (o posición de memoria RAM) *rr*. El operando *rr* se especifica en el byte siguiente a 1F, byte 84h, situado en la dirección ROM PC+1. Para implementar esta instrucción, el tratamiento básico sería:

' ACUMULADOR=RAMarray(ROMarray(PC+1)) '

Donde 'ROMarray(PC+1)' es el segundo byte de la instrucción (84h) y 'RAMarray(...) ' es el contenido de la dirección RAM especificada dentro del paréntesis. En resumen, en 'ACUMULADOR' queda almacenado el contenido del registro cuya dirección RAM es 84h.

Las restantes instrucciones del repertorio se implementan siguiendo un razonamiento análogo.

Cuando cualquiera de estas unidades finaliza el procesado de una instrucción, devuelve el control del programa al programa principal que, entre otras cosas, actualiza la información ofrecida en pantalla.

Una de las acciones más importantes que lleva a cabo el programa principal es la gestión de interrupciones. Esta operación se realiza tras la ejecución de cada instrucción. Mediante cadenas de saltos se detecta si se ha producido alguna interrupción. Cuando esto ocurre, se salva el valor actual del PC en la pila (STACKarray) y se reemplaza por la dirección reservada para el tratamiento de dicha interrupción. Posteriormente, al detectar en el programa de control la instrucción de retorno de interrupción (instrucción 'RETI', código 4Dh), se extrae de la pila la dirección de retorno.

Cada vez que se produce una interrupción, el usuario recibe un *feedback* por pantalla que le informa de tal acontecimiento.

Se ha pretendido reproducir lo más fielmente posible el comportamiento del monochip, con toda la exactitud que la información proporcionada por el fabricante ha permitido (*ST6 Family programming manual)*. Entre otros aspectos, se informa del tiempo que hubiera empleado el monochip en la ejecución del programa de control, tiempo calculado en base al número y tipo de instrucciones ejecutadas desde que se produjo el último RESET.

También ha sido simulado el tiempo de conversión que necesita el conversor A/D interno para ofrecer un resultado correcto. El retardo generado es el tiempo medio real de conversión de dicho circuito, indicado en las especificaciones del fabricante (60 µseg).

Las limitaciones que presenta MCSIM son prácticamente las mismas que presentaría cualquier simulador:

- Al no existir conexión física con el exterior, cuando en un punto del programa de control se debe realizar una conversión analógica a digital de una entrada, el usuario es el que debe introducir el resultado de esa conversión.
- No permite modificar el programa. Para hacerlo, habrá que salir de MCSIM, realizar la modificación con un editor de texto y volver a compilar.
- El tiempo de ejecución del programa de control no es real. No obstante, entre otra información ofrecida por pantalla aparece el tiempo real transcurrido.

3.- MCEM *(Emulador de monochip)*

El emulador de monochip es una herramienta diseñada, al igual que el simulador, para facilitar la depuración de programas para el monochip ST6225. La diferencia entre ambos es que MCEM, además del soporte software, incorpora un componente hardware, lo cual hace posible que su comportamiento se acerque más al del monochip que MCSIM.

Las operaciones más importantes que el usuario puede llevar a cabo desde el menú de MCEM son las mismas que se han indicado anteriormente para el MCSIM.

La parte software es muy similar a MCSIM, tanto en lo referente al aspecto interno o de programación como en la interfase de usuario. Cabe destacar la incorporación de la opción *TURBO*, que permite realizar una ejecución a una mayor velocidad, a costa de eliminar de pantalla toda la información que aparece en el modo normal, o no *TURBO*.

La parte hardware consta de una tarjeta y de un cable con un conector macho que simula las patitas del monochip. La tarjeta se introduce en cualquiera de los slots de expansión de cualquier PC (8086 o superior). El conector macho irá insertado, dentro del circuito diseñado por el usuario, en el zócalo destinado al monochip. El PC se encargará de suplir al monochip.

La tarjeta es una *protoboard*, o tarjeta de prototipos IBM, que incorpora de fábrica, básicamente, un decodificador de direcciones. En esta tarjeta se han instalado:

- un adaptador paralelo de periféricos (PPI, 8255)
- un conversor A/D (ADC 0808)
- un decodificador 74LS138
- un buffer triestado
- un generador de onda cuadrada (NE555). El conversor necesita recibir, para un correcto funcionamiento, una señal de este tipo de una frecuencia aproximada a 500 KHz.

El decodificador instalado permite asignar un rango de direcciones de activación a cada elemento o señal a controlar. Las direcciones se han asignado tal como se indica en la tabla 1.

Para acceder al PPI se necesitan las cuatro direcciones, pues dispone de cuatro registros: Registro de control (303h), registros de datos de los puertos A, B y C (300h, 301h y 302h respectivamente). Sin embargo, para dejar el resultado del bit del conversor EOC (fin de conversión) en el bus de datos sólo se necesita activar una dirección de las cuatro que se le han asignado. Se ha optado por la 304h, como se podría haber escogido cualquiera de las otras tres disponibles.

Sucede lo mismo con la señal OE (*output enable*), también del conversor, que vuelca en el bus de datos el resultado de la conversión. Se ha seleccionado la dirección 308h de entre las cuatro posibles.

Los puertos del PPI hacen las veces de puertos del monochip. Todas trabajan en modo 0 y su configuración como entrada o salida la realiza el software mediante la programación del registro de control del PPI, cuando el programa de control que se está depurando lo solicita. Las entradas que pueden ser también entradas analógicas se restringen a PC4-PC7, que se conectan además de a los puertos correspondientes del PPI, a las entradas IN0-IN3 del conversor A/D.

Como ya se sabe, el microcontrolador ST6225 sólo dispone para el puerto C de cuatro líneas (PC4-PC7), por lo que en el PPI, que dispone de ocho, se supone que sobrarían cuatro. De estas cuatro *sobrantes*, tres son usadas para controlar el conversor: dos (PC1 y PC2) para seleccionar la entrada del conversor y la otra (PC0) para activar la señal START, o de inicio de conversión. La cuarta señal (PC3), simula la patita TIMER del monochip.

El subprograma que gestiona el hardware se ha dividido en una serie de funciones, como pueden ser: Configurar_Registro_De_Control o Leer_Resultado_Conversión. Todas esas funciones son llamadas desde el programa principal, que es prácticamente idéntico a MCSIM. Se podría decir que el simulador es el *corazón* del emulador. De hecho, MCEM se desarrolló basándose íntegramente en MCSIM y una vez que éste estuvo perfectamente depurado.
MCEM presenta una serie de limitaciones, que se deben tener en cuenta a la hora de diseñar programas que se vayan depurar con esta utilidad:

- Sólo pueden usarse como entradas analógicas las líneas PC4-PC7.
- La línea TIMER (PC3) es sólo de salida.
- Los puertos deben ser unidireccionales
- Las línea del monochip RESET y NMI no han sido implementadas por hardware. Estas señales sólo se pueden activar desde menú.
- La ejecución de un programa, aún en el modo *TURBO*, es más lenta que en un monochip. Hay que tener en cuenta que por cada microinstrucción del programa de control a ejecutar, MCEM debe ejecutar una media de treinta instrucciones de alto nivel (lenguaje PASCAL).

Por contra, MCEM presenta unas ventajas importantes:

- La más importante, el tiempo que ahorra al usuario en la culminación con éxito de su trabajo, pues permite hacer un seguimiento del programa y observar los resultados de su ejecución en el propio entorno en el cual irá instalado el monochip posteriormente. Sería una depuración del programa más cercana a la realidad que la realizada con MCSIM.
- Hardware muy económico. La parte hardware de MCEM cuesta menos, para hacerse una idea, que un simple filtro antirradiación para monitor.

4.- Conclusiones

MCSIM y MCEM son herramientas que pretenden ser una ayuda para el usuario, ayuda que se traduce en un descenso en los tiempos de lanzamiento del programa y en quebraderos de cabeza. No son estrictamente complementarias, por lo cual, la elección de una u otra utilidad queda en manos del usuario. Sin embargo, se recomienda realizar la depuración en dos etapas:

1ª. En la primera, usando MCSIM para detectar los errores lógicos. Es recomendable elaborar previamente unos juegos de ensayo para contrastarlos después con la información suministrada durante la ejecución del programa.
2ª. En la segunda etapa, con la ayuda de MCEM, se realizarían los ajustes finos, tarea que se facilita mucho al poder observar la ejecución del programa en el que será su futuro entorno físico. Un ejemplo de ajuste a realizar en esta etapa sería el del retardo a generar entre la lectura de dos pulsaciones consecutivas de un pulsador.

5.- REFERENCIAS

[1] Gil Larrea, M.J., Gutiérrez Temiño, Lago Rey, A. "Microcontroladores, computadores integrados de bajo coste". *Eurofach electrónica*, nº 216, JUL/AGO.94

[2] Gil Larrea, M.J., Gutiérrez Temiño, Lago Rey, A., Tirado Uria, J.L. "Simulador y emulador de microcontroladores ST6225". *Eurofach electrónica*, nº 218, SEPT.94

[3] Gutiérrez Temiño, J.L., Etxebarria, M., Jiménez, P.M. "Micro'85". Ed. Rede, 1994. ISBN 84-247-0307-3