Cika Electrónica

Nota de Aplicación: CAN-080
Título: **Utilización de la EEPROM en HT48E**
Autor: Sergio R. Caprile, Senior Engineer

Revisiones	Fecha	Comentarios
0	04/07/07	

Los micros de la serie HT48E de Holtek incorporan una EEPROM equivalente a las memorias de la serie 93LCx6. En particular, hasta 48E30 se trata de una 93LC46, mientras que 48E50 y 48E70 incorporan una 93LC56. La presente nota de aplicación muestra la forma de utilizar una biblioteca de funciones para acceder a la EEPROM tanto desde C como assembler, aprovechando código desarrollado por Holtek. El código original en assembler ha sido modificado para soportar acceso desde C y assembler de manera indistinta, y la posibilidad de que el compilador modifique los registros BP y MP1, lo cual ocurre al acceder variables declaradas como externas en C.

Utilización desde C

Incluimos dos header files, *E93LC46.h* y *E93LC56.h*, de modo que el compilador pueda chequear la sintaxis y el usuario conozca las funciones que debe llamar.

Lectura

Para leer en una posición de la EEPROM, utilizamos la siguiente función:

```
var=READ(address);
```

pasando como parámetro la dirección en que se encuentra el dato.

Protección contra escritura

Para poder escribir y/o borrar, necesitamos habilitar la escritura. Una vez realizada la operación es conveniente volver a impedirla. Para ello utilizamos las siguientes funciones:

```
EWEN();         // habilita escritura
EWDS();         // impide escritura
```

Escritura

Para escribir en una posición de la EEPROM, utilizamos la siguiente función:

```
WRITE(address,data);
```

pasando como parámetros la dirección en que se escribe el dato y el valor del mismo.

Borrado

Podemos borrar una posición con la siguiente sentencia:

```
ERASE(address);
```

pasando como parámetro la dirección en que se encuentra el dato a borrar.

Borrado Total

Podemos borrar toda la memoria llamando a la rutina correspondiente:

```
ERAL();
```

Relleno

Podemos poner toda la memoria a un valor particular llamando a la rutina correspondiente:

```
WRAL(data);
```

pasando como parámetro el valor deseado.

El archivo adjunto muestra ejemplos de utilización

Utilización desde assembler

Incluimos dos header files, *E93LC46.inc* y *E93LC56.inc*, de modo que el ensamblador conozcas las funciones y variables y el usuario conozca las funciones que debe llamar. La dirección del dato se aloja en la variable *EEADDR*, mientras que el dato lo hace en *EEDATA*. La función de lectura retorna además el dato leído en el acumulador.

Inicialización

El usuario deberá poner los registros BP y MP1 al valor correcto para su utilización por las rutinas, lo cual se realiza con la siguiente porción de código:

```
mov A,01h
mov BP,A                        ; bank 1
mov A,040h
mov MP1,A                       ; MP1 = EECR
```

De ser necesario, se puede acceder a las rutinas para ser llamadas desde C, las cuales realizan una llamada a una rutina que realiza la función de setear estos registros. Dichas rutinas tienen el nombre precedido por '_', por ejemplo: _READ. en la mayoría de los casos en que el programador tiene el control y no utiliza o controla adecuadamente el registro MP1, esto no será necesario.

Lectura

Para leer en una posición de la EEPROM, utilizamos la siguiente porción de código:

```
mov A,address
mov EEADDR,A
call READ
```

a continuación, disponemos del dato en *ACC* o en *EEDATA*

Protección contra escritura

Para poder escribir y/o borrar, necesitamos habilitar la escritura. Una vez realizada la operación es conveniente volver a impedirla. Para ello utilizamos las siguientes operaciones:

```
call EWEN      ; habilita escritura
call EWDS      ; impide escritura
```

Escritura

Para escribir en una posición de la EEPROM, utilizamos la siguiente porción de código:

```
mov A,address
mov EEADDR,A
mov A,data
mov EEDATA,A
call WRITE
```

Borrado

Podemos borrar una posición con la siguiente porción de código:

```
mov A,address
mov EEADDR,A
call ERASE
```

Borrado Total

Podemos borrar toda la memoria llamando a la rutina correspondiente:

```
call ERAL
```

Relleno

Podemos poner toda la memoria a un valor particular llamando a la rutina correspondiente:

```
mov A,data
mov EEDATA,A
call WRAL
```

El archivo adjunto muestra ejemplos de utilización