

Temas Varios *del 68HC908....*

TBM Module

LVI Module

FLASH Memory

Uso de la FLASH como EEPROM...

Curso de Microcontroladores Familia HC908 Flash de
Motorola , **Parte II**

ING. DANIEL DI LELLA DDFAE For Motorola Products

Memoria FLASH de los 908.....

- Programable en circuito por medio de **un solo pin** (PTA0 / PTB0 según el MCU)
- Reprogramable en circuito **sin tensión elevada externa** (funciona con la misma VDD)
- **Puede ser usada como memoria de datos temporales similar a una EEPROM (del tipo 93Cxx)**
- 10.000 ciclos de Escritura/Borrado Mínimos garantizados (a – 40°C, la peor condición)
- **+ de 100.000 ciclos de Escritura/Borrado a temp. Ambiente (entre 20 y 30 °C)**
- Control de la programación por medio del registro **FLCR (Flash Control Register)**
 - **HVEN** -- Bit habilitación de la Bomba de Alta tensión p/ programación
 - **MASS** --- Bit habilitación de Borrado en Masa de la Flash (Mass Erase Control)
 - **ERASE** -- Bit habilitación para el Borrado de la Flash (interconectado con el bit PGM
para evitar que ambos estén en “ 1 ” al mismo tiempo
 - **PGM** ---- Bit de control de programación de la Flash (interconectado con ERASE)
- Tiempo de borrado en MASA (32K) = **4 mSeg.**
- Tiempo de programación de una fila = **1 mSeg.**

Memoria FLASH de los 908.....

- La memoria Flash está dividida en "PAGINAS" de 128 Bytes por página (para el GP32)
- El **mínimo tamaño de memoria a borrar** es una PAGINA o sea 128 Bytes (para el GP32)
- El borrado de la FLASH se hace de página en página o en forma masiva (MASS ERASE)
- La escritura de la flash** está organizada en "FILAS" de 64 Bytes por FILA (para el GP32).
- Una PAGINA es equivalente a 2 (dos) FILAS.
- Se escribe por FILAS en la FLASH de los HC908....
- La escritura se hace desde código escrito en RAM o ROM y NO sobre código en FLASH !!!**
- Tengo que transferir el código escrito en FLASH a RAM y ejecutarlo desde allí o si el MCU tiene en su ROM las sub-rutinas de manejo de la FLASH, se invoca a estas desde el código en FLASH (ver nota de aplicación **AN-1831** de Motorola) .
- Existe un mecanismo de protección contra escritura involuntaria muy efectivo
 - El registro **FLBPR (Flash Block Protect Register)** posee 8 Bits que permiten la protección de bloques de memoria a partir de una dirección prefijada por los mismos hasta el FIN DEL MAPA DE MEMORIA o sea \$ FFFF
- Si se habilita el **LVI**, se garantiza la NO CORRUPCION de la memoria FLASH
- Ver secciones "FLASH Memory" & "Monitor ROM (MON08)" en el manual del MCU elegido.

Curso de Microcontroladores Familia HC908 Flash de
Motorola , **Parte II**

LVI (Low Voltage Inhibit) ó supervisor de Tensión

- Produce un **RESET** por detección de **baja tensión** de **VDD**.
- Identificable por medio del **flag LVI** en el **SRSR** (SIM Reset Status Register)
- Trip point (punto de disparo) ajustable por soft para funcionamiento en 3 y 5V de VDD (2 puntos de disparo), registro CONFIG1, bit “LVI5OR3”.
- Histéresis incorporada en el comparador (asegura reposición cuando VDD está O.K)
- Mantiene al MCU en estado de RESET (no operativo) si la tensión es inferior al mínimo establecido
- El módulo puede **habilitarse / deshabilitarse** por soft en el registro CONFIG1 por medio del bit “LVIPWRD”.
- Puede generar un RESET o una indicación (Modo “Polled”) por medio del registro CONFIG1 de la siguiente forma:
 - **Modo RESET:** en CONFIG1 --- **LVIPWRD = 0 & LVIRSTD = 0**
 - **Modo POLLED ó “poleado” :** en CONFIG1 --- **LVIPWRD = 0 & LVIRSTD = 1**

Ver secciones “Configuration Register (CONFIG)” y “Low – Voltage Inhibit (LVI)” del respectivo manual del MCU elegido.

TBM (Time Base Module) ó Módulo de Base de Tiempo

- Base de Tiempo independiente del CPU, no lo utiliza para su funcionamiento.
- Produce una interrupción periódica según lo programado en un registro (**TBCR**)
- La fuente del clock es el cristal de **32,768 Khz** que se utiliza en el **PLL**
- Tiempos de interrupción de 1Hz, 4, 16, 256, 512, 1024, 2048, y 4096 Hz programables por medio de los bits **TBR2 / TBR1 / TBR0**.
- El registro **TBCR** (**Timebase Control Register**) controla el funcionamiento y habilitación del mismo.
- Util para generar un **RTC (Reloj de Tiempo Real)** o para despertar periódicamente al MCU de un estado **WAIT** o **STOP**, garantizando así un muy bajo consumo de energía, para aplicaciones que lo requieran.
- Ver sección “**Timebase Module (TBM)**” en el manual del MCU elegido.

Uso de la memoria FLASH de los MCUs HC08 para el Almacenamiento de Datos Temporales.

Por el Ing. Daniel Di Lella, Depto. Técnico Electrocomponentes S.A.

Una pregunta común cuando se comienza a trabajar con la familia de los nuevos microcontroladores HC08 FLASH de Motorola o bien cuando se está por decidir la Migración a uno de estos interesantes MCUs, es si es posible utilizar la memoria Flash de los mismos no solo para "grabar" programas en ella (memoria de programa) sino además para "almacenar" datos temporales (variables temporales, seteos, tablas transitorias, etc.) como si se tratara de una memoria "EEPROM" o similar NO - Volatil.

En los MCUs OTP ROM (One Time Programming ROM) esto era SOLO posible con el agregado de una memoria "externa" del tipo serial de 8 pines como las 93Cxx..

La memoria FLASH de 2da generación que poseen los MCUs HC08 es ideal para reemplazar ventajosamente el uso de memorias EEPROMs externas por los siguientes motivos:

- Al no utilizarse un chip externo de memoria, se dispone de mayor cantidad de Pines I/O para usos generales y NO para el control de la memoria.
- Ahorro en el costo del chip externo de memoria y en el espacio de PCB a utilizar.
- Flexibilidad en el tamaño de la memoria a utilizar (se utiliza solo lo necesario, dejando el resto para la memoria del programa).
- Circuitos adicionales integrados en el MCU como el LVI (Low Voltage Inhibit), necesarios para asegurar un funcionamiento confiable en sistemas con memoria No - Volatil .

Existen dos modos de "grabar" la memoria FLASH de los MCUs HC08, a saber:

1) En el "Modo Monitor" , que es un modo muy particular que poseen estos MCUs, puede grabarse la memoria flash de estos, además de emplearse este modo para el "debugging" en tiempo real de programas. Este modo permite la actualización del programa contenido en la memoria Flash aún cuando el chip se encuentre soldado a la placa del usuario. Esta modalidad se la conoce con el nombre de In - Circuit Programming (I.C.P.)

2) En el "Modo Usuario" , que es el modo de funcionamiento normal del MCU, (en el cuál se correrá el programa implementado por el usuario), es posible alterar el contenido de la memoria flash, sin necesidad de tensiones o uso de pines especiales para tal fin.

El primero de los modos, es ideal cuando se requiere grabar un programa (aplicativo) en la memoria de un dispositivo "virgen" (de fábrica o borrado totalmente) durante la producción o desarrollo y durante posteriores "actualizaciones" (revisiones) del software originalmente implementado en el mismo.

Para este modo, existen herramientas de software y hardware (**PROG08SZ, E-FLASH08, EVAL08xx, etc.**) que facilitan la Programación En - Circuito (I.C.P.) o la programación de la totalidad del chip para las operaciones anteriormente descritas.

En el segundo de los modos, y punto fundamental de nuestro artículo, **puede utilizarse la Flash para almacenar datos "temporales" en memoria No - Volátil**, con similares prestaciones a los chips EEPROM comunmente utilizados para tal fin.

Si tenemos en cuenta la estructura interna de la memoria FLASH de los HC08, la misma NO permite su alteración por medio de código escrito en su propia memoria, dicho de otra forma, no es posible alterar la flash con código "corriendo" en la misma flash. Esto que en principio parecería un "inconveniente" , garantiza (junto con otros recursos disponibles) la inalterabilidad de los datos ante programas del tipo "gusano" (que se "comen" código) frecuentes por errores en el programa u otras circunstancias.

Curso de Microcontroladores Familia HC908 Flash de
Motorola , **Parte II**

ING. DANIEL DI LELLA DDFAE For Motorola Products

Entonces, para alterar (grabar o Borrar) la flash, se dispone de dos (2) métodos, en el modo usuario:

- **Ejecución de Código desde RAM :**

Con este método, es posible grabar o borrar la memoria flash, con código "corriendo" desde memoria RAM del MCU, para ello el programa de usuario implementado en la flash, deberá incluir una sub-rutina que realice una "migración" (copia) de líneas de código desde la flash (estas líneas contendrán el código necesario para modificar la memoria flash) hacia la RAM. Una vez copiado el código, y cuando se quiera borrar o grabar la flash con algún valor, el programa del usuario deberá llamar al código en RAM como si fuera una o varias sub-rutinas, con la particularidad que su ubicación no es la usual, sino espacio RAM.

Este método es común de usar en derivados de la flia. HC08 con memorias RAM superiores a los 128 Bytes, por ejemplo los 908GP32, 908MR32/24/16/8, 908SR12, Etc. La ventaja principal, radica en la facilidad que dispone el usuario para mejorar o modificar las sub-rutinas encargadas del borrado o grabación de la flash. La desventaja es que se ocupa temporalmente espacio de RAM, por lo que hay que tener cuidado de ello.

- **Ejecución de Código desde ROM (ROM "Monitor") :**

Con este método, es posible grabar o borrar la memoria flash, con código "corriendo" desde memoria ROM del MCU, para ello el programa de usuario implementado en la flash, deberá invocar por medio de un salto a sub-rutina, la dirección de comienzo de cada una de las sub-rutinas incluidas en la ROM de los distintos derivados de HC08 que posean dichas rutinas.

Los derivados que poseen estas rutinas en ROM son, por ejemplo, 908JL3, 908JK3, 908JK1, 908GR8, 908KX8 / KX2, 908JB8, etc..

En estos MCUs, la memoria RAM es del orden de los 128 Bytes o similar, por lo que no es aconsejable ejecutar código desde esta, por el poco espacio disponible para tal fin.

Para mayor detalle de funcionamiento de cada uno de estos métodos, se sugiere al lector consultar las siguientes notas de aplicación de Motorola :

- [AN1831](#)
- [AN - HK - 32](#)
- [AN - HK - 33](#)
- [AN1770](#)

Estas notas se pueden obtener de los siguientes web site:

www.mcu.motpsps.com

www.electrocomponentes.com.ar

En el presente artículo, se explicará un programa ejemplo de cómo utilizar las rutinas incorporadas dentro de la ROM monitor de algunos derivados de la flia. HC08.

El programa se basa en la nota de aplicación [AN1831 "Using MC68HC908 On - Chip FLASH Programming Routines"](#) y en el mismo se utilizará un MC68HC908JK3, [pero es valido su uso para otros dispositivos.](#)

Curso de Microcontroladores Familia HC908 Flash de
Motorola , **Parte II**

ING. DANIEL DI LELLA DDFAE For Motorola Products

El Programa.

En el programa "demo3jl" , se efectua la "memorización" de datos temporales (variable Num_RAM auto-incrementada en cada corrida del programa) en un espacio reservado de la memoria FLASH (en este caso \$EC00, pero podría ser cualquier dirección de la Flash), que en el comienzo posee \$FF (en blanco) en toda su extensión (TABLA).

Se utilizarán las sub - rutinas "PRGRNGE" y "ERARNGE" contenidas en la ROM.

También se incluyó un "artilugio" para extender la "vida útil" de la Flash (que en condiciones normales, osea sin este artilugio, soporta unos 10.000 ciclos grab. / borr a -40 °C y más de 100.000 entre 20 y 30 °C) unas 64 veces más, ya que se irán desplazando los valores memorizados al próximo espacio "en blanco" disponible en la TABLA reservada para los valores temporales y solo se efectuará el borrado de la "página" cuando se complete la tabla (valores todos distintos de \$FF).

Debe hacerse notar, los espacios reservados para las variables utilizadas por las sub - rutinas en ROM en la RAM del MCU, para mayor detalle, se sugiere leer atentamente la nota **AN1831 de Motorola.**

```

*****
* DEMO3JL.ASM - PROGRAMA DEMO NUMERO 3 - FLASH PROGRAMMING *
*   SEMINARIO DE ACTUALIZACION MICROCONTROLADORES HC08 FLASH *
*   ELECTROCOMPONENTES S.A 2001 *
* *
* SE USARAN LAS RUTINAS CONTENIDAS EN LA ROM "MONITOR" SEGUN AN1831 *
*   DISPOSITIVOS JL3 / JK3 / JK1 *
*****
*****
* EQUATES *
*****

```

```

BASE 10T           ; BASE DECIMAL POR DEFAULT
INCLUDE 'j13regs.inc' ; equates registros 68HC908JL3

```

```

RAM      EQU $0080      ; Comienzo de la RAM para el JL3
DATSTRC  EQU RAM+8      ; RAM para uso del "MONITOR ROM"
RAMUSER  EQU DATSTRC+7  ; Comienzo de la RAM p/ usuario
DATAFLASH EQU $EC00     ; Comienzo de la FLASH para almacenar datos
RomStart EQU $ED00      ; Comienzo de la ROM/FLASH para el programa
VectorStart EQU $FFDE   ; Comienzo tabla de vectores varios
GETBYTE  EQU $FC00      ; Sub-Rutina en ROM TX/RX un Byte x port
RDVRRNG  EQU $FC03      ; Sub-Rutina en ROM lectura y verif. de un rango
ERARNGE  EQU $FC06      ; Sub-Rutina en ROM borrado row / mass Flash
PRGRNGE  EQU $FC09      ; Sub-Rutina en ROM escritura datos en FLASH
DELNUS   EQU $FC0C      ; Sub-Rutina en ROM DELAY "N" microsegundos

```

Curso de Microcontroladores Familia HC908 Flash de
Motorola , **Parte II**

* VARIABLES EN RAM *

ORG RAM ; Comienzo RAM para JK3
; reservada hasta DATSTRC P/ uso
; rutinas ROM prog/erase FLASH.
ORG DATSTRC ; RAM p/ uso en prog/erase flash
CTRLBYT RMB 1 ; BIT 6 -- MASS FLAG, 1 -- mass erase
CPUSPD RMB 1 ; CPU speed = 4 x Fop aprox.
LADDR RMB 2 ; Last Address p/ Read o Write un rango
DATA RMB 1 ; DATOS en RAM transitorios

ORG RAMUSER ; RAM para uso del programa gral.
Num_RAM RMB 1 ; Valor HEXA del numero ALMACENADO .
AUX RMB 1 ; Variable auxiliar Gral.

* FLASH DE DATOS *

ORG DATAFLASH
TABLA FCB \$FF,\$FF,\$FF,\$FF,\$FF,\$FF,\$FF,\$FF ; comienzo tabla FLASH de 64 Bytes
FCB \$FF,\$FF,\$FF,\$FF,\$FF,\$FF,\$FF,\$FF ; al principio tendrá TODO con \$FF
FCB \$FF,\$FF,\$FF,\$FF,\$FF,\$FF,\$FF,\$FF ; luego se ira llenando con valores
FCB \$FF,\$FF,\$FF,\$FF,\$FF,\$FF,\$FF,\$FF ; memorizados de esta DEMO
FCB \$FF,\$FF,\$FF,\$FF,\$FF,\$FF,\$FF,\$FF ; y vuelta a limpiarse .
FCB \$FF,\$FF,\$FF,\$FF,\$FF,\$FF,\$FF,\$FF
FCB \$FF,\$FF,\$FF,\$FF,\$FF,\$FF,\$FF,\$FF
FCB \$FF,\$FF,\$FF,\$FF,\$FF,\$FF,\$FF,\$FF

Curso de Microcontroladores Familia HC908 Flash de
Motorola , **Parte II**

```
*****
* FLASH DE PROGRAMA *
*****
```

ORG RomStart

```
*****
* Config. del MCU *
*****
```

```
START    MOV #$11,CONFIG1    ; Reg. CONFIG1->LVID=1,STOP=0,COPD=1
          MOV #$00,CONFIG2 ; Reg. CONFIG2->IRQPUD,LVIT1,LVIT0=0
          NOP
          NOP                ; NOPs de delay aseguran configuración.
          NOP
          MOV #$00,PTAPUE    ; sin Rs PULL UP en PortA
          MOV #$00,PORTA    ; PTA0 a PTA7->IN
          MOV #$00,DDRA
          MOV #$00,PORTA    ; Fuerzo ceros al PORTA
          MOV #$00,PORTB    ; PORTB todo como ENTRADA
          MOV #$00,DDRB    ;
          MOV #$00,PORTB    ; Fuerzo ceros al PORTB
          MOV #$00,PDCR    ; PORTD Control register todo cero
          MOV #$00,PORTD    ; PORTD todo como ENTRADA
          MOV #$00,DDRD
          MOV #$00,PORTD    ; Fuerzo ceros al PORTD
```

Curso de Microcontroladores Familia HC908 Flash de
Motorola , **Parte II**

* DESHABILITO *

* CONV. A/D E *

* INTERRUPCIONES *

MOV #\$1F,ADSCR ; ADC --- OFF

MOV #\$06,INTSCR ; IRQ1 --- DESHABILITADA !!!

MOV #\$06,KBSCR ; KBI --- DESHABILITADA !!!

MOV #\$00,KBIER

SEI ; I MASK=1 --- INT's deshabilitadas

* SETEO INICIAL DE *

* VARIABLES *

CLR Num_RAM ; LIMPIO VARIABLES

Curso de Microcontroladores Familia HC908 Flash de
Motorola , **Parte II**

ING. DANIEL DI LELLA DDFAE For Motorola Products

```

*****
* Busqueda en Flash del ultimo valor      *
* de la variable NUMERO y luego la       *
* traslado a RAM " Num_RAM "             *
* Busco ultimo valor en tabla flash      *
* distinto a $FF (lo uso como marca)    *
* TABLA --> 64 Bytes en Flash ($EC00)    *
*****

```

```

                CLRH                ;Limpio H (no se usa) (ver HC08)
                CLRX                ;Limpio X para usarlo como puntero
                LDA TABLA,X         ;Acc <-- tabla+X (inicio busqueda)
                STA Num_RAM         ;Num_RAM con valor de tabla
Searching      CBEQX #$40,End_Search ;X=64? -> si -> termino busqueda
                CBEQA #$FF,End_Search ;sino sigo buscando hasta Acc=$FF
                STA Num_RAM         ;guardo posible valor en RAM
                INCX                ;INCR. X para apuntar a nuevo valor
                LDA TABLA,X         ;Acc <-- nuevo valor de tabla
                BRA Searching       ;sigo buscando !!!

End_Search     NOP

```

* INCREMENTO EL VALOR RESCATADO *
* DE LA TABLA FLASH Y LO VUELVO *
* A GUARDAR EN LA MISMA PERO EN *
* EL PROXIMO LUGAR VACANTE !!! *

INC Num_RAM ; INCREMENTO EN 1 EL VALOR PARA
JSR WR_FLASH ; LUEGO IR A GUARDARLO EN FLASH

ENDLESS BRA ENDLESS ; LOOP SIN FIN P/ ESPERAR STOP MANUAL

* WR_FLASH - SUB-RUTINA DE ESCRITURA DE DATOS EN FLASH *
* , espacio reservado en TABLA (\$EC00) *
* para guardar la configuracion del sistema (NUMERO) (64 bytes para ello) *
* Para el JK3 se borrarán de a 64 bytes (page) y se grabará de a 1 byte por ciclo *
* ya que no necesito más que ello *

WR_FLASH CLRH ; Limpio índice H (no se usa)
CLRXL ; Limpio puntero de TABLA flash
NEWFLASH LDA TABLA,X ; ACC <- dato tabla a analizar
CPX #\$3F ; Se llegó al final de la tabla?
BEQ ERAPAGE ; si es así -> borrado de FILA !
CMP #\$FF ; SINO el byte de tabla es = \$FF?
BEQ WFLASH ; SI -> Write Flash
INCL ; NO -> busco nuevo byte "virgen"
BRA NEWFLASH ; con \$FF p/ escritura en el.

Curso de Microcontroladores Familia HC908 Flash de
Motorola , **Parte II**

* Invoco a rutina *

* en ROM PRGRNGE *

```
WFLASH      STX AUX          ; guardo X en AUX temporalmente
             CLRH           ; Limpio indice H (no se usa)

             MOV #00,CTRLBYT ; CTRLBYT = 0 -> MASS ERASE = 0
             MOV #10,CPUSPD  ; CPUSPD=10 -> 4xFop -> 4x 2,45
             LDHX #TABLA     ; HX <- #TABLA, FIRST ADDRESS

DESPLA      LDA AUX          ;
             CMP #$00        ;
             BEQ END_DESPLA  ;
             AIX #+1         ;
             DEC AUX         ;
             BRA DESPLA     ;

END_DESPLA  STHX LADDR       ; HX -> LADDR, LAST ADRESS
             LDA Num_RAM     ; Num_RAM -> DATA , vuelco a RAM
             STA DATA       ; Valor a guardar en FLASH !!
             LDA #$FF        ; Debo escribir cualquier cosa
             STA FLBPR       ; en el registro "FLBPR" antes...
             JSR PRGRNGE     ; rutina en ROM "PRGRNGE" p/ wflash
             NOP             ; DELAY P/ MEJORAR GRABACION
             NOP             ;
             BRA VUEL_FLASH  ; ***** retorno a ppal *****
```

Curso de Microcontroladores Familia HC908 Flash de
Motorola , **Parte II**

* borro PAGE *

* (64 BYTES) *

* FLASH *

```
ERAPAGE      MOV #00,CTRLBYT      ; CTRLBYT = 0 -> MASS ERASE -> P.E
              MOV #10,CPUSPD      ; CPUSPD=10 -> 4xFop -> 4x 2,45
              LDHX #TABLA          ; HX <- comienzo pagina cero
              JSR ERARNGE          ; Rutina en ROM "ERARNGE" p/ borrar
              NOP                  ; DELAY P/ MEJORAR BORRADO
              NOP                  ;
              NOP                  ;
              NOP                  ;
              MOV #00,CTRLBYT      ; CTRLBYT = 0 -> MASS ERASE = 0
              MOV #10,CPUSPD      ; CPUSPD=10 -> 4xFop -> 4x 2,45
              LDHX #TABLA          ; HX <- #TABLA, FIRST ADDRESS
              STHX LADDR           ; HX -> LADDR, LAST ADRESS
              LDA Num_RAM          ; Num_RAM -> DATA , vuelco a RAM
              STA DATA            ; valor a guardar en FLASH !!
              LDA #$FF             ;
              STA FLBPR            ;
              JSR PRGRNGE          ; rutina en ROM "PRGRNGE" p/ wflash
              NOP                  ;
              NOP                  ;
VUEL_FLASH   RTS                  ; ***** RETORNO *****
```

Curso de Microcontroladores Familia HC908 Flash de
Motorola , **Parte II**

```

*****
* INTERRUPT VECTORS TABLE - *
* RESET / SWI / IRQ1 / ADC / Y NO ASIGNADOS ---> START *
* KEYBOARD INTERRUPT VECTOR ---> START *
* TIM1 OVERFLOW VECTOR ---> START *
*****

```

```

ORG VectorStart

```

```

dw START ; ADC Conversion Complete Vector
dw START ; Keyboard Vector
dw START ; (No Vector Assigned $FFE2-$FFE3)
dw START ; (No Vector Assigned $FFE4-$FFE5)
dw START ; (No Vector Assigned $FFE6-$FFE7)
dw START ; (No Vector Assigned $FFE8-$FFE9)
dw START ; (No Vector Assigned $FFEA-$FFEB)
dw START ; (No Vector Assigned $FFEC-$FFED)
dw START ; (No Vector Assigned $FFEE-$FFEF)
dw START ; (No Vector Assigned $FFF0-$FFF1)
dw START ; TIM1 Overflow Vector
dw START ; TIM1 Channel 1 Vector
dw START ; TIM1 Channel 0 Vector
dw START ; (No Vector Assigned $FFF8-$FFF9)
dw START ; ~IRQ1
dw START ; SWI Vector
dw START ; Reset Vector

```

FIN DEL PROGRAMA !!

Curso de Microcontroladores Familia HC908 Flash de
Motorola , **Parte II**

“Corriendo” el programa en el sistema E-FLASH08.....



Fig. 1 - Ventana de Memoria en el sistema *E-FLASH08*, donde se observa el Bloque de memoria "TABLA" totalmente borrado (\$FF).

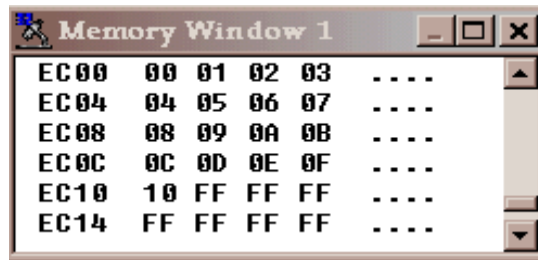


Fig. 2 - La misma ventana, donde se observa el bloque de memoria, pero con sucesivas corridas del programa (posiciones de memoria grabadas).

Grabación de memoria FLASH en el MC68HC908GP32

Por el Ing. Gabriel Dubatti

Adaptación Ing. Daniel Di Lella Dto. Técnico Electrocomponentes S.A.

Introducción:

Como hemos visto anteriormente, es posible utilizar la memoria FLASH de los microcontroladores HC908 de Motorola para almacenar datos temporales No - Volátiles como si usáramos la típica configuración de EEPROM externa del tipo 93Cxx.

En ese artículo, se mostró en detalle el uso de la memoria FLASH en los dispositivos de menos de 8Kbytes de memoria de programa flash., tales como los 908JL3 / JK3 / JK1 o 908KX8 / KX2, etc.

En el presente artículo, trataremos en detalle la grabación y borrado de la memoria FLASH de MCUs HC908xx con memorias superiores a 8Kbytes como los 908GP32, 908MR32 / MR16, 908SR12, etc. En particular se eligió el 908GP32 como ejemplo de uso de la memoria FLASH como EEPROM.

El MC68HC908GP32 es un poderoso microcontrolador de Motorola™ basado en la tecnología FLASH.

Entre sus características más importantes figuran:

- 32 KBytes de memoria FLASH (con un ciclo mínimo de 100.000 grabaciones a 20°C .)
- 512 Bytes de RAM
- SPI, SCI, 2 timers muy completos, etc.
- Protección contra lectura del código.

Curso de Microcontroladores Familia HC908 Flash de
Motorola , **Parte II**

ING. DANIEL DI LELLA DDFAE For Motorola Products

El procesador tiene dos modos de trabajo básicos: **monitor** y **usuario**.

Para ingresar en modo **monitor** el procesador debe tener el vector de reset borrado (\$FFFF) o se debe aplicar una tensión elevada en el PIN de IRQ y poner algunos ports en un valor determinado.

Luego se deben enviar los 8 bytes de seguridad, que si coinciden con los últimos 4 vectores grabados, permite la lectura y modificación de la FLASH y si no coinciden sólo permite un borrado masivo.

Los utilitarios de grabación trabajan en este modo. Si no se ingresa en modo monitor, se está en modo **usuario**.

Este artículo muestra como utilizar la memoria FLASH para grabar datos modificables durante la ejecución (modo usuario) como si fuera una memoria EEPROM.

Organización de la memoria FLASH:

La memoria FLASH está organizada en páginas de 128 bytes, permitiéndose la grabación mínima de 1 byte, el borrado mínimo de 1 página y el borrado completo (o masivo) en una sola operación.

Se pueden grabar hasta 64 bytes (un row) por operación pero el algoritmo presentado aquí lo hace de a un byte por vez para utilizar el mínimo posible de RAM.

Cuando un byte de la memoria está borrado se lee como \$FF y sólo se le pueden grabar bits en 0.

Para pasar un solo bit de 0 a 1, deben borrarse los 128 bytes de la página.

Mientras se modifica la FLASH esta no puede ser utilizada. Esto obliga a ejecutar el código de grabación

desde RAM (o ROM como ocurre en los otros micros de la familia como el JL3) y no pueden

atenderse las interrupciones. El algoritmo presentado **graba 1 byte en 100 microsegundos** y **borra una página (128 bytes) en 1 milisegundo**.

Los siguientes address están implementados en memoria FLASH:

\$8000 - \$FDFF (32256 bytes, páginas 0..251)

- **\$FF7E** (FLBPR, página 254)
- **\$FFDC - \$FFFF** (36 bytes, página 255)

El registro **FLBPR** (FLash Block Protect) permite indicar hasta que página se puede modificar en modo usuario (255 = toda la memoria, 0 = nada), para impedir sobregrabar o borrar por accidente el programa. Este registro sólo puede modificarse desde modo monitor.

Esto obliga a poner los datos que se deseen modificar en \$8000 (y bytes siguientes) y el código a continuación (hasta \$FFFF, cuidando de no usar una misma página para código y datos).

Mediante el siguiente código se puede programar el valor correcto del registro:

```
=====
ORG $8000                ;Comienzo de la memoria FLASH
Datos:                  ;
.....                  ; COLOQUE AQUI LOS DATOS A MODIFICAR
.....                  ;
=====

ORG {($+!127) & $FF80}  ;Alinea a la siguiente pagina de
                        ;128 bytes
CodigoStart:           ;Aqui comienza el codigo protegido
.....                  ;
.....                  ; COLOQUE AQUI SU CODIGO
.....                  ;
=====

ORG FLBPR                ;FLash Block PRotect
DB {(CodigoStart & $7FFF)/!128} ;donde empieza el CODIGO
                        ;a PROTEGER
=====
```

Nota: *El código que fuerza la alineación puede reemplazarse por una dirección fija.*

Curso de Microcontroladores Familia HC908 Flash de Motorola , **Parte II**

Algoritmo de grabación:

Para modificar un byte deben seguirse los siguientes pasos:

1. Tapar las interrupciones
2. Subir en el registro **FLCR** (FLash Control Register) el bit PGM (\$01)
3. Leer el registro **FLBPR** (FLash Block Protect)
4. Grabar cualquier valor en el byte a modificar (esto selecciona el row)
5. Delay TNVS = 10 microsegundos (PGM to HVEN setup time)
6. Subir en el registro **FLCR** (FLash Control Register) el bit HVEN (\$08)
7. Delay TPGS = 5 microsegundos (Program Hold Time)
8. Grabar el valor correcto en el byte a modificar
9. Delay TPROG = 30 microsegundos (Byte Program Time)
10. Bajar en el registro **FLCR** (FLash Control Register) el bit PGM (\$01)
11. Delay TNVH = 5 microsegundos (High-Voltage Hold Time)
12. Bajar en el registro **FLCR** (FLash Control Register) el bit HVEN (\$08)
13. Delay TRCV = 1 microsegundos (Return to read mode)
14. Recuperar estado de las interrupciones

Curso de Microcontroladores Familia HC908 Flash de
Motorola , **Parte II**

Algoritmo de borrado:

Para borrar una página deben seguirse los siguientes pasos:

1. Tapar las interrupciones
2. Subir en el registro **FLCR** (FLash Control Register) el bit ERASE (\$02)
3. Leer el registro **FLBPR** (FLash Block Protect)
4. Grabar cualquier valor en la página a borrar (esto selecciona la página)
5. Delay TNVS = 10 microsegundos (ERASE to HVEN setup time)
6. Delay TERASE = 1 milisegundo (Erase block time)
7. Bajar en el registro **FLCR** (FLash Control Register) el bit ERASE (\$02)
8. Delay TNVH = 5 microsegundos (High-Voltage Hold Time)
9. Bajar en el registro **FLCR** (FLash Control Register) el bit HVEN (\$08)
10. Delay TRCV = 1 microsegundos (Return to read mode)
11. Recuperar estado de las interrupciones

Implementación:

Las rutinas presentadas realizan estos dos algoritmos manteniendo al mínimo la cantidad de RAM necesaria. Para ello realizan la mayor cantidad posible de inicializaciones antes de grabar o borrar y no tocan la máscara de interrupciones, por lo que **requieren llamarlas con las interrupciones tapadas**.

Los **delays** están calculados para una **frecuencia de bus de 8 MHz** (lo que equivale a 1 microsegundo = 8 ciclos). Si decide utilizar otra frecuencia de bus **DEBERA** ajustar las constantes asegurándose que se mantengan los tiempos mínimos.

Por ejemplo: delay TERASE= 1 milisegundo

- Para 8MHz son 8000 ciclos y como cada loop tiene 33 ciclos, son 242.42.. iteraciones. La constante para el borrado debe ser **243** o sea: 8019 ciclos o 1.002 miliseg.
- Para 4MHz son 4000 ciclos y como cada loop tiene 33 ciclos, son 121.21.. iteraciones. La constante para el borrado debe ser **122** o sea: 4026 ciclos o 1.006 miliseg.

La rutina de grabación tiene un delay total de 101 microsegundos, dado que la copia de los 35 bytes a RAM requiere 350 ciclos que a 8MHz son 45 microsegundos (45% del tiempo total, que aumenta si se baja la frecuencia del bus). Si su aplicación no tiene problemas de RAM y sí de tiempo, puede modificar el código para que realice la copia a RAM una sola vez al inicializar y la ejecute directamente desde esa dirección (tenga cuidado en cargar los valores que requieren las rutinas en los registros antes de llamarlas).

Por su parte, **la rutina de borrado tiene un delay total de 1,068 milisegundos**. Aquí el tiempo de copia (igual que en la grabación) sólo representa el 4% del total, pero valen las mismas observaciones que en el caso anterior.

Como utilizar las rutinas:

Para grabar: indique el address a modificar en **FlashAddress** y el dato en **FlashData** y llame a la rutina: **FlashWRITE** (estas variables mantienen su valor al salir).

Para borrar: indique un address dentro de la página a borrar (por ejemplo el 1er byte de la página) en **FlashAddress** y llame a la rutina: **FlashERASE** (Atención: **FlashData** es usado de temporal y retorna con valor 0, en cambio FlashAddress mantiene su valor al salir).

Tenga en cuenta que cada vez que desee borrar un byte, deberá borrar 128 (la página completa).

Por lo tanto, según la cantidad de datos que desee grabar deberá implementar una rutina que se encargue de salvar los datos de la página que no cambian, la borre, regrebe los datos salvados y agregue los nuevos. Aquí tiene algunas opciones:

- Usar 1 byte por página (**contra:** desperdicia 127 bytes por página).
- Idem anterior, pero grabar en el 1er lugar vacío (\$FF) de la página. Esto disminuye el número de operaciones de borrado (ver el ejemplo) (**contra:** no se puede usar el valor \$FF como dato y al leer hay que buscar el último valor grabado).
- Usando RAM de backup. Copiar los datos a RAM, modificarlos, borrar la página y grabarla.
(**contra:** requiere RAM de backup)
- Usar 2 paginas. Usando el valor \$FF como indicador de vacío, puede intentar grabar en una página, si ese byte esta ocupado grabar en la siguiente página. Si ésta también esta ocupada, salvar los datos que estén en la 1era y no en la 2da. Borrar la 1era y grabar ahí los valores nuevos.
Hace falta agregar en la 1er página un flag que indique que página es la considerada 1era.
(**contra:** algoritmo más complejo y no permite usar el valor \$FF como dato, pero requiere muy poca RAM).

En el código fuente se incluyen las rutinas junto con un ejemplo de utilización e inicialización de la frecuencia de bus desde un cristal de 32,768KHz.

Curso de Microcontroladores Familia HC908 Flash de
Motorola , **Parte II**

```

; =====
;
; FlashGP.ASM      Programacion de memoria FLASH para 68HC908GP32
; =====      Por el Ing. Gabriel Dubatti (c) 2001
;                www.ingdubatti.com.ar
;
;                Se permite el uso y modificacion de este codigo
;                bajo su exclusiva responsabilidad.

; =====

; Las siguientes rutinas permiten grabar y borrar la memoria FLASH del
; GP32.
; Las mismas estan optimizadas para utilizar la menor cantidad posible
; de RAM:
;  rutina de grabacion o borrado: 35 bytes
;  stack de subrutina:           2 bytes
;  stack de llamada a la rutina: 2 bytes
;
;                =====
;  TOTAL de RAM necesaria:       39 bytes
; La optimizacion se realizo para una frecuencia de BUS de 8MHz pero
; pueden ajustarse las constantes para trabajar con otras frecuencias de
; clock si se respetan los delays indicados.
; Las funciones DEBEN llamarse con las INTERRUPCIONES TAPADAS por lo que
; pueden usarse normalmente estos 39 bytes de RAM como stack.
; =====

```

Curso de Microcontroladores Familia HC908 Flash de
Motorola , **Parte II**

ING. DANIEL DI LELLA DDFAE For Motorola Products

;Registros de control del PLL

PCTL EQU \$0036
PCTL_B_PLLON EQU 5
PCTL_B_BCS EQU 4
PBWC EQU \$0037
PBWC_B_AUTO EQU 7
PBWC_B_LOCK EQU 6
PMSH EQU \$0038
PMSL EQU \$0039
PMRS EQU \$003A
PMDS EQU \$003B

;Registros de control de la memoria FLASH

FLCR EQU \$FE08
FLCR_PGM EQU \$01
FLCR_ERASE EQU \$02
FLCR_MASS EQU \$04
FLCR_HVEN EQU \$08
FLBPR EQU \$FF7E

;COP

COPCTL EQU \$FFFF

```

;=====
    ORG      $0040                ;Comienzo de la RAM
FlashAddress:  DS  2                ;Address de grab. o borrado FLASH
FlashData:    DS  1                ;Dato a grabar en FLASH o temporal

flash_code:   DS  !35              ;Lugar donde se copian las rutinas
;de grabacion y borrado (se puede
;reutilizar como stack o temporales)

;=====
    ORG      $8000                ;Comienzo de la FLASH
PagDatos:     ;PAGINA de DATOS (para el ejemplo)
    ds      !128                  ;
NBYTES_WRITE EQU !4              ;Cuantos bytes grabar en la pagina
;antes de borrarla.

;=====
    ORG      $9000                ;Comienzo del codigo
CodigoStart:  ;(protegido p. Flash Block Protect)
;=====

```

```

;=====
; FlashWRITE:
; Graba 1 byte en memoria FLASH (GP32)
;
; REQUIERE:
; * BUS clock= 8MHZ (8 ciclos= 1 useg)
; * INTERRUPCIONES TAPADAS
; * FlashAddress= Address donde grabar el byte (2 bytes)
; * FlashData= Dato a grabar (1 byte)
;
; Descripcion:
; Carga en RAM el codigo de grabacion y lo ejecuta
; (en: flash_code)
; Requiere 35 bytes de RAM. (+4 de stack)
; Resetea el COP antes de grabar.
; Delay total = 809 ciclos = 101 microsegundos.
;=====

FlashWRITE:
    ldhx #{_flash_wr_fin - _flash_wr_com} ;numero de bytes a copiar,
                                           ; H=0

copy_write:                               ;copia funcion de GRABACION a RAM
    lda    {_flash_wr_com-1},x            ;
    sta    {flash_code-1},x              ;(-1 porque X= ultimo..1)
    dbnzz  copy_write                    ;
    lda    #FLCR_PGM                      ;A= Set PGM
exec_en_ram:                              ;
    sta    COPCTL                          ;resetea COP
    ldhx   FlashAddress                    ;carga address del bloque a
                                           ;grabar/borrar

    jmp    flash_code                     ;ejecuta codigo en RAM,sale por RTS

```

Curso de Microcontroladores Familia HC908 Flash de
Motorola , **Parte II**

ING. DANIEL DI LELLA DDFAE For Motorola Products

```

;-----
_flash_wr_com:                ;Comienzo del codigo de grabacion
                                ;(se ejecuta en RAM)
                                ;Al entrar: HX= address a GRABAR
                                ;          A= FLCR_PGM
    sta    FLCR                ;Flash Control Register= PGM
    lda    FLBPR               ;lee block protect
    sta    ,x                  ;selecciona el bloque a grabar

                                ;==== PGM to HVEN setup time ====
                                ;delay TNVS (10 useg.= 80 ciclos)
    lda    #{FLCR_HVEN+FLCR_PGM} ;2 ciclos (luego setea bit HVEN)
    ldx    #!24                ;2 ciclos
    bsr    DoDelayYStaFLCR     ;4+4 ciclos + 3*X= 80 ciclos
                                ;deja X=11 y A= FlashData

                                ;Program Hold Time
                                ;delay TPGS (5 useg.= 40 ciclos)

```

Curso de Microcontroladores Familia HC908 Flash de
Motorola , **Parte II**

```

delay_tpgs:
    dbnzx    delay_tpgs        ;9 + 3*X = 42 ciclos
                                ;GRABA BYTE (A= FlashData)

    ldhx    FlashAddress      ;(H:X= addrees a GRABAR)
    sta     ,x                ;

                                ;Byte Program Time (HI)
                                ;delay TPROG (30 useg.= 240 ciclos)

    ldx     #!78              ;2 ciclos
    lda     #FLCR_HVEN        ;2 ciclos (luego borra bit PGM)
    bsr     DoDelayYStaFLCR    ;4+4 ciclos + 3*X= 242 ciclos
                                ;delay extra= 9 ciclos
                                ;deja X=11
                                ;High-Voltage Hold Time
                                ;delay TNVH (5 useg.= 40 ciclos)

    clra                                ;1 ciclo (luego borra bit HVEN)
                                ;10 ciclos + 3*X= 43 ciclos

DoDelayYStaFLCR:
                                ;X= delay en ciclos = (X * 3)
    dbnzx    DoDelayYStaFLCR    ;3 ciclos
    sta     FLCR                ;A= nuevo valor FLCR
    ldx     #!11              ;2 ciclos
    lda     FlashData          ;3 ciclos
    rts                                ;delay extra= 5+4 = 9 ciclos
                                ;return to read mode
                                ;delay TRCV (1 useg.= 8 ciclos)

_flash_wr_fin:
                                ;== Fin del codigo de grabacion ==
;-----

```

Curso de Microcontroladores Familia HC908 Flash de
Motorola , **Parte II**

```

;=====
; FlashERASE:
; Borra una pagina de memoria FLASH (GP32, una pagina= 128 bytes)
;
; REQUIERE:
; * BUS clock= 8MHZ (8 ciclos= 1 useg)
; * INTERRUPCIONES TAPADAS
; * FlashAddress= address de un byte dentro del bloque a borrar
; * Usa "FlashData" de TEMPORAL (delay de borrado)
;
; Descripcion:
; Carga en RAM el codigo de borrado y lo ejecuta (en: flash_code)
; Requiere 35 bytes de RAM (+4 de stack)
; Reseta el COP antes de borrar.
; Delay total = 8545 ciclos = 1,068 milisegundos.
;=====

```

FlashERASE:

```

    ldhx #{_flash_er_fin - _flash_er_com} ;numero de bytes a copiar,
                                           ;H=0
copy_erase:                               ;copia funcion de borrado a RAM
    lda    {_flash_er_com-1},x            ;
    sta    {flash_code-1},x              ;(-1 porque X= ultimo..1)
    dbnzx  copy_erase                     ;
    mov    #!243,FlashData                ;setea delay de borrado
    lda    #FLCR_ERASE                    ;A= Set ERASE
    bra    exec_en_ram                    ;HX=FlashAddress, COP y jmp "flash_code"

```

----- Motorola , **Parte II**

```

_flash_er_com:                ;= Comienzo del codigo de borrado =
                                ;(se ejecuta en RAM)
                                ;Al entrar: HX= address en pagina a borrar
                                ;
                                ;          A= FLCR_ERASE
                                ;Flash Control Register= ERASE
    sta     FLCR                ;lee block protect
    lda     FLBPR
    sta     ,x                  ;selecciona el bloque a borrar

                                ;Erase to HVEN setup time
                                ;delay TNVS (10 useg.= 80 ciclos)
    lda     #!26                ;
delay_tnvs:                    ;
    dbnxx  delay_tnvs          ;2 + 3 x 26= 80 ciclos

    ldhx   #FLCR                ;
    lda     #{FLCR_HVEN+FLCR_ERASE} ;setea bit HVEN
    sta     ,x                  ;
                                ;Erase block time
                                ;delay TERASE (1 mseg.= 8000 ciclos)

```

```

delay_terase:
    bsr    delay_10_ciclos    ;4+10= 14 ciclos
    bsr    delay_10_ciclos    ;4+10= 14 ciclos
    dbnz  FlashData,delay_terase    ;+5 ciclos => 33x243= 8019 ciclos

    lda    #FLCR_HVEN        ;borra bit ERASE
    sta    ,x                ;
                                ;High-Voltage Hold Time
                                ;delay TNVH (5 useg.= 40 ciclos)

    lda    #!13              ;

delay_tnvh:
    dbnza delay_tnvh        ;2 + 3 x 13= 41 ciclos
                                ;"dbnza" deja A= 0

    sta    ,x                ;borra bit HVEN

                                ;return to read mode
                                ;delay TRCV (1 useg.= 8 ciclos)

delay_10_ciclos:
    nsa                                ;3 ciclos
    nsa                                ;3 ciclos
    rts                                ;4 ciclos

_flash_er_fin:
    ;== Fin del codigo de borrado ==
;-----

```

Curso de Microcontroladores Familia HC908 Flash de
Motorola , **Parte II**

```

;=====
; ; ; ; ; CODIGO DE PRUEBA DE LAS RUTINAS DE GRABACION y BORRADO ; ; ; ;
;=====
; RESET:
; 1) Setea frecuencia de BUS= 8MHz mediante el PLL con cristal de
; 32.768KHz.
; 2) Busca el ultimo valor distinto de $FF en la pagina de datos y
; graba en el byte siguiente $55 hasta completar los NBYTES_WRITE
; bytes.
; 3) Cuando se completan los NBYTES_WRITE bytes, BORRA la PAGINA y
; graba en el primer lugar.
; Las paginas son de 128 bytes en el GP32.
; 4) Entra en un loop infinito.
;=====

```

RESET:

;1)

```

    rsp                ;SP=$00FF
;setea PLL para generar 8 MHZ de frec.BUS desde cristal de 32K768
;R=1    N=3D1    P=0    E=2    L=D0
clr     PCTL                ;PLL:OFF
bset   1,PCTL                ;P= 0  E= 2
mov    #3,PMSH                ;N HI= 3
mov    #1,PMDS                ;R= 1 (default)
mov    #$D1,PMSL                ;N LOW= D1
mov    #$D0,PMRS                ;L= D0
bset   PCTL_B_PLLON,PCTL      ;PLL:ON
bset   PBWC_B_AUTO,PBWC       ;enganche automatico
wait_pll:                    ;esperar que LOCK sea = 1
brclr  PBWC_B_LOCK,PBWC,wait_pll;
bset   PCTL_B_BCS,PCTL        ;cambia a frecuencia del PLL

```

Curso de Microcontroladores Familia HC908 Flash de
Motorola , **Parte II**

ING. DANIEL DI LELLA DDFAE For Motorola Products

```

;2)
    ldhx    #PagDatos          ;busca ler $FF de la pagina
BuscaFF:
    lda     ,x                 ;
    cmp    #$FF               ;
    beq    EncontroFF        ;encontro un byte vacio, grabar $55
    aix    #1                 ;
    cphx   #{PagDatos+NBYTES_WRITE};se completaron NBYTES_WRITE bytes?
    bne    BuscaFF           ;NO, ver el siguiente byte

;3)
                                ;se completo el lugar, borrar pagina
    ldhx   #PagDatos          ;indica el ADDRESS a BORRAR
    sthx   FlashAddress       ;(cualquier address en la pag. sirve)
    jsr    FlashERASE         ;BORRA la pagina indicada
                                ;(todos sus bytes pasan a $FF)
    ldhx   #PagDatos          ;grabar en el 1er byte de la pagina

EncontroFF:
                                ;encontro un byte vacio, grabar $55
    sthx   FlashAddress       ;indica el ADDRESS a GRABAR
    mov    #$55,FlashData     ;indica el DATO a GRABAR
    jsr    FlashWRITE        ;GRABA el dato indicado

```

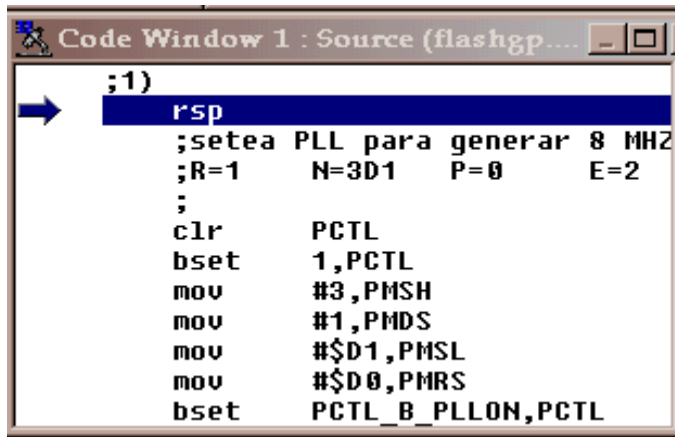
```

;4)
loop_final:                                ;espera el proximo reset para grabar
    bra    loop_final                      ;otro byte.
;=====
    ORG    FLBPR                            ;FLash Block PRotect
    DB    {(CodigoStart & $7FFF)/!128} ;donde empieza el CODIGO a
                                                ;PROTEGER
;=====
    ORG    $FFFE                            ;
    DW    RESET                            ;Vector de RESET
;=====

```

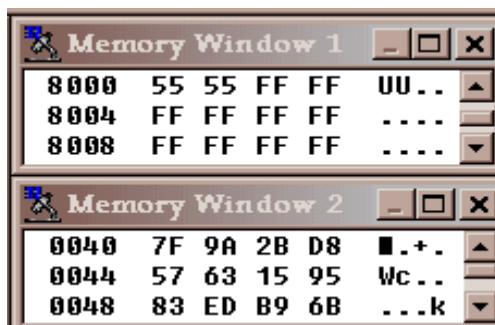
“Corriendo” el programa en los sistemas EVAL08GP / E-FLASH08.....

Las rutinas aquí presentadas se testearon satisfactoriamente en los sistemas de desarrollo para **HC08**, *EVAL08GP* y *E-FLASH08* , mostrandose las siguientes pantallas a modo de resumen.



```
Code Window 1 : Source (flashgp....)
;1)
rsp
;setea PLL para generar 8 MHZ
;R=1   N=3D1   P=0   E=2
;
clr    PCTL
bset   1,PCTL
mov    #3,PMSH
mov    #1,PMDS
mov    #$D1,PMSL
mov    #$D0,PMRS
bset   PCTL_B_PLLON,PCTL
```

Fig1. - Código fuente "Flasgp.asm" en una ventana del EVAL08GP / E-FLASH08



Memory Window 1				
8000	55	55	FF	FF UU..
8004	FF	FF	FF	FF
8008	FF	FF	FF	FF

Memory Window 2				
0040	7F	9A	2B	D8 ■.+. .
0044	57	63	15	95 Wc.. .
0048	83	ED	B9	6B ...k .

Fig. 2.- Ventanas de Memoria FLASH / RAM con valores \$55 grabados usando la rutina "Flashgp.asm" aquí presentada.

Curso de Microcontroladores Familia HC908 Flash de Motorola , **Parte II**