COMENTARIO TECNICO

Seguridad en la familia 68HC908

Por el Ing. Gabriel Dubatti
e-mail: info@ingdubatti.com.ar

Adaptación Ing. Daniel Di Lella / Electrocomponentes S.A.
e-mail: fae@electrocom.com.ar

Introducción

Los procesadores de la línea 68HC908 de Motorola™ cuentan con un mecanismo de protección del código y los datos grabados en la memoria FLASH mediante una clave de 8 bytes.
Esta clave se ingresa cada vez que se desea entrar en modo monitor (modo que permite ver y o modificar el contenido de la FLASH) luego de un "Power On Reset". Si la clave es correcta se permite ver, borrar y grabar la totalidad de la FLASH. Si alguno de los 8 bytes no coincide con los grabados en el procesador, se permite el ingreso a modo monitor con la salvedad que sólo puede realizarse un borrado total de la FLASH y si se intenta leer, se obtiene un valor fijo **($AD).**
Este artículo muestra como utilizar correctamente esta clave y discute que tan seguro es el método de protección.

¿Dónde se graba la clave?

La clave se graba en las 8 posiciones de FLASH: **FFF6 a FFFD**. Estas direcciones coinciden con los 4 vectores anteriores al RESET que según el modelo del procesador pueden estar implementados o no.

Por ejemplo:

- **68HC908GP32**
  - FFF6/7 : vector CH0 timer 1
  - FFF8/9 : vector PLL
  - FFFA/B : vector IRQ
  - FFFC/D : vector SWI
- **68HC908JL3**
  - FFF6/7 : vector CH0 timer
  - FFF8/9 : **LIBRE** (no tiene PLL)
  - FFFA/B : vector IRQ
  - FFFC/D : vector SWI

La idea de utilizar los vectores como clave surge de considerar que difícilmente dos programas coincidan en todos los vectores y por ende, no tener que destinar 8 bytes específicos para la protección. Esto tiene como efecto secundario que a medida que el programa se modifica, los vectores pueden cambiar de posición y por lo tanto, también la clave.

¿Cómo se graba la clave?

Para cada uno de los 4 vectores existen dos opciones:

- **Vector en uso:** Si utiliza el vector, el valor debe estar dentro del rango de la memoria FLASH y apuntar al código de la interrupción correspondiente (obviamente).
  Esto restringe el rango de valores posibles y permite a un atacante barrer un menor número de posibilidades para descubrir la clave correcta.

Lo mejor en este caso es no dejar todos los vectores en valores muy próximos, sino distribuirlos a lo largo de toda la FLASH. Esto puede realizarse mezclando las rutinas a lo largo del programa (si tiene la FLASH muy comprometida de espacio) o forzando posiciones "raras" mediante el comando "ORG" en lugares vacíos de la FLASH.

- **Vector libre:** Si tiene la precaución de deshabilitar la causa de interrupción o el procesador no la implementa, puede poner el valor que desee en los dos bytes correspondientes al vector con "DW $xxxx".
  Tenga presente que la **IRQ** por defecto está activa en el reset y debe deshabilitarse con el comando **mov #2,INTSCR**.
  NUNCA deje los vectores de la clave que no usa en $FF.

Algunos ejemplos:

1) **68HC908GP32** utilizando los vectores: RESET, TIMER1 CH0 y TIMER1 CH1:

```
    ORG $FFDC
    dw  VECTOR_VACIO    ;VECTOR_TIMEBASE
    dw  VECTOR_VACIO    ;VECTOR_ADC_COMPLETE
    dw  VECTOR_VACIO    ;VECTOR_KEYBOARD
    .......
    dw  VECTOR_TIM1_CH1 ;
;-------------------------- CLAVE --------------------------------
    dw  VECTOR_TIM1_CH0 ; <- valor fijado por el vector
    dw  $9D5B           ;VECTOR_PLL << no se usan:
    dw  $4722           ;VECTOR_IRQ <<  uso cualquier valor de clave
    dw  $1254           ;VECTOR_SWI <<
;-----------------------------------------------------------------
    dw  VECTOR_RESET
```

Por las dudas, VECTOR_VACIO apunta a una instrucción RTI.
Si VECTOR_TIM1_CH0= 8523, la clave será: **85 23 9D 5B 47 22 12 54**.

2) **68HC908GP32** utilizando los vectores: RESET, TIMER1 CH0, TIMER1 CH1, PLL, IRQ y SWI:

```
    ORG $FFDC
    dw  VECTOR_VACIO    ;VECTOR_TIMEBASE
    dw  VECTOR_VACIO    ;VECTOR_ADC_COMPLETE
    dw  VECTOR_VACIO    ;VECTOR_KEYBOARD
    .......
    dw  VECTOR_TIM1_CH1
;-------------------------- CLAVE --------------------------------
    dw  VECTOR_TIM1_CH0 ; <- valor fijado por el vector
    dw  VECTOR_PLL_SEG  ; <- se "desvía" el vector
    dw  VECTOR_IRQ_SEG  ; <- se "desvía" el vector
    dw  VECTOR_SWI      ; <- valor fijado por el vector
;-----------------------------------------------------------------
    dw  VECTOR_RESET

;Opción 1:
;--- se desvían los vectores a lugares no utilizados de memoria ---
    ORG $EDC3           ;usar lugares libres de memoria!!
VECTOR_IRQ_SEGUR:
    jmp    VECTOR_IRQ   ;salto al verdadero vector

    ORG $F17A           ;usar lugares libres de memoria!!
VECTOR_PLL_SEGUR:
```

```
    jmp    VECTOR_PLL   ;salto al verdadero vector

;Opción 2:
;--- se mueve el código directamente ---
    ORG $F729           ;usar lugares libres de memoria!!
VECTOR_SWI:
    ....
    RTI
```

Si VECTOR_TIM1_CH0= 8015, la clave será: **80 15 ED C3 F1 7A F7 29**.

3) **68HC908JL3** utilizando los vectores: RESET, TIMER CH1, IRQ:

```
    ORG $FFDE
    dw  VECTOR_VACIO    ;VECTOR_ADC_COMPLETE
    dw  VECTOR_VACIO    ;VECTOR_KEYBOARD
    .......
    dw  VECTOR_TIM_CH1  ;USO CH1 y dejo CH0 libre para la clave
;--------------------------- CLAVE --------------------------------
    dw  $D17F           ;no se usa (elijo valor)
    dw  $2673           ;NO TIENE PLL (elijo valor)
    dw  VECTOR_IRQ      ; <- valor fijado por el vector
    dw  $1F9B           ;no se usa (elijo valor)
;------------------------------------------------------------------
    dw  VECTOR_RESET
```

En este caso elijo usar el canal 1 del timer para dejar libre el vector del canal 0.
Si VECTOR_IRQ= ED5F, la clave será: **D1 7F 26 73 ED 5F 1F 9B**.

¿Qué tan seguro es el método de protección?

La seguridad del método radica en el tiempo necesario para probar distintas combinaciones hasta dar con la correcta.
La única forma de probar una clave es luego de un "Power On Reset" (POR), esto es, hay que quitarle la alimentación y esperar un tiempo antes de volver a conectarla y probar otra clave.
También el envío de la clave demora un tiempo ya que debe enviarse en forma serie (vea la salvedad en el modo paralelo del GP32) que a 9600 equivale a 8.33 milisegundos (forzando baudrates más altos puede reducirse a una cuarta parte).
La cantidad de combinaciones posibles se obtiene multiplicando la cantidad de combinaciones de cada uno de los 4 vectores, las cuales dependen de si se utilizan o puede usarse cualquier valor, dado que en el primer caso la cantidad de combinaciones es igual al tamaño en bytes de la FLASH y en el segundo es 65536 (todos los valores posibles con 2 bytes). Esta diferencia es más importante en los procesadores con muy poca FLASH.

NCombTotal= NCombV1 * NCombV2 * CombV3 * CombV4

Ejemplos:

1. En el mejor caso todos los vectores están libres: NCombTotal= 65536^4= **$1.8x10^{19}$**
2. Peor caso del **68HC908GP32**: NCombTotal= 32256^4= **$1.1x10^{18}$**
3. Peor caso del **68HC908JK1**: NCombTotal= $65536*1536^3$=**$2.4x10^{14}$**
   (ya que al no tener PLL uno de los vectores siempre está libre)

Lo cual traducido a tiempo da (considerando que cada clave se puede probar en 1 milisegundo y en la mitad de las pruebas encuentran el valor correcto):

1. Mejor caso: **292 millones de años**.
2. Peor caso del **68HC908GP32**: **17 millones de años**.
3. Peor caso del **68HC908JK1**: **3765 años**.

Este análisis es válido dado que no se conoce el resultado hasta no haber ingresado los 8 bytes. Si se pudiera saber si cada byte es correcto o no sin esperar al final, bastaría con probar 128 x 8 veces = 1024 pruebas y obtener la clave correcta en 1 segundo!.
A continuación se muestra como está implementado el algoritmo de verificación de la clave en la ROM Monitor del 68HC908GP32.

Extracto de la ROM MONITOR del 68HC908GP32:

```
;=====================================================================
                                ;== inicializa / lee modo de lectura ==
FF1C    MOV     #FF,$40         ;[$40].6=1  ==> por ahora clave correcta
                                ;[$40].7=1  ==> leer bytes PARALELO (PORTA)
FF20    LDHX    #FFF6           ;primer byte de la clave
FF23    BRSET   7,PORTA,FF28    ;(PA.7 == 0)? ==> leer SERIE
FF26    BCLR    7,$40           ;[$40].7=0  ==> leer bytes SERIE (PA.0)

                                ;== loop para clave ok hasta el momento ==
FF28    CBEQX   #FE,FF45        ;leyó los 8 bytes?
FF2B    LDA     PORTA           ;lee PORTA (por si es modo paralelo)
FF2D    BRSET   7,$40,FF33      ;modo paralelo? => ya leyó el byte
FF30    JSR     FE97            ;modo serie => lee un byte desde PA.0
FF33    NOP                     ;ajusta delay "NOP+CBEQ X+"= 1+4= 5 ciclos
FF34    CBEQ    X+,FF28         ;compara con el vector, igual => ver próximo
                                ;distinto => error, seguir leyendo

                                ;== loop para clave incorrecta ==
FF36    CBEQX   #FE,FF4B        ;repite lectura paralelo o serie del
FF39    LDA     PORTA           ;bloque anterior pero no compara porque ya
FF3B    BRSET   7,$40,FF41      ;sabe que hubo error en la clave.
FF3E    JSR     FE97            ;
FF41    AIX     #1              ;delay "AIX+BRA"= 2+3= 5 ciclos
FF43    BRA     FF36            ;

FF45    LDX     #FF             ;=== CLAVE CORRECTA ===   [$40].6=1
FF47    RSP                     ;
FF48    JMP     FE20            ;entrada que permite leer la FLASH

FF4B    BCLR    6,$40           ;=== CLAVE INCORRECTA === [$40].6=0
FF4D    LDX     #FF             ;
FF4F    RSP                     ;
FF50    JMP     FE21            ;entrada que NO PERMITE leer la FLASH
;=====================================================================
```

En la dirección **FF33** se puede ver la instrucción **NOP** insertada para que se produzca el mismo delay ante clave correcta e incorrecta en ambos lazos, no permitiendo inferir desde fuera el resultado del testeo. Sin esta instrucción podría medirse el tiempo desde los bytes enviados y el BREAK que genera el procesador al finalizar la verificación y obtener la clave en pocos segundos.

Ingreso de la clave en "paralelo" en el 68HC908GP32

En el listado de la ROM puede verse que el modo paralelo permite acelerar el ingreso de la clave vacía (todos $FF) poniendo 8 resistores de "pull up" en el PORTA. Utilizar este modo para ingresar otra clave es muy complicado (aunque posible) dado que en la dirección **FF23** se lee PA7 para ver si es serie o paralelo y en **FF2B** se lee el primer valor desde el PORT, no existiendo una referencia de tiempo desde el exterior que permita saber cuando cambiar el valor del PORTA. Una forma posible es mediante un circuito de reset muy preciso, sin usar el PLL y determinando el delay de todas las instrucciones comprendidas entre cada lectura.

Una vez finalizada la clave, el procesador envía un BREAK y desde allí en más todos los comandos deben enviarse en forma serie.

Conclusión

El esquema de protección implementado en la familia 68HC908 es **"prácticamente" muy seguro**. Los ejemplos mostraron que los tiempos obtenidos para acertar la clave son muy elevados si se toma la precaución de generar valores realmente "impredecibles", ya que cuanto más información posea el atacante, menor será la cantidad de combinaciones que deba probar y por lo tanto más probable que acierte la clave correcta.