Order this document
by EB294/D

Motorola Semiconductor Engineering Bulletin

EB294

How to Write to the 64-Cycle Time-Protected Registers on M68HC11 Development Tools

By Brian Scott Crow
Austin, Texas

Introduction

The MC68HC11 Family was designed with flexibility in mind. To achieve maximum flexibility, software can control a number of hardware options which customize the operating environment. Since software can control the operating environment, it is necessary to take precautions against run away software and its ability to change the hardware configuration unintentionally. The result is a set of registers known as the time-protected registers.

These registers are writable in normal operating modes (single-chip and expanded) only during the first 64 clock cycles after reset. This protection is disabled while operating in a special mode (special bootstrap or special test).

Time-Protected Registers

New registers have been added to the register block with each new member of the MC68HC11 Family. Table 1 shows the first three registers that are time protected on all MC68HC11s. The x in the upper

MOTOROLA

nibble of the register address shows that this nibble can be configured via the lower nibble of the INIT register. The x is a 1 out of reset, but may change if the INIT register lower nibble is changed from $1 to another value.

Table 1. Original Time-Protected Registers

Register address	Register name	Function
$x03D	INIT	RAM and I/O register remapping
$x024	TMSK2	Timer prescalar and interrupts
$x039	OPTION	System configuration options

Every bit in the INIT register is time protected, since all eight bits control the addresses of internal memory resources (RAM addresses and register block addresses).

Only bits 1 and 0 are time protected in the TMSK2 register. These two bits, PR1:0, select one of four divider chains for the free-running, 16-bit timer which provides the clock to the real-time interrupt, output compares, and input captures.

Bits 5, 4, 1, and 0 of the OPTION register also are time protected. Bit 5 controls whether IRQ is edge-sensitive or level-sensitive. Bit 4 controls whether a delay is imposed coming out of stop mode to allow the oscillator to stabilize. Bits 1 and 0 are CR1:0, which select the divider chain for the COP watchdog timer.

When the family expanded to include the MC68HC11E9 and its other E-series derivatives, an EEPROM block protection register was added to the register block at address $1035. Bits 0 through 4 of this register are implemented. Bits 0 through 3 protect the user EEPROM and bit 4 protects the CONFIG register from programming or erasure. If a bit is a logic 1, the protected area cannot be programmed or erased, and if it is a logic 0, the area can be modified.

This register can only have bits cleared during the first 64 clock cycles, but bits may be set at any time to protect an EEPROM area. **Table 2** provides descriptive information about the BPROT register.

Table 2. BPROT Register

Register Address	Register name	Function
$x035	BPROT	Protect EEPROM blocks

Newer MC68HC11 Family members have outgrown the current register set and include many new registers. Of these, the INIT2 and the OPT2 have some write protection built in.

Table 3. INIT2 and OPT2 Registers

Register Address	Register name	Function
$x035	INIT2	EEPROM mapping
$x038	OPT2	System configuration options 2

The INIT2 register can only be written once and then becomes read only. The OPT2 register bit 4 can only be written once and then becomes readable only. These registers are not 64 cycle time protected like the others; however, they have similar protection interlocks which should be addressed at initialization.

For detailed description of each register for a particular device in the MC68HC11 family, refer to the technical data book for that particular device.

How to Modify Time-Protected Registers on the M68HC11EVBU

This board is shipped with a MC68HC11E9FN1 ROM device as the resident processor. This ROM device contains the BUFFALO monitor. In single-chip or expanded multiplexed modes of operation, BUFFALO comes up out of reset and sends a sign-on message to the terminal.

However, BUFFALO eats up all of the first 64 cycles before control is passed to the user. So how can a user modify the 64-cycle, time-protected registers while under the control of the BUFFALO monitor?

BUFFALO provides a hook to execute user code within the first 64 cycles out of reset. The EVBU board uses jumper J2 to implement this feature. BUFFALO tests the digital state of input pin port E bit 0 immediately out of reset. If this pin is grounded, BUFFALO continues executing normally which uses up the first 64 cycles. If BUFFALO detects a logic 1 on this pin, then it jumps to address $B600 in the internal EEPROM. The user can program his code to modify the time-protected registers and jump back to BUFFALO or continue executing user code.

A sample code segment could look like this:

```
regbas         equu   $1000
```

NOTE: *The address $E019 to return to BUFFALO is valid for BUFFALO version 3.2 only. This address is subject to change on different versions of BUFFALO without notice. The stack pointer is not initialized in this code segment because it is initialized when execution returns to BUFFALO. If you continue on with your application software, remember to initialize the stack pointer.*

Assemble this code and program it into the M68HC11E9 EEPROM at address $B600. Place jumper J2 on pins 1 and 2. Press the reset button. The BUFFALO sign-on message should appear on your terminal; however, this time, you have modified the time-protected registers before BUFFALO uses up the first 64 cycles.

How to Modify Time-Protected Registers on the M68HC11EVB

This process is nearly identical to that of the M68HC11EVBU. This board is shipped with an MC68HC11A1FN as the resident processor, and the BUFFALO monitor is located in an external EPROM in socket U3. The same EEPROM execution hook is in this version of BUFFALO.

Use this code segment for the EVB board, since it does not contain the BPROT register.

```
regbas         equ     $1000
tmsk2          equ     $0024
option         equ     $0039
init           equ     $003D

               org     $B600
start          ldx     #regbas          ;get the register base address in X
               lda     #$03
               staa    tmsk2,x          ;change timer prescale to /16
               ldaa    #$93
               staa    option,x         ;IRQ is level-sensitive, delay out
                                        ;of stop mode, and COP prescale to /64
               ldaa    #$01
               staa    init,x           ;leave RAM at $0000 and registers at $1000
               jmp     $E014            ;go back to BUFFALO at the point after it
                                        ;has written the time registers
```

NOTE: *The address $E014 to return to BUFFALO is valid for BUFFALO version 2.5 only. This address is subject to change on different versions of BUFFALO without notice. The stack pointer is not initialized in this code because it is initialized when execution returns to BUFFALO. If you continue with your application software, initialize the stack pointer.*

As you can see, the process is similar. Assemble this code and program it into the EEPROM of the 68HC11A1 at $B600. Place jumper J4 on pins 2 and 3. Press reset. The BUFFALO sign on message should appear on your terminal; however, this time, you have modified the time-protected registers before BUFFALO uses up the first 64 cycles.

Modifying the Time-Protected Registers on the M68HC11EVM, M68HC11EVS

These two boards operate similarly with respect to time-protected operations. These boards operate using EVMBug and EVSBug respectively for monitor programs and also operate independent of the choice of resident processor. (An EVM may have an A1, A0, E1, E0, etc.)

These boards have a user reset and abort button in common. The user reset switch may be used to begin executing user code. Because pressing this switch really resets the resident processor, time protection is restarted, and user code will begin executing immediately following the fetch of the user reset vector. The result is better emulation of target system software behavior, because the user code executed out of reset will have only 64 cycles to modify the time-protected registers. A sample code segment follows.

NOTE: *That this code writes to registers which may not be implemented on some versions of the MC68HC11. These writes to unimplemented registers will not affect operation, but are unnecessary if your device does not contain these registers.*

```
regbas         equ     $1000          ;may be $0000 on some devices
tmsk2          equ     $0024
bprot          equ     $0035
option         equ     $0039
opt2           equ     $0038
init           equ     $003D
init2          equ     $0037

               org     $FFFE          ;address of reset vector
               fdb     start          ;insert address of code beginning

               org     $E000          ;start of user code
```

```
start          lds     #$00FF         ;always initialize the stack pointer
               ldx     #regbas        ;get the register base address in X
               ldaa    #$03
               staa    tmsk2,x        ;change timer prescale to /16
               ldaa    #$00
               staa    bprot,x        ;clear the Block Protect register
               ldaa    #$93
               staa    option,x       ;IRQ is level-sensitive, delay out
                                      ;of stop mode, and COP prescale to /64
               ldaa    #$01           ;may be $00 on some devices
               staa    init,x         ;leave RAM at $0000 and registers at $1000
                                      ;some devices may leave RAM at $0000 and
                                      ;registers at $0000
               ldaa    #$00
               staa    init2,x        ;leave EEPROM at $0D80 - $0FFF
               ldaa    #$00
               staa    opt2,x         ;IRV is disabled

                                      ;continue with rest of user code or insert
                                      ;a dummy infinite loop to wait until user
                                      ;presses the abort switch like below

loop           bra     loop
```

As you may notice, this operation is slightly different than that of the other boards. To execute this code, press the user reset switch. The newly reset processor will fetch the user reset vector from the user map at address $FFFE and begin execution at the address pointed to by the reset vector.

The user's code must write to the protected registers within the first 64 clock cycles. Then the user may insert an infinite loop or get on with the rest of his application. If an infinite loop is inserted, press user reset to begin execution, and press the abort switch to return control to the monitor.

At this point, you may continue de-bugging with the monitor and the protected registers will have taken on your values. Remember that any future reset will change the registers to their initial state, and this process may need to be employed again.

This technique is not valid when the above code segment is started with the GO command from the monitor. In this case, the processor is not truly reset, so time protection is still enforced, making any writes to protected bits useless.

Conclusion

Writing to the time-protected bits is not difficult on Motorola development systems. The code segments presented here should demonstrate the concepts for each M68HC11 development tool. Remember to check for device-specific information in the technical data book for your specific target processor to see which registers are implemented and which are time protected or writable once only.

Attention to detail will help solve most problems, and writing to the time-protected registers is no exception. Experiment to see how you can use these tricks in your own development environment.

MOTOROLA