SECTION 8: SOFTWARE CONTROL

Introduction

Several features have been incorporated into the Secure Microcontroller to help insure the orderly execution of the application software in the face of harsh electrical environments. Any microcontroller which is operating in a particularly noisy environment is susceptible to loss of software control. Electrical transients such as a glitch on the clock or a noise spike on an I/O pin can cause software problems like the loss of key variables in internal registers and/or execution of code out of its logical sequence. Such transients can send the microcontroller into an indefinite period of seemingly random software execution.

Timed Access, Watchdog Timer and CRC hardware features have been built in to help provide control and recovery under difficult operating conditions. The operation of these features is described below.

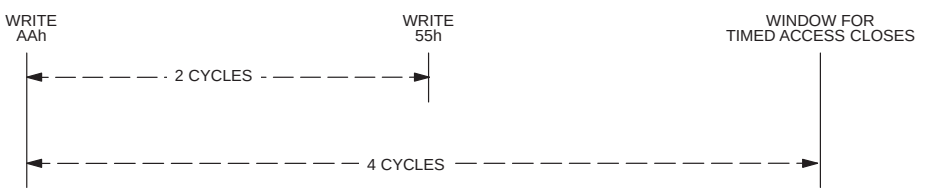
Timed Access

The Timed Access feature is provided to help insure controlled access by software to critical configuration bits in the Special Function registers. These protected bits may only be written through the execution of a specific multiple instruction software sequence which involves the Timed Access register. This restriction is designed to help prevent a potentially catastrophic change in the configuration by an inadvertent write during times when software control has been lost.

In order to modify the protected bits listed in Table 8–1, a pattern of two bytes must first be written to the Timed Access register at location 0C7h. The first write should be a value of 0AAh and the second should be a value of 55H. After this sequence is performed, the protected bits may be modified. Upon receiving a 0AAH in the Timed Access register, two timers are initiated. The first timer allows two instruction cycles to write a 55H. This means a one– or two–cycle instruction may be used. If 55H is not written within two cycles, Timed Access is reset. The second timer requires that the protected bit be modified within four instruction cycles. Since this timer started prior to writing 55H, the remaining time depends on which type of instruction was used to write 55H. If a one–cycle instruction was used to write 55H, then three cycles remain to modify protected bits. In the same way, if a two–cycle instruction was used to write 55H, then two cycles remain. This is depicted in Figure 8–1. The following code sequences demonstrate this procedure.

In the rare case that back to back Timed Accesses are performed, the user must be aware that the four–cycle Timed Access window must close before another Timed Access can begin. This is only an issue if a one–cycle instruction is performed after the MOV TA, #55h instruction, leaving one cycle remaining in the four–cycle count. The user can eliminate this problem by either using a two–cycle instruction after the MOV TA, #55h instruction, or by inserting one other instruction between the two Timed Access procedures. Violation this rule will result in a failure of the second Timed Access procedure, leaving the bit(s) unmodified.

TIMED ACCESS Figure 8–1



This code allows the reset of the Watchdog Timer:

```
MOV     0C7H,#0AAH      ; 1st TA Value
MOV     0C7H,#055H      ; 2nd TA Value              2 Cycles
SETB    IP.7            ; Reset Watchdog Timer      1 Cycle
```

The Watchdog Timer bit may have been set using ORL IP, #80H which takes two cycles.

This code allows the reset of the Watchdog Timer using a different approach:

```
MOV     A, #55H         ; Setup Acc for fast write
MOV     0C7H, #0AAH     ; 1st TA Value
MOV     0C7H, A         ; 2nd TA Value              1 Cycle
MOV     A, IP           ; Get Current IP            1 Cycle
ORL     A, #80H         ; Prepare for fast write    1 Cycle
MOV     IP, A           ; Reset Watchdog Timer      1 Cycle
```

Note that a new value for IP could have been retrieved from any direct register instead of the current IP.

The bits which are write access–protected by the Timed Access function are listed in Table 8–1.

TIMED ACCESS PROTECTED CONTROL BITS Table 8–1

BIT NAME	MICRO VERSION	LOCATION	DESCRIPTION
EWT	All Secure Micro	PCON.2	Enables the Watchdog Timer Reset function
RWT	All Secure Micro	IP.7	Resets the Watchdog Timer count
STOP	All Secure Micro	PCON.1	Stop Mode Enable
POR	All Secure Micro	PCON.6	Power On Reset
PAA	DS5000 series	MCON.1	Partition Address Access bit (protects PA3–0)
PA3–0	DS5001, DS5002 series	MCON.7–4	Partition Address bits
AE	DS5001, DS5002 series	RPCTL.4	Access Enable

The Secure Microcontroller family has a variety of control bits that are critical to the correct operation of the processor. Several of these are nonvolatile and will not be altered by a reset. Thus they must be protected from an accidental write by software that has gone out of control. This is a possibility in all microprocessor based systems, especially those in an industrial environment. While the Watchdog Timer will recover from this condition, the critical bits must be protected during the interval before the time–out of the Watchdog Timer.

The Secure Microcontroller family actually has two levels of protection for these critical bits. The most critical SFR bits can only be altered using the Bootstrap Loader. An example is the Range function that determines the total memory. There is no need for an application to modify this bit during normal operation. For those critical bits that might need to be modified during normal operation, the Timed Access procedure protects against an inadvertent write operation.

Timed Access provides a statistical protection. It is unlikely that randomly generated states will correctly match the sequence and timing required to bypass the Timed Access logic. Presented below is a brief justification for each bit that is protected by Timed Access.

The EWT bit is protected to prevent errant software from disabling the Watchdog Timer. The Watchdog is one of the important mechanisms that assure correct operation and should not be turned off accidentally. RWT is the bit that software uses to restart the Watchdog time–out. The Secure Microcontroller makes this more difficult by Timed Access protecting the bit. Thus software must "really" intend to reset the time–out in order to do so. Note that the Watchdog Timer is disabled in Stop mode. Critical applications which rely on the Watchdog Timer should exercise caution if the application will utilize Stop mode.

POR informs the software of the power supply condition. Specifically, it means the power has previously dropped below the V_{CCMIN} level and returned to normal. In many systems, this is a unique condition that requires interaction with external hardware. Protecting this bit with a Timed Access procedure prevents the micro from accidentally performing a power on reset procedure.

On a DS5000 series device, the PAA bit allows software to alter the Partition. If this is done accidentally, the resulting configuration could be unrecoverable without human intervention. This could mean selecting a Partition that is outside of the user's plan and that causes the system to fail. In a like manner, the PA3–0 bits on a DS5001 series device are protected through Timed Access. As the DS5001 does not have a PAA bit, the Partition control bits are directly protected. The motivation for protecting the AE bit is similar. This bit invokes a Partitionable configuration where one had not been selected during Bootstrap loading. While there are several valid reasons to select AE, accidentally selecting this condition might be unrecoverable without manual intervention.

Note that the Timed Access logic protects against the possibility of a single inadvertent write modifying a critical control bit. It does not protect against inadvertently entering a section of code that contains the correct sequence to modify a protected bit. However, the statistical protection does greatly improve the system's resilience to a crash.

Watchdog Timer

The on–chip Watchdog Timer provides a method of restoring proper operation during transients that cause the loss of controlled execution of software. When the Watchdog Timer is enabled, it will eventually reach a timeout condition after 122,800 machine cycles unless it is reset by the application software. An internal reset to the CPU will be generated if the timeout condition is ever reached. Software which utilizes the Watchdog Timer must periodically reset the RWT bit so that it will never be reached during normal operation. The reset operation(s) should be inserted at critical check points in the program. The Watchdog Timer will monitor program execution to insure that these check points are reached, indicating proper operation. If controlled execution of the software is lost so that these check points are not encountered within the timeout period, then the Watchdog Timer will provide an automatic reset. A block diagram of the Watchdog Timer is shown in Figure 8–2.

The Special Function Register bits that are used to control the Watchdog include the Enable Watchdog Timer bit (EWT; PCON.2), the Reset Watchdog Timer bit (RWT; IP.7), and the Watchdog Timer Reset status flag (WTR; PCON.4). The Watchdog Timer incorporates a free–running counter that starts counting as soon as the clock oscillator begins operation following a Power On Reset. If a 12 MHz crystal is used as the time base element, this gives a timeout period of 122.88 ms. The Watchdog Timer Reset function is enabled with a Timed Access write operation which sets the EWT bit to a 1. A Watchdog Timer Reset will then occur the next time that the free–running counter reaches its timeout condition.

Regardless of whether the Watchdog Timer will be used, it should be initialized after each reset. If the Watchdog Timer is desired, then the first step is to reset the timer count. This is necessary since the timer is free running and may be about to time–out. Set the RWT bit to a logic 1 using a Timed Access procedure. This will restart the timer with the full interval. Then enable the Watchdog Timer reset function by setting the EWT bit to a logic 1, again with a Timed Access procedure. Note that the EWT bit only controls whether the reset is issued, not whether the timer runs. The Watchdog Timer must now be reset prior to 122,800 machine cycles or it will reset the CPU. If the Watchdog Timer is not used, then clear the EWT bit to a logic 0 using a Timed Access procedure. Since the EWT bit is nonvolatile, this makes certain that the Watchdog reset function remains disabled.

During subsequent program execution, the Watchdog Timer can be reset by a Timed Access write operation which sets the RWT bit to a 1. This will cause the Watchdog Timer to begin counting machine cycles again from an initial count of 0. The RWT bit itself is automatically cleared immediately after the Watchdog Timer is reset. An instruction sequence which performs this operation is as follows.

This code allows the reset of the Watchdog Timer:

```
MOV     0C7H, #0AAH   ; 1st TA Value
MOV     0C7H, #055H   ; 2nd TA Value
SETB    IP.7          ; Reset Watchdog Timer
```

If the timeout period is ever reached without the timer being reset by the software, the Watchdog Timer will reset the CPU, set the WTR status flag, and will begin counting again. The WTR flag allows the application software to distinguish this type of reset from other possible sources so that special processing can be performed to accommodate this case. This flag will be set in response to a timeout, regardless of whether the reset is enabled. The WTR bit is cleared only by a read of the PCON register. Therefore, this register should be read during initialization following a reset in order to properly interpret the source of the reset.

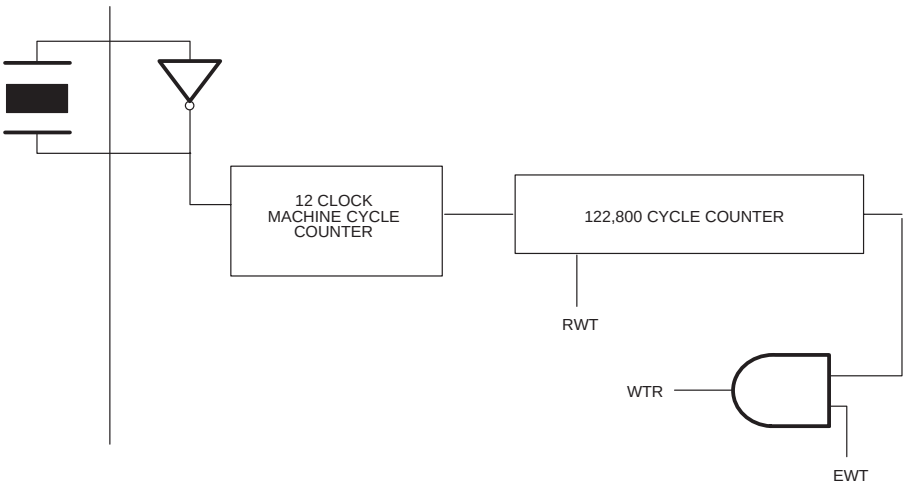
The Watchdog Timer Reset Bit (WTR) is held in a logic 1 state for 8192 clock cycles following the time–out of the watchdog 122,880 cycle counter. During this time, the bit may be read but attempts to clear the bit will fail. This condition will not be noticed if the Enable Watchdog Timer bit (EWT) is set, because the 8192 cycle count will be reset during the device reset triggered by the watchdog time–out. The bit may then be cleared, if desired, during application's power–on reset routine.

Some applications may use the watchdog timer but not set the EWT bit, preferring instead to poll the WTR bit in software to detect a watchdog time–out. In this case, one approach is for the application software to continually read the EWT bit as long as it is set. When the 8192 clock cycle period is complete, the last read of the EWT bit will successfully clear the bit and exit the routine. Alternatively, software can poll the WTR bit until it is set, then reset the watchdog via the RWT bit to clear the 8192 cycle count. The next read of the PCON register will clear WTR bit as expected.

The Watchdog Timer is also reset whenever any other type of reset is issued to the CPU and will begin its count as soon as the reset condition is released and the application software begins execution.

If operation without the Watchdog Timer is desired, then the EWT bit should be cleared following any type of reset by using the Timed Access register. This will insure that the Watchdog Timer will never cause an undesired reset during execution of the application software.

WATCHDOG TIMER Figure 8–2



WATCHDOG TIMER CONTROL BITS

Bit Description:

PCON.4:	**WTR**
"Watchdog Timer Reset"	Set to a 1 when a Watchdog Timer timeout occurs. If Watchdog Timer Reset is enabled, this will indicate the cause of the reset. Cleared to 0 immediately following a read of the PCON register.
Initialization:	Set to a 1 after a Watchdog Timeout. Cleared to a 0 on a No–V_{LI} Power On Reset. Remains unchanged during other types of resets.
Read Access:	May be read normally anytime.
Write Access:	Cannot be written.
PCON.2:	**EWT**
"Enable Watchdog Timer Reset":	Used to enable or disable the Watchdog Timeout Reset. The Reset is enabled if EWT is set to a 1 and will be disabled if EWT is cleared to a 0. This bit affects the generation of a reset condition, not the running of the Watchdog Timer.
Initialization:	Cleared to a 0 on a No–V_{LI} Power On Reset. Remains unchanged during other types of resets.
Read Access:	May be read normally anytime.
Write Access:	Can be written only by using the Timed Access register.
IP.7:	**RWT**
"Reset Watchdog Timer":	When set to a 1, the Watchdog Timer count will be reset, and counting will begin again. The RWT bit will then automatically be cleared again to 0. Writing a 0 into this bit has no effect. This bit should be set prior to EWT, as the timers are free–running.
Initialization:	Cleared to a 0 on any reset.
Read Access:	Cannot be read.
Write Access:	Can be written only by using the Timed Access register.

CRC MEMORY VERIFICATION

When using nonvolatile memory, there is always the potential for a catastrophic event to alter the memory contents. These events include lightning, massive ESD, severe mistreatment, etc. No nonvolatile technology is immune to these events. To compensate, the DS5001 series contains a CRC function that allows for automatic verification of memory on power up. The CRC function is also available to the user for application software use. Note that this is not available on DS5000 series devices [DS5000(T), DS2250T, DS5000FP].

If the CRC option is selected through the Bootstrap Loader, then on power up or after a Watchdog Timer reset, the microcontroller will automatically perform a CRC–16 on the memory. The range over which it is performed is selected by the user, and the result is compared to a pre–stored value. If the CRC–16 is in error, the DS5001 series microcontroller will enter the Bootstrap Loader and wait. From the perspective of the system, the appears held in a reset condition.

To support this function, the CRC register shown below is accessible through the Bootstrap Loader. Setting the CRC bit (LSB) enables the power–up CRC function. The loader command "W" is used to write to this register. The upper nibble of the CRC register (a hex value between 0 and F) defines the address space in 4K

blocks over which the CRC calculation is performed. For example, if the nibble is set to 0001b, the CRC range is from 0000 to 0FFFh. Once the LSB of the CRC register is set, the loader "I" command will cause the CRC of the specified block to be computed. The result is automatically stored in the last two bytes of the specified block. These bytes should not be used by the application. This computation will be correct provided that the CRC range is less than or equal to the partition if PM=0. If PM=1, using 32K RAMs, the CRC range must be less than or equal to the program range.

If CRC is enabled, the DS5001FP will automatically invoke the Bootstrap Loader on either power–up or a Watchdog timeout and the CRC check will be performed. If an error is detected, the Bootstrap Loader will wait for reloading. If there is no error, the application will begin at address 0000h following a reset. Automatic checking of the CRC can be disabled by writing a 0 to the CRC register LSB. As mentioned above, this is done using the "W" command in loader mode. The CRC hardware uses registers 0C3h and 0C2h for most and least significant byte intermediate storage. The DS5002FP and DS2252T do not perform a CRC check to ensure software security.

DS5001 CRC REGISTER (Address 0C1h)

RNGE3	RNGE2	RNGE1	RNGE0	—	—	MDM	CRC

CRC.7–4: **RANGE 3–0**
Determines the range over which a power–up CRC will be performed. Addresses are specified on 4K boundaries.

Initialization: Reset to 0 on a No V_{LI} reset.

Read Access: Can be read at any time.

Write Access: Cannot be written by application software. Can be written via the Bootstrap Loader.

CRC.1: **MDM**
When set to 1, the Bootstrap Loader will attempt to use a modem (UART) on $\overline{PE4}$ if CRC is incorrect. This feature is no longer useful following the obsoletion of the corresponding modem devices.

Initialization: Reset to 0 on a No V_{LI} reset.

Read Access: Can be read at any time.

Write Access: Cannot be written by application software. Can be written via the Bootstrap Loader.

CRC.0: **CRC**
When set to 1, a CRC check will be performed on power–up or watchdog timeout. CRC will be checked against stored values. An error will initiate Program Load mode. This bit will not be present in the DS5002 as the device does not support the power–on CRC function.

Initialization: Reset to 0 on a No V_{LI} reset.

Read Access: Can be read at any time.

Write Access: Cannot be written by application software. Can be written via the Bootstrap Loader.

CRC CODE EXAMPLE Figure 8–3

This routine tests the CRC–16 circuit in the DS5001FP

```
crcmsb      equ     0C3h
crclsb      equ     0C2h

            org     00h             ;after reset, CRC regs = 0000
begin:
            mov     p2,crcmsb       ;p2=00 read crcmsb register
            mov     p3,crclsb       ;p3=00 read crclsb register
            mov     crclsb, #075h   ;check crc register operation
                                    ;data in = 75 result = E7C1
            mov     crclsb, #08Ah   ;data in = 8A result = 37A7
            mov     crclsb, #00Bh   ;data in = 0B result = 7D37
            mov     crclsb, #075h   ;data in = 75 result = 31FD
            mov     crclsb, #0C7h   ;data in = C7 result = 13B1
            mov     crclsb, #0AAh   ;data in = AA result = 0B53
            mov     crclsb, #075h   ;data in = 75 result = DA8A
            mov     crclsb, #0C7h   ;data in = C7 result = 351A
            mov     crclsb, #055h   ;data in = 55 result = F474
            mov     crclsb, #043h   ;data in = 43 result = D6B5

            nop                     ;delay after last write and before first read
                                    ;let CRC finish
            mov     p0 ,crcmsb      ;p0=D6 read CRCMSB register
            mov     p1 ,crclsb      ;p1=B5 read CRCLSB register
            mov     crclsb  ,crclsb ;clear CRC, data in = B5 result = 00D6
            nop                     ;need delay
            mov     crclsb  ,crclsb ;cleared, data in = D6 result = 0000

            nop
            mov     p2 ,crcmsb      ;p1=00 read crcmsb register
            mov     p3 ,crclsb      ;p1=00 read crclsb register
end_loop:
            sjmp    $
            end
```

As mentioned, the CRC–16 function is optionally available to the application software. This is available regardless of whether the automatic power–on CRC is used. Although a CRC could be computed completely in software, it would take much longer than using the DS5001 facility. Using the CRC–16 hardware, the DS5001 series can perform a CRC–16 on 64K bytes of memory in approximately 500 ms. The CRC–16 logic resides behind the two SFRs mentioned above. These display the current CRC result and also serve as the input locations. The software must sequentially write the memory values into the CRC LSB at location 0C2h. After a delay of one instruction cycle, the 16–bit result will be available at 0C3h and 0C2h. The CRC–16 is a superior method of checking the file validity compared to a checksum. Using the DS5001 hardware, it can be computed quickly. When using the CRC–16 hardware as part of an application, the existing CRC should first be cleared. This is done by writing the CRC back on itself. This process makes the CRC–16 result equal to 0000h. The LSB is written back twice with a delay in between for computation. The code example shown in Figure 8–3 displays the CRC–16 result on ports 0 and 1.