SECTION 13: TIMED ACCESS PROTECTION

The High–Speed Microcontroller uses a protection feature called Timed Access to prevent accidental writes to critical SFR bits. These bits could cause a system failure or prevent the Watchdog Timer from doing its job if improperly written. The Timed Access involves opening a timing window during which the protected bit can be modified. If the window is opened correctly, it remains open long enough to alter one protected bit. This section explains which bits are protected, why, and how to use the Timed Access feature.

PROTECTED BITS

Bits which are protected by the Timed Access feature are shown below. Only critical function bits which are unique to the High–Speed Microcontroller family are protected, assuring code compatibility with the original 80C51 or 80C52. A full description of the function of each bit is provided in Section 4.

EXIF.0	BGS	Band–gap Select
WDCON.6	POR	Power–on Reset Flag
WDCON.1	EWT	Watchdog Reset Enable
WDCON.0	RWT	Reset Watchdog Timer
WDCON.3	WDIF	Watchdog Interrupt Flag
TRIM.7	E4K	4096 Hz RTC Output
TRIM.6	X12/$\overline{6}$	12pF/6pF Crystal Select
TRIM.5	TRM2	Capacitance Trim Bit 2
TRIM.4	$\overline{\text{TRM2}}$	Inverse Capacitance Trim Bit 2
TRIM.3	TRM1	Capacitance Trim Bit 1
TRIM.2	$\overline{\text{TRM1}}$	Inverse Capacitance Trim Bit 1
TRIM.1	TRM0	Capacitance Trim Bit 0
TRIM.0	$\overline{\text{TRM0}}$	Inverse Capacitance Trim Bit 0
ROMSIZE.2	RMS2	ROM Size Select Bit 2
ROMSIZE.1	RMS1	ROM Size Select Bit 1
ROMSIZE.0	RMS0	ROM Size Select Bit 0
RTCC.2	RTCWE	RTC Write Enable
RTCC.0	RTCE	RTC Enable

PROTECTION SCHEME

Each bit mentioned above is protected against an accidental write by requiring the software to perform a procedure before writing the bit. Timed Access requires the software to write two specific values to the Timed Access register during two consecutive instruction cycles. The values AAh, then 55h, must be written in consecutive instructions to the TA register at SFR location C7h. If the writes are performed correctly, the write access window will open for three machine cycles. During this window, the software may modify a protected bit. The suggested code to open a Timed Access window is:

```
MOV   0C7h, #0AAh
MOV   0C7h, #55h
```

The procedure to modify a Timed Access protected bit begins by writing the value AAh to the Time Access register (TA;C7h). The value 55h must then be written to the Timed Access register within three machine cycles of writing AAh. This opens a three machine cycle window, after the write of 55h, during which any Timed Access protected bits may be modified. Failure to complete any of the required steps will also require the procedure to begin again, starting with the write of AAh to the Timed Access register. Attempts to modify Timed Access protected bits after the window has closed will be ignored. This is regardless of whether any bits were modified. Figure 13–1 illustrates a number of examples of correct and incorrect use of the Timed Access procedure.

TIMED ACCESS EXAMPLES Figure 13–1

three machine cycles	three machine cycles	two machine cycles
MOV 0C7h, #0AAh	MOV 0C7h, #55h	SETB EWT

three machine cycles	three machine cycles	one machine cycle	two machine cycles
MOV 0C7h, #0AAh	MOV 0C7h, #55h	NOP	SETB EWT

three machine cycles	three machine cycles	three machine cycles
MOV 0C7h, #0AAh	MOV 0C7h, #55h	MOV WDCON, #02h

VALID TIMED ACCESS PROCEDURES

three machine cycles	one machine cycle	three machine cycles	two machine cycles
MOV 0C7h, #0AAh	NOP	MOV 0C7h, #55h	SETB EWT

*Second write to TA register does not occur within 3 cycles of first write.

three machine cycles	three machine cycles	one machine cycle	three machine cycles
MOV 0C7h, #0AAh	MOV 0C7h, #55h	NOP	MOV WDCON, #02h

*Modification of protected bit did not occur with 3 cycles of second write to TA register.

three machine cycles	three machine cycles	two machine cycles	two machine cycles
MOV 0C7h, #0AAh	MOV 0C7h, #55h	SETB EWT	SETB RWT

*Modification of second protected bit did not complete within 3 cycles of second write to TA register.

INVALID TIMED ACCESS PROCEDURES

TIMED ACCESS PROTECTS WATCHDOG

Any microcontroller–based system can be faced with environmental conditions that are beyond its designed abilities. These include external signal transients due to component failure, fluctuating power conditions, massive electrostatic discharge (ESD), and other unexpected system events. When a microcontroller is exposed to such conditions, program execution can become corrupted. Members of the High–Speed Microcontroller family which incorporate a Watchdog Timer can initiate a reset to recover from these conditions. The primary function of the Timed Access feature is to protect against accidental disabling of the watchdog timer by an "out–of–control" device. This will allow the watchdog timer to reset the system in the event of program execution failure.

The following hypothetical example demonstrates how a single bit change can corrupt program execution. The Timed Access procedure protects against an accidental write to the EWT bit by the errant code, allowing the watchdog timer reset function to reset the device. While this is a purely fictitious example, it illustrates how the watchdog timer and Timed Access feature make the High–Speed Microcontroller minimize the effect of accidental code corruption. *Note: Timed Access is not optional and must be supported if the protected bits are used. This example simply helps explain the category of problem that the Timed Access prevents.*

EXAMPLE: A TRANSIENT CAUSES THE WATCHDOG TO BE DISABLED

```
        TABLE_READ:
C2D2    90 0A 00        MOV     DPTR, 0A00H     ;LOAD TABLE POINTER
C2D5    79 FF           MOV     R1, #0FFH       ;LOAD COUNTER
C2D7    78 90           MOV     R0, #90H        ;DESTINATION POINTER

                        LOOP:
C2D9    E0              MOVX    A, @DPTR        ;READ DATA BYTE
C2DA    F6              MOV     @R0, A          ;STORE IT IN RAM
C2DB    06              INC     R0              ;NEXT TABLE LOCATION
C2DC    A3              INC     DPTR            ;NEXT DATA VALUE
C2DD    D9 C2 D9        DJNZ    R1, LOOP        ;NEXT BYTE OR DONE ?
```

A transient occurs while the opcode is being fetched for the first instruction. The transient causes one bit of the opcode in the first instruction to be read as a 0 instead of 1. The resulting program is what the microcontroller would actually execute:

```
        TABLE_READ:

C2D2    80 0A 00        SJMP    0BH             ;RELATIVE JUMP BY 10 LOCATIONS
C2D5    79 FF           MOV     R1, #0FFH       ;LOAD COUNTER
C2D7    78 90           MOV     R0, #90H        ;DESTINATION POINTER

                        LOOP:
C2D9    E0              MOVX    A, @DPTR        ;READ DATA BYTE
C2DA    F6              MOV     @R0, A          ;STORE IT IN RAM
C2DB    06              INC     R0              ;NEXT TABLE LOCATION
C2DC    A3              INC     DPTR            ;NEXT DATA VALUE
C2DD    D9 C2 D9        DJNZ    R1, LOOP        ;NEXT BYTE OR DONE ?
```

The resulting jump is to address C2DE. This is not even a real opcode, but would be treated as such. The resulting fetch is the value C2 D9. This is the opcode for CLR D9h. The bit addressable location D9h corresponds to the EWT – Enable Watchdog Timer. If the Timed Access procedure did not prevent it, this errant instruction would disable the Watchdog. Note that now, the program execution is completely lost. Real opcodes are being replaced by operands, data and garbage. In the High–Speed Microcontroller, the Watchdog will recover from this state as soon as it times out since it could not have been disabled in this way.

In the High–Speed Microcontroller it is very hard to contrive a situation that will accidentally disable the Watchdog. Note, the Timed Access prevents accidentally writing a bit. It can not prevent accidentally calling the correct code that writes a bit. This is much more unlikely however.