19-4056; Rev: 0; 2/08

ERRATA SHEET
MAXQ7665

*The errata listed below describe situations where MAXQ7665 components perform differently than expected or differently than described in the data sheet. To obtain an errata sheet on another MAXQ7665 die revision, visit our website at **www.maxim-ic.com/errata**.*

1) FLASH ERASE/PROGRAMMING OPERATION CAN RETURN FALSE ERROR FLAG

Description:
During a flash program/erase operation, the MAXQ7665 flash interface logic may return a false erase/write failure error flag even though the operation completes successfully. This may occur occasionally due to an incorrect internal status read in a specific timing combination and does not occur every time.

Workaround:
Re-issue erase/program command on error flag.

2) UTILITY ROM FILL CODE DOES NOT STOP RUNAWAY CODE EXECUTION

Description:
The MAXQ7665 utility ROM fill code is byte reversed. Unused locations of the utility ROM are filled with 0xFF0C instead of 0x0CFF. This is “move DP[01h], IP” instruction instead of the planned “SJUMP $” instruction. The SJUMP instruction is a better choice and would have stopped runaway code execution compared to IP move to one of the data pointers.

Workaround:
Do not jump to unused ROM code locations. The user should only have cause to jump to the published subroutine start addresses.

3) SIGN EXTENSION OF TEMP SENSOR OFFSET (TSO) REGISTER VALUE NOT IMPLEMENTED

Description:
According to specifications, the effective length of the TSO register is 12 bits, the upper four bits are sign extended. The sign extension is not implemented and a negative offset would result in incorrect arithmetic. As a result, for a negative offset, the utility ROM temperature conversion routine adjustment to temperature result is incorrect.

Workaround:
Re-adjust temperature result returned by utility ROM routine if TSO offset is negative.

4) CPU DETECTION OF ANALOG STATUS COULD BE SLOW WHEN USING LOW-POWER MODES

Description:
The analog interface module is connected to the divided system clock instead of the undivided clock. In general, when synchronizing asynchronous signals, the interface module uses the fastest clock available (8MHz) to resolve signals rapidly. This allows the CPU to see analog status signals (e.g., from the voltage monitors) almost as soon as they occur because the synchronization delay is very small. In MAXQ7665, the analog interface module is driven by the divided clock. As a result, if the CPU is using a divided clock, for example, low-power PMM mode (clock is /256 - 31kHz, 32µs period), then an analog event that causes switchback to fast mode (e.g., a DVDD brownout interrupt, assuming it is enabled) may take longer (up to approximately six 32µs clock periods, instead of approximately six 125ns clocks) to become evident to the CPU. The CPU would still switch back to 8MHz and then service the interrupt, but the delay before switchback could be longer than expected.

Workaround:
None.

5) RESTRICTIONS IN ENTERING AND EXITING PMME MODE

Description:
Only PMME mode 100 should be used. Other clock-divide ratios cannot be used in conjunction with the PMME bit.

Workaround:
Before setting the PMME bit in CKCN register to enter low-power mode, ensure the clock-divide control bits (CD1, CD0) in the CKCN register are set to 0.

6) RESTRICTION ON SETTING BREAKPOINT FOLLOWING IGE MODIFICATION

Description:
This problem appears only at debug time. If a breakpoint is set on the instruction immediately following IGE bit modification, the modification does not take effect. The following code snippet illustrates the issue:

```
;
; IGE test
;
; This test shows that setting a breakpoint after modifying IGE
; will cause the new value to be lost.
;
;
$include (maxq7665.inc)

  org 0
  move IC, #1
  nop             ; Breakpoint here. New value gets thrown away here.
  move a[0], IC
  nop             ; Breakpoint here. A[0] should be 1 but comes back 0.
  move IC, #1
  move a[1], IC   ; No breakpoint after mod of IC to insure new value sticks.
                  ; A[1] will be 1.
  move IC, #0
  nop             ; Breakpoint here. New value gets thrown away here.
  move a[2], IC
  nop             ; Breakpoint here. A[2] should be 0 but comes back 1.
  end
```

Workaround:
Do not put a breakpoint on the instruction immediately following IGE bit modification.