DS80C390
Dual CAN High-Speed Microprocessor

REVISION B2 ERRATA

The errata listed below describe situations where DS80C390 revision B2 components perform differently than expected or differently than described in the data sheet. Dallas Semiconductor intends to correct these errata in subsequent die revisions.

This errata sheet only applies to DS80C390 revision B2 components. Revision B2 components are branded on the topside of the package with a six-digit code in the form yywwB2, where yy and ww are two-digit numbers representing the year and workweek of manufacture, respectively. To obtain an errata sheet on another DS80C390 die revision, visit the website at **www.maxim-ic.com/errata**.

1. CURRENT CONSUMPTION EXCEEDS SPECIFICATIONS

Description:
Current consumption can exceed the data sheet specifications. Listed below is the maximum current consumption in various operating modes.

Active at 40MHz: 160mA
Idle at 40MHz: 100mA
Stop: 10mA

Work Around:
None

2. I_{TL} EXCEEDS SPECIFICATIONS

Description:
I_{TL} maximum can be as high as -700µA.

Work Around:
None

3. R_{RST} EXCEEDS SPECIFICATIONS

Description:
R_{RST} maximum can be as high as 300kΩ.

Work Around:
None

4. FREQUENCY MULTIPLIER DISRUPTS WATCHDOG TIMER OPERATION

Description:
Using the watchdog-reset function while the CTM bit is set causes unpredictable device operation.

Work Around:
Do not use the watchdog-reset function with the crystal clock multiplier.

5. SERIAL PORT 0 DOES NOT OPERATE CORRECTLY UNDER CERTAIN CONDITIONS

Description:
Serial Port 0 does not operate correctly under the following conditions:

1) Timer 2 is used as the time base (RCLK = TCLK = 1), and
2) The SMOD bit for serial port 0 is cleared, and
3) Timer 1 is running (TR1 = 1).

Work Around:
Ensure that these conditions never occur simultaneously while using serial port 0. The easiest way to accomplish this is to use the serial port with the serial port doubler bit set (SMOD = 1).

6. RESTRICTIONS ON ACALL OR LCALL INSTRUCTION IN 24-BIT CONTIGUOUS MODE

Description:
While operating in the 24-bit contiguous addressing mode, the execution of an ACALL or LCALL instruction when the return address is located within 4 bytes of a 64kB address boundary can result in incorrect data being pushed onto the stack.

Work Around:
Ensure that the return addresses are not located within 4 bytes of a 64kB page boundary.

7. RI BIT CAN BE UNINTENTIONALLY SET IN SERIAL PORT MODE 3

Description:
When either serial port 0 or 1 is operated in mode 3, the RI bit of that serial port can be unintentionally set under the following conditions:

- Multiprocessor communication mode is enabled (SM2 = 1), and
- A byte is received by that serial port with the RB8 bit set, and
- The SADEN register associated with that serial port is non-zero, and
- The received byte, masked by the SADEN register, does not match the associated SADDR register.

Work Around:
Ensure RB8 = 0 any time a byte is received. If using the serial port in mode3 with SM2 = 1, clear RB8 when initializing the serial port and after every byte is received. If the RI bit is set and RB8 = 0, then the received byte did not match and can be ignored.

8. RESTRICTIONS ON DIV AB INSTRUCTION

Description:

The DIV AB instruction can return erroneous results if the A register is accessed immediately preceding the DIV AB instruction.

Work Around:

It is common programming practice to load both the A and B registers right before executing the DIV AB. Either of the following two programming approaches execute correctly.

```
MOV B, #data
MOV A, #data
DIV AB ;*INCORRECT*
```

```
MOV A, #data
MOV B, #data
DIV AB ;CORRECT
```

```
MOV B, #data
MOV A, #data
NOP
DIV AB ;CORRECT
```

9. INTERRUPT WHILE SETTING IDLE BIT CAN CORRUPT OPERATION

Description:

Device operation may be corrupted if an interrupt occurs during the execution of the instruction that sets the idle bit (PCON.0). Interrupts that are enabled and acknowledged after the idle state has been entered are handled normally and cause an exit from the idle state.

Work Around:

Software can be used to detect which interrupts are imminent and to put off entering the idle state until the interrupt. The detection technique is specific to the type of interrupts that are used/enabled in the application:

Serial Port: If a serial port interrupt is enabled, poll the least significant nibble of the Status (C5h) SFR. If all 4 bits are clear, then no serial port transmit or receive operation is in progress and the idle bit can be safely set. If any of the 4 bits are set, loop until clear.

Timer 0, 1, or 2: Read the timer value, and, if close to its rollover value, delay until the timer has rolled over.

CAN0/1 and CAN Bus Activity Interrupts: Under certain circumstances it is possible to have CAN interrupts enabled and use the idle feature at the same time. This can be accomplished by polling the CAN bus active bits (CAN0BA and/or CAN1BA) to confirm if CAN activity, which might generate an interrupt, is ongoing. If both bits are clear, then no CAN operation is in progress and the idle bit can be safely set. Note that this scheme may not work in some networks because of the high rate of CAN activity.

Watchdog Timer: Reset the watchdog timer before entering idle mode.

Power-Fail Interrupt: No work around. Do not use the power-fail interrupt if the idle mode is also used.

External Interrupts: By their nature, external interrupts are asynchronous and very difficult to predict. By careful examination of the application it is possible to find windows of time during which the idle mode can be safely invoked without fear of an external interrupt occurring at the same time. In this way, it may be possible to use external interrupts and the idle mode in the same application.

10. INT0 AND INT1 PINS REQUIRE SYNCHRONIZED INPUT

Description:

The input pins of INT0 and INT1 can fail to detect transitions in edge mode if the transitions occur at specific times during the sampling process. This situation does not occur with interrupt signals that are synchronized to processor operation.

Work Around:

Qualify the input on the previously mentioned signals so that they do not transition during the sampling window. This can be easily done by gating the input signal with a D-type flip-flop using the falling edge of ALE as a clocking signal.

11. UPDATE OF CAN INTRQ, DTUP BITS BLOCKED BY SIMULTANEOUS SOFTWARE WRITE

Description:

Under the following conditions, the INTRQ and DTUP bits for a message center can fail to be updated by an incoming message:

1) An incoming message has been received in a message center,
2) The internal CAN hardware sets the INTRQ bit of that message center, and
3) Software simultaneously clears the INTRQ bit of any other message center in the same CAN module.

There is no direct way to detect this condition because of the invisibility of the internal CAN hardware operations and the asynchronous nature of CAN communications.

Work Around:

Although the INTRQ and DTUP bits are not set, CAN receive message-stored register (CxRMS0 and CxRMS1) and the CAN transmit message-acknowledgment register (CxTMA0 and CxTMA1) bits are still updated to show activity associated with the corresponding message center. After clearing any INTRQ bit, interrogate the CxRMS0, CxRMS1, CxTMA0, and CxTMA1 registers. Indication of activity in those registers points to either a missed setting of the INTRQ or new activity in another message center. In either case, service the message centers as appropriate.

12. UPDATE OF CAN MTRQ BIT BLOCKED BY SIMULTANEOUS SOFTWARE WRITE

Description:

Under the following conditions, the MTRQ bit for a message center can fail to be updated by the CAN hardware. This can cause multiple transmissions of a message until the condition is cleared.

1) The internal CAN hardware attempts to modify the MTRQ bit of a particular message center, either to signal that a message has been transmitted, or is transmitting in response to a remote frame request.

2) Software simultaneously modifies the MTRQ bit of any other message center in the same CAN module.

The effect of this condition depends on the use of the MTRQ bit:

1) If hardware attempted to clear the MTRQ bit to signal that a message has been successfully transmitted, the MTRQ bit is not cleared. As a result, the message be resent until the condition clears, at which time the MTRQ bit will be cleared.
2) If hardware attempted to set the MTRQ bit in response to a remote frame request, then the remote frame request is received but the MTRQ bit is not automatically set.

There is no direct way to detect this condition because of the invisibility of the internal CAN hardware operations and the asynchronous nature of CAN communications.

Work Around:

The work around depends on the use of the MTRQ bit:

1) *The MTRQ bit is being used to signal that a message has been successfully transmitted:* Ensure that the system level software can tolerate multiple resends of the same message. Alternatively, a polling routine can be used to wait until all MTRQ bits of all message centers are 0. When all MTRQ bits of all message centers are 0, there is no pending CAN activity that could simultaneously write to the MTRQ bit while software sets it to initiate a transmission.
2) *The MTRQ bit is being used to in conjunction with a remote frame request:* As a result of this erratum, the automatic frame request reply feature cannot be supported. Use the nonautomatic frame-request reply feature.

13. RI BIT SET EARLIER THAN EXPECTED

Description:

In mode 3, the RI bit associated with serial port 0 and 1 is set 1/16 of a bit time earlier than expected. This condition is only noticed when using polled serial communications. In this situation, reading the contents of SBUF immediately after the RI flag is set can result in the retrieval of incorrect data.

Work Around:

When performing polled serial communications, an instruction sequence such as the following tests the RI bit to determine when a character has been received.

```
JNB      RI, $       ;Loop until RI flag is set
MOV      A, SBUF     ;Read serial buffer and store off contents
```

The solution adds a delay between the setting of the RI flag and accessing the SBUF register. At 9600bps, 1/16 of a bit time would be 6.5µs. If the main oscillator frequency is 11.0592MHz, it equals 18 machine cycles. The following sequence provides the necessary delay.

```
JNB      RI, $       ;Loop until RI flag is set
MOV      A, #05h     ;Initialize delay counter     [3 machine cycles]
DJNZ     A, $        ;Delay loop [5x 3 machine cycles]
MOV      A, SBUF     ;Read serial buffer and store off contents
```

At fast baud rates, as few as one or two NOP instructions can provide sufficient delay.

14. SCON0 AND SCON1 SFRS CANNOT BE MODIFIED WHEN TRANSMITTING

Description:

SCON0 and SCON1 cannot be modified while their respective transmitter circuitry is active. This is most likely an issue when attempting to clear the RI flag during full-duplex serial port activity. This erratum applies to both serial ports.

Work Around:

This erratum can be overcome with two simple modifications. First, software should check the state of the RI after attempting to clear it. If no transmit activity was in progress, the bit was cleared and software can proceed. If the bit was not cleared, then the software should repeat the attempt until successful. The following code fragment illustrates the procedure:

```
ri_loop:
      clr  RI
      jb   RI, ri_loop
```

The second condition avoids the possibility of a receiver overrun while waiting to write to SCON. This is easily remedied by adding a character-pacing requirement to the incoming data stream. Specifically, the following time delay should be inserted between characters in the incoming data stream to avoid a receiver overrun during the preceding code example:

$$\text{Delay} = \left\lceil \frac{1}{\text{baud\_rate}} \right\rceil \text{ seconds}$$

The same effect can be achieved by configuring the transmitter to send 2 stop bits per character.

15. RB8 BIT INCORRECT IN SERIAL PORT MODE 1 OR 2

Description:

In mode 1 or 2, the RB8 bit associated with serial port 0 and 1 does not reflect the state of the ninth bit of the received data.

Work Around:

None

16. READS OF C0RMSX OR C0TMAX CAN UNINTENTIONALLY AFFECT OTHER SFRs

Description:

Reads of the CAN 0 transmit message-acknowledgement registers or the CAN 0 receive message-stored registers can unintentionally clear the corresponding SFR in the CAN 1 module. For example, reading C0RMS0 clears C1RMS0. Affected registers include C0RMS0, C0RMS1, C0TMA0, C0TMA1, C1RMS0, C1RMS1, C1TMA0, and C1TMA1.

Work Around:

When reading one of the above CAN 0 SFRs, first read and store the corresponding CAN 1 SFR to preserve its contents. For example, if software desires to read C0RMS0, first read the C1RMS1 and store off the contents to a dedicated shadow memory location. Then read C0RMS0 normally. When it is desired to read C1RMS1, read it and logically OR its contents with the shadow register. This workaround requires 4 bytes of memory to shadow the contents of the C1RMS0, C1RMS1, C1TMA0, and C1TMA1 registers.

The software examples below use direct RAM to shadow the SFRs. One example routine is shown for reading affected SFRs associated with CAN 0, another routine for reading affected SFRs associated with CAN 1.

```
; Reading C0TMAx or C0RMSx. This example reads C0TMA0
MOV R0, #C1TMA0_addr ; Point to shadow location for C1TMA0.
MOV @R0, C1TMA0      ; Copy C1TMA0 to shadow location.
MOV A,   C0TMA0      ; Read C0TMA0.

; Reading C1TMAx or C1RMSx. This example reads C1TMA0
MOV R0, #C1TMA0_addr ; Point to shadow location for C1TMA0.
MOV A,  C1TMA0       ; Get current value of C1TMA0.
ORL A,  @R0          ; ORL with prev value stored in shadow loc.
```

17. FIRST CODE FETCH FOLLOWING RESET CAN EXHIBIT DIFFERENT TIMING

Description:

The first code fetch following a reset occurs early. This could result in incorrect program execution.

Work Around:

The work around depends on whether multiplexed or nonmultiplexed addressing is used.

Multiplexed: The first instruction of the program, located at 000000h, must be an NOP (00h). Because the reset vector is only 3 bytes long, the next instruction must be an SJMP to another location where an LJMP redirects program flow to the start of the main program. The combination of the address bus and op code being all 0s ensures that the NOP and subsequent instructions are executed correctly.

Nonmultiplexed: Assume that for the first cycle following a reset the t_{MCS} factor is 3 t_{CLCL}.

18. EXTERNAL INTERRUPTS 2–5 WILL NOT EXIT STOP MODE

Description:

External interrupts 2–5 cannot be used to cause the device to exit Stop mode.

Work Around:

Use external interrupts 0 and 1 if an external interrupt must be used to exit Stop Mode. It may also be possible to use the Idle mode of operation in place of Stop mode. This erratum does not affect DS80C390 revision C devices.

19. CAN AUTOBAUD MODE RXS BIT FUNCTION CLARIFIED

Description:

When either CAN is operating in autobaud mode, the RXS bit in the CxS SFR will only be set upon reception of a valid (i.e., no bus errors) identifier that matches one or more of the message IDs programmed into the CAN module. The documentation implies that the RXS bit should be set if a valid identifier is received, even if the identifier did not match any of the message IDs programmed into the CAN module.

Work Around:

If it is desired to use the autobaud mode to monitor bus activity and set RXS when any message is successfully received, the user should enable a message center to receive messages with any ID. Upon exit from autobaud mode, this message center can be reconfigured for other uses.