TMS470R1VF448, TMS470R1VF4B8

TMS470 Microcontroller

Silicon Errata

Silicon Revision A

SPNZ125B
August 2006

REVISION HISTORY

This silicon errata revision history highlights the technical changes made to the SPNZ125A revision to make it an SPNZ125B revision.

PAGE(S) NO.	ADDITIONS/CHANGES/DELETIONS
8	Added errata ADC#15−18
9	Added erratum FP#9
18	Added errata MIBSPI#50 and MIBSPI#51
19	Added errata MIBSPI#52 and MIBSPI#53
20	Added erratum RTI#7

Contents

1 Known Design Marginality/Exceptions to Functional Specifications

The following is a list of advisories on modules in this version of silicon. Documentation may differ from the user guide or data sheet. The advisory reference number is shown first (i.e.; ADM#8), followed by a description and any known workarounds. The reference numbers may not always be sequential for this device.

Modules include the following:

Multi-buffered analog-to-digital converter (ADM)

Flash pump (FP)

Flash wrapper (FW)

General purpose input/output (GIO)

High-end timer (HET)

Multi-buffered serial peripheral interface (MIBSPI)

Real-time interrupt (RTI)

Standard CAN controller (SCC)

Serial peripheral interface (SPI)

Zero-pin phase-locked loop (ZPLL)

Advisory ADM#8	*Stopping and Starting ADC When Conversions Are Ongoing*

Description: When used in FIFO mode, if the A to D module is disabled or if the channel select registers are cleared while conversions are still ongoing, the operation will be unpredictable when the module is restarted.

Workaround: Stop the ADC by clearing ADCR1(5).
Restart the ADC by setting ADCR1(5) again.
Configure all groups to be in single conversion mode.
Configure one channel to be converted in all three groups and start the conversions.
Wait for these conversions to end by polling the ”conversion end” flags in the ADSR register.
Clear the channel select registers for the three groups.
Continue with the desired configuration for the ADC.

Advisory ADM#9 — *Freeze Feature Error for Conversion Groups*

Description: When multiple conversion groups are being used and the ADC is used in the multi-buffered mode, the use of the freeze feature for conversion groups can lead to conversion results being written to the wrong FIFO. If a conversion group (say group A) is configured to be ”freezable”, and if there is a request for servicing another conversion group (say group B) while group A conversion is still ongoing, then the conversion result for the last channel converted in group A will be written to the FIFO for group B.

Workaround: Do not use the freeze ability for the conversion groups.
OR
For applications that must use the freeze ability, please use only the compatibility mode of the ADC.

Advisory ADM#10 — *Data Not Written to MibADC FIFO*

Description: In buffered mode, if the channel select register is written and if no other MibADC registers are accessed, the converted data does not get written to the FIFO but the threshold counter is updated upon each conversion. This occurs only when the ICLK/SYSCLK ratio is more than 2.

Workaround: Do a read operation from the same group input channel select register after writing it.

Advisory ADM#11 — *Disabling EN Bit Does Not Properly Stop and Restart the ADC*

Description: Disabling the ADC by clearing the ADC EN bit of the ADCR1 register does not reset some internal flags used to store the conversion statuses of the three groups, causing problems when stopping and restarting the ADC conversions in buffered mode.

Workaround: TBD

Advisory ADM#12 — *Clearing Channel Selection Register Does Not Clear FIFO Completely*

Description: Clearing the channel selection register does not clear FIFO completely under certain conditions.

Workaround: Depending upon the number of selected Channels, the over−head on ”time” is in the increasing order with each work−around. Depending upon the requirement and criticality, any of them can be chosen.

1. Irrespective of whether a group is in continuous or single conversion mode, read−out the corresponding group’s FIFO data until it’s empty, write all ’0’s to the Group Channel Select Register. After this, wait for the duration of one Channel Conversion completion including the sampling and the conversion time, before writing the next set of channels to the Channel Select Register.

2. Alternatively, if the group conversion is in continuous mode, and an application wants to change the Group Channel Select Register, then first change the mode of the group to single conversion. Read−out all the conversion data from the FIFO, until the FIFO becomes empty. Write a single channel into the Channel Select Register. Wait until the Group Conversion End flag gets set. Write the next set of Channels to the Group Channel Select Register.

3. If the group is in Single conversion mode, wait until the Group Conversion End flag is set, read out the FIFO data until it’s empty and then write the new set of Channels to the Group Channel Select Register.

Advisory ADM#13 — *Current Channel Conversion Does Not Complete When Suspend Mode is Entered*

Description: Ongoing conversion is not completed on entering suspend mode when COS = 1.

Workaround: TBD

Advisory ADM#14 — *Calibration Results Are Written to Group FIFO*

Description: If calibration is started during a group conversion, the result of the calibration is written to that group FIFO until the end of group conversion.

Workaround: Do not do calibration during a group conversion. The documentation will be updated to reflect this requirement. (SPNU193, 9/2002)

Advisory ADM#15 — *Multiple FIFO Flags Appear as '1'*

Description: Both <gp>FIFO Empty and <gp>FIFO overrun flags in AD<gp>SR register appear as '1'.

Workaround: Write a new Channel Select information into the corresponding AD<grp>SEL register to reset this status.

Advisory ADM#16 — *Write Pointer Incremented While Data Not Written When FIFO Is Full*

Description: If a FIFO read occurs exactly two ICLK cycles before a FIFO write while FIFO is already full, then the data does not get written, but the FIFO write pointer gets incremented.

Workaround: Avoid the FIFO overrun condition. Start reading out the results from the FIFO before it becomes full. The Interrupt Threshold Counter can be used to trigger reads at a specified boundary without reaching FIFO full condition.

Advisory ADM#17 — *Single Conversion Group Converted Twice*

Description: A single conversion group may get converted twice when all three groups have FREEZE bit set and other two groups are in continuous mode.

Workaround: This bug results in two sets of conversion results for the conversion group that is triggered first. The workaround is to discard one set of conversion results for this group.

This is only possible if the exact number of conversions results are properly tracked. The AD<grp>INTCR can be used to keep track of the number of conversions in each group.

Advisory ADM#18 — *Group Trigger On ADEVT Missed*

Description: Group trigger on ADEVT pin might be missed if the event edge lines up with an internal clock

Workaround: Do not use any external triggers for Event Group conversions. Event triggers can be triggered internal to the device by driving the I/O pin from the device itself through software.

Advisory FP#9 — *Increase Voltage to Improve Erase Yield*

Description: Increase VHV voltage at “B” setting to 9.67V to improve erase yield.

Workaround: None

Advisory FW#3 — *Configuration Mode Required for Sleep or Standby*

Description: The configuration mode must be set to enter sleep or standby modes.

Workaround: The documentation will be updated to reflect this requirement. (SPNU213, 12/2002)

Advisory FW#13 — *Fails Initial Read of 0x0−0x7 in Pipeline Mode*

Description: Immediately after entering pipeline mode, a data read of location 0x04 immediately following a data read of location 0x0 will cause 0x04 to read as all 0’s.

Workaround: Do a dummy data read of any location other than zero or four immediately after entering pipeline mode. The documentation will be updated to reflect this requirement. (SPNU213, 12/2002)

Advisory GIO#1 — *Reading the Interrupt Offset Registers*

Description: When either of the two interrupt offset registers are read, and a higher priority interrupt occurs in the same cycle, the interrupt pending flag for the higher-priority interrupt is wrongly cleared, but the offset for the lower-priority interrupt is read. As a result, the lower-priority interrupt will be serviced twice and the higher-priority interrupt will not be serviced at all.

Workaround: Do not read the interrupt offset register to identify the pending interrupt with the highest priority. Instead, read from the interrupt pending flag register and use bit tests to decode the pending interrupt with the highest priority by software. An additional write to the flag register is necessary to clear the pending interrupt flag.

Advisory HET#19	*Reading the Interrupt Offset Registers*

Description: When either of the two interrupt offset registers are read and a higher priority interrupt occurs in the same cycle, the interrupt pending flag for the higher-priority interrupt is wrongly cleared, but the offset for the lower-priority interrupt is read. As a result, the lower-priority interrupt will be serviced twice and the higher-priority interrupt will not be serviced at all.

Workaround: Do not read the interrupt offset register to identify the pending interrupt with the highest priority. Instead, read from the interrupt pending flag register and use bit tests to decode the pending interrupt with the highest priority by software. An additional write to the flag register is necessary to clear the pending interrupt flag.

Advisory MIBSPI#2	*Additional Trailing Wait State Needed When Reading Buffers with LDM Instructions*

Description: The software needs to add one trailing wait state whenever it needs to read the MibSPI buffers using the LDM instruction and all its variants.

Workaround: Documentation will be updated to reflect this requirement. (SPNU217, 5/2004)

Advisory MIBSPI#4	*MibSPI Registers Only Support 16-Bit Accesses*

Description: The MibSPI module registers only support half-word (16-bit) accesses, not word (32-bit) accesses.

Workaround: Documentation will be updated to reflect this requirement. (SPNU217, 5/2004)

Advisory MIBSPI#5	*Transfer Group Triggered Incorrectly by CS0 in 4-Pin Slave Mode*

Description: If the MibSPI is in slave mode 4-pin configuration with chip select CS0 as a functional pin, then the transfer group TG0 is triggered when CS0 is active (Low) and transfer group TG1 is triggered when CS0 is inactive (High).

Workaround: Documentation will be updated to reflect this requirement. (SPNU217, 5/2004)

Advisory MIBSPI#6	*Minimum/Maximum Delay Time Between Transfers*

Description: The minimum and maximum delay between transfers is not clear in the specification.

Workaround: Documentation will be updated to reflect this requirement. (SPNU217, 5/2004)

Advisory MIBSPI#7	*Receive Buffer Can Not Be Disabled in Slave Mode*

Description: The receive buffer can not be disabled in slave mode.

Workaround: Documentation will be updated to reflect this requirement. (SPNU217, 5/2004)

Advisory MIBSPI#9	*Setup Interrupt Incorrectly Set*

Description: When the suspend interrupt is disabled and the “suspend to wait” buffer mode is encountered, the suspend interrupt flag for TG (Transfer Group) is set but no interrupt occurs. When TG is finished, the interrupt occurs for that TG, thus exhibiting the incorrect behavior. At this time the suspend bit in the TGINTVECT register is set without checking the interrupt enable bit. The setting of this bit gives the impression that the “suspend to wait” interrupt has occurred. Thus the TG finished interrupt is never detected.

Workaround: TBD

Advisory MIBSPI#10	*MibSPI RAM and Register Access Problem During Buffer Initialization*

Description: During the MIBSPI buffer initialization, the CPU can not access the MibSPI RAM and some of the MibSPI registers (e.g. TGxCTRL).

Workaround: The software must delay after peripheral reset before accessing the MibSPI RAM and registers.

On fixed devices, a new “Buffer Initialization Active” flag is added. The CPU can test this flag to see if the buffer initialization is complete.

Advisory MIBSPI#11	*Transfer Group Start Address Corruption*

Description: If a triggered transfer group is not serviced before its trigger goes off (due to other active transfer groups), and then comes back again and is serviced, the transfer group will start from buffer0, ignoring its programmed PSTART. This behavior can be irrespective of transfer group priorities. This can happen only if there are multiple active transfer groups.

Workaround: TBD

Advisory MIBSPI#12	*All TG Buffers Transferred When Trigger Event Set to 'Always'*

Description: When a transfer group is being serviced, if the TGENA bit is cleared, the current buffer is transferred and then the transfer group should not be serviced any more. However, when the trigger event is set to "always", all the transfer group buffers get transferred before disabling the transfer group.

Workaround: TBD

Advisory MIBSPI#13	*MibSPI Sequencer Might Hang When TG0 is Disabled*

Description: If transfer group TG0 is level triggered, transfer group TG1 is edge triggered, and TG0 has a "suspend to wait" buffer, then TG1 might not be transferred if TG0 is disabled.

Workaround: TBD

Advisory MIBSPI#14	*PCURRENT Does Not Show Next Buffer*

Description: The PCURRENT is not incremented when the MibSPI is suspended on the RX_EMPTY flag. PCURRENT still shows the current buffer.

Workaround: Documentation will be updated (SPNU217, 10/2003)

Advisory MIBSPI#15	*DMA ICOUNT of 16 Bits Now Supported*

Description: The MibSPI has been enhanced to support a DMA ICOUNT of 16 bits.

Workaround: None

Advisory MIBSPI#16	*nENA Pin Remains Tri-Stated in 4-Pin Slave Mode*

Description: In 4-pin nENA MibSPI slave mode, the enable signal of nENA output buffer remains de-asserted if ENABLE HIGHZ = '1'. This means that the nENA pin from slave remains tri-stated.

Workaround: TBD

Advisory MIBSPI#17	*CSHOLD Not Supported in 5-Pin Slave Mode*

Description: CSHOLD is not supported in 5-pin slave mode.

Workaround: Documentation will be updated to reflect this requirement. (SPNU217, 5/2004)

Advisory MIBSPI#19	*Slave Chip Select De-assertion in 4-Pin Slave Mode Results in Data Lost*

Description: When the MibSPI is configured as a slave in 4-pin mode with the slave chip select functional, and if the slave chip select is de-asserted for three to four ICLK cycles in between two transfers, the data received in the first transfer is lost. This occurs only if there is an expansion bus memory transaction to the MibSPI RAM during that time.

Workaround: TBD

Advisory MIBSPI#20	*Additional DMA Transfer on Transmit Channel*

Description: When a buffer is configured for DMA transfer with the ONESHOT bit cleared and the NOBRK bit set in the DMAxCTRL register, an additional DMA request is observed on the transmit channel. The total number of TXDMA requests is ICOUNT+2.

Workaround: TBD

Advisory MIBSPI#21	*Transfer Group Priority Conflict*

Description: If two transfer groups are both enabled and triggered, and the higher priority transfer group is de-asserted after some transactions, the lower priority transfers will start. But if the higher priority TG is triggered again, the lower priority transfers will continue until the end of the lower priority transfer group.

Workaround: TBD

Advisory MIBSPI#22	*Incorrect Parity Error Detected on First Transaction*

Description: In 3-pin enhanced mode when the SPI is a master, polarity is 1, odd parity is enabled, and CHARLEN is 2, a parity error is incorrectly detected because the data is shifting in before the clock starts. This happens only for the first transaction.

Workaround: TBD

Advisory MIBSPI#23	*Trigger Change Causes Unpredictable Behavior of the MibSPI Sequencer*

Description: If a trigger change occurs while the first NOBRK RXDMA buffer is being transferred, the trigger change will be accepted by the sequencer since until the receive is completed the sequencer doesn't get NOBRK indication. This will create unpredictable behavior of the sequencer FSM.

Workaround: TBD

Advisory MIBSPI#24	*MibSPI Sequencer Hangs in "Suspend to Wait" State*

Description: If a buffer N is set as NOBRK RXDMA and if buffer (N+1) is set as "suspend to wait" until TXFULL, then the MibSPI sequencer FSM hangs in "suspend to wait" state only if level trigger goes off at the time. The TXFULL flag of the prefetched buffer (N+1) is cleared during prefetch. Since level goes off and comes back again, the buffer is read again, and this time the TXFULL flag is found cleared.

Workaround: TBD

Advisory MIBSPI#26	*Buffer Following a NOBRK RXDMA is Not Transferred*

Description: If a buffer is set as NOBRK RXDMA, then the next buffer (which is prefetched) is not transferred if it Is set as "suspend to wait" until TXFULL mode.

Workaround: TBD

Advisory MIBSPI#27 — *Incorrect Buffer Delay Behavior When CSHOLD is Set*

Description: When CSHOLD is set for a buffer, T2C delay is inserted at the end of the CSHOLD buffer and not inserted at the end of the next non-CSHOLD buffer.The correct behavior should be that T2C delay is not inserted at the end of CSHOLD buffer and C2T delay is not inserted at the beginning of the next buffer.

Workaround: TBD

Advisory MIBSPI#29 — *nENA Pin Set to Active Value in 5-Pin Compatibility Mode*

Description: In compatibility mode (5-pin mode), the MibSPI asserts the nENA pin to active value even if it is neither selected by master using nSCS(0), nor if new data is written to its transmit shift register.

Workaround: TBD

Advisory MIBSPI#30 — *Setup/Hold Times Are Affected if Baud Rate is Less Than ICLK/2*

Description: The MibSPI SPInSCS() signals to SPICLK setup/hold times have dependence on the frequency of SPICLK. This determines the time it takes for a master SPI to start SPICLK after it has selected a slave. For baud rates less than ICLK/2 (for example, ICLK/3 or 4 or more), the throughput of the SPI will be affected.

Workaround: TBD

Advisory MIBSPI#31 — *Must Initialize Buffer Before Enabling a Transfer Group*

Description: If a higher priority transfer group interrupts a lower priority transfer group and the data in the higher priority transfer group's buffer has not been initialized, then the first data received may not be written into the MibSPI RAM.

Workaround: The buffer must be initialized before enabling the transfer group.

Advisory MIBSPI#32 — *Status Flag Update Arbitration*

Description: The SPISTAT register status flag update has arbitration issue. The flag does not get cleared if a CPU read occurs at the same ICLK cycle when MibSPI is trying to clear the RXEMPTY flag indicating that receive is complete. This results in a situation where a receive complete goes undetected. Similar issues exist on other status flags such as TIMEOUT, PARITYERR, BITERR, RCVROVRN, and DESYNC. The RXEMPTY flag affects the basic functionality of MibSPI in compatibility mode, whereas the other bits affect only MibSPI Mode. Continuous polling of SPISTAT by the CPU could result in this malfunction and eventual failures.

Workaround: TBD

Advisory MIBSPI#33 — *False Setting of Suspend Interrupt Flag*

Description: When suspend interrupt is disabled and “suspend to wait” buffer mode is encountered, the suspend interrupt flag for the TG (Transfer Group) is set, but no interrupt occurs. When the TG is finished, the interrupt occurs for that TG, thus exhibiting the incorrect behavior. At this time, the suspend bit in the TGINTVECT register is set without checking the interrupt enable bit. This gives the impression that a “suspend to wait” interrupt has occurred. Thus the TG finished interrupt is never detected.

Workaround: TBD

Advisory MIBSPI#35 — *Incorrect Transfer Behavior if TGENA is Cleared in the Middle of Transfer*

Description: When a TG is being serviced, suppose the TGENA is made zero. The current buffer must be transferred and then the TG should not be serviced any more. BUT, if the trigger event is set as “always”, all the TG buffers get transferred before disabling the TG.

Workaround: TBD

Advisory MIBSPI#38 — *Data Corruption When Using ICOUNT=0*

Description: If transfer of a normal buffer is in progress and higher priority transfer group is triggered, and if the first buffer of the higher priority transfer group is a DMA buffer with icount 0, then TX_SHIFT_REG will be updated with new data before the transfer of the normal buffer is completed. This corrupts the data transferred for normal buffer.

Workaround: Do not use ICOUNT=0

Advisory MIBSPI#40 *4-Pin nEnable Mode With a Slave Not In Compatibility Mode*

Description: In 4-pin nENA mode with a slave in MIB slave mode, the enable signal of nENA output buffer remains low if ENABLE−HIGHZ = ’0’

Workaround: Do not use 4-pin ’nEnable’ mode with a slave which is not in compatibility mode.

Advisory MIBSPI#44 *Limitations of Higher Priority TG Interruption*

Description: The specification requirement of ”any higher priority transfer group can break any TG” has limitations in the current implementation. In the current implementation, once a TG has interrupted another TG, until the completion of this new TG, no higher priority TG can break-in. This affects the multibuffered mode of MibSPI only.

Workaround: None

Advisory MIBSPI#45 *Changing Clock Polarity Corrupts Data*

Description: Writing to change the clock polarity at the same time as the transmit data buffer is written (one 32-bit write) will cause the Master received data to be corrupted.

Workaround: Clock polarity must be changed by a CPU write between transfer groups.

Advisory MIBSPI#46 *Receive Char Length Counter Corrupted by Clock Edges*

Description: When used as a slave, the receive char length counter will be corrupted by clock edges even if the nSCS is inactive

Workaround: None

Advisory MIBSPI#47 *PCURRENT Value Shows Zero Until After the First Buffer*

Description: PCURRENT value in TGxCTRL register shows zero until after the first buffer of the transfer group has been transmitted.

Workaround: None

Advisory MIBSPI#48	*MibSPI Waits for the nENA Pin When WaitENA=1*

Description: MibSPI in multi-buffer mode waits for the nENA pin even if the nENA pin is not configured as functional if either C2T or T2C delays are used and WaitENA=1

Workaround: Do not select WaitENA=1 if the nENA pin is not used.

Advisory MIBSPI#49	*Receive Overrun Flag is Erroneously Set*

Description: When used as a multi-buffered slave, the receive overrun flag is erroneously set if a previous overrun condition has occurred, and the buffer is read while another transfer is occurring to the same buffer

Workaround: None

Advisory MIBSPI#50	*Chip Select Must Go High Between Transmissions in Slave Mode*

Description: In slave mode, the chip select must go high between transmissions.

Workaround: None

Advisory MIBSPI#51	*Higher Priority TG Completion Interrupt Lost*

Description: If a lower priority Transfer Group completion interrupt occurs and when reading the SPIINTVECT0 or SPIINTVECT1 register a higher priority TG completion interrupt occurs, then the bit of TGINTFLG register corresponding to the higher priority TG gets cleared but the vector read will indicate the lower priority interrupt. In effect, higher priority TG completion interrupt will be lost.

Workaround: None

Advisory MIBSPI#52	*Master Fails to Detect De-assertion of nENA Pin*

Description: If nENA pin is de-asserted for 1 or 2 ICLK cycles (if SPICLK frequency is slower than three ICLK cycles) and asserted again by the Slave, then the Master fails to detect the de-assertion of the nENA pin (depending upon the alignment of the nENA signal with respect to SPICLK) and DESYNC Error flag gets set. If T2E_DELAY count is set to '0', this issue does not show up.

Workaround: The nENA pin needs to be de-asserted for at least one SPICLK duration at the end of a transfer to guarantee that there's no false DESYNC Error if T2E_DELAY timer is used.

Advisory MIBSPI#53	*SPIPC2 Register Does Not Display Pin Value*

Description: The SPIPC2 register reads the internal DOUT value and not the actual pin value when the pin is in O/P mode.

Workaround: None

Advisory RTI#4	*Tap Interrupt When Clearing Counter in Suspend Mode*

Description: Write accesses to the RTICNTR register will clear the CNTR (21 bit counter), which causes a Tap interrupt if the corresponding bit switches from a "1" to a "0" when the suspend signal is asserted.

Workaround: This is the same problem as RTI#3, however, on the initial fix of RTI#3, the case where the suspend signal is asserted because of an emulator breakpoint was not considered. This problem occurs when the emulator has set a breakpoint on one of the instructions closely following the instruction which writes to the counter.

Advisory RTI#6	*Asynchronous Clear of the Tap Flag May Interfere With Tap Interrupt*

Description: Using the oscillator to clock the RTI counter, any asynchronous clear of the RTI Tap flag could cause an arbitration condition between the clear and the RTI module trying to set the flag. This will cause the Tap flag to not get set, and therefore the Tap interrupt will not occur.

Workaround: Before attempting to clear the RTI Tap flag, the RTI counter must be checked to make sure that the RTI module is not about to set the flag again.

Advisory RTI#7	*RTI Tap and Compare Interrupts Not Reliably Generated*

Description: When using the oscillator to clock the RTI counter, RTI TAP and RTI compare interrupts are not reliably generated.

Workaround: Use SYSCLK instead of OSCIN except when in STANDBY mode.

Advisory SCC#4	*CAN Does Not Perform a Resynchronization as Expected*

Description: Due to the proposed update of the ISO-WD-16485 CAN Test specification (2001−05−31), the HCC on this device has a non-conformance to the Bosch CAN Specification and the ISO-11898 Standard as described below.

If the following conditions are met, the CAN does not perform a re-synchronization as it is expected.

Conditions:

1. The node must be transmitter
2. The node must transmit a dominant bit
3. The dominant bit must be sampled back as recessive
4. A recessive to dominant edge must be detected after the sample point

But since the recessive sampling of the bit transmitted as dominant is an error anyway, an error frame will be transmitted at the beginning of the following bit.

Therefore, the effect of the non-conformance is a delay of this error frame. The maximum for this delay is five (max(SJW) + 1 Tq) time quanta.

Workaround: This non-conformance is classified as non-serious and does not have any impact on proper communication and inter-operability with other nodes. See above description.

Advisory SCC#5	*Pins Are High Impedance in Low Power Mode*

Description: Regardless of how the CANSTX or CANSRX pins are configured, they become general purpose inputs when entering low power mode.

Workaround: If the pin is not driven externally, which is usually the case with the CANSTX pin, an external pull-up or pull-down resistor should be added to avoid consuming extra current in low power mode.

Advisory SCC#6 — *CANSRX Must be High During Self-test*

Description: The CANSRX pin must be high during self-test.

Workaround: The CANSRX pin is usually driven high by the bus transceiver. As long as there is no bus activity during the self-test, this is not a problem. If there is nothing driving the CANSRX pin, it can be configured as a digital output and set high during the self-test.

Advisory SCC#7 — *Abort Acknowledge Bit Not Set After Transmission Request Reset*

Description: After aborting a message using the Transmission Request Reset (TRR) register bit, there are some rare instances where the TRR bit will clear without setting the Abort Acknowledge (AA) bit.

In order for this rare condition to occur all the following three conditions must happen:

1. The current message has a message error or lost arbitration. This message does not need to have the same mailbox number as the following TRR bit mailbox.
2. The TRS bit of the same mailbox as the TRR mailbox must be set from either this current message, prior to the current message and still pending, or just set.
3. The TRR bit must be set in the exact ICLK cycle were the wrapper state machine is in IDLE for one cycle. (One ICLK before or after and the condition will not occur). This IDLE state can occur just after the current message. It can also occur just a few ICLKs after setting the TRS bit of any mailbox after the current message (point 1 above).

If these conditions occur then the TRR and TRS bits for the mailbox will clear t_{clr} ICLKs after the TRR bit is set where:

$$t_{clr} = ((16 - mailbox\_number) * 2) + 3 \quad \text{ICLK cycles}$$

The TA and AA bits will not be set if this condition occurs. Normally, either the TA or AA bit sets after TRR bit goes to zero.

Workaround: When this problem occurs, the TRR and TRS bits will clear within t_{clr} ICLK cycles. To check for this condition, first disable the interrupts. Check the TRR bits' t_{clr} ICLK cycles after setting the TRR bits to make sure that they are still set. A set TRR bit indicates the problem did not occur. If TRR is cleared, then maybe it was the normal end of a message and the TA or AA bits are set. Check both the TA and AA bits. If they are both zero, then the conditions did occur. Handle the condition like the interrupt service routine would, except that the AA bit does not need clearing now. If the TA or AA bit is set, then the normal interrupt routine will happen when the interrupt is re-enabled.

Advisory SPI#1 — *Slave Baud Rate Setting*

Description: When the SPI is operated in slave mode, the SPI clock must be configured to a baud rate as close to the master's baud rate as possible. If the baud rate is too slow, the enable signal will not be generated in time to keep the master from sending additional data. If the baud rate is too fast, the slave will capture the data before the last bit is shifted in.

Workaround: The documentation will be updated to reflect this requirement. (SPNU195C, 7/2003)

Advisory SPI#2 — *Clearing, Setting SPI EN Bit Does Not Clear Internal Flag*

Description: Clearing and then setting the SPI EN bit does not clear an internal flag that indicates there is valid data in the SPI data register. This could lead to an inadvertent overrun error. The software should do a dummy read of SPIBUF after setting SPI EN bit to clear the internal flag.

Workaround: The documentation will be updated to reflect this requirement. (SPNU195C, 7/2003)

Advisory ZPLL#1 — *Interrupt Requests to the CIM Module Must Be Disabled*

Description: All interrupt requests coming to the CIM module must be disabled when changing between multiply by 4 and multiply by 8.

Workaround: Disable the interrupt request at the peripheral source if possible.

IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, modifications, enhancements, improvements, and other changes to its products and services at any time and to discontinue any product or service without notice. Customers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All products are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its hardware products to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by government requirements, testing of all parameters of each product is not necessarily performed.

TI assumes no liability for applications assistance or customer product design. Customers are responsible for their products and applications using TI components. To minimize the risks associated with customer products and applications, customers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any TI patent right, copyright, mask work right, or other TI intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information published by TI regarding third-party products or services does not constitute a license from TI to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. Reproduction of this information with alteration is an unfair and deceptive business practice. TI is not responsible or liable for such altered documentation.

Resale of TI products or services with statements different from or beyond the parameters stated by TI for that product or service voids all express and any implied warranties for the associated TI product or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

Following are URLs where you can obtain information on other Texas Instruments products and application solutions:

Products		**Applications**	
Amplifiers	amplifier.ti.com	Audio	www.ti.com/audio
Data Converters	dataconverter.ti.com	Automotive	www.ti.com/automotive
DSP	dsp.ti.com	Broadband	www.ti.com/broadband
Interface	interface.ti.com	Digital Control	www.ti.com/digitalcontrol
Logic	logic.ti.com	Military	www.ti.com/military
Power Mgmt	power.ti.com	Optical Networking	www.ti.com/opticalnetwork
Microcontrollers	microcontroller.ti.com	Security	www.ti.com/security
Low Power Wireless	www.ti.com/lpw	Telephony	www.ti.com/telephony
		Video & Imaging	www.ti.com/video
		Wireless	www.ti.com/wireless

Mailing Address: Texas Instruments
Post Office Box 655303 Dallas, Texas 75265