NS9360

Errata

90000677-88_F

Release date: March 2008

- Unused USB module can cause failures
- SPI boot fails intermittently - updated
- SPI slave data output high impedance control
- UART gap timer
- UART CTS-related transmit data errors
- Ethernet receive data overflow (ethernet receiver stall)
- Linked Ethernet TX buffer descriptors do not work with late collisions

Unused USB module can cause failures

The USB logic inside the NS9360 must be disabled when it is not used. Other sections, like the SPI DMA, fail because they enable the DMA logic that is also used by the USB, and with the USB in a random state due to being enabled and un-initialized can start random DMA operations.

There are two ways to disable the USB logic. One uses software and one uses hardware.

Software --Turn off the system clock to USB, and hold the USB Host and Device in Reset.

Be aware reading any USB register results in a bus error, but with fewer clocks running less power is consumed.

Software Workaround Code:

Addition to startup code.

```
/*
 * Hold USB Host and Device in Reset. Master Reset register Address: 9060 0000.    * D11 = 1 holds Host, and D12 = 1 holds Device.
 */
bbus_reset:
      LDR     R0, = BBUS_RESET_BASE
      LDR R1, [R0]                    /* Load the current value into R1 */
      LDR R2, =0x1802
      AND R1, R2, R1                  /* Mask out the bits we care about */
      STR R1, [R0]                   /* Put it back */
/*
 * Shut off the USB clocks for Host and Device. Clock Configuration register
 * Address: A090 017C. D14 = 0 disables Host, and D15 = 0 disables Device.
 */
LDR     R0, = CLOCK_CONFIGURATION_REGISTER
      LDR R1, [R0]                    /* Load the current value into R1 */
      LDR R2, =0xFFFF3FFF
      AND R1, R2, R1                  /* Mask out the bits we care about */
      STR R1, [R0]                   /* Put it back */
```

Hardware - Reset the USB controller at power-up by providing an external clock.

Hardware workaround:

Always provide an external USB clock into pin F18 (x1_usb_osc). The USB controller uses synchronous resets, and needs this clock running during power-up reset.

This clock does not need to be 48MHz. A lower frequency clock used elsewhere in the same design can be used. For example, the system clock or Ethernet clock.

SPI boot fails intermittently - updated

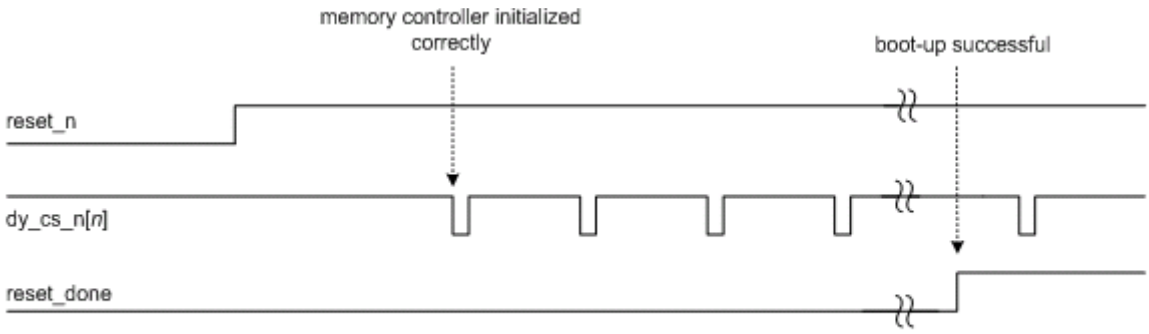
SPI boot fails intermittently during power-on because of an initialization error. As a result, a manual reset to the NS9360 can sometimes be required after the initial power-up reset. SPI boot works reliably from push button reset.

A failure can also occur if the software changes the system endian bit after boot-up (bit 3 in the Miscellaneous System Configuration register). This bit is set according to the bootstrap condition on gpio[44].

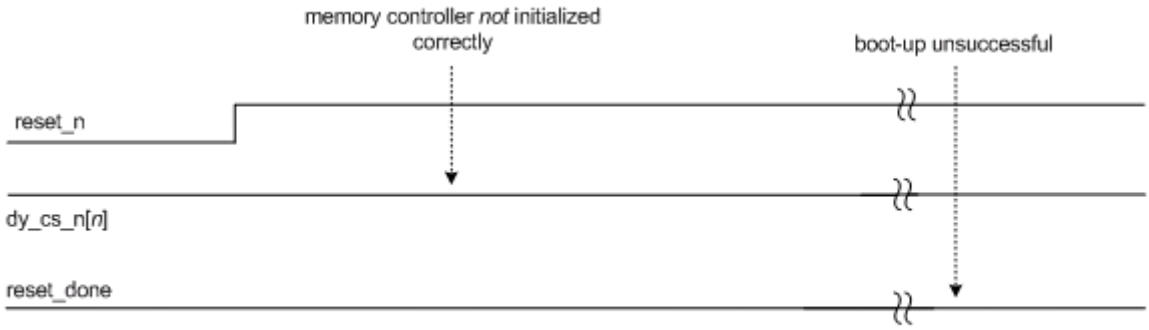
Hardware workaround:

A board hardware work around can detect and recover from the error.

The next diagrams show the timing for a proper boot-up and for an errored boot-up sequence:



Successful Boot-up

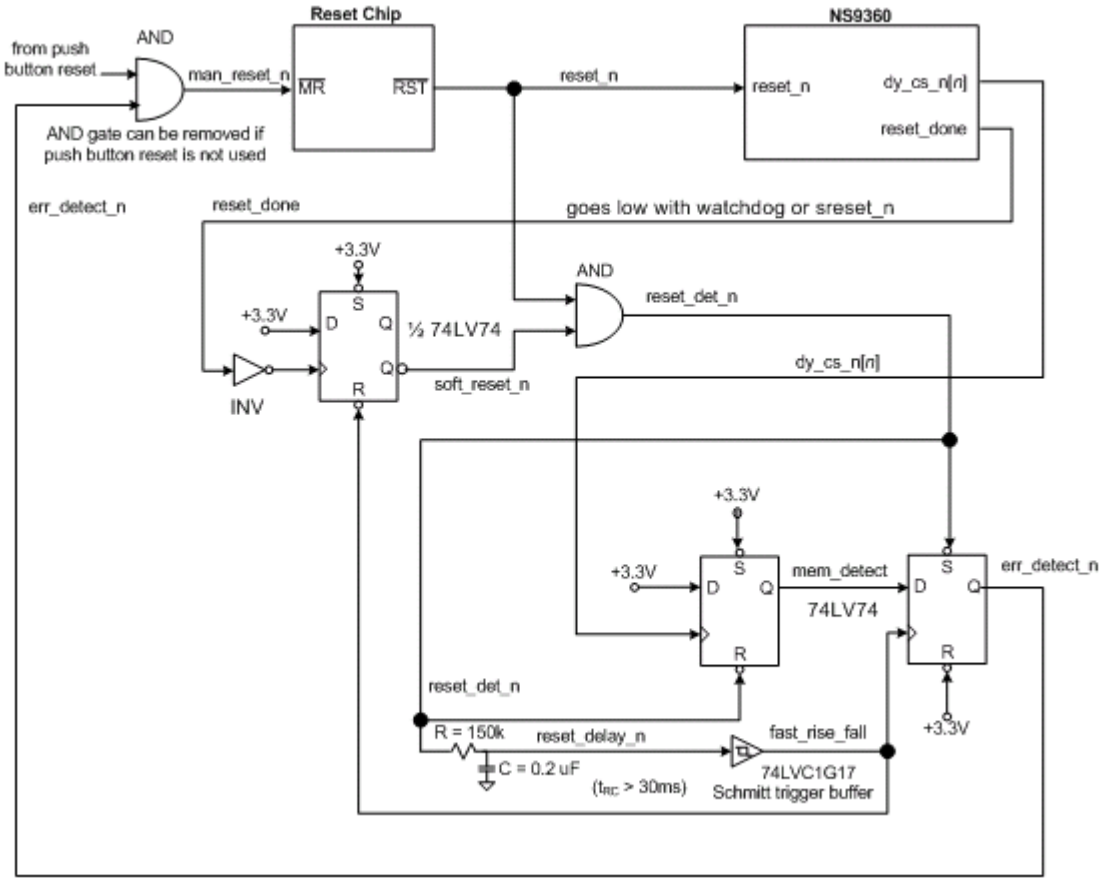


Errored Boot-up

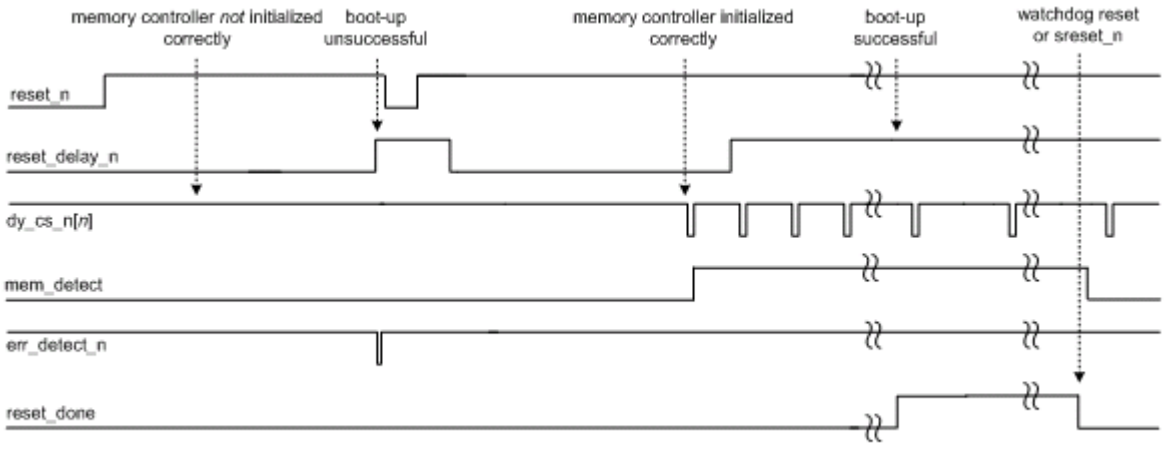
The board hardware workaround circuit that detects the unsuccessful boot-up is shown next. Digi recommends that you use a spare dynamic memory chip select in the workaround circuit.

If the endian bit is never changed (bit 3 in the Miscellaneous System Configuration register), the circuit fed by reset_done (inverter, flip-flop and AND gate) can be omitted. If the err_detect_n flip-flop has a Schmitt trigger clock input the Schmitt trigger buffer can also be omitted.

Note: The Schmitt trigger buffer is required when both the err_detect_n (clock) and soft_reset_n (reset) inputs are driven.



This timing diagram illustrates the recovery process:



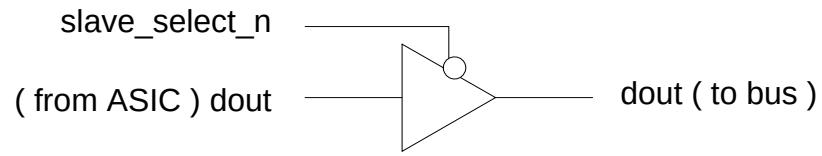
Errored Boot-up Recovery

SPI slave data output high impedance control

There is a problem that occurs in slave mode when there are other slaves on the SPI bus. When the slave select signal is de-asserted, the data output pin fails to go into a high impedance state and interferes with any other slave that is trying to drive the data signal.

Workaround: Do this for each SPI port. Externally buffer the data_out signal with a tri-state buffer and connect the select signal to the active low tri-state control pin. There should be a pullup resistor on the output of the buffer to prevent floating when no slaves are selected.

Example:



UART gap timer

The start bit of a new character may not be detected when the character or buffer gap timer expires. Framing, parity, or data corruption occur when a start bit is missed.

Software workaround: Three conditions have been identified for this erratum:

- Applications with a steady stream of receive data are not affected if the buffer gap timer is disabled.

- Applications where the gap between characters is fixed and the character gap timer period is configured to be less than the fixed period. The buffer gap timer must be disabled.
- Applications that have higher-level protocol error detection and recovery such as PPP can use both the buffer and character gap timers.

Hardware workaround:

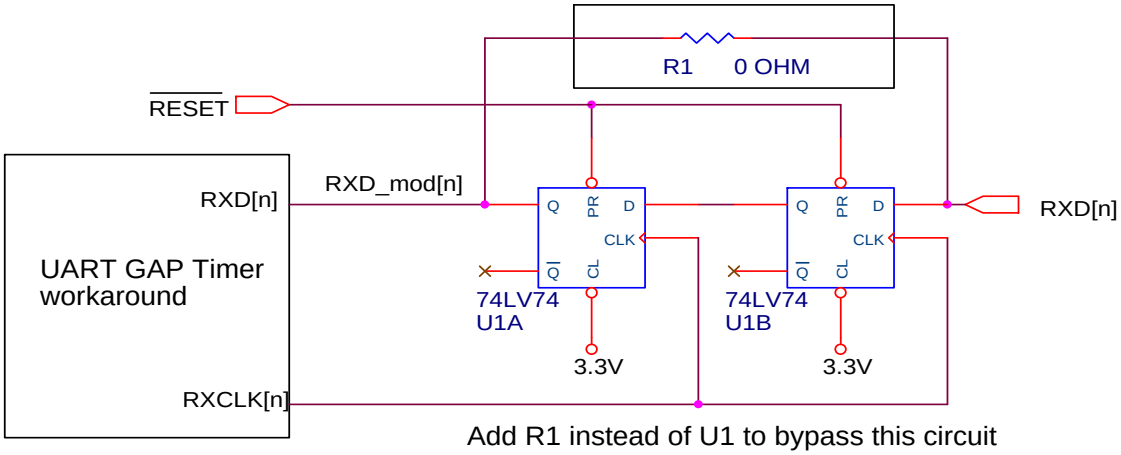
Note: The hardware workaround requires that you have installed the appropriate software patch found in the NETOS SW Toolkit (on the Web).

A hardware workaround eliminates the possibility of receiving a start bit when a character or buffer gap timer is expiring. The workaround drives the baud clock off-chip and synchronizes the incoming data with this clock. As a result, the buffer and character gap timers and the next start bit have a fixed and known relationship with each other.

Limitations:

- The TMODE bit in the Bit-rate register must be cleared.
- Baud rates are limited to those available in x16 mode
- Baud rates with a divisor of 0 are not possible for all CPU frequencies. This translates to a maximum baud rate of 460k in x16 mode.
- Baud rates with a divisor of 1 are not possible for CPU frequencies below 147MHz. This translates to a baud rate of 230k in x16 mode.
- The buffer and character gaps must be an even multiple of the sample clock period. For a 96000bps UART in x16 UART mode, the equation is:
- Timer increment `= ((1/9600) / 16) x 2 = 13.020us`
- One GPIO per UART is required to output the baud clock. The baud clock can be output on the RI pin on each UART. This function is controlled by the RXEXT bit (27) in each Serial Bit-rate register.
- If multiple UARTs are running at the same baud rate, one baud clock can be used for the multiple UARTs.

Workaround drawing



UART CTS-related transmit data errors

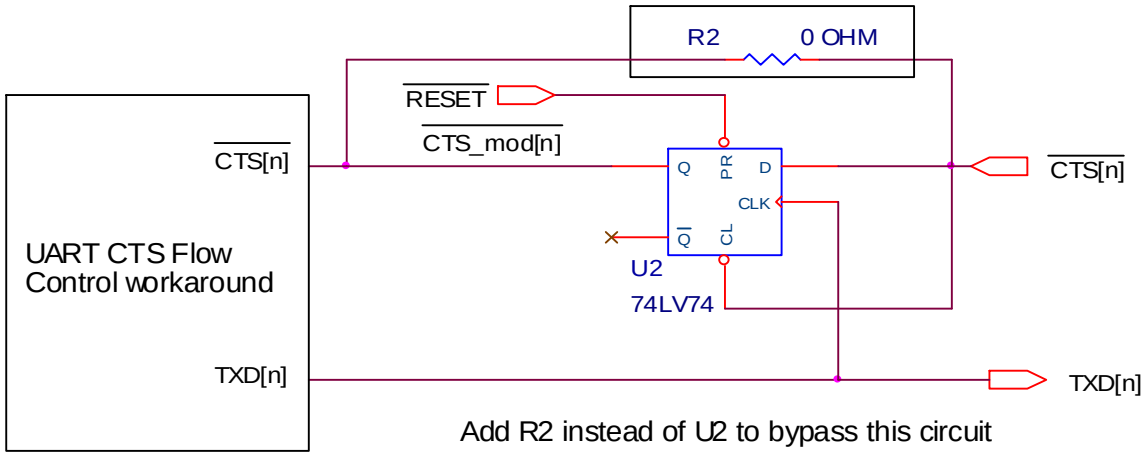
A problem occurs when the CTS flow control signal is de-asserted during the BCLK that begins processing a new character. This problem causes the previous character to be re-transmitted instead of getting the next character from the transmit FIFO.

Software workarounds:

- Modify these bits in Serial Channel Control register A:
  - Set the CTSTX bit (bit 23) to 0 to disable hardware-controlled CTSTX.
  - Set the ERXCTS bit (bit 4) to enable the software CTS signal change interrupt.
  - Update the serial transmit ISR to handle the CTS signal change.
- The maximum bytes in each DMA buffer descriptor is limited to 16 bytes.
- The maximum skid rate can be up to 16 characters.
- The serial monitor thread is changed to handle the missing CTS interrupt.

See the appropriate (6.0 or 6.3) NET+OS SW toolkit (on the Web) for the required software workarounds.

Hardware workarounds: For each UART, externally clock the CTS signal with the Txd_n signal to guarantee that CTS will not be seen de-asserting at the start of a character.



Ethernet receive data FIFO overflow (Ethernet receiver stall)

The Ethernet receiver intermittently locks up in 100 Mbps half-duplex applications due to an overflow in the RX data FIFO.

Workaround: Reset the RX Ethernet logic when an RX_OVFL_DATA interrupt is generated. Go to http://ftp1.digi.com/support/documentation/ and search for "ethernet_rx_lockup" to read the application note for instructions.

Linked Ethernet TX buffer descriptors do not work with late collisions

If the Ethernet transmitter locks up when a late collision occurs while transmitting an Ethernet packet consisting of multiple linked buffer descriptors, one of these situations occurs:

- The WRAP bit in the last entry of the embedded TXBD RAM is cleared.
- The WRAP bit is set in both the first and last entries in the TXBD RAM.

Workaround: Software keeps a shadow copy of the TXBD RAM flags in main memory, and updates the copy only when the CPU accesses the TXBD RAM. When the CPU receives an Ethernet TX ERROR interrupt of any kind (that is, the TXERR bit is set in the Ethernet Interrupt Status register), which includes a late collision, the CPU takes this action:

- Reads the TX Error Buffer Descriptor Pointer register (TXERBD) which, in the case of the logic error, points to the location that has the bad W (wrap) bit.
- Copies the flags from the shadow TXBD RAM to the real TXBD RAM starting at the location pointed to by TXERBD and ending at the first shadow location that has the L (last) bit set, indicating the last buffer descriptor for the packet. This corrects the W bit errors.

Digi International
World Headquarters
11001 Bren Road East
Minnetonka, MN 55343

Information in this document is subject to change without notice and does not represent a committment on the part of Digi International.

Digi provides this document "as is," without warranty of any kind, either expressed or implied, including, but not limited to, the implied warranties of, fitness or merchantability for a particular purpose. Digi may make improvements and/or changes in this manual or in the product(s) and/or the program(s) described in this manual at any time.

This product could include technical inaccuracies or typographical errors. Changes are made periodically to the information herein; these changes may be incorporated in new editions of the publication.

Online problem reporting: www.digi.com/support/eservice/login.jsp

Documentation updates: Digi occasionally provides documentation updates on the Web site (www.digiembedded.com/support).

Be aware that if you see differences between the documentation you received in your package and the documentation on the Web site, the Web site content is the latest version.

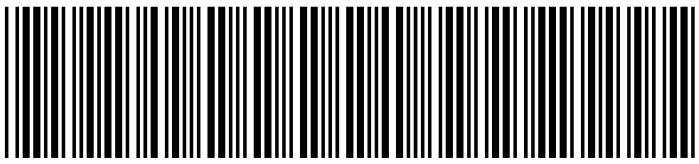
Support: To get help with a question or technical problem with this product, or to make comments and recommendations about our products or documentation, use the contact information listed in this table:

For Contact information: Technical support www.digiembedded.com/support

United States: +1 877 912-3444

Other locations: +1 952 912-3444

Fax: 952-912-4952



PN (1P): 90000677-88 F