NS9775B-0

Errata

90000527, Rev E

Release date: January 2006

Use in conjunction with:

NS9775 Hardware Reference, Rev. C

Part number: 90000524_C
Released: December 2004

If you are using an earlier version of the *NS9775 Hardware Reference*, see these documents:

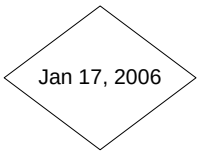
- *NS9775B-0 Errata, Rev. D* (December 2005)
- *NS9775 Errata, Rev. C* (August 2004)
- *NS9775 Errata, Rev. B* (June 2004)
- *NS9775 Errata, Rev. A* (March 2004)

Technical Support:

Phone: 1.877.912.3444

Web: techpubs@digi.com

- UART gap timer
- UART CTS-related transmit data errors
- USB OVR and USB PWR
- PCI arbiter senses false request
- Ethernet receive data FIFO overflow (Ethernet receiver stall)
- System PLL instability
- 1284 nibble ID negotiation problem
- Modem signals are inverted
- Serial port buffer GAP timer non-functional in PLL bypass mode
- RTS not asserted during transmit on serial channels A and D
- Linked Ethernet TX buffer descriptors do not work with late collisions



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 displayed.
- Applications that have higher-level protocol error detection and recovery such as PPP can use both the buffer and character gap timers.

Hardware workaround:

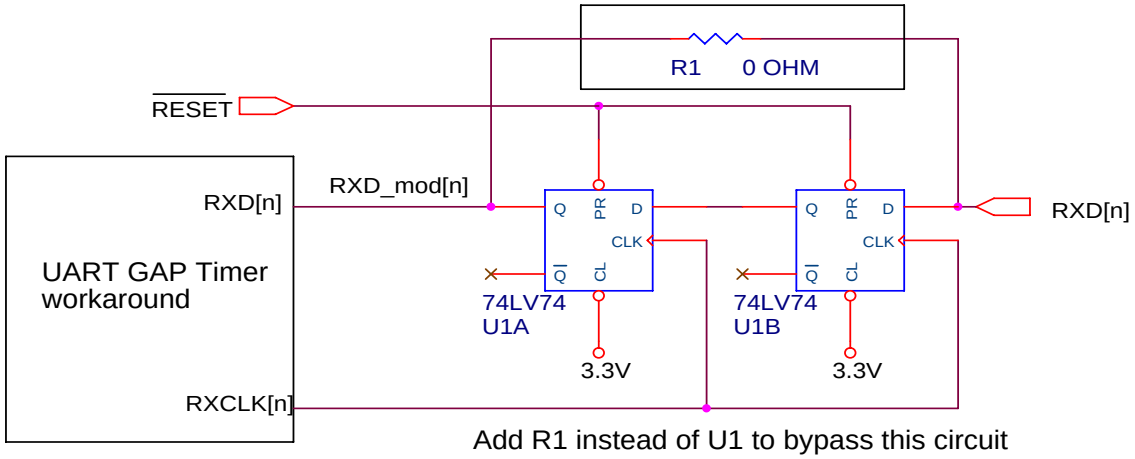
Note: The hardware workaround requires that you have installed the appropriate software patch found in the NETOS SW Toolkit (on the Web).

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.



Add R1 instead of U1 to bypass this circuit

Jan 17, 2006

UART CTS-related transmit data errors

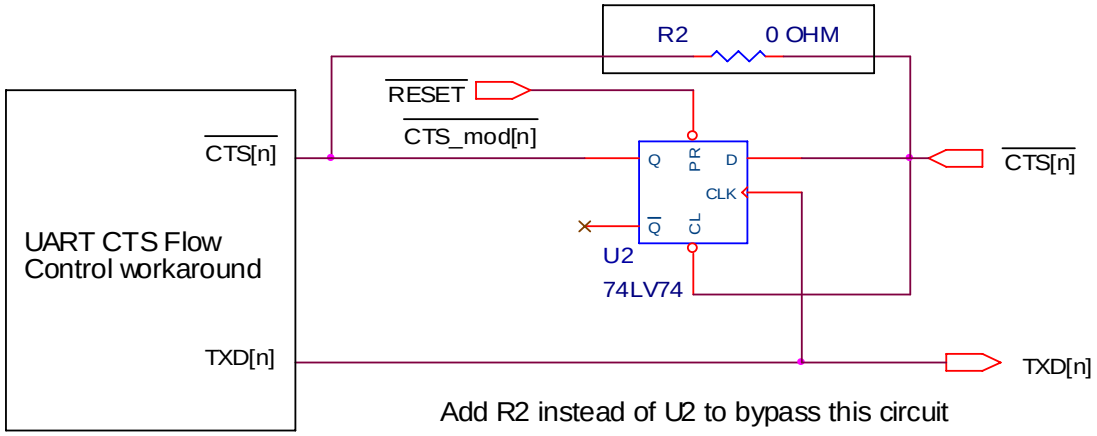
A problem occurs when the CTS flow control signal is de-asserted during the BCLK that begins processing a new character. This problem causes the previous character to be re-transmitted instead of getting the next character from the transmit FIFO.

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 workaround: 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.



Jan 17, 2006

USB OVR and USB PWR

The USB OVR (over current) input on gpio[16] is not shown in the GPIO MUX pinout tables. The polarity of this signal cannot be changed. It is true high, normally requiring an external inverter and noise filter to prevent false indications of over current.

This table shows the corrections to GPIO[16] in the pinout table:

Signal	Mode	Description
gpio[16]	00	Reserved output
	01	1284 nFault (peripheral driver, duplicate)
	02	Timer 11 (duplicate), USB OVR
	03 (default)	GPIO 16

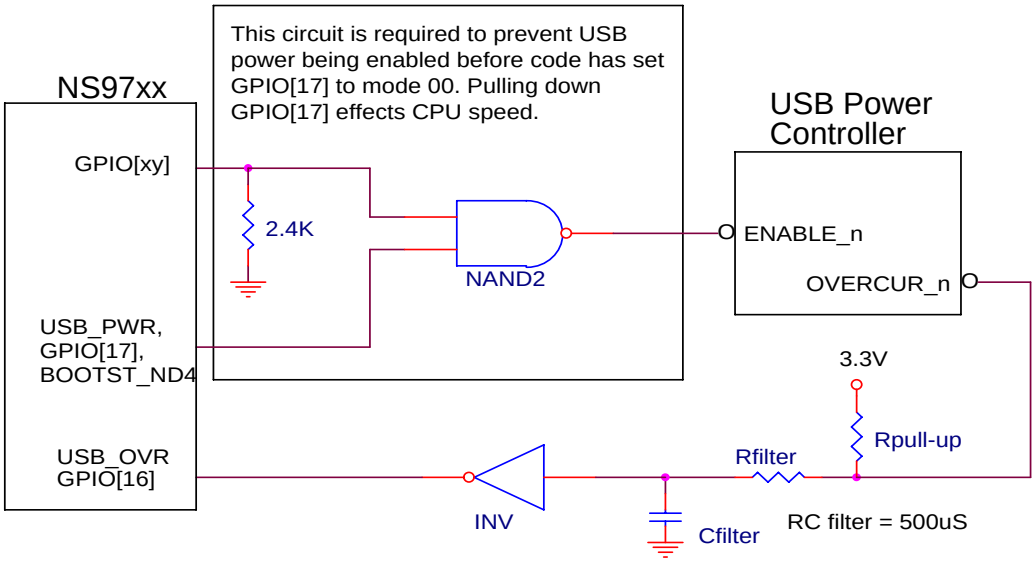
Updates are required to the BSP to change gpio[16] from mode 00 to mode 02. Timer 11 is disabled, allowing USB OVR to be recognized by the USP IP. Even though Timer 11 is disabled, the Timer clock select field must be set to 111-External pulse event to set GPIO[16] to an input.

If you are using USB Host, GPIO[16] *must be reserved* for the USB OVR function.

The USB PWR (power) control output requires additional external logic and GPIO to prevent USB power from being enabled from the time power is valid until the code has set the USB registers and selected gpio[17] to mode 00. The polarity of USB PWR is true high. Pulling down GPIO[17] to disable the USB power control changes the CPU bus speed because this pin is also bootstrap — ND4.

Designs that add the additional GPIO to prevent USB power-on during startup would require the BSP, as a final step, to set this GPIO to output a 1.

This drawing shows a recommended logic workaround:



Changed Jan 17, 2006

PCI arbiter senses false request

For complete information and workarounds for this issue, see the related application note at `http://www.netsilicon.com/support/appnotes.jsp`

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://www.netsilicon.com/support/appnotes.jsp to read the related Application note for instructions for doing this.

System PLL instability

The NS9775 system PLL shows frequency instability under various operating conditions, including operation from external system oscillators and crystals. The instability is eliminated when the NS9775 is operated in PLL bypass mode using an external 400 MHz system oscillator.

Workaround: Use external crystal oscillators.

1284 nibble ID negotiation problem

The NS9775 should set the Xflag (Select) line low at event #6 (see the IEEE 1284 standards specification). Instead, this signal is being driven high. The host uses this signal to determine whether the requested mode is supported. Therefore, the host incorrectly concludes that the NS9775 does not support nibble ID mode.

Workaround: None.

Modem signals are inverted

These serial signals — RTS, CTS, DSR, DTR, DI, and DCD — are inverted from conventional serial logic. As a standard, a 1 in the control or status register is an active state for the signal. Serial signals are active in the *space* condition, which is a logic low, or a positive voltage on a 232 line after passing through the line driver. The NS9775 has this standard reversed, so a 1 in a register matches with a logic high and a mark (negative voltage) condition on the 232 line.

Workaround: When used, the affected signals must be inverted externally.

Serial port buffer GAP timer non-functional in PLL bypass mode

Serial port buffer GAP timers do not function in PLL bypass mode.

Workaround: None.

RTS not asserted during transmit on serial channels A and D

The control signal that asserts "RTS-only" while transmitting is miswired for serial channels A and D, but is wired correctly for channels B and C. This means that RTS is not asserted properly while transmitting when the RTSTX control bit is set in Control Register B.

Workaround: None.

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 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 the first shadow location that has the LAST bit set, indicating the last buffer descriptor for the packet. This corrects the WRAP bit errors.

Digi International
11001 Bren Road East
Minnetonka, MN 55343 U.S.A.
United States: +1 877 912-3444
Other locations: +1 952 912-3444
Fax: +1 952 912-4960
www.digi.com/support
www.digi.com

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