

Multi-Drop Serial Device Driver

Windows NT 4.0

User Manual

***Written by:
Terje B. Nilsen***



***Experts in Hardware
&
Embedded Software Development***

Dwg. Revision A

NILSEN CONSULTING

7115 W. Tidwell, #102
Houston, Texas 77092
Voice: (713) 690-8120
Fax: (281) 552-9566
www.ncon.com
Email: tnilsen@ncon.com

September 15, 1999

1 PURPOSE OF DOCUMENT.....3

2 INSTALLATION NOTES.....3

3 DEFINITION OF MULTI-DROP MODE AS IMPLEMENTED IN SERIAL.SYS.....3

4 USING SERIAL.SYS FROM YOUR C OR C++ APPLICATION4

5 DOCUMENT REVISION HISTORY4

1 Purpose Of Document

This document is to define the additions that Nilsen Consulting, Inc. made to Windows NT 4.0's Serial.sys Device Driver in order to support Multi-Drop communications.

NOTE: these additions to serial.sys does NOT affect its normal behavior and therefore will not affect previous communications packages.

2 Installation Notes

There are two files included on the disk:

1. Serial.sys
2. Ntddser.h

Place the Serial.sys file into the following directory: **\\WinNT\System32\Drivers**

This will overwrite NT's normal Serial.sys. Optionally, place the Ntddser.h file in directory where your application can access it (for 'C' and C++ development.).

Reboot your PC in order to start the new Serial.sys (or use Device Manager to stop and restart Serial.sys).

NOTE: this installation will NOT affect the operation of any of your present Communications Applications.

3 Definition of Multi-Drop Mode as implemented in Serial.sys

Whenever you make use of the Multi-Drop parity feature within this device driver, the following applies:

1. The line control register is configured for Space parity, 8 data bits, 1 stop bit. All other settings are left as configured by User.
2. The RTS line will idle low. NOTE: set flow control to SERIAL_TRANSMIT_TOGGLE if RTS is needed for transceiver.
3. A nine-bit message is a series of sequential characters (bytes) transmitted, where the first character is sent with the parity bit stuck high ('1'), and the rest of the characters (bytes) have the parity bit stuck low ('0').
4. The device driver considers any single WriteFile access a 9-bit message. (I.e. make sure each entire message is sent to the device driver in one call to WriteFile).
5. The device driver will turn the RTS line high and wait 5 milliseconds before the first character starts transmission. This allows Multi-Drop network transceivers to settle before transmission starts.
6. The RTS line remains high during the entire message transmission.
7. The RTS returns low *immediately* following the complete transmission of the last character in the message.

8. All responses (i.e. received characters) are received using Space parity. This means if your slave node responds with its first transmitted character's parity as high ('1'), then the UART will flag this character with a Parity error.
9. When the port is closed (using CloseHandle) the driver will reset to SPACE Parity in order to avoid the chance of 'older' applications getting confused with a parity setting of 5 (i.e. Multi-Drop parity). Should you re-open the port, make sure to set parity back to Multi-Drop.

4 Using Serial.sys from your C or C++ Application

To make use of the Multi-Drop feature, follow these steps:

1. Add **#include "mmsystem.h"** to your application and call **timeBeginPeriod(1)**; as soon as your application starts (i.e. before opening the serial port). This will modify the NT timer to a one-millisecond base. This gives the device driver more responsive timers (and less timing jitter, especially on the RTS line).
2. Add **winmm.lib** as a library input file to your linker setup. This allows linking the above routine.
3. Open the serial port as you would normally under your application using the CreateFile function.
10. Configure the driver as you normally would using whatever method under Win32 you wish (e.g. IOCTLs, SetCommConfig, etc), but set the PARITY to MULTI_PARITY (a macro within the Ntddser.h routine which is the number 5). (Normal parity settings are NONE=0, ODD=1, EVEN=2, MARK=3, SPACE= 4, and now MULTI-DROP parity=5). NOTE: set flow control to SERIAL_TRANSMIT_TOGGLE if RTS is needed for transceiver on/off behavior.
4. Construct your 9-bit messages in accordance to your protocol, and send them using the Win32 WriteFile function. NOTE: make sure to pass the entire message in one access to WriteFile.
5. Use ReadFile to receive messages from your serial network nodes. Note: if you are required to detect the 9-th bit on responses, we recommend using the IOCTL_SERIAL_LSRMST_INSERT method, which is described by Microsoft in their help.

NOTE: a most frequent overlooked aspect of Win32 serial communications is the correct setting of the serial timeouts. Make sure these are set to fit your protocol.

5 DOCUMENT REVISION HISTORY

Revision A (September 15, 1999)

Document created (TBN).