DOS Device Driver

The Intellicon-NT960 DOS device driver provides an interface between the DOS operating system and an Intellicon-NT960 Host Adapter. Under DOS you can install a maximum of 128 ports using one NT960 Host Adapter.

Installation

To install the DOS device driver follow these procedures:

> ***Note:***
> *The diskettes may also include README files. Please examine these files for technical tips or release notes concerning installation and configuration of the device driver*

> ***Technical Tip:***
> *When you install the NT960 subsystem for the **first** time, you **must** run from DOS the* **ntload** *program with either:*
>
> 1. *The HEX file* **fdnis.hex** *(for NT960 Host adapters with up to 32 ports connected).*
>
> *OR*
>
> 2. *The HEX file* **fdnid.hex** *(for NT960 Host adapters with over 32 ports connected.*
>
> *You will find these files on the NT960 DOS Support diskette. You run this **before** installation of the NT960 DOS device driver. Please refer to the **ntload Program** section on page 3-1 of the **Software Installation** chapter*
>
> ***Examples:*** **ntload f=fdnis** *(up to 32 ports)*
> **ntload f=fdnid** *(over 32 ports)*

1. Insert the Intellicon-NT960 DOS device driver diskette into a floppy drive and copy the file **NT960.SYS** to your system's boot drive. You will find this file in the root directory of the DOS Support diskette.

> ***Example:***
> *To copy the files from floppy drive "A:" to the root directory on hard drive "C:" the syntax is:*
>
> **copy a:\NT960.SYS c:**

2. Add the following entry to your **CONFIG.SYS** file:

Technical Tip:
Please prevent the use of the NT960's base memory address segment by DOS memory management software. You do this by adding the following statement to your **CONFIG.SYS** *:*

device= c:\dos\emm386.exe x=[memory segment]

Where **emm386.exe** *is in your* **DOS** *subdirectory and* **[memory segment]** *equals the NT960 base memory segment (in hex).*

DEVICE=NT960.SYS A=nnnnn P=nnn N=nn I=nn D=name

Where:

A= The memory segment used for the dual ported memory. This will default to D0000 (hex) if not specified. Note that the dual ported memory only takes up a 8 KB footprint (i.e., you use the region D0000 to D1FFF when you select D0000 as the base address).
default is D0000

P= The port address used by the NT960 Host adapter. This defaults to 300 (hex).

N= The offset for "COM" ports. When using **NT960.SYS** along with standard serial ports, you must enter an offset where the Intellicon-NT960 ports begin. For example the setting **N=3** will start the NT960 ports at COM3. Default is **N=1**

I= The interrupt request line (IRQ) used by the Host Adapter and specified in decimal. The default is 10.

D= The prefix to be used in device names (maximum of 5 characters). The default is **"COM"**.

Notes:
1. *You require only one* **DEVICE=** *statement in the* **CONFIG.SYS** *for the Intellicon-NT960, even when using multiple Host Adapters.*
2. *A space is used as a delimiter between each parameter.*
3. *In some applications, the name* **COMn** *invokes special device handling (i.e., BASIC). The software will assume a standard serial port rather than a character device. To avoid this, you can use the* **D=** *option to change the device name prefix.*

Example:
Installing a single NT960 adapter with the following parameters:

memory segment: *D0000h*
port address: *300h*
port offset: *COM1*
interrupt - IRQ: 10
device name prefix: *COM*

The **DEVICE=** *statement for the* **CONFIG.SYS** *file should read:*

DEVICE=NT960.SYS a=D0000 p=300 n=1 i=10 d=COM

Example:
Installing one NT960 Host Adapter and one ACM/16 or ACM/Flex16 module in a system that has two serial ports (installed as COM1 and COM2). The **DEVICE=** *statement for the* **CONFIG.SYS** *file should read:*

DEVICE=NT960.SYS a=D0000 p=300 n=3 i=7 d=COM

Where:
memory segment is: *D0000h*
port address is: *300h*
port offset is: *3*
interrupt is: *7*
device name prefix is: *COM*

Note:
The two previously installed serial ports remain as COM1 and COM2, while the NT960 ports will be COM3 through to COM18.

Accessing Ports

There are two methods of accessing the extra serial ports on the Intellicon-NT960 subsystem.

- Character Device Interface
- INT 14h Functions

Character Device Interface

The simplest method of access is through file pointers (FCB or handle). The **NT960.SYS** driver provides a DOS character device interface that allows opening and reading or writing as a file named **COMn.**

> ***Technical tip:***
> *In some applications, the name* **COMn** *invokes special device handling (i.e., BASIC). The software will assume a standard serial port rather than a character device. To avoid this, you can use the* **d=** *option to change the device name prefix.*

In order to configure the extra serial ports on the Intellicon-NT960 subsystem, please refer to the **CTIMODE** section below.

CTIMODE

The **CTIMODE** command allows you to configure the extra serial ports on the Intellicon-NT960 subsystem. It provides the same functionality and defaults as the DOS **MODE** command with the following exceptions:

- Does not support redirection of LPTs
- Does not support time out retry parameter P
- Supports extra baud rates
- Supports setting of protocols

You will find the **CTIMODE.EXE** file in the root directory of the DOS Support diskette.

Syntax
Type in the following statement or add it to your **AUTOEXEC.BAT** file

CTIMODE COMn[:] [baud[,[parity] [,[databits] [,[stopbits][,[protocol]]]]]

Where **[protocol]** can be:

N= none

X[,nn[,mm]]= Xon-Xoff **nn** = XON, **mm** = XOFF in hex (default is 11h 13h)

C= CTS/RTS

P= Xon-Xoff for PC Term

Example:

CTIMODE COM4 9600, ODD, 8, 1, X

The above is an example of a **CTIMODE** *command statement configuring a NT960 serial port as:*

COMn = COM4
baud = 9600
parity = odd
databits = 8
stopbit = 1
protocol = XON/XOFF

Technical Tip:
By typing **CTIMODE COMn,** *where* **n** *is the port number, you can query the current settings of the port.*

INT 14h Functions

This section outlines all the INT 14h function calls of the NT960 DOS driver. The INT 14h function calls are a more powerful and complicated method of accessing and configuring the extra ports of a NT960 subsystem. The first four functions are identical to the standard BIOS functions for INT 14h. We provide an additional 13 function calls for increased functionality.

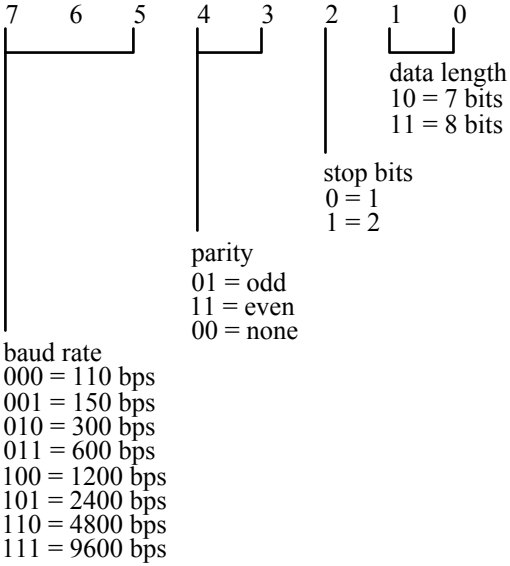
Technical Tip:
Please see the compressed file **SAMPLES.ZIP** *(in PKzip v.2.04g file format). This file contains the "C" source and executables of various sample programs using the INT 14h function calls. You will find the* **SAMPLES.ZIP** *file in the root directory of the DOS Support diskette.*

Function 0 - Initialize the port

Input

ah = 0
al = baud, parity, length
dx = communication channel number(i.e. 0 = COM1, 1 = COM2,)

Bit designations for al are:



Return Value

ah = port status (see **Function 3**)
al = modem status

Description

This function allows for the selection of traditional port parameters. For more complete port initialization see **Function 4** (i.e., baud rates above 9600 bps).

Function 1 - Output character

Input
ah = 01h
al = character to send
dx = communication port number

Return Values
ah = port status (bit 7 set if unable to send)
al = character to send

Description
This function attempts to send a character. The NT960 adapter buffers the data when it is busy. When it is free, the NT960 transfers the data from the buffer to its queue.

Function 2 - Receive character

Input
ah = 02h
dx = communication port number

Return Values
ah = port status (bit 7 set if timeout has occurred)
al = character received

Description
This function tries to get a character from a port. If an empty input buffer receives no character, then bit 7 of ah is set to indicate a timeout has occurred (182 ticks).

Note: *1 tick = 55 ms*

Function 3 - Return port status

Input

ah = 03h
dx = communication port number

Return Values

ah = port status
al = modem status

Description

This function returns the port status. Bit designations are as follows:

ah bit
0 = data ready
1 = data overrun
2 = parity error
3 = framing error
4 = break interrupt
5 = transmitter holding register empty
6 = transmitter shift register empty
7 = timeout

al bit
0 = change in CTS
1 = change in DSR
2 = change in RI
3 = change in CD
4 = CTS
5 = DSR
6 = RI
7 = CD

Function 4 - Extended port initialization

Input

ah = 04h
al = parity, stop bits, length (described in **Function 0**)
cx:bx = baud rate (cx contains highword)
dx = communication port number

Return Values

ah = 0 on success

Description

This function allows the setting of all Intellicon-NT960 supported baud rates, including the ones not supported by **Function 0**.

Example:

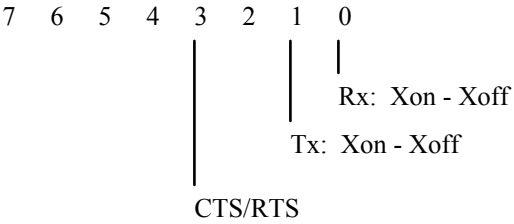
19200 bps: cx = 0, bx = 4B00h (19200 decimal = 0004B00 hex)
9600 bps: cx = 0, bx = 2580h (9600 decimal = 0002580 hex)

Function 5 - Set protocol

Input

ah = 05h
al = protocols (0 for none)
bl = XON character
bh = XOFF character
dx = communication port number

Bit designations for al are:



Return Values

ah = FFh on invalid protocol specification

Description

This function allows the user to set protocols for each port. If you specify the Xon-Xoff protocol and bx is non-zero, then bx becomes the specified XON/XOFF characters.

Function 6 - Identification

Input

ah = 06h
dx = communication port number

Return Values

ah = high bit set
al = highest function number supported

Description

This function allows you to identify the NT960 driver as an intelligent communication board interface.

Function 7 - Break modem connection

Input

ah = 07h
al = when bit 0 is set, bx contains break duration in ticks
when bit 0 is clear, break length will be 9 timer ticks
bx = number of timer ticks to hold break on line if al bit 0 is set (up to a maximum of 23 [1.275 seconds])
dx = communication port number

Return Values

None

Description

This function allows a program to send a **BREAK** on the communication line.

Note: *1 tick = 55 ms*

Function 8 - Nondestructive read

Input

ah = 08h
dx = communication port number

Return Values

al = next character in queue if zero flag is clear, else no characters are waiting

Description

This function allows the user to check the queue for the next character to read. This sets a zero flag if no characters are waiting.

Function 9 - Flush buffers

Input
ah = 09h
dx = communication port number

Return Values
None

Description
This function will flush(throw away the data) all the Intellicon-NT960 input and output buffers for the port.

Function 10 - Input queue check

Input
ah = 0Ah
dx = communication port number

Return Values
ax = number of characters waiting for input

Description
This function allows a program to determine the number of characters waiting in the input buffer.

Technical Tip:
You should minimize the use of this function. You should also use this function with ***Function 13 - Register a port****.*

Function 11 - Disable a port

Input
ah = 0Bh
dx = communication port number

Return Values
ah = FFh if an invalid port number was specified

Description
This function allows the user to disable a given port, freeing up its resources for use by other ports.

Function 12 - Read current parameters

Input

ah = 0Ch
dx = communication port number

Return Values

ah = FFh if an invalid port number was specified
al = parameters as described in **Functions 0, 4**
cx, bx = baud rate (cx high order)

Description

This function allows the user to query a port for its parameters.

Function 13 - Register a port

Input

ah = 0Dh
dx = communication port number

Return Values

es:bx = a pointer to a byte flag. This memory location is non-zero if there is a character waiting in the input queue.
ah = FFh if an invalid port number was specified

Description

This function returns a byte pointer to a flag indicating that a character is waiting for input. When data is waiting, the memory location pointed to is non zero. Implementation of polled operation is far faster than a nondestructive read by registering a port and testing the memory location pointed to. The reason for this is that you check the flag without the overhead of an INT 14h call.

Function 14 - String output

Input

ah = 0Eh
cx = maximum number of characters to transfer
dx = communication port number
es:bx = pointer to string
si = timeout length

Return Values

ax =number of characters transferred. If the zero flag is clear (0), the transfer was successful, else a timeout occurred during the transfer, (i.e., the output buffer is full).

Description

This function allows for the transfer for multiple characters via one "INT" call rather than a single character in one call, thereby reducing software interrupt overhead. If si is zero, the function uses an internal timeout. If si is non-zero, si is the number of timer ticks allowed to send one character. If any one character in the string takes more than si ticks to send, the function returns with the zero flag set and ax will contain the count of characters sent before timing out.

Function 15 - String input

Input

ah =	0Fh
cx =	maximum number of characters to read
dx =	communication port number
es:bx =	pointer to receiving buffer
si =	timeout length

Return Values

ax =number of characters transferred. If the zero flag becomes (1), a timeout occurred waiting for a character from the port.

Description

This function allows for the transfer for multiple characters via one "INT" call rather than a single character in one call, thereby reducing software interrupt overhead. If si is zero, the function uses an internal timeout. If si is non-zero, si is the number of timer ticks allowed to receive one character. If it takes more than si ticks to receive any one character in the string, the function returns with the zero flag set and ax will contain the count of characters received before timing out.

Function 16 - Reserved

Function 17 - Lower/raise DTR & RTS

Input

ah = 11h
al = when bit 0 is set, raise DTR
when bit 0 is reset, lower DTR
when bit 1 is set, raise RTS
when bit 1 is reset, lower RTS
dx = communication port number

Return Values

None

Description

This function allows the user to change the states of the RTS and DTR handshake lines.

Function 18 - Reserved

Function 19 - Read port metrics

Input

ah = 13h
dx = communication port number

Return Values

ax = number of characters waiting for input
bx = size of input buffer
cx = number of characters waiting for output
dx = size of output buffer

Description

This function allows the user to query a port's input and output buffer status. The values reported are NOT the total space available or used, but deal only with the buffers used for high speed reads or writes.

Example:

If a call to Function 19 yields the following values:

ax = 0010 (hex)
bx = 0200 (hex)
cx = 0100 (hex)
dx = 0300 (hex)

This means:

1. *You can read 10 hex (16 decimal) characters immediately without waiting on the NT960 driver or the NT960 Host Adapter.*
2. *There is space for 200 hex (512 decimal) characters in the input buffer.*
3. *You can pass 100 hex (256 decimal) waiting characters to the NT960's secondary buffers.*
4. *You can write 300 hex (768 decimal) characters to the port without waiting.*

Note:

Subtracting the cx value from the dx value yields the value for maximum size of a non waiting string write to a port. In the example above it means you can write 200 hex (512 decimal) more characters before more NT960 secondary buffer space becomes available.

NTSET

The **NTSET** command allows the user to set on or off the features of the Intellicon-NT960. You will find this file in the root directory of the DOS Support diskette. The command syntax is:

NTSET [COM]n[:] [options]

where **[options]** are:

+/-autorts	turns on or off the RTS signal. This feature is useful in multi-drop RS-422/485 applications. (default is off)
+/-autodsr	turns on or off the DSR signal. This feature is useful in half duplex RS-422/485 applications. (default is off)
rxtimeout = n	is the value when the receiver will timeout. The value **n** is in milliseconds (default is 5).

Technical Tip
If you want the fastest response time you should set the value low(such as 1), while the most efficient setting is a high setting (such as 300).

Examples:
The first example shows Intellicon-NT960 port COM3 set with autorts and autodsr on for a half duplex, point to point or multi-drop RS-422/485 application:

NTSET COM3: +autorts +autodsr

The second example shows Intellicon-NT960 port COM3 set with a timeout value of 50 for RxD

NTSET COM3: rxtimeout=50

Note:
You will display the current settings if you give no arguments to **NTSET***, other than the com port number.*

Utilities

ntdiag

ntdiag is a utility that allows you to examine various aspects of the NT960 subsystem under DOS. With **ntdiag**, you can perform tests such as:

- *NT960 Host Adapter tests*
  - PC dual ported RAM test
  - NT960 interrupt test
  - PC interrupt test
  - NT960 memory test
  - Flash memory test
  - NT960 dual ported RAM test
- *ACM/16 and ACM/Flex16 tests:*
  - Active port test
  - ACM/16 and ACM/Flex16 interrupt test
  - ACM/16 and ACM/Flex16 address switch test
  - Loopback tests

For more information on **ntdiag**, please refer to the file **ntdiag.txt** in the root directory of the DOS Support diskette.

Error Messages

NT960 Error! Bad Reason = nn

The NT960 reports an interrupt condition that is invalid. Causes may be a hardware conflict or the Intel i960 processor is malfunctioning.

NT960 Startup Error! Dual Ported RAM at *nnnn* not responding...Check for memory conflict

At startup the NT960 did not respond to the startup sequence correctly. Possible causes include: incorrect loading of NT960 firmware; incorrect setting of the adapter's memory segment (***nnnn*** represents the memory segment setting); or memory/IO port conflicts.

NT960 Startup Error! Card not responding

At startup the NT960 did not respond. Possible causes include: incorrect I/O port setting; and memory conflicts.

NT960 interrupt not responding! Check board settings
At startup an interrupt did not occur. Possible causes are: IRQ conflicts; or I/O port conflicts.

NT960 Startup Error # nn
The NT960 firmware is not functioning correctly. Possible causes include: you have no ACM/16 or ACM/Flex16 external modules connected to the Host Adapter; the NT960 firmware loaded improperly; or a hardware malfunction occurred.

Invalid Option *ccc*
Invalid option ***ccc*** specified in the **DEVICE= NT960.SYS** statement. Syntax error.

Invalid Port Address *nnn*
Invalid port address ***nnn*** specified with the **p=** parameter in the **DEVICE= NT960.SYS** statement.

Device name too long
Too many characters used in the **d=** parameter in the **DEVICE= NT960.SYS** statement. You can use a maximum of five characters.

Invalid IRQ *nn*
Invalid interrupt number ***nn*** specified with the **i=** parameter in the **DEVICE= NT960.SYS** statement.