Plug and Play External COM Device Specification

Version 1.00

February 28, 1995

Revision History

Revision Code, Issue Date	Comments
0.80, 12/23/93	1st Draft, edited from PnP COM Enumerator Specification 0.8
0.82, 1/10/94	Second Draft, simplified COM ID specifications
0.83, 1/19/94	Add EISA references, 1st public version
0.84	Add patent disclaimer, etc
0.86, 2/16/94	Revise as per 1st comments: a) two phase enumeration: simple mice and modems b) Increase timer tolerances to 35 ms c) fix format of Product ID and Serial Number d) add provisions for multiple ID sets, w/example e) Add checksum to optional IDs f) revise Patent Disclaimer as per MS legal g) retitle to External COM Device, Draft Specification
0.86a	a) fix assorted typos, particularly discrepancies in description of PnP ID strings. b) modified definition of checksum.
0.87, 4/28/94	Define Timer T4 in 2.1.6, to limit the time the enumerator will wait for the complete PnP ID string.
0.90, 6/6/94	a) reverse order of enumeration phases: - revise Figure 1, reverse 2.1.2 and 2.1.4 b) increase T4 timeout c) define T5 per-character timeout d) clearup conflict in multiple ID delimiter character (;) e) prohibit BeginPnP and EndPnP characters within IDs f) allow <CR><LF> characters within IDs g) various other italicized notes included. h) correct Table A-1 Title in TOC i) change byte order of checksum to match rest of ID
0.92, 6/18/94	a) add Annex C, to list defined Chicago Class names b) correction: comma character is 0x2C, not 0x27.
0.93, 7/8/94	a) add checksum examples b) add note on * in PNP compatible IDs
0.94, 7/26/94	a) add notes about problems with UPS devices b) add acknowledgements c) add note prohibiting use of invented class names
0.95, 8/19/94	a) corrections to handle some modems, including new 2.1.2 to check for any device, add Verify Disconnect b) renumber timers c) corrections to sample responses, Tables 4 and 5 d) implementation notes in Annex A.4 on &D3 issues
0.96, 9/7/94	a) remove asterisks from compatible IDs. b) revise Abstract c) revise descriptions of Enumerator internal behavior d) simplify Table 5
0.99, 10/20/94	a) added TABLET Class name in Table C-1. b) release (?)
0.99C, 1/3/95	a) revised Table C-1
0.99D, 2/17/95	a) deleted “Another ID capability” b) deleted Table 5 b) added modem control standard references

Patent Disclaimer

Neither Microsoft nor Hayes make any representation or warranty regarding this specification or any product or item developed based on this specification. Microsoft and Hayes disclaim all express and implied warranties, including but not limited to the implied warranties of merchantability, fitness for a particular purpose and freedom from infringement. Without limiting the generality of the foregoing, neither Microsoft nor Hayes make any warranty of any kind that any item developed based on this specification, or any portion of it, will not infringe any copyright, patent, trade secret or other intellectual property right of any person or entity in any country. It is your responsibility to seek licenses for such intellectual property rights where appropriate. Neither Microsoft nor Hayes shall be liable for any damages arising out of or in connection with the use of this specification, including liability for lost profit, business interruption, or any other damages whatsoever. Some states do not allow the exclusion or limitation of liability for consequential or incidental damages; the above limitation may not apply to you.

Acknowledgements

In addition to employees of Hayes Microcomputer Products and Microsoft Corporation, we received useful contributions from the following organizations: Logitech, Black Diamond, Kensington, APC, and Tripplite.

Table of Contents

List of Tables & Figures

ABSTRACT

This specification presents a mechanism to provide automatic configuration capability to peripheral devices that connect to a PC using Asynchronous Serial Data Interchange on standard serial ports, commonly known as COM ports. This enables full Plug and Play for the PC system, including external serial peripherals, referred to here as COM Devices.

The essential elements of Plug and Play COM are:

- Detect attachment of serial devices
- Identify the device, and notify the operating system of its arrival
- Detect detachment of serial devices, and notify the operating system of its removal

----

1. Introduction

The PC system includes devices installed within the PC system unit, and with external peripherals. The most common external PC peripherals are those connected to printer ports or serial ports. This document is concerned with detecting and identifying devices connected to the serial ports, also known as COM ports. COM ports are the common means for connecting with COM port devices, such as modems, pointing devices like mice, serial printers and other bi-directional devices.

This specification defines mechanisms that each Plug and Play COM device must implement to support identification, which in turn supports automatic configuration of the entire PC system without end-user intervention.

1.1. Goals

The following are the architectural goals to support full Plug-and-Play of COM devices:

1. Focus on ease-of-use for the end-user
2. Maintain backward and forward compatibility as follows:
   - Plug and Play COM devices will electrically and functionally inter-operate with standard CCITT V.24, EIA/TIA-232-E, EIA/TIA-574 and equivalent serial ports and with existing software for compatible serial devices.
   - Any software that does Plug and Play COM Enumeration will detect Plug and Play compatible COM devices, but will not damage, incapacitate or confuse older non-Plug and Play COM devices.

1.2. Device Implementation Considerations

The solution of the COM device identification problem addresses major concerns of end-users, system integrators, and operating system vendors. It also presents an excellent business opportunity for providing measurable value and differentiation in the short term, by reducing the user difficulties and associated technical support costs.

There are two ways that serial device manufacturers can implement this specification. For simple hardware based devices (e.g. serial mice) the hardware (e.g. ASICs) will need revision. For intelligent serial devices (e.g. modems and printers) these changes can be implemented by controller code firmware changes. However, care has to be taken in the controller code to avoid false-positive (erroneous) detection of the COM Enumerator (confusing the application software with unexpected PnP ID) or failure to detect the COM Enumerator.

Note that for devices that are logically serial but are installed in a PC system bus, particularly ISA, EISA, MCA (MicroChannel Architecture) or PCMCIA modems, the requirements of the appropriate bus Plug and Play specification must be met for Plug and Play compatibility. Otherwise, important bus resources (address space, Interrupts, DMA, etc) cannot be allocated by the Plug and Play system software, and resource collisions may result, negating the value of Plug and Play.

However, if a bus based COM device does meet the requirements of ISA Plug and Play, then the requirements of COM Plug and Play are optional. For example, an ISA-bus internal modem must support ISA Plug and Play; if it does, COM Plug and Play is not required, although it might provide useful additional information.

1.3. References

Plug and Play ISA Specification, Microsoft 1993
Plug and Play BIOS Specification, Microsoft 1993
Plug and Play PCMCIA Specification, Microsoft 1993

ITU (CCITT) V.24, List of definitions for interchange circuits between data terminal equipment and data circuit terminating equipment.
EIA/TIA-232-E Interface between data terminal equipment and data communication equipment employing serial binary data interchange on unbalanced circuits.
EIA/TIA-574 9-position Non-synchronous Interface Between Data Terminal Equipment and Data Circuit-terminating Equipment Employing Serial Binary Data Interchange
EIA/TIA-578 Asynchronous Facsimile DCE Control Standard - Service Class 1
TIA/EIA-592 Asynchronous Facsimile DCE Control Standard - Service Class 2
TIA/EIA-602 Serial Asynchronous Automatic Dialing and Control
TIA IS-131 Extensions to Serial Asynchronous Dialing and Control
ITU T.class1 Asynchronous Facsimile DCE Control Standard - Service Class 1
ITU T.class2 Asynchronous Facsimile DCE Control Standard - Service Class 2
ITU V.25ter Serial Asynchronous Automatic Dialing and Control
PCCA STD-101 Data Transmission Systems and Equipment - Serial Asunchronous Automatic Dialing and Control for Character Mode DCE on Wireless Data Services.

1.4. COM Port Circuit Definitions

Several common serial circuits will be referenced in this specification. These serial circuits are described in a number of public standards (see 1.4) and implemented in several types of physical ports. These are defined here with respect to the following common serial ports:

- EIA/TIA-232-E (current version of EIA RS-232, DB25 connector)
- EIA/TIA-574 (common "IBM AT" style, DB9 serial port)
- CCITT V.24 (equivalent of EIA/TIA-232-E)

Since these interfaces were originally developed for modems, the names reflect communications. DTE = Data Terminal Equipment (e.g. PC); DCE = Data Circuit-terminating Equipment (e.g. modem).

Table 1 - Referenced Standard Serial Port Circuits

name	V.24	232E	574	source	Description
COM	102	7	5	both	signal common (ground)
TXD	103	2	3	PC	transmit serial data
RXD	104	3	2	device	received serial data
RTS or RFR	105 or 133	4	7	PC	Request to Send or Ready to Receive
CTS	106	5	8	device	Clear to Send
DSR	107	6	6	device	DCE ready (e.g. COM device)
DTR	108/2	20	4	PC	DTE ready (e.g. PC or terminal)
RLSD	109	8	1	device	Carrier detect
RI	125	22	9	device	Ring Indication

2. Detection Method

This section will describe the method which may be used to detect and identify COM peripheral devices. Section 2.1 specifies a software “COM Enumerator,” which runs on a computer to detect Plug and Play COM devices. Note that this is a sample description which will vary in actual implementations; section 2.2 specifies the COM ID string serial data format. (The COM ID character strings are specified in section 3.) It also includes sample COM device implementations (2.3-2.5). Tutorial information about the basis for the design is contained in Annex B.

2.1. COM Enumerator Detection

The COM Enumerator software shall do the following things visible on the COM port:

- Initialize the COM port, with DTR ON , RTS OFF, and TXD set to Mark Idle.
- Wait for DSR=1 to indicate presence of COM device.
- Stimulate the COM port control leads in a defined manner.
- Collect the ID information if available.
- (Optional) Monitor the port when idle to detect device attach/detach.

The COM Enumerator will make two attempts to elicit a PnP ID:

- First, it will try a means with a DTR-RTS time signature, so that devices like modems (which have traditional uses for DTR and RTS) are not confused.

- If the first try does not generate a reponse, it will assume it is a device that will begin to respond within 200ms of being powered (e.g. COM port powered mice, non-modems).

The entire interval can be drawn as follows:

Figure 1 - Example COM Enumerator External Timing Diagram

```
interval---->|check|<---1st--->|<1st>|<2nd>|<2nd>|<--Idle----
             | dev |   Setup   | Wait|Setup| Wait|
             |     |     |     |     |     |     |
time         |  T1 |  T2 |  T3 |  T4 |  T2 |  T4 |
             ______       ___________       _____________
DTR(108)  XXX      \_____/           \_____/
             |     |     |     |     |     |     |
                                ______      ______
RTS(105)  XXX__________________/      \_____/      \_______
             |     |           |     |     |
reference    |2.1.2|   2.1.3   |2.1.4|2.1.5|2.1.6|
```

T1: minimum interval to hold DTR high while waiting for DSR
T2: minimum interval for external device to power down or detect the port state
T3: DTR-RTS enumerator signature delay
T4: maximum interval to wait for DSR and/or first received character
T5: PNP COM ID per/character timeout (not shown)
T6: PNP COM ID EndPNP timeout (not shown)
T7: Disconnect Verification timout (not shown)
Note: Figure 1 is drawn with characters to survive translation to plain ASCII.

2.1.1. COM port initialization, check for port availible

- Attempt to acquire the port via defined operating system services.
- If the port is busy (e.g. the modem or mouse is in use), don't enumerate this port.
- If the port IRQ is in use by another device driver, don't enumerate this port.

2.1.2. COM port initialization, check for device enumerate

- leave TXD = mark idle.
- set DTR=1, set RTS=0, start T1 = 200 ms (+/- 35 ms tolerance).
- If T1 expires and DSR=0, go to Disconnect Idle.
- else, go to Com port Setup, 1st phase.

2.1.3. COM port Setup, 1st phase

- Set the serial port for 1200 bit/s, 7 data bits, no parity, one stop bit.
- set DTR=0, set RTS=0, start T2 = 200 ms (+/- 35 ms tolerance).
- When T2 expires, set DTR=1, start T3 = 200 ms (+/- 35 ms tolerance).
- When T3 expires go to Wait for response, 1st phase.

2.1.4. Wait for response, 1st phase

- set RTS=1, start T4 = 200 ms (+/- 35 ms tolerance).
- If any character received, go to Collect PnP COM device ID(s).
- If T4 expires and no character received, go to COM Port Setup, 2nd Phase.

2.1.5. COM port Setup, 2nd phase

- Set DTR=0, RTS=0, start T2 200ms (+/- 35 ms tolerance).
- When T2 expires go to Wait for response, 2nd phase

2.1.6. Wait for response, 2nd phase

- set DTR=1 and RTS=1, start T4 = 200 ms (+/- 35 ms tolerance).
- If any character received, go to Collect PnP COM device ID(s)
- If T4 expires and DSR=0, go to Verify Disconnect
- If T4 expires and DSR=1, go to Connect Idle

2.1.7. Collect PnP COM device ID(s)

- Set T5 = 200ms (-0ms/+40ms), for per-character timeout.
- Set T6= 2.2 seconds; (256 char x 10bits/char)/(1200 bit/s) = 2.13 seconds.
- Receive and buffer any characters (see section 3.3 and section 4). Each time a new character is received, reset T5=200ms.
- If bit errors or framing errors are detected after the first character go to Connect Idle.
- If T5 expires after the first character, check for valid PnP ID string. If valid notify appropriate operating system service of the new device. If invalid go to Connect Idle.
- If **End ID** character detected, check for valid PnP ID string. If valid notify appropriate operating system service of the new device. If invalid go to Connect Idle.
- If T4 expires without **Begin ID** character detection go to Connect Idle.
- If T6 expires without **End ID** character detection go to Connect Idle.
- If DSR=0, go to Verify Disconnect.

2.1.8. Verify Disconnect

This step verifies that the device has actually been removed. This accounts for modems and other devices that lower DSR briefly when DTR drops in order to reset, etc. The enumerator assumes the device is not present if DSR is low after a timeout.

- Set DTR=1; set RTS=0.
- Set T7 = 5 seconds.
- When T7 expires, if DSR=1 go to Disconnect Idle.
- IF T7 expires and DSR=0, go to Disconnect Idle.

2.1.9. Connect Idle

- Set DTR=1; set RTS=0.
- Set the serial port for 300 bit/s, 7 data bits, No parity, one stop bit.
- Wait (forever): If DSR=0, notify appropriate operating system service of device removal, go to Disconnect Idle.
- (Optional) Monitor port for events caused by miscellaneous non-Plug and Play devices, e.g. GIDEI input devices (not documented here).

NOTE: GIDEI = General Input Device Emulating Interface. These devices are used to provide accessibility for some computer users with disabilities.

2.1.10. Disconnect Idle

This step is optional, if the enumerator software is going to monitor idle ports for dynamic device removal, or attempt to detect non-Plug and Play devices.

- Notify appropriate operating system service of device removal, if a device was previously present.
- Set DTR=1; set RTS=0.
- Set the serial port for 300 bit/s, 7 data bits, No parity, one stop bit.
- Wait (forever): If DSR=1, go to COM port Setup, first phase.
- (Optional) Monitor port for events caused by miscellaneous non-Plug and Play devices.

2.2. COM Device ID Serial Data Format

All Plug and Play compatible devices shall report device ID at 1200 bit/s second. (This is an artifact of common Serial Mice.) The data format shall be compatible with a receiver set for a nine bit frame: one start bit, 7 data bits (LSB first) and one stop bit. Example compatible formats:

- nine bit frame: one start bit, 7 data bits, no parity, one stop bit
- ten bit frame: one start bit, 7 data bits, no parity, two stop bits
- ten bit frame: one start bit, 7 data bits, Mark parity, one stop bit
- ten bit frame: one start bit, 8 data bits with MSB set to 1; one stop bit.

The Plug and Play ID string formats are defined in section 3.

2.3. COM Enumerator Recognition by Serial Mice

This is a sample implementation. This design requires the mouse to treat the RTS lead as a reset lead, with mouse operation enabled by RTS=1.

a) Echo DTR to DSR, always, in hardware.
   On Power-up (from DTR=1 and TXD=Mark)

c) If RTS=0, wait (forever) for RTS=1

d) If RTS=1, go send COM ID (e.g. Table 3)

e) Go be a mouse

f) If RTS=0, go back to state (c).

2.4. COM Enumerator Recognition by Modems

This is a sample implementation. This contains two departures for modems:

- The modem treats (DTR=0 & RTS=0) while IDLE as a special case, regardless of the state of an modem commands that condition the interpretation of the DTR lead (V.24 circuit 108/1 or 108/2) and the RTS lead (V.24 circuit 105 or 133).

- The modem uses the time signature between DTR=1 and RTS=1 to detect the COM Enumerator and distinguish from all other applications.

After the modem finishes the detection process, with success or failure, it should "go be a modem" and use the relevant command settings (e.g. TIA-602, etc) to condition susequent responses to DTR and RTS.

Sample Modem COM Enumerator Detection States

a) on power up, set DSR=1

IDLE state (b-e)

b) check for new commands, Ringing, check RTS and DTR.

c) If AT command received, go be a modem.

d) If Ringing detected, report event at default speed and go be a modem.

e) if (DTR=0 & RTS=0) (unplugged or computer off or computer rebooted)
and IDLE (e.g. not in an active phone call)
wait (forever) for DTR=1

COM Enumerator detection (f-j)

f) IF DTR=1, start T1=150ms, T2=250ms and check RTS (h)

g) IF DTR=0, go back to IDLE (b)

h) IF RTS=1 and T1 not expired, quit - go to IDLE (b)

i) IF RTS=1 and T1 expired and T2 not expired
go send COM ID (e.g. Table 4)
go to IDLE (b)

j) IF RTS=0 and T2 expired, quit - go to IDLE (b)

2.5. COM Enumerator Recognition by Other COM Devices

Serial devices other than modems could respond to the COM Enumerator with a simple state machine, particularly if they don't have the modem problem of using DTR and RTS for other meanings. In this case, the timing measurement can be omitted.

a) on power up, set DSR=1

IDLE state (b-e)

b) check for new commands, external events, internal events, RTS and DTR.

c) If PC commands received, go execute commands.

d) If events occur, process events (reports, actions, etc).

e) if (DTR=0 & RTS=0) (unplugged or computer off or computer rebooted)
and IDLE (e.g. not executing a command)
wait (forever) for DTR=1

COM Enumerator detection (f-h)

f) IF DTR=1, check RTS (h)

g) IF DTR=0, go back to IDLE (b)

h) IF RTS=1, go send COM ID (e.g. Table 4)
go to IDLE (b)

3. Identification Methods

3.1. COM Device ID Fields

A Plug and Play COM Device shall report ID information as defined in Table 2. The first field is reserved for old serial mice and other devices that spontaneously generate characters on power-up. The next four fields and the last field are mandatory, and all other fields are optional. If optional fields are used, the Checksum field is required.

For compatibility with old serial mouse drivers, all mouse-compatible pointing devices must restrict themselves to a 6 bit character set, for all fields except the Mouse ID. Therefore, all old-mouse-compatible strings are limited to values of 0x00 to 0x3F; character strings are ASCII codes from 0x20 to 0x5F, offset by subtracting 0x20. This is indicated by the Begin PnP ID.

Table 2, Plug and Play COM Device ID Fields

Field Name	size	required	Short Description
Other ID	<17	no	reserved for short non-PnP ID (e.g. "0x4D") May be saved or ignored by the COM Enumerator
Begin PnP	1	yes	begin PnP ID. This is "(": either 0x28 or 0x08
PnP Rev	2	yes	Plug and Play revision (e.g. 0x00,01)
EISA ID	3	yes	EISA determined unique Mfr Identifier
Product ID	4	yes	Mfr determined unique Product Identifier
Extend	1	no	"\": either 0x5C or 0x3C. see note below.
Serial Number	8	no	optional device serial number
Extend	1	no	
Class Name	<33	no	Plug and Play Class Identifier (Annex C)
Extend	1	no	
Driver ID	<41	no	Compatible Device IDs
Extend	1	no	
User Name	<41	no	end-user legible Product Description
Checksum	2	yes, if any optional fields	8 bit arithmetic checksum of all characters from Begin PnP to End PnP inclusive, exclusive of the checksum bytes themselves, represented as a two character hexadecimal number
End PnP	1	yes	End PnP ID. This is ")": either 0x29 or 0x09

Note 1: for all optional fields, the character string begins with the Extend character "\". If an optional field is not present but subsequent fields are, that field shall be represented by a single "\" character. "\" is coded as 0x3C (6-bit characters, e.g. if this is a mouse-compatible pointing device) or 0x5C (7-bit ASCII, all other devices). If the subsequent fields are not provided, then the "\" characters may be omitted (e.g. Table 5).

Note 2: The entire length of the Plug and Play ID string and Other ID string, including all fields and delimiters, shall not exceed 256 characters. This is to minimize delays in system boot-up time.

Note 3: Values for the PNP ID fields, enclosed within the BeginPnP and EndPnP characters, should be static for a given device.

3.2. Definitions

Other ID: the first field, which shall not exceed 16 characters, may contain an optional ID. An example usage is for serial mice, which would ordinarily generate one or more characters in response to the COM Enumerator. This may be saved or ignored by the enumerator. For new Plug and Play COM pointing devices, this can be a single character (e.g. "M", 0x4D, for Mice); other devices should omit this field. Warning: If either **Begin PnP** or **End PnP** characters (0x08 or 0x28, 0x09 or 0x29) are included, the Plug and Play Enumerator will be confused.

This field may include 7-bit <CR> and <LF> characters for human legibility.

Begin PnP: This marks the beginning of the Plug and Play ID set. This is the open parentheses character, 0x28 (7-bit IDs), or the offset value 0x08 (6-bit IDs). The COM Enumerator will look for a matching **End PnP** character (see below).

PnP Rev: the two byte (12 bit) revision code for the Plug and Play COM specification this device conforms to. The twelve bits are extracted from [first byte: bits 5-0] concatenated with [second byte: bits 5-0]. This number divided by 100 decimal yields the version number, ranging from 0.00 (0x00,00) through 1.0 (0x01,24) to 40.95 (0x3F,3F). See examples in Tables 4-6. Note that PnP Rev codes that contain 0x09 or 0x29 will not be used. This would preclude "1.05" which would be coded as 0x01, 0x29, or "1.37", which would be coded 0x02,0x09.

EISA Mfr ID: Each device manufacturer must have a unique three character EISA identifier. Plug and Play compatible COM devices shall report this ID here. If **Begin PnP** is 0x08 (6-bit), then each character is offset by subtracting 0x20 from ASCII.

To request the form to use for obtaining an EISA Mfr ID, FAX or call BCPR Services, PO Box 11137, Spring, TX 77391-1137, T:713-251-4770, F:713-251-4832. There is no charge as of this writing.

Product ID: Plug and Play compatible COM devices shall report a unique product ID here, defined as a 16-bit value in 4-character hexadecimal code, with first character representing the most significant 4 bits. (See PnP ISA Specification.) These hexadecimal characters are offset for 6-bit IDs, see example in Table 3.

Extend: This character indicates that more optional fields follow. This is the backslash character "\", 0x5C (7-bit), or the offset value 0x3C if **Begin PnP** is 0x08 (6-bit).

Product Serial Number: In this optional field, Plug and Play compatible COM devices may report a unique product serial number, defined as a 32 bit value in 8 character hexadecimal code, with the first character representing the most significant bits. (See PnP ISA Specification.)

Class Name: This optional field, not exceeding 32 characters, is selected from a set of device types defined in Annex C. If the device does not fit a defined Windows 95 class name, this field should be omitted; device manufacturers should not invent new class names.

Compatible Device IDs: This optional field, including one or more strings separated by comma characters (0x0C or 0x2C), may identify compatible device IDs, for which the driver(s) are applicable. Each Device ID has the same format as defined for EISA ID (three characters) and Product ID (four characters). For example, a pointing device compatible with the Microsoft Mouse would indicate the seven character ID of a known compatible Microsoft Mouse.

Microsoft will define generic drivers, and define the corresponding device ID strings; the prefix will be “PNP”. These will include, for example, Microsoft compatible mice, TIA-602 compatible modems, and TIA-578 compatible Class 1 FAX modems. Refer to the Plug and Play Forum on Compuserve and download “DEVIDS.ZIP” for a current listing of device IDs for legacy devices.

User Name: This optional field, not exceeding 40 characters, shall consist of a user-readable and user-recognizable product device description. It is expected that this would be displayed by a Plug and Play User Interface. Note that this string should not include either of the **End PnP** characters, 0x09 or 0x29.

For legibility, the User Name field may include <CR><LF> characters for 7-bit IDs.

Checksum: This optional field, a modulo-8, 8 bit arithmetic checksum of all characters from **Begin PnP** to **End PnP**, exclusive of the checksum characters themselves, represented as a two character hexadecimal number. The first character is the Most Significant Digit; the second is the Least Significant Digit.

End PnP: This mandatory character indicates the end of the Plug and Play ID strings. If **Begin PnP** is 0x28, then this is 0x29; if **Begin PnP** is 0x08, this is 0x09. Note that the **End PnP** character should not appear in the option information fields, so that the Enumerator can correctly detect the end of the ID string.

For legibility, the EndPnP character may be followed by <CR><LF> characters for 7-bit IDs.

3.3. Example PnP ID Responses

The following are example responses that a Plug and Play compatible COM device might generate. <u>All ID entries are fictitious, unless otherwise noted</u>.

Table 3 - Example PnP ID response, Mouse, without optional fields, 6-bit characters

Field Data (hex)	Field Name	Description
4D	Other ID	Identifies a mouse for old drivers
08	Begin PnP	"(" indicates PnP IDs will follow
00,01	PnP Rev	identifies PnP version 0.01
21,2D,23	EISA ID	"AMC" (A Mouse Company)
11,12,13,14	Product ID	"1234" (a random example)
09	End PnP	")" indicates PnP IDs are complete

Table 4 - Example PnP ID response, Modem, with optional fields, 7-bit characters

Field Data (hex)	Field Name	Description
-	Other ID	-omitted, not a mouse-
28	Begin PnP	"(" indicates PnP IDs will follow
01,24	PnP Rev	identifies PnP version 1.0
4D,44,43	EISA ID	"MDC" (Modem Design Company)
30,32,38,38	Product ID	"0288" (random example)
5C,30,30,33,31,34,31,35,39	Serial #	"\00314159" (random example)
5C,4D,4F,44,45,4D	Class Name	"\MODEM" (a real Class name)
5C,4D,44,43,30,31,34,34, 2C,41,54,4D,30,30,39,36	Device ID	"\MDC0144" (e.g. previous product) ",ATM0096" (e.g. generic AT modem)
5C,5A,49,50,20,32,38,38	User Name	"\ZIP 288" (random model name)
43,34	Checksum	"C4" mod-256 2 byte hex checksum
29	End PnP	")" indicates PnP IDs complete

Annex A (Informative) - Additional Modem Information

If the COM port device does not provide a Plug and Play ID string, other applications or systems software can interrogate a device that is known to be a modem, and determine useful information to support loading drivers.

A.1 Modem Service Classes

Several Telecommunications Industry Association (TIA) modem control standards define services classes for a PSTN modem. Some of these are being made into international (International Telecommunications Union, ITU) Recommendations. (ITU was formerly known as CCITT.) Other organizations (including Microsoft) are also defining modem service classes (e.g. VoiceView). TIA and Portable Computer and Communications Associations (PCCA) are defining devices that look like modems for Cellular or private Wireless Networks.

The capabilities of the modem can be read, test and set using the AT+FCLASS command. The syntax to test the capabilities is AT+FCLASS=?. The modem will respond with ERROR if it does not recognize the command. If it does, the result will be one or more values separated by comma (0x3C) characters. A table of defined values and sources is presented here:

Table A-1 - Serial Device Standards and Service Classes

+FCLASS value	specification	description
0	TIA-602, TIA IS-131, ITU V.25ter	common data modem, interim standard extensions CCITT/ITU equivalent
(0)	PCCA XSTD-101 PCCA STD-101	Character Mode Wireless Modems revision
1	TIA-578, TIA-578-A	Service Class 1 FAX modem, first revision of Class 1
1.0	ITU T.class1	draft CCITT/ITU equivalent
2	TIA SP-2388	obsolete Service Class 2 FAX modem
2.0	TIA-592, ITU T.class2, TIA IS-135, TIA IS-99	standard Service Class 2 FAX modem draft CCITT/ITU equivalent TDMA Digital Cellular FAX modem (IS-54 network) CDMA Digital Cellular FAX modem (IS-95 network)
8	TIA IS-101	Voice DCE (aka Service Class 8, or AT+V)
80	VoiceView 1.0	Voice View compatible modem (Radish Communications)

A.2 TIA and ITU Standard ID Strings

In most of the above referenced standards, there are ID string comands defined, which could provide useful information to meet the requirements of PnP.

For those implemented, there is one command for Manufacturer ID, one for Model ID and one for revision code. These commands are commonly implemented as a handful of characters on one line, but they are allowed to be several lines, up to 2048 characters.

Table A-2 - +FCLASS values, command syntax, and implementation

+FCLASS value	specification	Mfg ID	Model ID	Revision ID
0	TIA-602, etc	-	-	-
0	TIA IS-131	+GMI?	+GMM?	+GMR?
0	ITU V.25ter	+GMI?	+GMM?	+GMR?
0	PCCA XSTD-101	+FMI?	+FMM?	+FMR?
0	PCCA STD-101	+GMI?	+GMM?	+GMR?
1	TIA-578	-	-	-
1	TIA-578-A	+FMI?	+FMM?	+FMR?
1.0	ITU T.class1	+GMI?	+GMM?	+GMR?
2	TIA SP-2388 (1990 Class 2)	+FMFR?	+FMDL?	+FREV?
2.0	TIA-592	+FMI?	+FMM?	+FMR?
2.0	ITU T.class2	+GMI?	+GMM?	+GMR?
2.0	TIA IS-99	+FMI?	+FMM?	+FMR?
2.0	TIA IS-135	+FMI?	+FMM?	+FMR?
8	TIA IS-101	+FMI?	+FMM?	+FMR?
80	VoiceView 1.0	+FMI?	+FMM?	+FMR?

Note that TIA IS-131 and ITU-T Recommendation V.25ter also include +GSN?, to request the modem's serial number, and +GOI, to request the modem's X.208 formatted Global Object Identifier, if provided.

A.3 ATIn commands

There are no universal standards for ATI commands. However, it is recommended that Plug and Play modems include an ATI9 command. If implemented, the modem shall respond to this command by returning information text at the same speed and parity as the ATI command.

That information text should include all the characters it would deliver in a complete Plug and Play ID character string, in standard ASCII. However, if a modem manufacturer implements an ATI9 command, that format may be adjusted for accessibility. The two character PNP ID could be converted to a legible string (e.g. “1.00”). Also, <CR>,<LF> pairs could be inserted. The checksum is optional.

A.4 AT&D3 Implementation Issues

The AT&D3 command presents special problems for Plug and Play COM enumeration.

In many implementations, the hardware DTR lead is logically wired to the reset pin on the modem control processor. In this case, the modem will be reset on either the level (DTR=0) or on an edge (depending on implementation).

The useful effect is that this modem can be hard reset under PC software control. The adverse effect is that the modem controller may not be able to recover from a hard reset fast enough to respond to the enumerator, or to assert DSR if a reset clears it, and the enumerator will miss it.

There are at least two alternative implementations of &D3 that can function with the PNP COM enumerator, but they require redesign of the modem controller firmware. These are:

- Firmware-only implementation of &D3
- Hybrid implementation of &D3 (software ON-Hook, hardware OFF-Hook)

Modem controller firmware: (Some implementations already work this way.) If possible, this would allow the modem controller to be sufficiently initiallized to be functional when the COM enumerator begins toggling DTR. Also, this would mean that DSR wasn't automatically dropped if DTR dropped. The disadvantage is that the useful feature of &D3 would be lost, and a modem in controller firmware failure could not be reset without operator intervention.

Hybrid implementation: while ON-Hook, the modem would implement &D3 in firmware; while OFF-Hook, the modem would implement &D3 in hardware. This modem should also initialize DSR=1 within 200ms, and initialize to a software &D3 response. This would remove the capability for PC controlled hard reset while the modem is in a stable state (On-hook, idle) but retain the PC-controlled reset capability while the modem is in a vulnerable state (Off-hook, carrier on, data carrier active, possible error control and data compression engines running). Note that if the modem is Off-hook, the com port will be closed except in system failure modes, and the enumerator will not disturb the modem (see 2.1.1).

Note that a modem in recovery from a hard reset initiated while off-hook could still recognize the enumerator from the details of DTR-RTS on and off timing, and respond to the 2nd phase.

Annex B COM Enumerator Constraints

COM port devices are diverse in their requirements and behavior. The COM Enumerator is designed with these constraints in mind. The easy part of enumeration is collecting the ID information. The hard part is for the COM device to reliably detect the COM Enumerator and to distinguish it from applications. The goals are:

- Allow a PnP-compatible COM device to reliably detect the COM Enumerator.

- Allow a PnP-compatible COM device to avoid false detection of the COM Enumerator. In other words, don't mistake a application for the COM Enumerator, and thereby confuse the non-PnP aware application with unexpected ID information.

- Allow a non-PnP-compatible COM device to avoid being confused by the COM Enumerator.

B.1 Serial Mice

Common serial pointing devices like mice are simple: they are powered by the serial leads themselves. This requires that at least one of the two static control leads (usually DTR) be held true, to provide a positive DC supply, and requires that the transmit data lead (TXD) should be left in the Mark-Idle (negative voltage) condition to provide a negative DC supply. The latter constraint prohibits the use of character strings from the COM Enumerator to make the initial detection. (Some other mice use RTS as well for positive DC power.) Therefore, the COM Enumerator is constrained to using the only remaining control lead, RTS, if it is not tied to DTR.

The only common output lead from serial mice is the RXD lead. However, some also assert the DSR lead by tying it to the DTR lead. The COM Enumerator uses DSR to detect a mouse, and RXD to collect the COM ID.

Common mice generate RXD serial data at a fixed speed and format: start-stop asynchronous, 1200 bit/s, 9 bit frames, one start bit, 7 bits data (LSB first), no parity bit, one stop bit.

Common mice also generate an ASCII upper-case M (0x4D) to indicate a mouse. Most application software will look for that "M" to identify the mouse.

Serial mice also use bit 6 of characters delivered on RXD to indicate the beginning of a motion report. Therefore, to avoid confusing an older mouse driver that is unaware of Plug and Play, PnP ID strings for serial mice shall avoid setting bit 6, particularly if the device in question is a mouse. In this case, ASCII strings should be offset by 0x20, and lower case characters should not be used.

B.2 Modems

Common modems have their own power supply, or are powered from the PC bus; they do not depend on the serial leads for power.

Serial Data leads

Common modems connect to the TXD and RXD leads, and use asynchronously-framed character-string "AT" commands and responses (e.g. "Hayes (tm) -compatible, TIA-602, etc) for control.

Common modems can adaptively detect the asynchronous serial rate, over a range of supported speeds, by measuring the start bit time in the AT (or "at") command string prefix. This method is called "Autobauding". All but very old modems can support 1200 bit/s; some can autobaud to 57,600 bit/s. TIA-602 requires that all compliant modems be able to detect commands at 1200 or 9600 bit/s.

Control Leads

Common modems will also respond to the two control leads, DTR and RTS. The modem's response to these leads is typically programmable using in-band AT commands. For example, the common AT&D command selects one of the following responses to a negative transition on DTR:

Table B-1 - AT&D responses

&D setting	response to DTR-OFF transition
0	Modem ignores DTR
1	Modem escapes from ON-LINE state to ON-LINE COMMAND state.
2	Modem exits ON-LINE state and goes On-Hook (IDLE).
3	Modem resets (not in TIA-602, but common) See Annex A.4 above for further information.

The "RTS" lead is commonly used to represent the Ready-to-Receive function, for use in flow control of data from the modem, during On-Line state.

Because modems try to interpret DTR and RTS for traditional uses, the COM Enumerator is constrained to use those leads very distinctly from those traditional uses. The method described in section 2.1 depends on using these leads while the modem is IDLE (On-hook) and with a distinct time-signature.

Note that the implementation of the &D3 command poses special problems. See Annex A.4.

Status Leads

Common modems will drive four serial status leads, DSR, CTS, RLSD and RI (see details in section 1.5). Some modems echo DTR to DSR; other assert it automatically when powered. The AT&C command conditions RLSD behavior. The RI lead is typically driven only in response to actually ringing.

B.3 Serial Printers and other devices

Other serial devices may be smart (like modems) or dumb. Modem control commands may be ignored by some devices, but misinterpreted by others. For example, a non-Plug and Play serial printer would likely print out an "AT<tell_me_your_ID> command on a fresh page, which would annoy the user while failing to elicit a Plug and Play ID.

Serial printers are usually configured to a fixed serial port rate of 9600 bit/s or faster. They do not Autobaud.

B.4 Uninterruptible Power Supplies

Some computer systems need to run reliably, unattended, even in the presense of transient AC power outages; network servers are an obvious example. Many of these are connected to Uninterruptible Power Supplies (UPS).

For control reasons, some of these UPS devices are connected to the dependent PC through a serial port. Unfortunately, it is common to connect these devices to the standard serial control leads in non-standard ways. The effect is that in some cases, the Serial enumerator could shut down the power in the UPS under unusual circumstances.

Since the control ports on these UPS devices are not standard serial, the operation of the Enumerator might not be benign, in which case some precaution needs to be taken. There are several examples; here is a non-exclusive list:

- the user should not cold-start the system while dependent on UPS battery power.
- to use means embedded in Windows 95 to lock out the Enumerator from the port.
- to use an adaptor cable wired differently (e.g. don't connect DTR to a shut off relay).
- to use an adaptor cable that filters DTR (e.g. short pulses on DTR are ignored).
- to use a new adaptor cable with embedded Plug and Play response logic embedded in it.

Annex C (Informative) - Class Names

This specification refers to device Class Names. The Microsoft Windows 95 DDK contains a document that describes Device Installers. This list of defined class names is excerpted from that document set.

Table C-1 - Example Windows 95 Device Class Names

Class name	Description
MODEM	Modems (Data, FAX, Voice, etc)
MOUSE	Mouse
PRINTER	Printers

Note 1: other Class Names may be defined.