CHAPTER 11

Communications

Windows 95 features a new 32-bit communications subsystem that provides higher throughput, better reliability, and greater device independence for communications operations than Windows 3.1. The new communications subsystem serves as the underlying architecture on which Windows 95 provides communication services for supporting telecommuting and dial-up network access, Microsoft At Work fax services, access to on-line information services, computer-telephone integration, conferencing and remote access to mail.

The communications architecture addresses problems that users have encountered with communications support in Windows 3.1 to provide a powerful, robust, and flexible communications architecture.

Summary of Improvements over Windows 3.1

Changes made in the kernel and communications architecture in Windows 95 provide improvements and benefits to the Windows 3.1 user including:

- Robust and reliable high baud rate communications throughput
- Better multitasking of communications applications
- Simpler centralized setup and configuration
- Broader device support, and
- Better support for sharing communication devices on a PC (e.g., modems) among different communication applications
- Telephone network independence

Communications Architecture

Around the time when Windows 3.0 was first developed, 2400 baud modems were the mainstream and 9600 baud modems were just becoming affordable. Windows was able to handle receiving data at these relatively slow rates without much difficulty. However, as mechanisms to transfer communications information at faster rates (e.g., higher baud rates or the through the use of data compression) are becoming more popular, the communications architecture of Windows needed to be examined closely.

Around the time when Windows 3.1 shipped, 9600 baud modems were extremely popular, and communications under Windows 3.1 had many barriers that limited the overall effectiveness of reliable high data throughput and support for multitasking when running communications applications. These barriers included high interrupt latency and overhead affecting high speed communications, and a monolithic driver architecture that made it necessary for some third-parties to replace the communications driver provided with Windows to allow their devices to run efficiently in the system.

Windows 95 greatly improves upon the Windows 3.1 architecture to support communications applications and supports high speed communications, as well as a modular communications architecture to allow third-parties and communications device manufacturers to easily plug in new communications device drivers. This section describes the communications architecture used in Windows 95.

Communication Goals of Windows 95

The goals of communications support in Windows 95 are focused around supporting an architecture that delivers better performance than Windows 3.1, and supports ease-of-use enhancements through Plug and Play communications. The communications architecture of Windows 95 delivers the following performance benefits over Windows 3.1:

- **High-speed reliability**

Windows 95 supports reliable high-speed communications by keeping up with data coming in from the communications port, and thus resulting in no lost characters due to interrupt latency. In addition, the use of a 32-bit protected mode file system and network architecture results in less impact on the communications system by reducing required mode transitions and interrupt latency.

- **Higher data throughput**

The 32-bit communications subsystem leverages the preemptive multitasking architecture of Windows 95 to provide better responsiveness to communications applications, and thus supporting higher data throughput. Communications

transfers in 32-bit applications are not as affected by other tasks running in the system as Win16–based applications under Windows 3.1.

- **Provide support for time critical protocols**

The communications architecture for Windows 95 provides support for time critical protocols and allows for real-time serial device control.

- **Independence of underlying telephone networks**

Windows 95 allows applications developers to build telephony applications that can run on a wide variety of different types of telephone networks, including analog, proprietary digital PBXs, key systems, ISDN and cellular.

The Plug and Play initiative provides ease-of-use enhancements system-wide in Windows 95 and communications support is no different. Plug and Play support for communications delivers:

- **Broad device support**

Windows 95 features a new communication driver architecture that makes it easier for third-parties to extend the communications support provided as part of the operating system, without sacrificing functionality or stability. In addition, the new communications architecture features APIs that are extended to support more robust communications devices beyond base RS-232 devices (e.g., ISDN).

- **Easy to install and use communication devices**

Windows 95 features centralized modem installation and configuration support to simplify setup for end-users, and simplify communications development efforts for application developers. Windows 95 leverages the use of a single universal modem driver (UNIMODEM) to provide a consistent mechanism for communicating with modem devices. Windows 95 also provides detection support for Plug and Play modems. It also provides support for existing hardware by including mechanisms for detecting legacy modems.

- **Enables device sharing among communications applications**

Through the use of the Telephony API (TAPI), Windows 95 provides consistent device-independent access to control communication devices for operations such as dialing and answering incoming calls. Arbitration for sharing of communication ports and devices is also handled through TAPI. For example, while dial-up networking in Windows 95 is waiting for an incoming call, a TAPI-aware fax communications application can send an outgoing fax without having to first terminate the already running communications application.

To further understand the improvements related to the new 32-bit communications subsystem in Windows 95, we'll examine the components that make up the communications support.

Kernel Improvements in Windows 95 Makes Communicating More Responsive

When data is coming into the system from a serial communications port, an interrupt occurs telling the system that a piece of data has just been received. Under Windows 3.1, if information is being received at a high rate it was possible that the system could not keep up with the incoming data, thus resulting in errors or lost information at the port.

What is unique about serial communications I/O is that one interrupt occurs on the system for *each* incoming character, versus disk or network I/O in which they manipulate blocks of information at a time. The burden on the communications driver to keep up is quite high.

To support high speed throughput of information from a communications device, the system must be able to respond quickly to incoming data. In Windows 3.1, real-mode drivers sometimes disabled system-wide interrupts for a "long" period of time (usually in terms of milliseconds), during which no incoming information can be received.

To directly address the issue of supporting higher sustained communications throughput, the Windows 95 development team focused on areas in the Windows kernel that resulted in periods of time that interrupts were disabled by the system. In Windows 3.1, the Windows kernel and other components was limited to reliable serial communications at rates of 9,600 bps or slightly higher (dependent upon CPU speed), due to high interrupt latency or other systems design limitations. In addition, the use of real-mode file system and networking drivers would block the system when Windows 3.1 had to execute real-mode code, thus preventing the system from being able to keep up with incoming data.

To improve performance and the rate at which the system can accept incoming data reliably, the Windows 95 team reduced code that can only be used by one process at a time (critical sections), and reduced interrupt latency in the core system. In addition, the use of new 32-bit protect mode components for the implementation of the file system and network subsystem also helped to improve the system responsiveness. Windows 95 is now truly limited in baud rate only by the hardware characteristics of the PC such as the processor speed, and type of communications port.

Driver Architecture

The communications subsystem consists of a modular 32-bit protect mode architecture with *new communications drivers*. VCOMM is a new layer that provides protected mode services that allow Windows-based applications and device drivers to use ports and modems. To conserve system resources, communications device drivers are loaded into memory only when in use by applications. Also, VCOMM uses the new Plug and Play services in Windows 95 to assist with configuration and installation of communications devices.

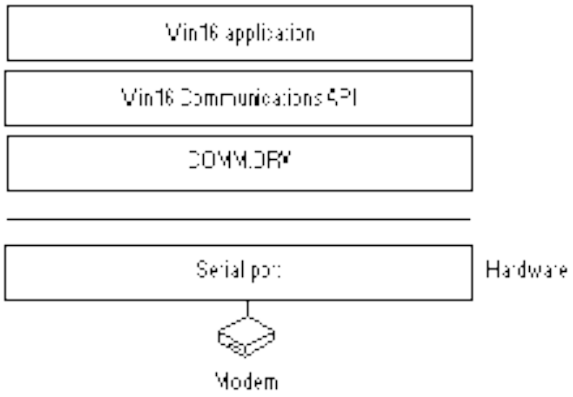

Figure 1. Windows 3.1 Communications Architecture

Windows 3.1 uses a monolithic communications driver, COMM.DRV, that provides an API interface for Windows–based applications to interact with communications devices and the code that serves as the communications port driver. The monolithic approach made it necessary to completely replace the Windows communication driver if new functionality was required by a hardware device. Figure 1 shows the relationship between the COMM.DRV driver and the hardware device in Windows 3.1.

Windows 95 provides a more flexible communications architecture than Windows 3.1, separating communications operations into three primary areas—Win32 communication APIs and TAPI, the universal modem driver, and communication port drivers.

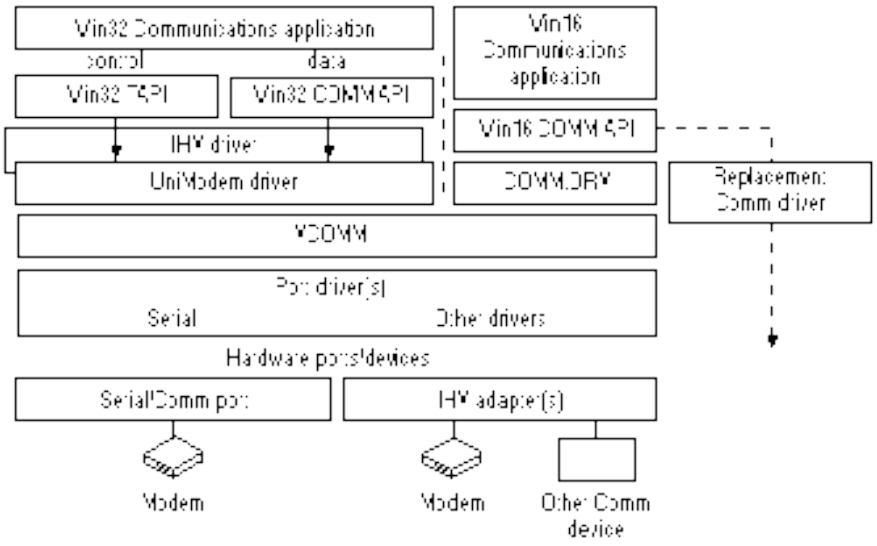

Figure 2. Communications Architecture in Windows 95

Figure 2 shows the relationship between the VCOMM communications driver and the port drivers to communicate with hardware devices. The flow path for a Win16–based application is also illustrated to show how compatibility is maintained for existing Windows–based applications. Compatibility is maintained for IHVs and ISVs that replace the Windows 3.1 COMM.DRV driver, however the vendor-specific communications driver communicates directly with the I/O port, rather than going through VCOMM.

A description of the primary areas that make up the architecture are described below.

u Win32 Communication APIs and TAPI

The Win32 Communications APIs in Windows 95 provide an interface to use modems and communications devices in a device-independent fashion. Applications will call the Win32 Communication APIs to configure modems and perform data I/O through them. Through the Telephony API, applications will be able to control modems or other telephony devices for operations such as dialing, answering, or hanging up a connection, in a standard way. TAPI-aware communication applications no longer need to provide their own modem support list, as interaction with a modem is now centralized by Windows 95. The communications functionality provided with Windows 95 utilizes these services.

- **Universal Modem Driver**

Also new in Windows 95 is the universal modem driver, UniModem, which is a layer for providing services for data and fax modems, and voice. Users and application developers will not have to learn or maintain difficult modem "AT" commands to dial, answer, and configure modems. Rather, UniModem does these tasks automatically, using mini-drivers written by modem hardware vendors. Application developers can utilize TAPI to perform modem control operations in a modem-independent manner.

- **Port Drivers**

Port drivers are specifically responsible for communicating with I/O ports. I/O ports are accessed through the VCOMM driver and provide a layered approach to device communications. For example, Windows 95 will provide a port driver to communicate with serial communications and parallel ports, and third-parties and IHVs can provide port drivers to communicate with their own hardware adapter such as multiport communications adapters. With the port driver model in Windows 95, it will no longer be necessary for third-parties to replace the communications subsystem as they did in Windows 3.1.

Telephony API (TAPI)

The Windows Telephony API is part of the Microsoft Windows Open Services Architecture (WOSA), which provides a single set of open-ended interfaces to enterprise computing services. WOSA encompasses a number of APIs, providing applications and corporate developers with an open set of interfaces to which applications can be written and accessed. WOSA also includes services for data access, messaging, software licensing, connectivity and financial services.

WOSA services such as the Windows Telephony API consist of two interfaces. Developers write to an applications programming interface (API). The other interface, referred to as the service provider interface (SPI), is used to establish the connection to the specific telephone network. This model is similar to the computer industry model whereby printer manufacturers provide printer drivers for Windows-based applications. Figure 3 shows the relationship between the "front-end" Windows Telephony API and the "back-end" Windows Telephony SPI.

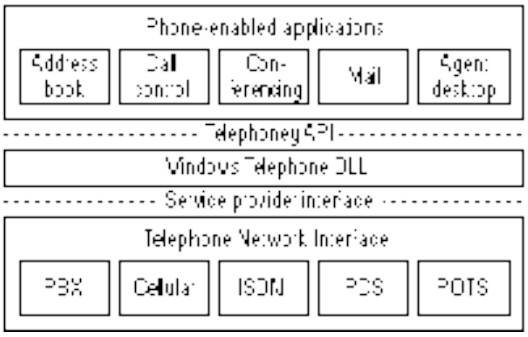

Figure 3. Windows Seamlessly Integrates Applications and Telephone Networks

The Windows Telephony API provides a standard way for communication applications to control telephony functions for data, fax, and voice calls. The API manages all signaling between a PC and a telephone network. This includes such basic functions as establishing, answering and terminating a call. It also includes supplementary functions, such as hold, transfer, conference and call park found in PBXs, ISDN and other phone systems. The API also provides access to features that are specific to certain service providers, with built-in extensibility to accommodate future telephony features and networks as they become available.

The Telephony API also supports multiple models for connecting Windows 95 machines with telephone networks. There are four models for integrating PCs with telephones, as illustrated in Figure 4. Applications using the Telephony API can work in any of these four connection models, whether they involve a physical connection between PC and phone on the desktop such as the phone or PC-centric models, or a logical connection in either of the client-server models.

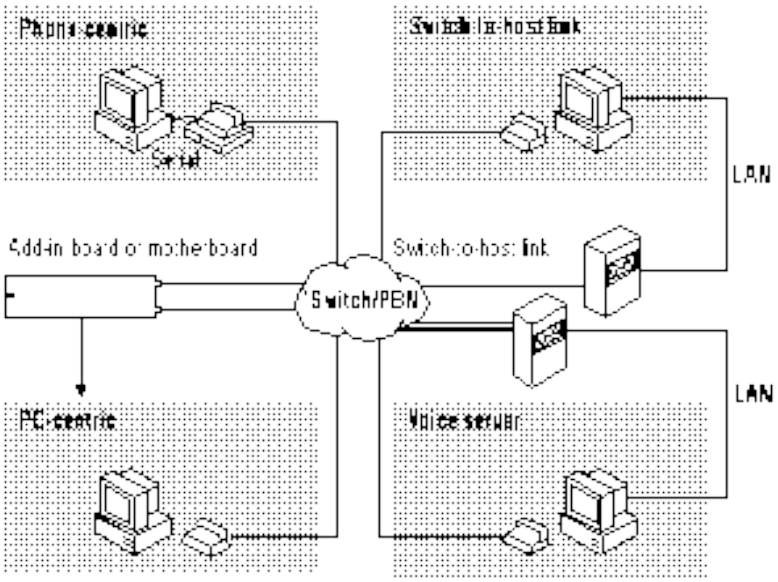

Figure 4. Four Models for Integrating PCs Running Windows 95 With Telephones

Through the use of the TAPI services, applications that support communication services have a device-independent means for interacting with telecommunications networks. TAPI also provides a common access mechanism for requesting the use of communication ports and communication devices, thus providing a means for multiple communications applications to share a single modem (voice, fax, or data) in the computer.

Windows 95 includes TAPI support in the base operating system, thus allowing application developers to leverage this functionality in their Windows 95-aware applications. In addition, all communication components included as part of Windows 95 are TAPI clients.

Better Sharing of Communication Devices Between Communication Applications and Services

Through the TAPI interface, communications applications can ask for access to the modem or telephone device, allowing the communications subsystem in Windows 95 to arbitrate device contention and allow applications to share a communications device in a cooperative manner.

Win32–based applications can utilize TAPI functionality to allow applications to be able to make outgoing calls, while others are waiting for inbound calls. For example, while a dial-up network service is configured for auto-answer mode and is waiting for an incoming call, a Win32–based communications application can call the TAPI services to request the use of the modem and perform an outgoing call. Of course, only one call can be performed at a time, but users no longer have to terminate other applications that are using a communications port in order to run a different communication. The TAPI services arbitrate requests to share communication ports and devices.

Centralized Modem Setup and Configuration

Support for installing and configuring a modem under Windows 95 is greatly simplified over Windows 3.1. No longer is it necessary to configure each individual communications program for the proper serial port, modem type, and other related modem configuration parameters. Windows 95 provides central configuration of communications devices through an icon in Control Panel. Win32–based applications that take advantage of the TAPI services implemented in Windows 95 can completely leverage the user's configuration of their communications hardware, making subsequent configuration of communications-based applications easy.

Windows 95 brings the following benefits to modem configuration:

- Easy modem configuration for use by entire system for new communications applications
- Centralized communications port status and configuration
- Supported by TAPI & Win32 Communication APIs
- Support for 100+ modems in Windows 95

Modem Configuration in Windows 3.1

To understand the implications of improved modem support in Windows 95, let's first examine the issue of modem configuration under Windows 3.1. Under Windows 3.1, when a user adds a new communications program to their computer, the user must first configure the application to communicate with their modem by specifying the COM port to use, the type of modem they have, in addition to other communication parameters. Communications and modem configuration is either handled by the applications vendor and specified as a series of default modem "AT" command statements, or it is up to the user to read through his/her modem manual and type in the appropriate command strings. Figure 5 shows the Modem Commands dialog box in Terminal provided with Windows 3.1. Furthermore, given the number of modems available on the market, many Windows 3.1–based communications applications support a limited set of

recognized modems because of the increased burden on the applications developer to provide this support.

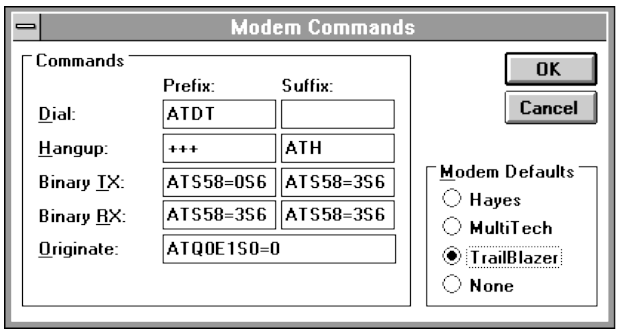

Figure 5. Modem Configuration in Windows 3.1 Terminal

Modem Configuration in Windows 95

As with support for printers, the use of modems in Windows 95 is centralized by the operating system. When a user first installs Windows 95, the user is prompted to detect or identify the modem device that they have connected to or installed in their computer. Once a modem has been selected and configured, any communications application that supports TAPI services can interact with the modem in a device-independent way. Users no longer need to know or understand complicated "AT" command sequences to customize their communications application.

Configuring a modem under Windows 95 is as easy as performing three simple steps: identifying the new modem device, configuring the modem device, and configuring the Telephony services.

Identifying a New Modem Device

If the user doesn't select a modem when Windows 95 is first installed, a new modem can be identified from the Modems control panel icon. When the Modems dialog box is displayed, the user can have Windows 95 detect the modem connected to the PC, or can manually select a modem from the list of known manufacturers and modem models. The detect option makes it easy for users by using Plug and Play detection or querying the modem to configure the correct device. If Windows 95 can not detect the device, then the user can manually select the proper modem to use.

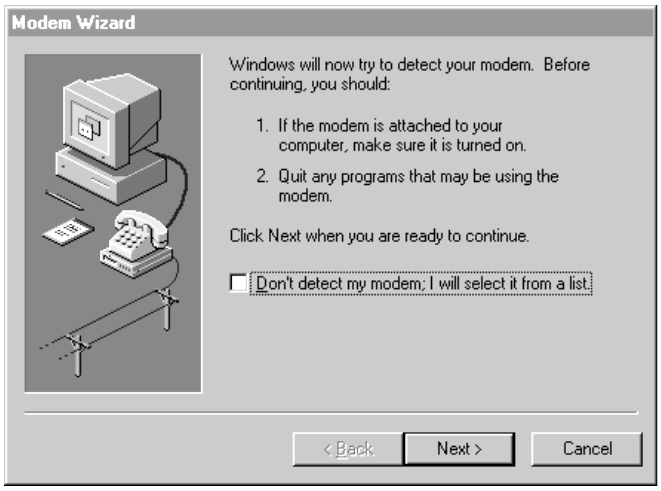

Figure 6. Modem wizard for Detecting and Installing a Modem

Configuring a Modem Device

Once the proper modem has been selected, the user can optionally change the properties for the device to set configuration parameters such as the volume for the modem speaker, the time to wait for the remote computer to answer the call, and the maximum baud rate to use. (The maximum baud rate is limited by the speed of the PC's CPU and the speed supported by the communications port.)

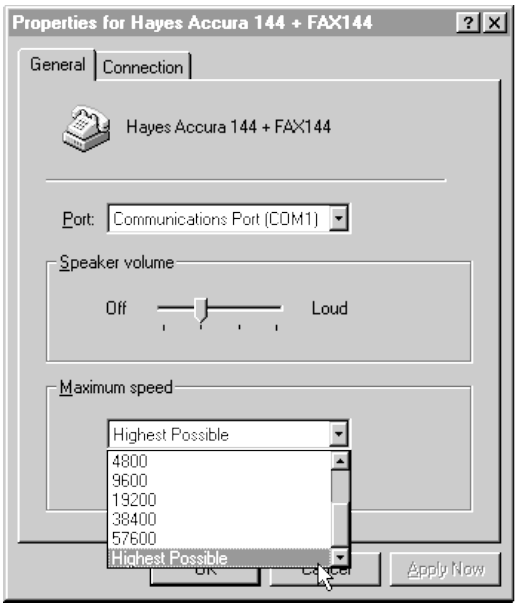

Figure 7. Sample Modem Property Sheet

Configuring Telephony Services

In addition to configuring the modem device, the user configures telephony services to identify the various dialing parameters associated with the different locations where the computer will be used.

For each location where the PC will be used, information is stored for use by TAPI-aware applications to simplify dialing a local call, a long distance call, the area code for the location for use in determining whether the call is inside or outside the calling area code, and calling card information. For a desktop PC, the location would commonly use the "default location" or perhaps change the default name to "in the office," whereas a mobile user would add several different locations to match where the laptop computer is commonly used.

For example, a mobile user may use the computer in the office, on the road, or in a remote city as part of his/her business needs. Figure 8 shows several different locations configured and selectable depending on the location where the computer is being used.

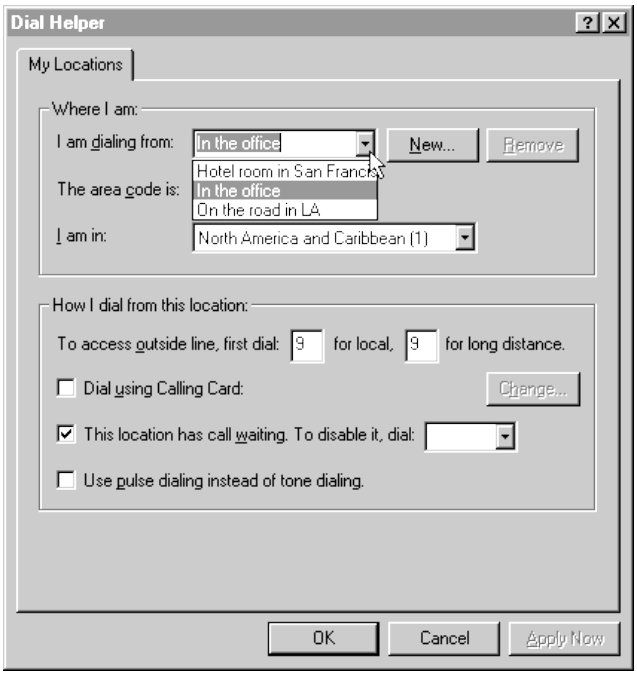

Figure 8. Dial Helper Property Sheet for Configuring Location Information

Improved Device/Hardware Support

Windows 95 provides improved communications device and hardware support over Windows 3.1. A few areas of improvement are discussed below.

16550A UART FIFO Support

Windows 95 will provide greater robustness and performance at high baud rates for MS-DOS-based and Windows-based communications applications using local serial ports with 16550A compatible UARTs. The 16550A UART contains a 16 byte FIFO buffer to prevent character overflow due to interrupt latency, and helps to reduce interrupt overhead overall. The Windows 3.1 communications driver did not fully support the use of the 16550A UART, requiring some third-party communications vendors to replace the Windows 3.1-provided communications driver. Communications in Windows 95 should alleviate the need for third-party vendors to replace communications driver components.

More Ports Supported

Windows 3.1 imposed a limit to the number of logical names that could be used to address or refer to serial communication ports and to parallel ports, of 9 serial ports and 4 parallel ports. The communication APIs present in Windows 95 have been enhanced to support the same number of logical ports as MS-DOS, which is 128 serial ports, and 128 parallel ports. This limit inhibited the use of multi-port serial devices in Windows 3.1. The actual limitations to the number of ports usable is still based on the physical number of ports available to the system.

Support for Future Parallel Modems

Windows 95 also provides support for Enhanced Capabilities Ports (ECP) to facilitate higher speed communications than is possible over a serial device. This support allows the use of future parallel port modems.

Plug and Play Support

Plug and Play support for communications devices in Windows 95 facilitates the detection of connected modem devices and assigning of system resources (e.g., IRQs and I/O addresses for communication ports), simplifying configuration and setup. In addition to Plug and Play detection, Windows 95 provides for manual detection of non-Plug and Play communication devices such as modems. Since there is presently no standard for obtaining device information using the "AT" modem command strings, detection of legacy modems is handled by performing a manual query of the modem device and checking information returned against a database of known modem information. Microsoft is working with other leading industry manufacturers to standardize the modem command set as part of a Telecommunications Industry Association (TIA) proposed standard called IS-131. When this proposal is adopted, Windows 95 will support the standardized command set and this will aid detection of legacy modems.

Modems

Plug and Play detection for external modems requires new firmware in the modem to return the required Plug and Play ID information, while internal Plug and Play modems utilize the ISA Plug and Play specification. Windows 95 supports the use of PCMCIA communication devices as part of the Plug and Play services for the PCMCIA specification. Some modem manufacturers will improve their communications product offerings by revising their existing modem lines, whereas others will produce a new line of Plug and Play modems. Detection for Plug and Play serial devices such as modems is handled when Windows 95 is initially installed, during the boot process, or when a new modem device is connected to the system. As with other Plug and Play devices, the user is notified that the new device has been detected and is asked to confirm the installation and configuration of the device.

Support for legacy modems is provided by using device-specific information about a modem to provide a manual detection mechanism, or providing a list of supported modems from which a user can choose the appropriate one. Once the modem is identified for the system, it is available for use by TAPI-enabled communication applications including dial-up networking, Microsoft At Work fax services, and the new HyperTerminal communications application.

New Communications Application: HyperTerminal

Windows 95 includes a new 32-bit communications application called HyperTerminal that demonstrates what it's like to be a good communication application under Windows 95. HyperTerminal offers the same base communication capabilities as Terminal included with Windows 3.1, but integrates well with the UI in Windows 95 and demonstrates how the Win32 communication APIs and TAPI services support more flexible communications applications than Windows–based applications under Windows 3.1.

Good communication applications under Windows 95 will utilize the following services and capabilities to offer a more robust and powerful solution:

- Win32–based application that uses the Win32 Communication APIs.

Internal architecture that uses multiple threads of execution to provide good responsiveness to the user and great error-free high speed communications. Multiple threads allows for full preemptive multitasking of communication tasks, supporting concurrent interaction with the user, downloading of remote data, and display of communication status.

- Support for TAPI services for making remote connections and controlling the modem device.

New Communications Application: Dialer

The Dialer application in Windows 95 provides basic support for making telephone calls. It includes a telephone dial pad, user programmable speed dials and a call log. Increasingly, new communications hardware will support voice communications in addition to data and fax. The next generation of modems will support the AT+V standard (TIA IS-101) which adds voice support to the standard AT command set, effectively turning the modem into a telephone designed to be a PC peripheral. Other devices, such as those built on digital signal processors (DSPs) will also include voice telephony support.

Figure 9. Dialer Application in Windows 95

Windows 95 communications applications will bring control of the telephone to PC, enabling programmable “smart” answering machines, dynamic call filtering and routing, dialing from any PC application or directory, dragging-and-dropping to set up conference calls and other examples of computer-telephone integration.

Try It!

To see how the improvements made to communications support in Windows 95 will help users of existing MS-DOS and Windows-based communications applications, you've got to try it!

Background Multitasking of Communications Applications

To see how support for communications application in Windows 95 is improved over Windows 3.1, try the following under both Windows 3.1 and under Windows 95:

- Run an MS-DOS–based communications application in the background with other foreground activities.
- Run a Win16–based communications application and perform other CPU or disk intensive tasks, such as copying files, accessing a network, or accessing/formatting a floppy disk.
- Run the 32-bit HyperTerminal communications application and perform other CPU or disk intensive tasks, such as copying files, accessing a network, or accessing/formatting a floppy disk.

Power of the Telephony API

To see how support for communications application in Windows 95 is simplified over Windows 3.1, try the following under Windows 95:

- Install and configure a modem for use on your system.
- Run TAPI-enabled applications such as Phone Dialer, HyperTerminal, Dial-Up Networking, and Microsoft At Work Fax software, and note that once the modem was configured you didn't have to change modem settings in any of these applications.