CHAPTER 12

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 communications services that support telecommuting and dial-up network access, Microsoft Fax services, access to online information services, computer-telephone integration, conferencing, and remote access to mail.

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

Summary of Improvements over Windows 3.1

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

- Robust and reliable, high-baud-rate communications throughput
- Better multitasking of communications applications
- Simpler centralized setup and configuration
- Broader device support
- Better support for sharing communication devices, such as modems, among different communications applications
- Telephone network independence

The 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—for example, higher baud rates or the use of data compression—became more popular, the communications architecture of Windows needed to be examined closely.

When Windows 3.1 was released, 9600-baud modems were extremely popular, but because of communications barriers under Windows 3.1, the overall effectiveness of reliable high data throughput was limited, and the efficiencies of multitasking were eroded when running communications applications. These communications barriers included high interrupt latency and overhead that affected 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 on the system.

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

Communications Goals of Windows 95

The goals of communications support in Windows 95 are to deliver better performance than Windows 3.1, and to enhance ease-of-use 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, thereby incurring no lost characters because of interrupt latency. In addition, the use of a 32-bit protected mode file system and network architecture has less impact on the communications system because required mode transitions and interrupt latency are reduced.
- **Higher data throughput.** The 32-bit communications subsystem leverages the preemptive multitasking architecture of Windows 95 to provide better responsiveness to communications applications and support 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.
- **Support for time-critical protocols.** The communications architecture provides support for time-critical protocols and allows for real-time serial device control.
- **Independence of underlying telephone networks.** Windows 95 allows application developers to build telephony applications that can run on a wide variety of telephone networks, including analog, proprietary digital PBXs, key systems, ISDN, and cellular.

The Plug and Play initiative provides ease-of-use enhancements throughout Windows 95, and communications support is no exception. Plug and Play support for communications delivers the following benefits:

- **Broad device support.** Windows 95 features a new communications 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 support more robust communications devices beyond base RS-232 devices—for example, ISDN.

- **Easy-to-install and easy-to-use communications devices.** Windows 95 features centralized modem installation and configuration to simplify setup for 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. It also provides detection support for Plug and Play modems and supports existing hardware by including mechanisms for detecting legacy modems.
- **Device sharing among communications applications.** Through the use of the Telephony API (TAPI), Windows 95 provides consistent, device-independent mechanisms for controlling communications devices for operations such as dialing and answering incoming calls. Arbitration for the sharing of communications 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 describe the improvements resulting from the new 32-bit communications subsystem in Windows 95, the rest of this section examines the components that comprise the communications support.

Kernel Improvements

When data comes into the system from a serial communications port, an interrupt tells the system that a piece of data has been received. If information was received at a high rate under Windows 3.1, the system sometimes could not keep up with the incoming data, resulting in errors or lost information at the port.

Whereas disk I/O and network I/O manipulate blocks of information at a time, serial communications I/O generates one interrupt on the system for *each* incoming character. 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, but in Windows 3.1, real-mode drivers sometimes disabled system-wide interrupts for “long” periods of time (usually milliseconds), during which no incoming information could be received.

To address the issue of supporting higher, sustained communications throughput, the Windows 95 development team focused on areas in the Windows 3.1 kernel that resulted in periods of time when interrupts were disabled by the system. The Windows 3.1 kernel and other components were limited to reliable serial communications at rates of 9600 bps or slightly higher (dependent on CPU speed) because of high interrupt latency or other systems design limitations. In addition, when Windows 3.1 had to execute real-mode code, the use of real-mode file system and networking drivers blocked the system, 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 code that can be used by only one process at a time (critical sections) was reduced and interrupt latency in the core system was also reduced. In addition, the use of new 32-bit protected-mode components for the implementation of the file system and network subsystem helped to improve the system responsiveness. Windows 95 is now truly limited in baud rate only by PC hardware characteristics, such as the processor speed and type of communications port.

Driver Architecture

The Windows 95 communications subsystem consists of a modular, 32-bit protected-mode architecture with new communications drivers. A new layer called VCOMM 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. VCOMM uses the Windows 95 Plug and Play services to assist with configuration and installation of communications devices.

In Windows 3.1, a monolithic communications driver called COMM.DRV provided an API interface, through which Windows–based applications interacted with communications devices, and the code that serves as the communications port driver. The monolithic approach made it necessary to completely replace the Windows communications driver if new functionality was required by a hardware device. Figure 77 shows the relationship between the COMM.DRV driver and the hardware device in Windows 3.1.

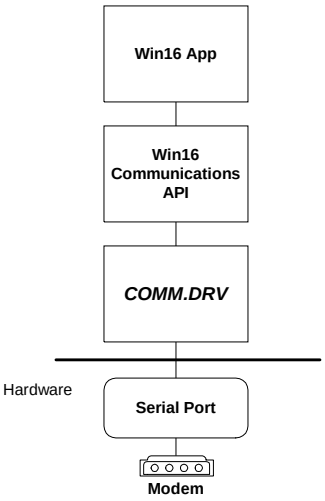

Figure 77. The communications architecture of Windows 3.1

Windows 95 provides a more flexible communications architecture than Windows 3.1, separating communications operations into three primary areas: Win32 communications APIs and TAPI, the universal modem driver, and communications port drivers. Figure 78 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.

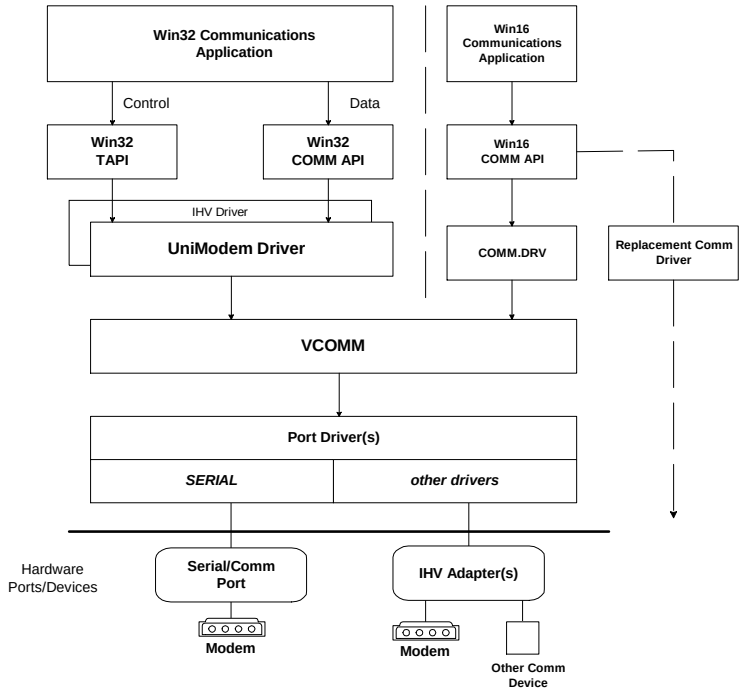

Figure 78. The communications architecture of Windows 95

The primary areas that make up this architecture are the following:

- **Win32 communications APIs and TAPI.** The Win32 communications APIs in Windows 95 provide an interface for using modems and communications devices in a device-independent fashion. Applications call the Win32 communications APIs to configure modems and perform data I/O through them. Through the Telephony API, applications can control modems or other telephony devices for operations such as dialing, answering, or hanging up a connection, in a standard way. TAPI-aware communications applications no longer need to provide their own modem support list because 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 no longer have to learn (and application developers no longer have to maintain) difficult modem AT commands to dial, answer, and configure modems. UniModem handles these tasks automatically, using mini-drivers written by modem hardware vendors. Application developers can use TAPI to perform modem control operations in a modem-independent manner.

- **Port drivers.** Port drivers are responsible for communicating with I/O ports, which are accessed through the VCOMM driver and provide a layered approach to device communications. For example, Windows 95 provides 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 adapters, such as multiport communications adapters. With the port driver model in Windows 95, third parties no longer have to replace the communications subsystem as they did in Windows 3.1.

The 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 application 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.

Like other WOSA services, the Windows Telephony API consists of two interfaces: the applications programming interface (API) that developers write to, and the service provider interface (SPI) that is used to establish the connection to the specific telephone network. This model is similar to the one whereby printer manufacturers provide printer drivers for Windows–based applications. Figure 79 shows the relationship between the "front-end" Windows Telephony API and the "back-end" Windows Telephony SPI.

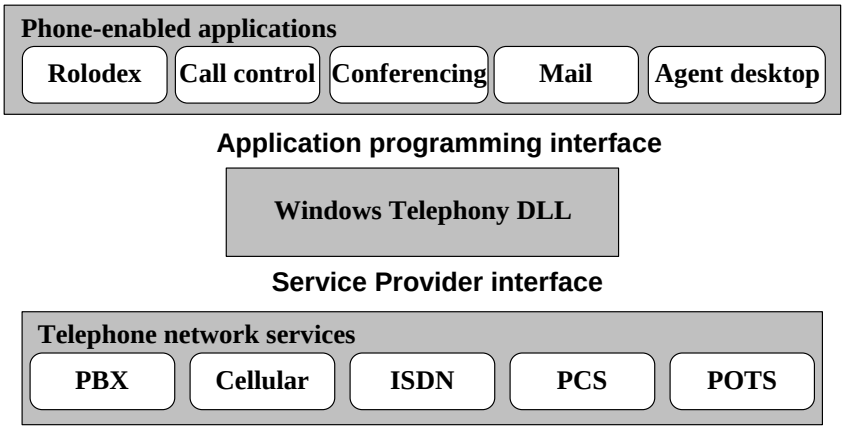

Figure 79. The seamless integration of applications and telephone networks by means of the Windows Telephony API and the Windows Telephony SPI

The Windows Telephony API provides a standard way for communications applications to control telephony functions for data, fax, and voice calls. The API manages all signaling between a PC and a telephone network, including 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. In addition, the API 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 supports four models for integrating Windows 95 PCs with telephone networks, as illustrated in Figure 80. Applications using the Telephony API can work in any of these four connection models, whether they involve a physical connection between a 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.

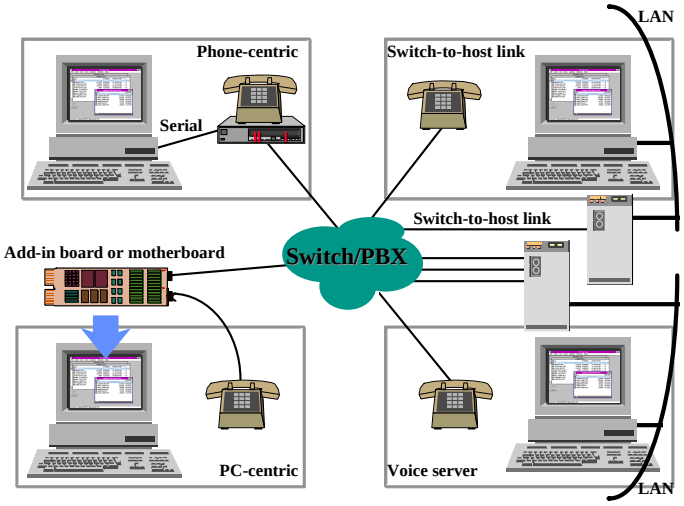

Figure 80. Four models for integrating Windows 95 PCs with telephones

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

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

Sharing Communications Devices

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 the communications device in a cooperative manner.

Win32–based applications can utilize TAPI functionality to allow some applications to make outgoing calls while others are waiting for inbound calls. For example, while a dial-up network service that is configured for auto-answer mode is for an incoming call, a Win32–based communications application can call the TAPI services to request the use of the modem to perform an outgoing call. 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 application. The TAPI services arbitrate requests to share communications ports and devices.

Try It!

Test the Power of the Telephony API

1. In Windows 95, install and configure a modem for use on your system.

2. Run TAPI-enabled applications, such as Phone Dialer, HyperTerminal, Dial-Up Networking, and Microsoft Fax software, and note that after the modem is configured you don't have to change modem settings in any of these applications.

Centralized Modem Setup and Configuration

Support for installing and configuring a modem under Windows 95 is greatly simplified over Windows 3.1. Configuring each individual communications application for the correct serial port, modem type, and other related modem configuration parameters is no longer necessary. Windows 95 provides central configuration of communications devices through a tool in the 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 applications easy.

Windows 95 brings the following benefits to modem configuration:

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

Modem Configuration in Windows 3.1

With Windows 3.1, when users added a new communications application to their computer, they first had to configure the application to communicate with their modem by specifying the COM port to use and the type of modem, in addition to other communication parameters. Communications and modem configuration was either handled by the application developer and specified as a series of default modem AT commands, or users had to read through the modem manual and type in the appropriate command strings. For example, Figure 81 shows the Modem Commands dialog box in Terminal. Many Windows 3.1–based communications applications support only a limited set of modems because, given the number of modems available on the market, the burden on the application developer of providing global support is too great.

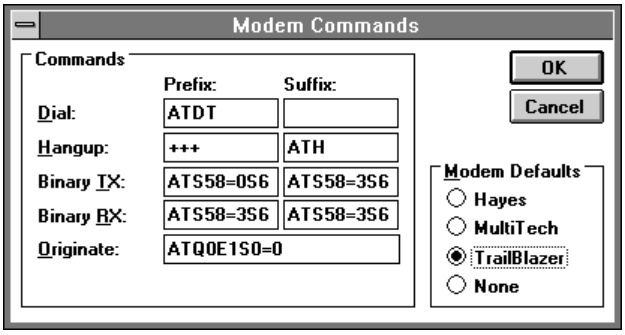

Figure 81. Configuring a modem in Windows 3.1 Terminal

Modem Configuration in Windows 95

As with support for printers, the support for modems in Windows 95 is centralized. When users first install Windows 95, they are prompted to detect or identify the modem device that they have connected to or installed in their computer. When 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 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 a modem is not selected when Windows 95 is first installed, The Modem Wizard can be used to identify a new modem, by using the Modems tool in the Control Panel. When the Modem Wizard dialog box is displayed, as shown in Figure 82, the user can have the Wizard 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 uses Plug and Play to configure the correct device. If the Wizard cannot detect the device, the user can still manually select the correct modem.

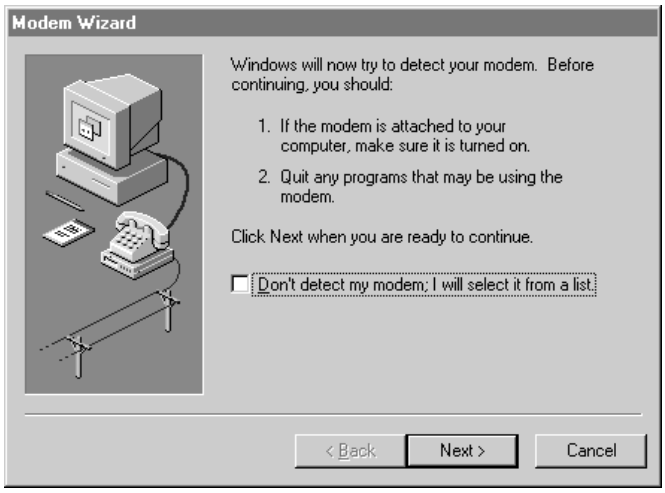

Figure 82. The Modem Wizard, which can detect and install a modem

Configuring a Modem Device

After the correct modem has been selected, users can optionally change configuration parameters, such as the volume for the modem speaker, the time to wait for the remote computer to answer a call, and the maximum baud rate to use, on a property sheet like the one shown in Figure 83. (The maximum baud rate is limited by the speed of the PC's CPU and the speed supported by the communications port.)

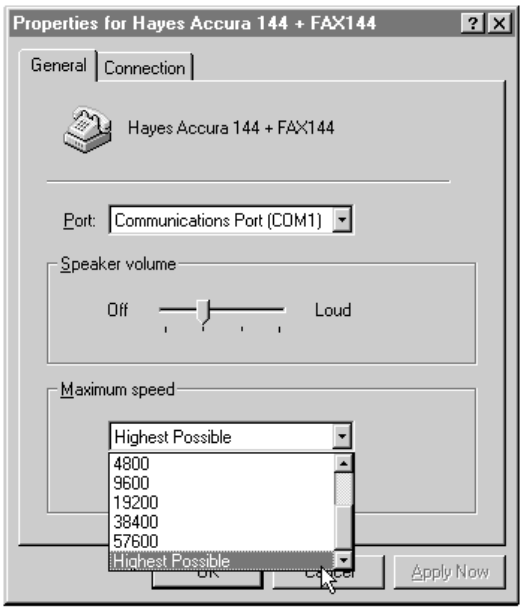

Figure 83. A Modem property sheet

Configuring Telephony Services

In addition to configuring the modem device, users configure telephony services to identify the various dialing parameters associated with the different locations where the PC will be used. For each location, information is stored for use by TAPI-aware applications, including information needed to dial a local call and a long distance call, the location's area code (for use in determining whether the call is inside or outside the calling area code), and calling card information. For a desktop PC, the default location would commonly be used—the default name could be changed to *in the office*—whereas for a portable computer, a mobile user might add several different locations to accommodate those where the computer is commonly used. For example, a mobile user might use the computer in the office, on the road, or in a remote city. Figure 84 shows three location configurations that are selectable depending on the location where the computer is being used.

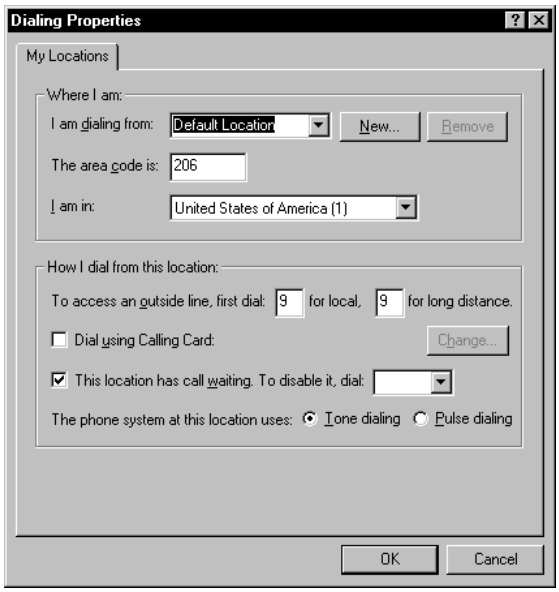

Figure 84. Dialing Properties for configuring location calling information

Device/Hardware Support

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

Support for 16550A UART FIFO

Windows 95 provides 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 resulting from interrupt latency, and help to reduce overall interrupt overhead. Because the Windows 3.1 communications driver did not fully support the use of the 16550A UART, some third-party communications vendors had to replace the driver. Improvements in Windows 95 communications should alleviate this problem.

Support for More Ports

Windows 3.1 limited the number of logical names that could be used to address serial communications ports to nine and the number that could be used to address parallel ports to four. This limit inhibited the use of multiport serial devices in Windows 3.1. The communications APIs in Windows 95 have been enhanced to support the same number of logical ports as MS-DOS: 128 serial ports and 128 parallel ports. Obviously, the number of usable ports is still a function of the number of physical ports available to the system.

Support for Future Parallel Port Modems

Windows 95 supports Enhanced Capabilities Ports (ECP) to facilitate higher speed communications than is possible over a serial device. This support provides for 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 assignment of system resources—for example, IRQs and I/O addresses for communications ports—which simplifies configuration and setup. In addition to Plug and Play detection, Windows 95 provides for manual detection of non–Plug and Play communications devices, such as modems. Because no standard for automatically obtaining device information using the AT modem command strings presently exists, detection of legacy modems is handled manually by querying the modem device and checking the information returned against a database of known modem information. As part of a Telecommunications Industry Association (TIA) proposed standard called IS-131, Microsoft is working with other leading industry manufacturers to standardize the modem command set. When this proposal is adopted, Windows 95 will support the standardized command set, which will aid detection of legacy modems.

Modems

External modems require new firmware to return the required Plug and Play ID information, whereas internal modems utilize the ISA Plug and Play specification. PCMCIA communications devices are supported 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, while others will produce a new line of Plug and Play modems.

Detection of 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 to provide a manual detection mechanism, or by displaying a list of supported modems from which a user can choose the appropriate one. After the modem has been identified for the system, it can be used by TAPI-enabled communications applications, including dial-up networking, Microsoft Fax services, and the new HyperTerminal communications application.

HyperTerminal

Windows 95 includes a new 32-bit communications application called HyperTerminal that has all the qualities of a good Windows 95 communications application. HyperTerminal offers the same base communication capabilities as the Terminal program included with Windows 3.1, but integrates well with the UI in Windows 95 and demonstrates how the Win32 communications APIs and TAPI services support more flexible communications applications than Windows 3.1–based applications.

Good communications applications utilize the Windows 95 services and capabilities to offer a more robust and powerful product, as follows:

- They are Win32–based applications that use the Win32 communications APIs.

- Their internal architecture uses multiple threads of execution to provide good responsiveness to the user and great error-free high-speed communications. Multiple threads allow full preemptive multitasking of communications tasks and support concurrent interaction with the user, downloading of remote data, and display of communications status.
- They take advantage of TAPI services for making remote connections and controlling the modem device.

Try It!

Run Communications Applications in the Background

1. Start Windows 3.1.
2. Run an MS-DOS–based communications application in the background with other foreground activities.
3. Run a Win16–based communications application and perform other CPU or disk-intensive tasks, such as copying files, accessing a network, or formatting a floppy disk.
4. Now start Windows 95 and repeat steps 2 and 3.
5. Run the 32-bit HyperTerminal communications application and perform other CPU or disk-intensive tasks, such as copying files, accessing a network, or formatting a floppy disk.

Phone Dialer

The Phone Dialer application in Windows 95 provides basic support for making telephone calls. As shown in Figure 85, it includes a telephone dial pad, user programmable speed dials, and a call log.

Figure 85. Phone Dialer Application

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. Windows 95 communications applications will bring control of the telephone to the PC, enabling programmable "smart" answering machines, dynamic call filtering and routing, dialing from any PC application or directory, drag-and-drop setup of conference calls, and other types of computer-telephone integration.