



(19) **United States**
(12) **Patent Application Publication** (10) **Pub. No.: US 2001/0033271 A1**
Pabon et al. (43) **Pub. Date: Oct. 25, 2001**

(54) **COMPUTER-TELEPHONY INTEGRATION
EMPLOYING AN INTELLIGENT
KEYBOARD AND METHOD FOR SAME**

Publication Classification

(51) **Int. Cl.⁷** **G09G 5/00**

(76) Inventors: **Reinaldo Pabon**, Whitehouse Station,
NJ (US); **Richard Mauro**, Flemington,
NJ (US)

(52) **U.S. Cl.** **345/168**

Correspondence Address:
Gerald T. Bodner, Esq.
HOFFMANN & BARON, LLP
6900 Jericho Turnpike
Syosset, NY 11791 (US)

(57) **ABSTRACT**

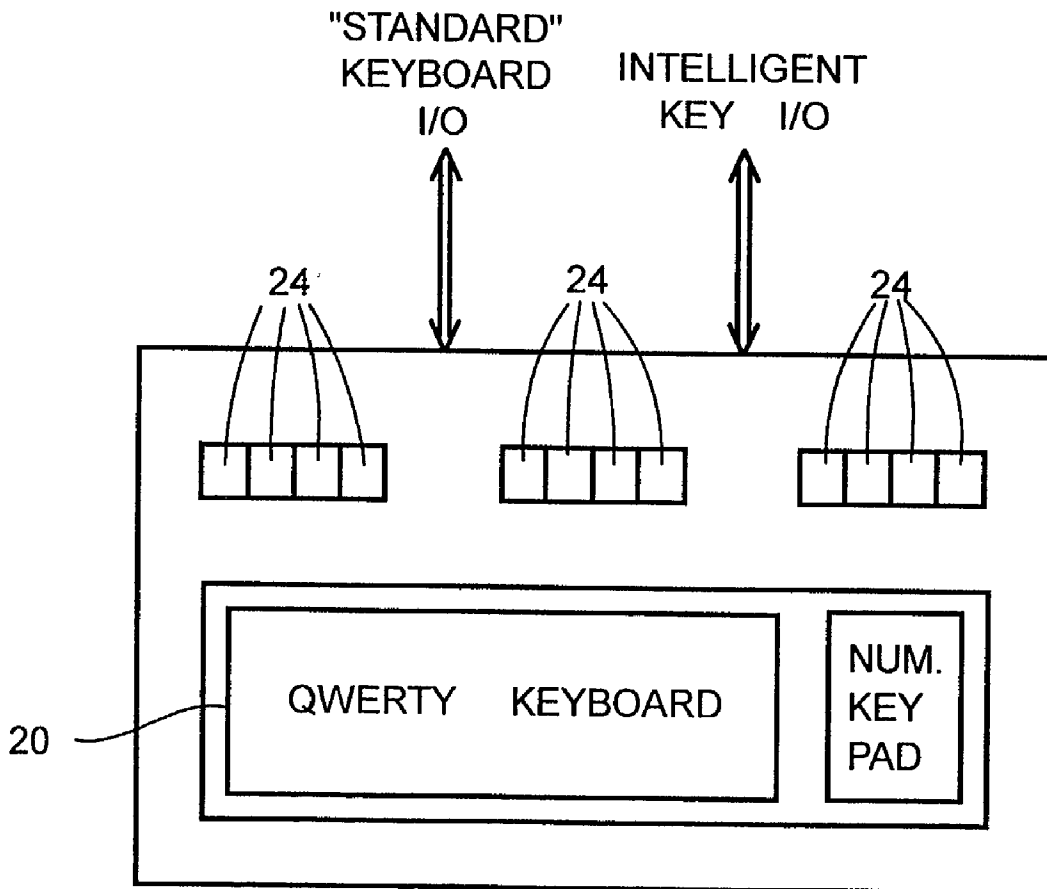
An improved computer-telephony integrated user console includes a computer system electrically connected to a telephone system. The computer-telephony integrated user console further includes an intelligent integrated keyboard which includes at least one programmable display key. The intelligent integrated keyboard is electrically connected to the computer system. Each display key includes an electrical switch and a programmable electrical display element, such as a bit-mapped LCD array. The display is responsive to commands received from the computer system and displays text and/or graphics on the key to indicate a current function assigned to that key. The function of the key is dynamic, and is controlled by a software automation server operating in the computer.

(21) Appl. No.: **09/859,110**

(22) Filed: **May 16, 2001**

Related U.S. Application Data

(63) Continuation of application No. 09/052,650, filed on Mar. 31, 1998, now Pat. No. 6,256,020, which is a non-provisional of provisional application No. 60/041,767, filed on Mar. 31, 1997.



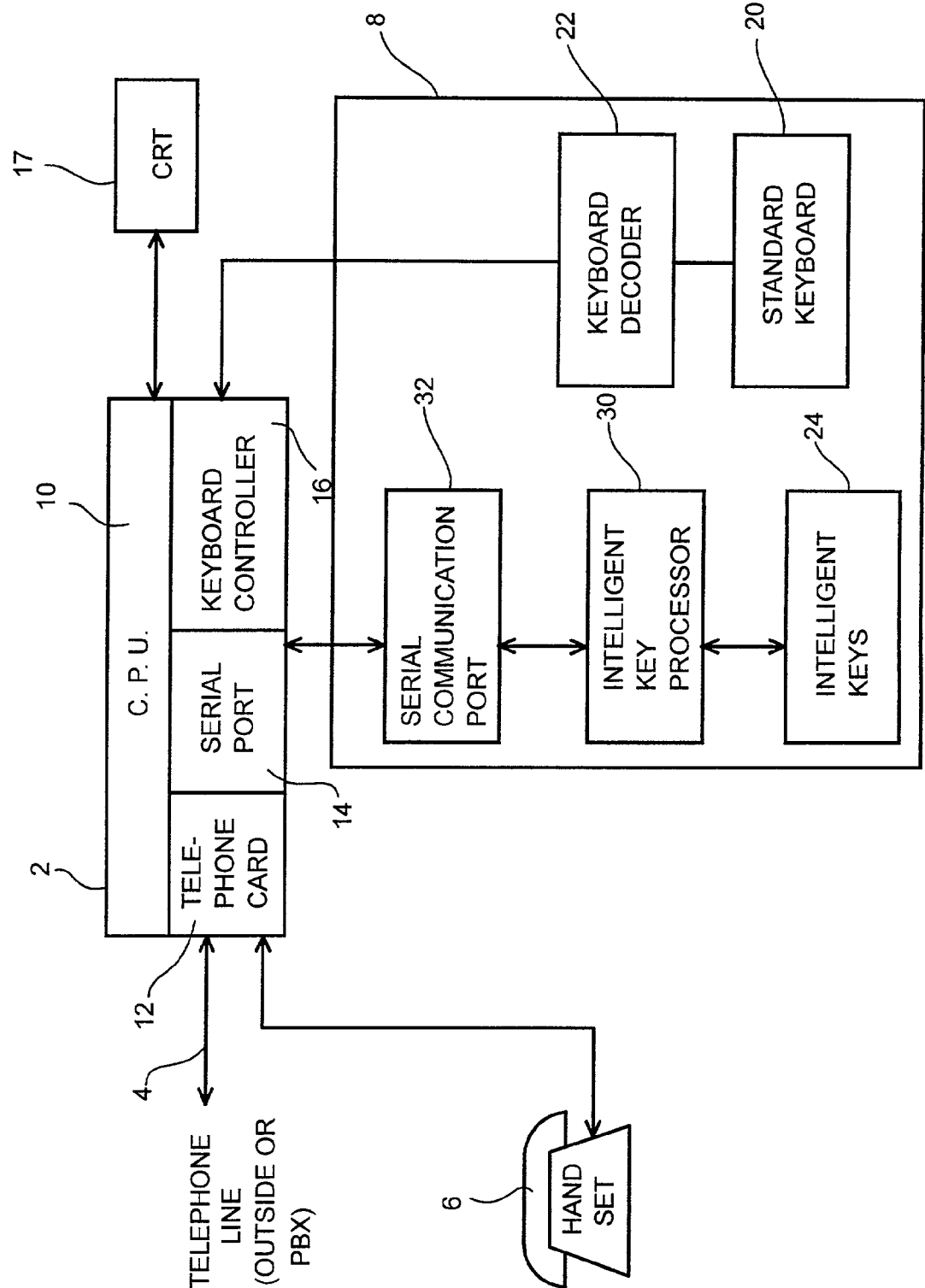


Fig. 1

Fig. 2

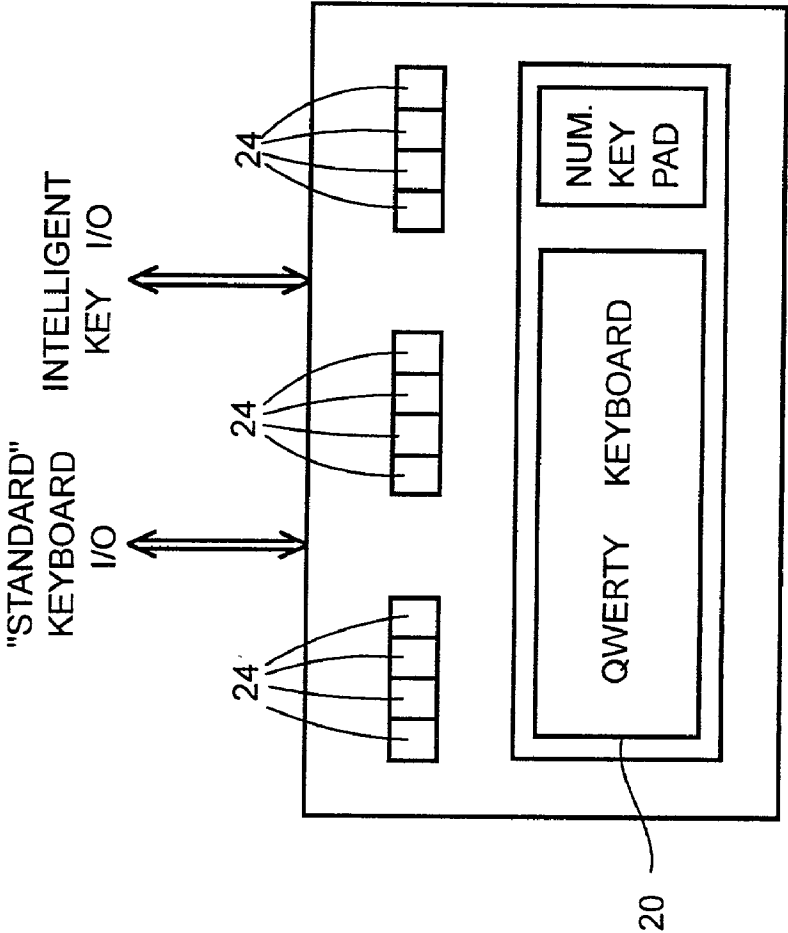


Fig. 3

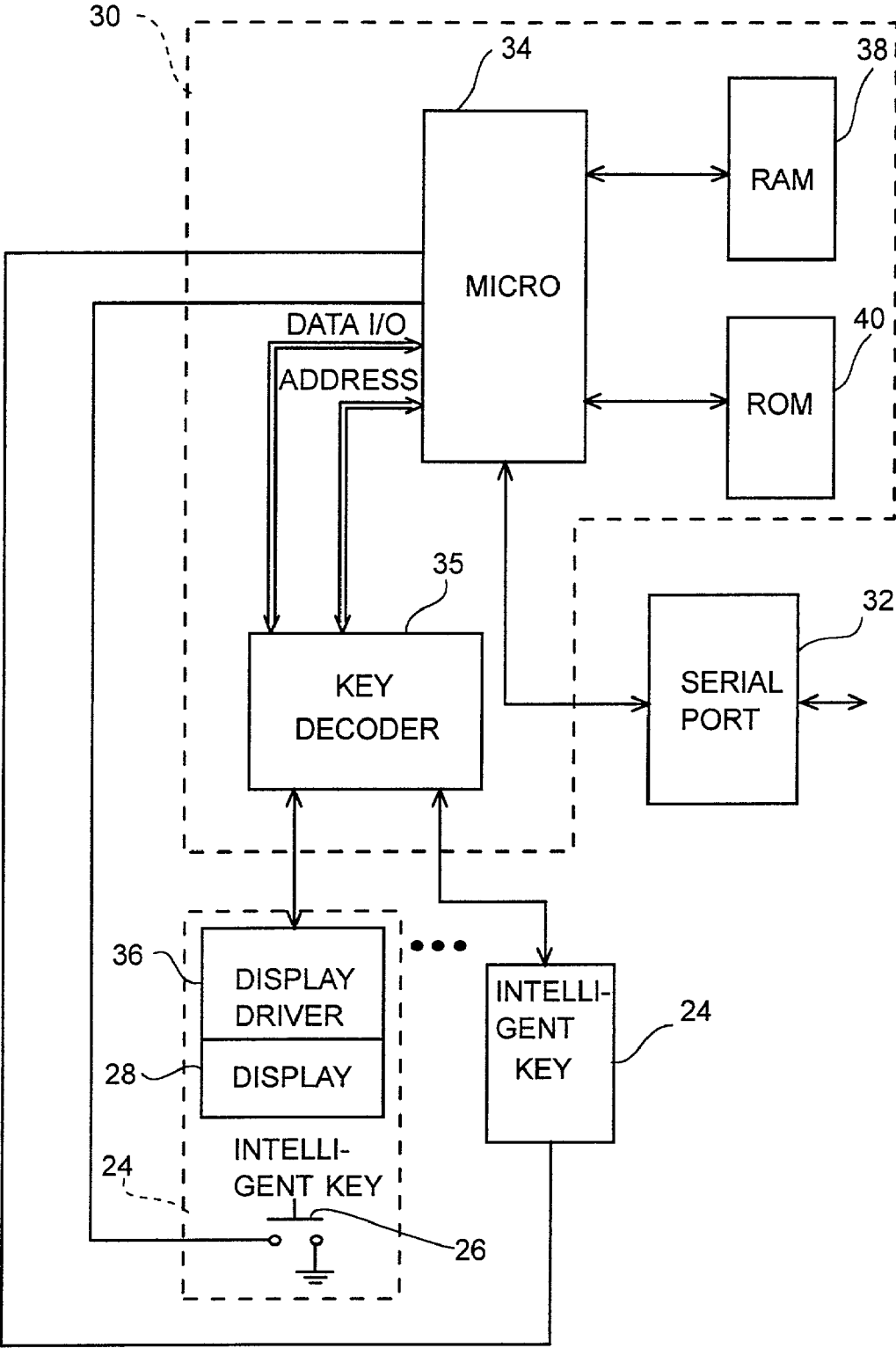


Fig. 4

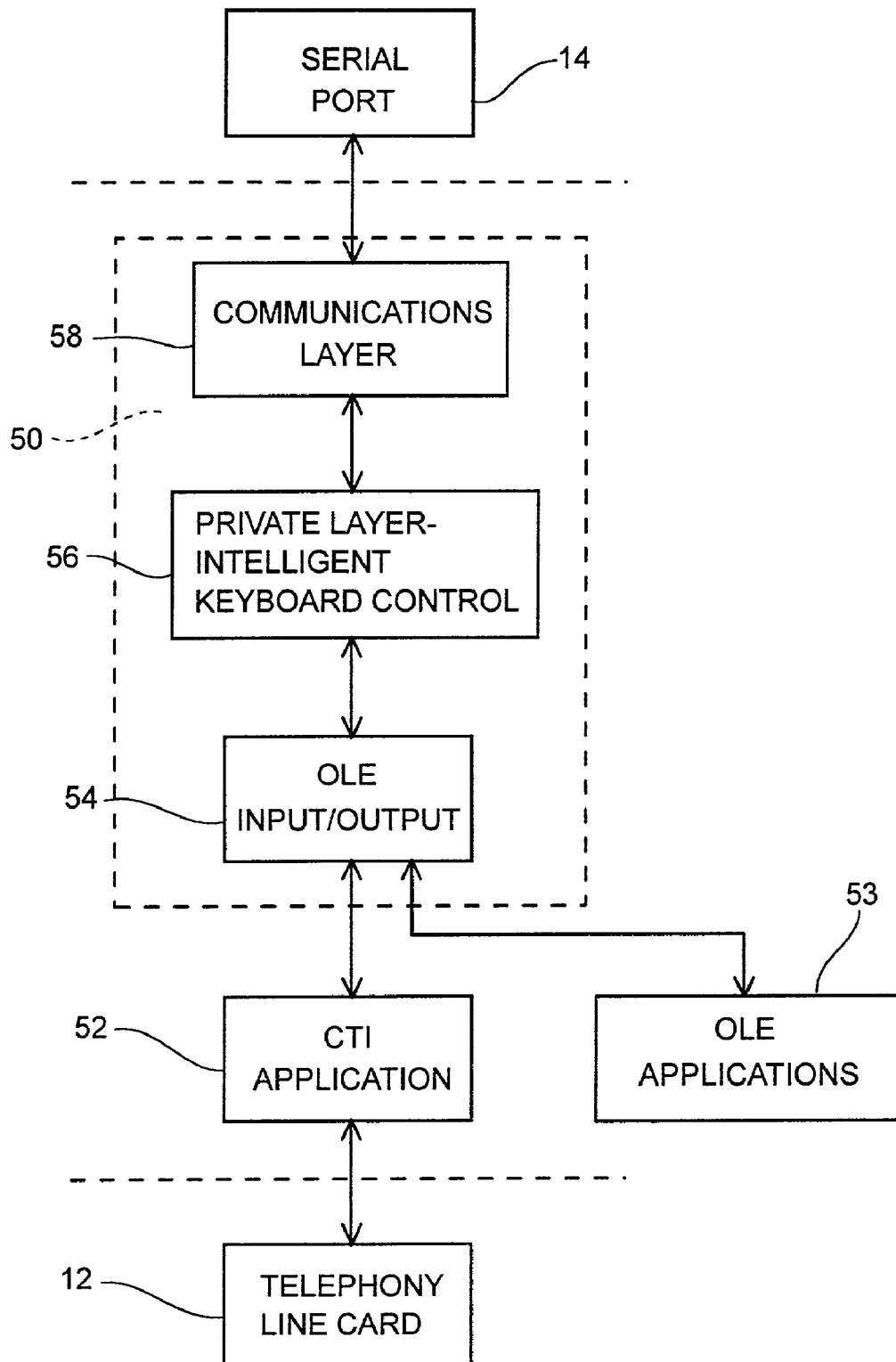


Fig. 5

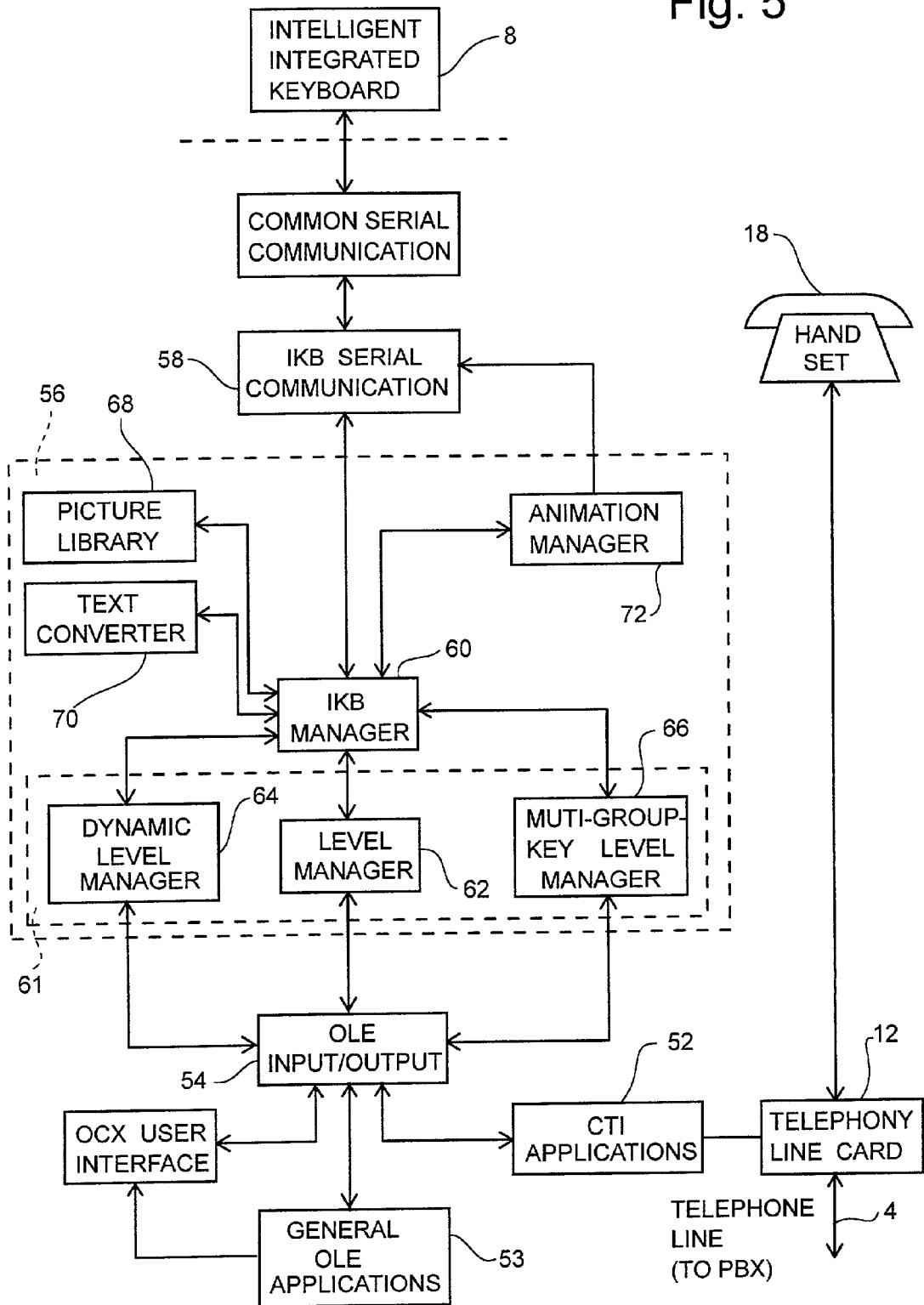


Fig. 6A

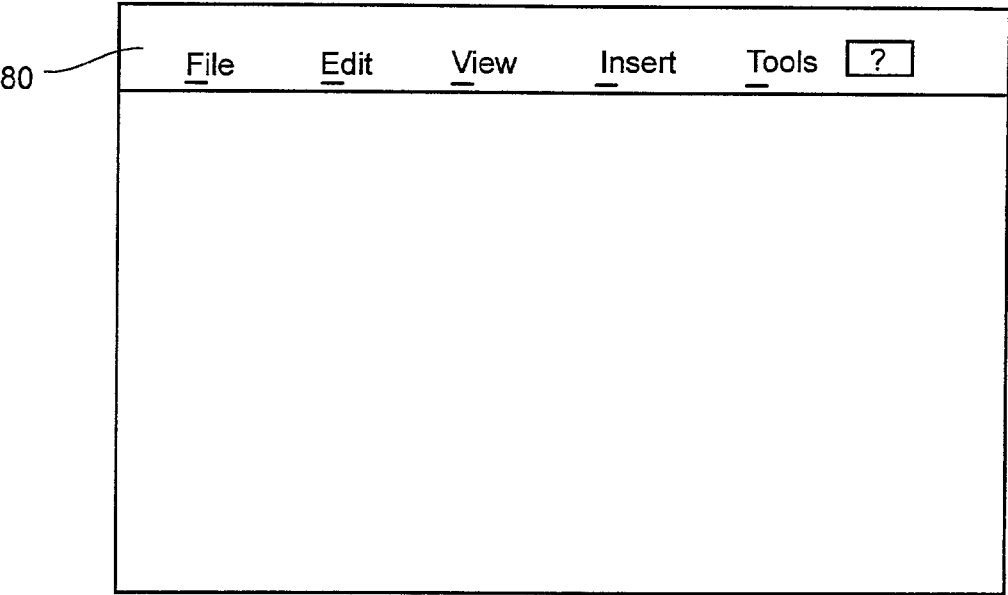


Fig. 6B

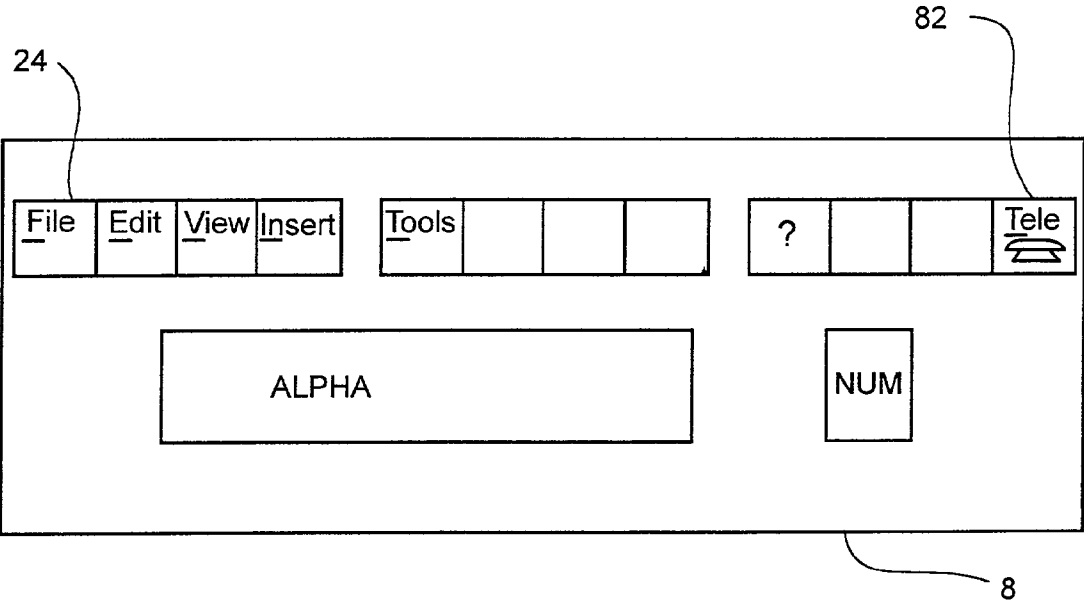


Fig. 7A

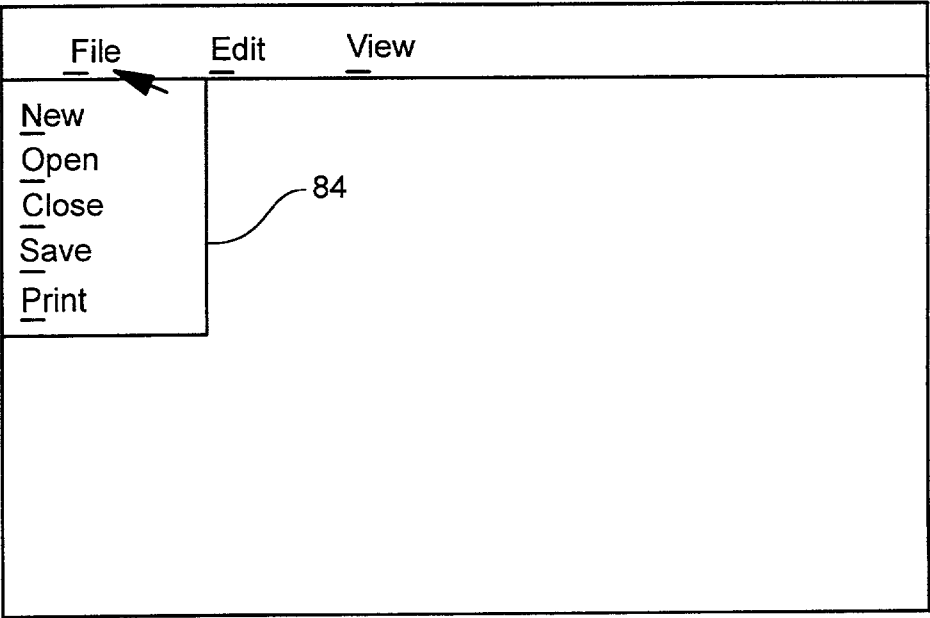


Fig. 7B

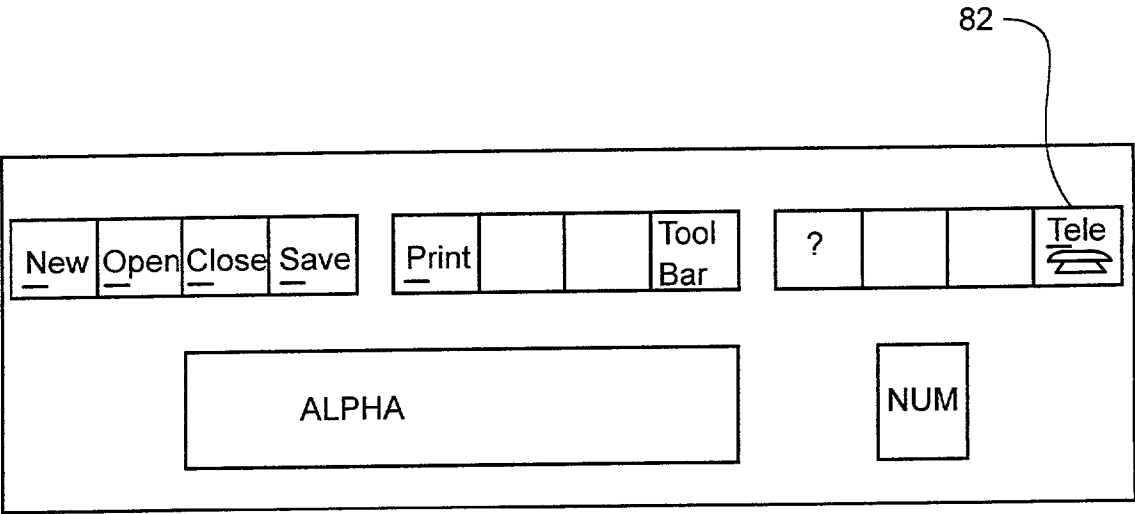


Fig. 8A

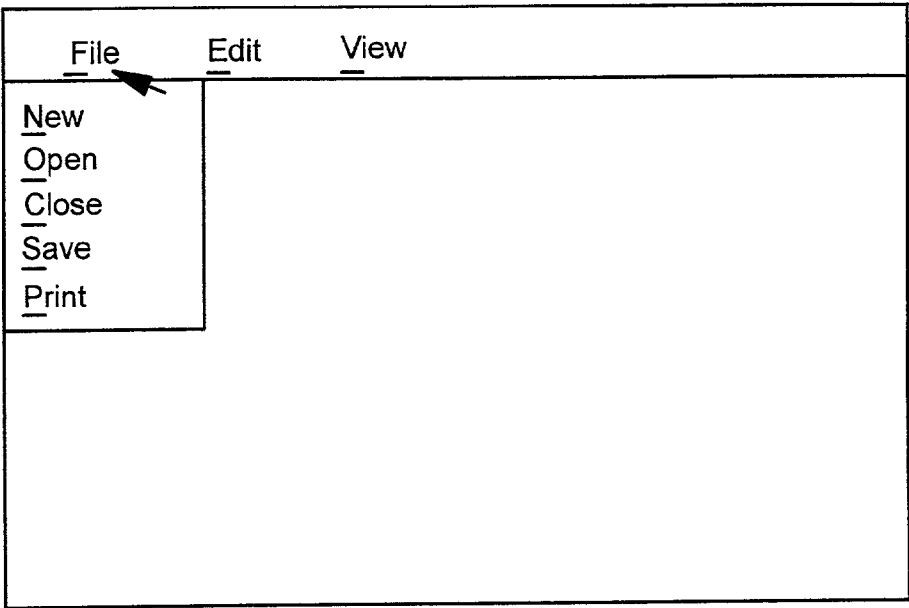


Fig. 8B

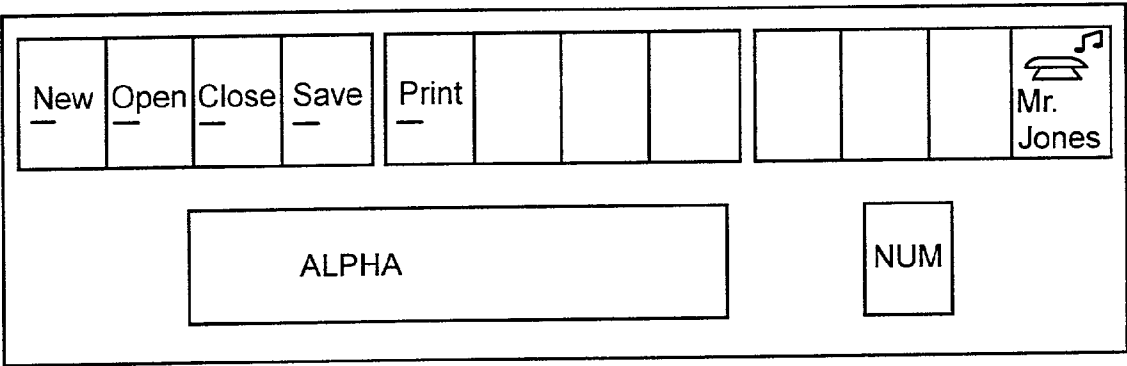


Fig. 8C

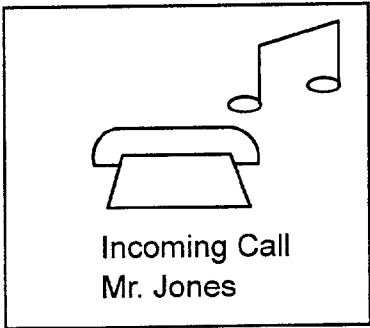


Fig. 9A

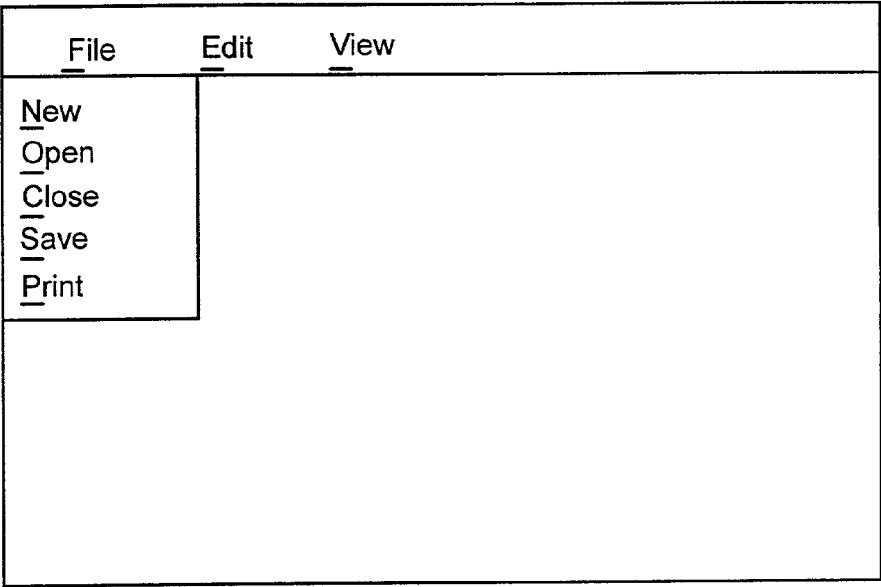


Fig. 9B

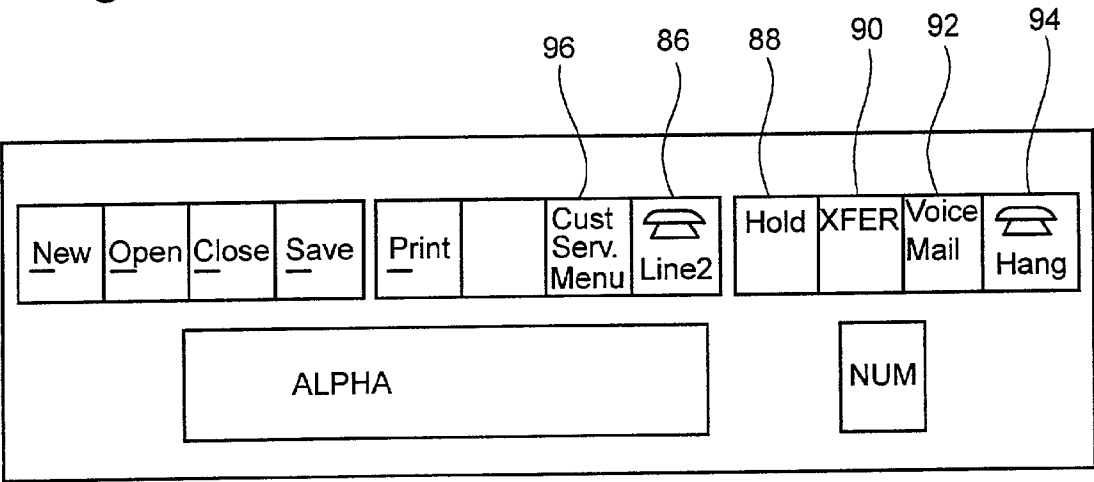
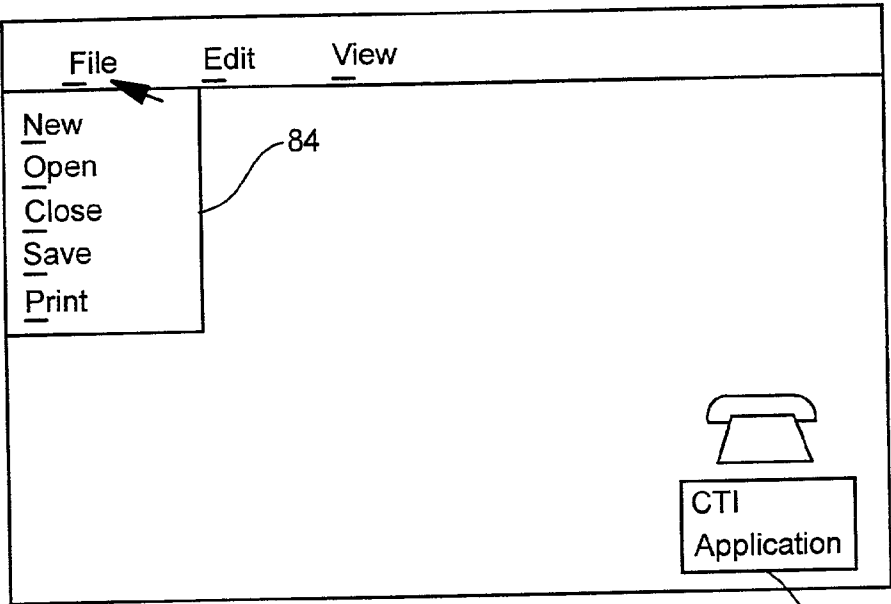
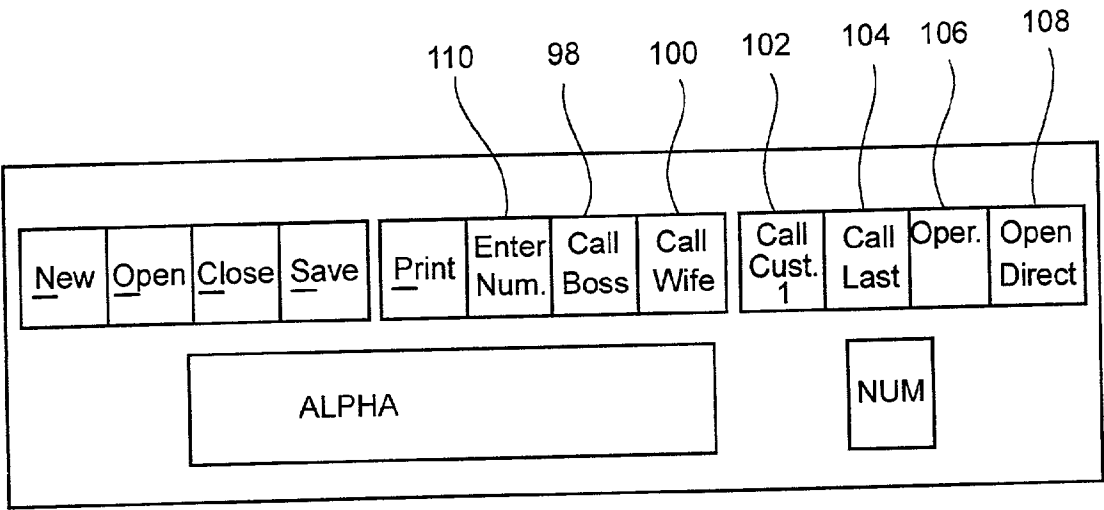


Fig. 10A



112

Fig. 10B



COMPUTER-TELEPHONY INTEGRATION EMPLOYING AN INTELLIGENT KEYBOARD AND METHOD FOR SAME

CROSS-REFERENCE TO RELATED APPLICATIONS

[0001] This application claims the benefit of U.S. Provisional Application No. 60/041,767, filed on Mar. 31, 1997 entitled "Computer-Telephony Integration Employing an Intelligent Keyboard and Method For Same."

BACKGROUND OF THE INVENTION

[0002] 1. Field of the Invention

[0003] The present invention relates generally to telephone systems, and more particularly relates to a computer-telephony integration employing an intelligent keyboard to facilitate improved user interface.

[0004] 2. Description of the Prior Art

[0005] Modern business practice relies heavily on the use of both telephone systems and computer systems as important tools. Quite often, the simultaneous use of these powerful tools is required. Typical examples of such applications include customer service centers, telephone based sales, telephone surveys and the like.

[0006] Therefore, it is often desirable to provide a degree of interaction between the telephone system and the computer system to improve efficiency. In this case, the telephone system is capable of responding to commands received by the computer, such as dialing a telephone number from a database stored in a computer. Similarly, the computer is able to respond to commands received from the telephone system, such as providing a visual indication that a call is incoming.

[0007] Business telephony systems have become quite complex and provide a number of features in addition to local switching and control of incoming calls. These systems, often referred to as private branch exchanges (PBX), are capable of multiplexing a large number of voice and data signals from a large number of user consoles. Often, the PBX includes a caller identification (ID) function to indicate the phone number of an incoming call on the user console.

[0008] Further, many PBX also include automated touch-tone answering systems. These systems prompt the caller through a series of questions, then route the call according to answers which are entered by the caller on a touch-tone telephone. The answers, in digital form, can also be forwarded along with the incoming telephone call. This provides the user at the console with certain information about the caller, such as account number and the nature of the call, even before the call is answered. A sophisticated user console interfaced to the PBX can display such information on a computer terminal associated with the user console.

[0009] One such user console, known in the prior art, includes a telephone handset combined with a digital terminal employing a touch-sensitive cathode ray tube (CRT). The digital terminal is part of a data processing system which stores business related information, such as a database of customers. In addition, information regarding the incoming call may also be displayed on the CRT. During the course of a call, information may be entered into the data

processing system by the user at the console by use of a keyboard associated with the digital terminal. Further, the touch sensitive CRT associated with the terminal allows a variety of commands to be selected by the user by touching the screen where these commands are displayed.

[0010] The use of the touch screen allows the user to enter commands with fewer key strokes on the keyboard. However, the use of a touch sensitive CRT has the disadvantage of requiring the user to remove his or her hands from the keyboard to use the touch screen. As a result, the data entry process is interrupted, resulting in inefficiencies in the call answering process. In addition, while selections are being made on the touch screen, the field of view of the screen is obscured to the user by his or her own hand. A final disadvantage associated with the use of the touch screen is that the commands which are available to the user are limited to those currently being displayed on the CRT by an active software application.

[0011] As an alternative to the use of a touch sensitive CRT, other computer-based PBX user consoles in the prior art rely on a digital pointer, such as a mouse, to control the operation of both a telephone system and a data processing system. As with the case of the touch sensitive CRT, commands related to the telephone system and data processing system are displayed on the CRT of a digital computer terminal. To select one of the commands, the digital pointer is moved until an indicator which is displayed on the CRT is proximate to the desired command. Typically, a switch on the digital pointer is then depressed to select the desired command.

[0012] While the use of the digital pointer alleviates the problem of obscuring the field of view during command selection, this system still requires the user's hands to leave the main keyboard to manipulate the digital pointer. As with the touch screen, the digital pointer can only access those commands currently being displayed by the active application on the CRT. Also, digital pointers often require additional space on a desktop on which to move the digital pointer. A further disadvantage of digital pointers is that a degree of dexterity is required to manipulate the pointer. While seemingly trivial, in the process of responding to hundreds of calls a day, this process can fatigue the user and possibly result in a repetitive motion injury.

OBJECTS AND SUMMARY OF THE INVENTION

[0013] It is an object of the present invention to provide a computer-telephony integrated system which includes an improved user console.

[0014] It is another object of the present invention to provide a computer-telephony integrated system which includes a user console which provides for more efficient manipulation of information in a data processing system and control of a telephone system.

[0015] It is yet another object of the present invention to provide a computer-telephony integrated system which includes a user console that allows a user to easily manipulate information in a data processing system as well as control a telephone system without the use of a touch sensitive screen or a digital pointer, such as a mouse.

[0016] It is a further object of the present invention to provide an intelligent keyboard suitable for use with a computer-telephony integrated user console.

[0017] It is still a further object of the present invention to provide a method of operating a computer-telephony integrated user console employing an intelligent keyboard.

[0018] In accordance with one form of the present invention, an improved computer-telephony integrated user console is formed employing a personal computer, a telephone line interface, and an integrated intelligent keyboard system. The integrated intelligent keyboard system preferably includes a standard alphanumeric keyboard and a plurality of intelligent function keys. The intelligent function keys include a switch and an integral display associated with each switch. The display associated with the intelligent function keys, as well as the function of each of the intelligent function keys, is dynamic. The function and display are responsive and adaptive to events detected by the computer from the telephone system and/or applications software.

[0019] The improved computer-telephony integrated user console of the present invention is controlled by software. The control software is resident in the personal computer and controls the interface between the telephone system, general purpose applications software and the integrated intelligent keyboard. The control software enables the integrated intelligent keyboard to respond and adapt to events detected by other software applications and to events detected within the telephone system.

[0020] These and other objects, features and advantages of the present invention will become apparent from the following detailed description of illustrative embodiments thereof, which is to be read in connection with the accompanying drawings.

BRIEF DESCRIPTION OF THE DRAWINGS

[0021] FIG. 1 is a block diagram of a computer-telephony integrated user console formed in accordance with the present invention.

[0022] FIG. 2 is a pictorial diagram of an exemplary embodiment of an integrated intelligent keyboard formed in accordance with the present invention.

[0023] FIG. 3 is an electrical block diagram of circuitry associated with an intelligent keyboard formed in accordance with the present invention.

[0024] FIGS. 4 and 5 are block diagrams illustrating the preferred interrelation of software modules which operate a computer-telephony integrated user console formed in accordance with the present invention.

[0025] FIGS. 6A and 6B are pictorial diagrams of an exemplary computer CRT display and corresponding intelligent integrated keyboard system display, respectively, illustrating the cooperative operation between the intelligent keyboard system and computer during operation of a computer-telephony integrated user console formed in accordance with the present invention.

[0026] FIGS. 7A and 7B are pictorial diagrams of a computer CRT display and corresponding intelligent keyboard display, respectively, further illustrating the cooperative relationship between the intelligent keyboard and computer system of a computer-telephony integrated user console formed in accordance with the present invention.

[0027] FIGS. 8A and 8B are pictorial diagrams of a computer CRT display and corresponding intelligent key-

board display, respectively, further illustrating the cooperative relationship between a computer system and intelligent keyboard of a computer-telephony integrated user console formed in accordance with the present invention.

[0028] FIG. 8C is a pictorial diagram further illustrating an exemplary display of both text and a graphical icon on one key of an intelligent keyboard, formed and operating in accordance with the present invention.

[0029] FIGS. 9A and 9B are pictorial diagrams of a computer CRT display and corresponding intelligent keyboard display, respectively, illustrating the operation of an intelligent keyboard in a computer-telephony integrated user console in response to a telephone call being answered, in accordance with the present invention.

[0030] FIGS. 10A and 10B are pictorial diagrams of a computer CRT display and corresponding intelligent keyboard display, respectively, illustrating the operation of an intelligent keyboard in a computer-telephony integrated user console in response to a telephone call being originated by a user, in accordance with the present invention.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

[0031] FIG. 1 is a simplified electrical block diagram of a computer-telephony integrated (CTI) user console formed in accordance with the present invention. The CTI user console includes a computer 2 which interfaces to a telephone line 4. The computer 2 further interfaces to a conventional audible telephone headset or handset 6. An important aspect of the present invention is that the CTI user console further includes an intelligent integrated keyboard 8.

[0032] Preferably, the computer 2 will take the form of an open architecture personal computer, conforming to International Standards Association (ISA) standards, such as that manufactured by IBM and other compatible manufacturers. The computer 2 further includes a central processing unit (CPU) 10 which coordinates the activities of the computer 2. Preferably, the CPU 10 is interfaced to a standard bus architecture such as an ISA bus. Preferably, also interfaced to the bus are at least one telephone line card 12, at least one serial port 14 and a standard keyboard controller 16. The computer 2 further includes a conventional cathode ray tube (CRT) 17 or other suitable computer display.

[0033] The telephone line card 12 provides an interface between the telephone line 4, the CPU 10 and the handset 6. The telephone line card 12 receives signals from the telephone line 4 such as ring detect, dial tone, caller identification (ID) information and other information related to the telephone system. In addition, the telephone line card 12 connects the telephone handset 6 to the telephone line 4 to allow a user to place and receive calls. In this configuration, the computer 2, telephone line card 12 and handset 6 interact to operate as a conventional telephone. A suitable telephone line card 12 is the BRI adapter-ISA, manufactured by Teloquent, Inc. of Billerica, Mass.

[0034] The intelligent integrated keyboard 8, illustrated in FIGS. 1-3, provides an enhanced user interface for the computer-telephony integration. The intelligent integrated keyboard 8 preferably includes a standard keyboard 20, such as a 101 key QWERTY keyboard commonly used with the IBM personal computer (PC) architecture. The standard

keyboard 20 is interfaced to the keyboard controller 16 by a conventional keyboard decoder 22, as is well known in the art.

[0035] The intelligent integrated keyboard 8 further includes a plurality of intelligent keys 24. FIG. 2 illustrates a preferred embodiment of the intelligent integrated keyboard 8. In this embodiment, twelve intelligent keys 24 are used. The intelligent keys 24 are physically arranged as three key groups with four keys per key group. It will be appreciated that while the keyboard is illustrated and described in the preferred, integrated form, the standard keyboard 20, 22 may be formed as a separate assembly from the intelligent keyboard 24, 30, 32.

[0036] The intelligent keys 24 are under the control of a dedicated intelligent key processor 30. The intelligent key processor 30 interfaces to the computer 2 via a serial communication port 32 associated with the intelligent integrated keyboard 8. Preferably, the serial communication port 32 interfaces with a serial port 14 associated with the computer 2.

[0037] Referring to the electrical block diagram of FIG. 3, each intelligent key 24 further includes a switch mechanism 26 and integral display means 28. A suitable integrated switch/display is model number LC24 manufactured by HE Electronics of Neunkirchen, Germany. The switch 26 is electrically coupled to a microprocessor 34 which is part of the intelligent key processor 30. While not illustrated, switch buffering and debouncing circuits, which are well known in the art of digital circuit design, can be interposed between each switch 26 and the microprocessor 34. Alternatively, software switch debouncing may be employed to improve closure detection.

[0038] The display means 28 of the intelligent keys 24 are connected to a display driver 36 which is electrically connected to the microprocessor 34 through a key decoder 35. In the LC24 switch/display, manufactured by HE Electronics, the display driver 36 is integral to the intelligent key 24. This display driver 36 includes a serial data interface to communicate with the microprocessor 34. The key decoder 35 includes an input/output (I/O) port electrically coupled to the microprocessor 34 and an address input port electrically coupled to the microprocessor 34. The key decoder 35 also includes a plurality of I/O ports which are connected to the display driver 36 of each intelligent key 24. The microprocessor 34 selects one of the intelligent keys by applying an appropriate digital signal to the address port of the key decoder 35. The key decoder 35 responds by electrically connecting the microprocessor to the selected intelligent key display driver 36.

[0039] Preferably, the display means 28 is a bit mapped, liquid crystal display (LCD). The LCD is a matrix of points, or pixels, which darken in response to a display driver signal for each pixel. To display graphics and text on the LCD, these symbols must first be converted to digital pixel-based bit maps. The pixel-based bit maps are received by the microprocessor 34 and are converted to appropriate digital signals for the display driver 36. The display driver 36 is responsive to the received digital signals and provides suitable driving signals to the LCD 28.

[0040] The intelligent key processor 30 further includes conventional memory elements of random access memory

(RAM) 38 and read only memory (ROM) 40. The RAM 38 and ROM 40 are electrically coupled to the microprocessor 34 in a manner which is conventional in the art of computer design. The RAM 38 is employed by the microprocessor 34 for temporary, volatile data storage and retrieval during operation. The ROM 40 is programmed with permanent storage such as the operating program for the microprocessor 34 and permanent data required by the keyboard 8. To reduce size and cost, the RAM 38 and ROM 40 can be integrated in a single integrated circuit with the microprocessor 34. A suitable device which includes the functions of the microprocessor 34, RAM 38 and ROM 40 may be selected from the 8051 family of microcontrollers commonly manufactured by several companies, including Intel and Signetics.

[0041] Through the serial port 32, the intelligent key processor 30 receives commands from the computer 2. These commands are generated in response to events detected on the telephone line 4 and applications software running in the computer 2. In response to these commands, the microprocessor 34 provides signals to the display driver 36 which effect a change in the display on the LCD 28. Preferably, the text and graphics displayed correspond to a function which is currently assigned to the switch 26 of each intelligent key. The microprocessor 34 is also responsive to closures in each switch 26 and provides signals through the serial port 32 to the computer 2 when such closures are detected. In this way, bidirectional communication between the integrated intelligent keyboard 8 and computer 2 is established.

[0042] One important aspect of the present invention is the software which runs on the computer 2 to control the CTI user console of the present invention. Furthermore, the keyboard, software and communications drivers of the present invention, in combination, preferably form a self-contained object component which can be embedded in other object-oriented software applications or used with object-oriented programming languages (e.g., Visual Basic® and Visual C++®, which are registered trademarks of Microsoft Corporation).

[0043] FIG. 4 is a simplified block diagram illustrating the relationship of the preferred functional software modules which collectively make up the control software of the present invention. Preferably, the control software will operate in an open architecture software platform which uses a standard messaging structure to support multiple applications. Exemplary software platforms include Windows®, JAVA (TM) and the Unix® operating systems (Windows is a registered trademark of Microsoft Corporation, JAVA is a trademark of Sun Microsystems, Inc. and Unix is a registered trademark of Novell, Inc.). While the operation of the software will be described in the context of the Windows® platform, operation in other operating environments is contemplated as within the scope of the present invention.

[0044] In the Windows® operating system, multiple software applications can be operational simultaneously. In addition, these multiple applications can exchange information and even be embedded in one another through the use of object linking and embedding (OLE). OLE is a standard Windows® program protocol which facilitates the interoperability between various software applications. While not described in detail, it is anticipated that other software

standard messaging protocols including the Active-X and OCX formats for Windows® and the Corba format for Unix®, could also be used to implement the present invention.

[0045] An overview of the functional modules of the software used to implement the CTI user console formed in accordance with the present invention will be described with reference to FIG. 4. The software includes an automation server module 50 and a computer-telephony integration (CTI) application software module 52. The CTI application software module 52 interfaces with the telephone line card 12 (FIG. 1). Preferably, the CTI application software module 52 complies with the OLE protocol and transfers information to and from the automation server module 50 using this protocol.

[0046] The OLE automation server module 50 preferably includes three virtual operating layers. The first layer is an OLE input/output layer 54. The OLE input/output layer 54 communicates with the CTI application module 52 and other OLE compatible applications 53 using the Windows® OLE standard. OLE compatible applications include word processors, data base programs, spreadsheets and the like. The OLE input/output layer 54 is thus considered a public layer of the automation server 50 in that other software applications 52, 53 can interact with this functional layer.

[0047] Directly interfaced with the OLE input/output layer 54 is the second layer of the automation server 50, which is a private layer 56. The private layer 56 receives commands from the OLE input/output layer 54 and forms appropriate commands for the intelligent integrated keyboard 8. The private layer 56 is dedicated to the control of the intelligent integrated keyboard 8 and can only be accessed by software applications running on the computer 2 through the OLE input/output layer 54.

[0048] The third layer is a communications layer 58 which controls communication between the intelligent integrated keyboard 8 and the automation server module 50. The communications layer 58 receives commands from the private layer 56 and formats these commands into a protocol which is compatible with the intelligent integrated keyboard 8. Once the commands are formatted into the appropriate protocol, the communications layer 58 interfaces with the hardware serial port 14 and ultimately to the intelligent integrated keyboard 8.

[0049] The communications layer 58 also receives data from the intelligent integrated keyboard 8 indicative of switch 26 closures. This data is transferred to the private layer 56. The private layer 56 determines which software application 52, 53 is currently assigned to the key being depressed and routes the switch press information through the OLE input/output layer 54 to the appropriate OLE software application 52, 53.

[0050] The intelligent integrated keyboard 8 provides an enhanced user interface for the CTI user console. This enhanced user interface is achieved by dynamically reconfiguring the functionality and the associated text and graphical icon displayed on the intelligent keys 24 in response to detected events. The events may take the form of signals from the telephone line 4, commands from the CTI application software 52 or commands from other OLE application software 53. The text and graphical icons can also be

animated to further enhance the user interface. In addition, the intelligent integrated keyboard 8 may be partitioned into key groups, if necessary, with priorities established by the automation server module 50 among various software applications 52, 53 competing to invoke one or more intelligent key 24 of the integrated intelligent keyboard 8.

[0051] FIG. 5 further illustrates the preferred operating modules which form the private layer 56 of the automation server module 50. The main operating module of the private layer 56 is the intelligent keyboard (IKB) manager 60. The IKB manager 60 receives event information from the OLE input/output layer 54 and manipulates this information to control the intelligent integrated keyboard 8. The IKB manager 60 also receives switch 26 closure data from the intelligent integrated keyboard 8 and routes the data to the appropriate software application 52, 53.

[0052] As previously stated, the intelligent keys 24 of the intelligent keyboard 8 can be reassigned in total, in groups of keys or individually. Key reassignments are typically event driven. Events may be detected from several sources, such as the telephone line card 12, the CTI application software 52, or other software application 53. Typically, these reassignments can be characterized as progressive command layers, i.e., the current key assignments predictably resulting from the previous assignment and a detected event. To facilitate the management of the various levels of intelligent key 24 functionality and display, the private layer 56 employs a level manager software module 61 operating between the IKB manager 60 and the OLE input/output layer 54.

[0053] Preferably, the level manager software module 61 includes a primary level manager 62, a dynamic level manager 64 and a multi-group-key level manager 66. Operating through the OLE input/output layer 52, the level manager software module 61 receives requests for intelligent key 24 assignments from the CTI application software 52 and various other application software packages 53. These requests are routed to the IKB manager 60 which determines if these requests can be serviced. The IKB manager 60 prioritizes conflicting requests for the intelligent keys 24 and makes the assignments according to either a dynamic or a predetermined priority hierarchy.

[0054] As previously discussed in connection with FIG. 3, each intelligent key 24 includes a bit mapped liquid crystal display (LCD) 28. To display graphics and text on the LCD 28, these symbols must first be converted to digital pixel-based bit maps which activate or deactivate the individual pixels on the LCD 28. To facilitate this conversion, the IKB manager 60 employs a picture library module 68 and a text converter module 70.

[0055] The picture library module 68 includes a digital storage table which contains a plurality of preestablished bit mapped graphical icons which can be selected by the IKB manager 60 at the request of an OLE application 52, 53. These bit mapped graphical icons are passed by the IKB manager 60 to the IKB serial communications layer 58 and ultimately to the intelligent key processor 30 for display on the LCD 28. In addition, the picture library module 68 can also receive a location from an OLE application 52, 53 indicating where a graphical icon is remotely stored. Using this location information, i.e. disk directory and file name, the picture library module 68 can import the desired graphi-

cal icon and add this icon to the storage table. The text converter 70 receives (ASCII) text characters from the IKB manager 60 and converts each character into a bit mapped graphic suitable for display on an LCD 28.

[0056] To further enhance the user interface provided by the intelligent integrated keyboard 8, the private layer 56 further includes an animation manager 72. In response to a request from the CTI application module 52 or other OLE software application 53, the IKB manager 60 transfers a plurality of related bit mapped graphics from the picture library 68 to the animation manager 72. These related bit mapped graphics are selected such that when presented sequentially, an animated sequence will be displayed on at least one of the intelligent keys 24.

[0057] Once the animation manager 72 receives the sequence of graphics from the IKB manager 60, the animation manager 72 operates directly with the IKB serial communications layer 58 to generate the desired animation sequence. The use of the animation manager 72 removes the burden of animation from the IKB manager 60 and allows the IKB manager 60 to perform the core keyboard control functions.

[0058] The interaction between the CTI applications software 52 and the integrated intelligent keyboard 8 will now be further described in connection with a series of illustrations in FIGS. 6-10. FIGS. 6-10 sequentially illustrate an exemplary series of events and responses typical of the present invention.

[0059] FIG. 6A is a pictorial diagram illustrating an exemplary CRT 17 display of a Windows® based software program having a toolbar 80 with a number of function commands displayed. FIG. 6B is an illustration of the intelligent integrated keyboard 8 with several of the intelligent keys 24 displaying the same commands which are displayed on the toolbar 80 of the CRT 17 display. In addition, one intelligent key 82 is illustrated with a telephone graphic icon displayed along with illustrative descriptive text, TELE.

[0060] As is conventional with Windows® software, the various functions of the application software can be selected by directing a digital pointer, i.e., a mouse, to one of the commands on the toolbar and depressing a button to select that function menu. Alternatively, with the use of the integrated intelligent keyboard 8 of the present invention, each function menu may be selected simply by depressing the appropriate intelligent key 24 displaying the desired function. This allows the user of the CTI user console to quickly select a desired function with only minimal hand movement and required dexterity. This cooperative relationship between the CRT 17 and the intelligent keyboard 8 is achieved through the interaction between the application software 53 and the automation server 50 formed in accordance with the present invention.

[0061] In response to an intelligent key 24 being depressed, a signal is sent from the intelligent keyboard 8 to the computer 2 indicative of this event. This signal, in the form of serial data, is received through the serial port 14 and is routed to the IKB serial communications layer 58.

[0062] The IKB serial communications layer 58 routes the key press information to the IKB manager 60 which correlates the detected event to an assigned software application.

Working through the OLE input/output layer 54, the IKB manager 60 interfaces with the OLE applications 52, 53 which is assigned to the currently pressed key 26. Preferably, the OLE application 52, 53 responds to this event in a manner consistent with the same function being selected by the digital pointer.

[0063] FIG. 7A illustrates a typical pull down menu which results when a "File" command is selected. This command can be selected from the exemplary. Windows® application software toolbar or by depressing the intelligent key 24 displaying the "File" icon or text on the intelligent keyboard 8. Once this function is selected, a new series of commands is made available in a pull down menu 84 by the application software 53. FIG. 7B illustrates that the IKB manager 60 has reassigned several of the intelligent keys 24 such that the new commands are available on the integrated intelligent keyboard 8. These commands may be selected simply by depressing the intelligent key 24 displaying the appropriate command.

[0064] It should be noted that key 82, displaying the telephone icon, has not changed functions. In this case, the multi-group-key level manager 66 has requested an assignment of this key to the CTI application module 52 which is running in the background. The IKB manager 60 has given this function priority over the general applications software 53 currently running in the foreground. Preferably, the CTI application module 52 can be immediately invoked by depressing this key 82.

[0065] FIGS. 8A and 8B illustrate a similar state of the keyboard 8 and CRT 17 display as in FIGS. 7A and 7B respectively. However, in FIGS. 8A and 8B the CTI application module 52 has detected an incoming call from the telephone line card 12. In response to the detected incoming call, the CTI application module 52 interrupts the IKB manager 60 and requests a change in both the function and display in the telephone key 82.

[0066] FIG. 8C is an enlarged illustration showing an exemplary icon and text associated with the detection of an incoming call. The CTI application 52 can further instruct the IKB manager 60 to animate this icon and change the color of this key in response to the incoming call. For example, the key 82 can change from green to red and musical notes can alternatively be displayed and removed to correspond to the ringing of the phone. Preferably, the CTI application 52 can determine the identity of the caller from the telephone line card 12 and display the ID information on the key as well.

[0067] In response to a signal that key 82 has been depressed, the CTI application software 52 directs the telephone line card 12 to couple the handset 6 to the telephone line 4, thus answering the call. In addition, a CTI application menu is enabled on the intelligent integrated keyboard 8. FIG. 9B illustrates exemplary key assignments established by the CTI application module 52 in response to the incoming call being answered. Typical key assignments may include: the status of a second telephone line 86 (requiring an additional line card 12 and telephone line 4), a function key to place the active call on hold 88, a function key to transfer the active call to another number 90, a function key to transfer the active call to voice mail 92, a function key to hang up the active call 94 and the like. In addition, a function

key can be assigned on the intelligent keyboard **8** to activate additional software applications, such as a customer service menu **96**.

[0068] In response to each of these function keys being depressed, a further layer of key functions will then be displayed. Further, each of the keys may include animation to further enhance the user interface. For example, if the caller was placed on hold by depressing the hold function key **88**, that key **88** could change color and an animated timer could be added as a graphic under the control of the animation manager **72** as a reminder to the user.

[0069] In the illustrations of **FIGS. 9A and 9B**, the application software **53** which was operating at the time the incoming call arrival is still available in the active window of the computer screen and the functionality of the intelligent keys **24** previously assigned to this program are still available. In this example, no conflict between the CTI applications module **52** and the application software **53** occurred in their respective intelligent key **24** assignments. If multiple applications were competing for one or more keys, the IKB manager **60** would make appropriate key assignments based on a priority hierarchy. These assignments would be passed to the applications **52, 53** through the level managers **62, 64, 66** and OLE input/output layer **54**.

[0070] In addition to receiving a call, a user can initiate an outgoing call by depressing key **82** to activate an outgoing call menu of the CTI application module **52**. **FIGS. 10A and 10B** illustrate exemplary displays on both the CRT **17** display and intelligent integrated keyboard **8** in response to key **82** being depressed in the absence of an incoming call. Exemplary functions assigned to the intelligent integrated keyboard **8** in response to the CTI application module **52** include: dialing preassigned numbers **98, 100, 102**, calling the last number previously dialed **104**, requesting the operator **106**, opening a computer-based telephone directory, manually entering a number **110** on the standard keyboard **20** and the like. In addition, the CTI application software module **52** preferably brings a minimized icon to the foreground of the CRT **17** display to facilitate access to this application by use of a digital pointer. It shall be noted that this functionality has been activated without interrupting the function of the other active software application **53** operating in the foreground.

[0071] When an intelligent key **24** with a preassigned telephone number is activated by the user, the automation server **50** will direct the CTI application module **52** to

initiate an outgoing call. In response, The CTI application module **52** directs the telephone line card **12** to seize the telephone line **4** and dial the number which is assigned to the selected key. The CTI application module **52** further directs the telephone line card **12** to connect the handset **6** to the telephone line **4**. Preferably, the CTI application module **52** directs the automation server **50** to assign appropriate functions and displays to one or more intelligent keys **24**. Such functions include hanging up the call and transferring the call to another user console.

[0072] It is to be appreciated that the number of key function layers illustrated, and the specific functions associated with the individual keys in the previous examples are merely exemplary. The intelligent integrated keyboard **8** of the present invention is capable of fully programmable operation with a large number of possible variations. The levels of functionality for the intelligent integrated keyboard **8** are a function of the software complexity of the private layer **56** and the hardware capabilities of the host computer **2**.

[0073] Through the use of an intelligent keyboard operating under the control of software formed in accordance with the present invention, an improved computer-telephony integrated (CTI) user console is formed. The improved CTI user console provides an enhanced user interface to both a telephone system and data processing system without requiring the use of a digital pointer, such as a mouse. This enhanced user interface provides for commands to be easily entered by a user without removing his or her hands from a main computer keyboard. Further, through the use of a standard software interface, such as OLE, the enhanced user interface can control standard applications software such as wordprocessors, database programs, spreadsheet programs and the like.

[0074] A computer program of the operation of the computer-telephony user console in accordance with the present invention is provided herewith and is incorporated herein as part of the disclosure of the invention.

[0075] Although illustrative embodiments of the present invention have been described herein with reference to the accompanying drawings, it is to be understood that the invention is not limited to those precise embodiments, and that various other changes and modifications may be effected therein by one skilled in the art without departing from the scope or spirit of the invention.

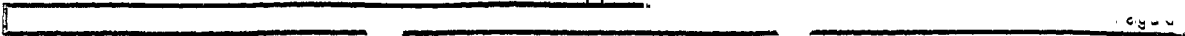
7
Hex file for Figure 3 (all items)

:100B06000008040C020A060E0109050D030B070FA7
:0A0BD60000FEFDFBF7EFDFFB7F001C
:10002600C206756303120ADFD20AD209753CFF7550
:10003600354175416C300603756302E4F540F53DC4
:1000460075B0031200037580FF858068E568120CA1
:1000560000007600007E01008602008E04009608ED
:10006600009E1000A62000AE4000B6800000000BE34
:1000760075140112082E804675140212082E803E51
:1000860075140412082E803675140912082E802E57
:1000960075140E12082E802675141312082E801E53
:1000A60075141C12082E801675143912082E800E2F
:1000B60075140912082E800675141C12082ED2BC5F
:1000C6001208DAC298C299756E01E4F54212001B55
:1000D600C20CD20F753E50753728C213C208753B45
:1000E600FFE4F567120AD0751A02120970C207C238
:1000F60011E4F564F565F539F53AC210C20DC2147E
:0E010600C205120114E542B401F802003022D5
:1001140012067A300808751A82120970C208301360
:10012400051209CFC21330101030090A751A0212D1
:100134000970C2108003120902300716751A0612DC
:100144000970C207756304754005C20C75416C756E
:100154003CFF301113751A15120970C21175630230
:10016400D20675416C753CFF300F0B120A03C20FA7
:1001740075416C753CFF300A1120050EE54F600A8D
:10018400F51B853F1C12098CC20A300B0A856A33A1
:08019400120B7BC20BD20A2200
:03000B0002019C53
:10019C00C0E0C0F0C083C082C0D0C000C001C002AB
:1001AC00C003C004C005C006C007C3E53C940130C1
:1001BC00D202B2E720E702153CE53C700D753CFF1E
:1001CC0075416CD211D20F754201C3E541940130D7
:1001DC00D202B2E720E7021541E541700AD2107550
:1001EC00416C200A02D205E56760021567758CB870
:1001FC00758AB8E581D395354003858135D007D014
:10020C0006D005D004D003D002D001D000D0D0D07D
:08021C0082D083D0F0D0E03263
:03002300020224B2
:10022400C0E0C0F0C083C082C0D0C000C001C00222
:10023400C003C004C005C006C007C309905756E012F
:10024400C299209803020657859908753C6C75413C
:10025400FFC298E53D120C00027E0002D00102FEAE
:100264000203CE0304E404051B0A05DC0B0287F039
:100274000290F1029EF200000246E50870C4753D4A
:10028400F080BFE50870BB753DF180B6E5087005E8
:10029400753DF280ADE4F53D80A8E50860A4753DA8
:1002A40001E508C394C0500AD211757002D20FE45C
:1002B400F53DE508543FF543E543D394074083D225
:1002C40011757003D20FE4F53D020246753D0285B7
:1002D4000844757100854372AE71AF727C017D0074
:1002E400120BEC8E718F72E5442572F572E4357150
:1002F400F571E4F564F565020246E572457170082E
:10030400D211E4F53D020246E572157270021571D0
:10031400850866E5082565F565E43564F564E566F4

:1009320004F5171208DEC21075416CE56364047099
:0F0942000D1540E5407007756302D206D20F22F3
:10095100E4F5171208DEE4F5171208DEE4F51712C4
:0F09610008DEE4F5171208DE7517C01208DE2253
:100970001209517517011208DE851A171208DEE4F4
:0C098000F5171208DE851A171208DE2297
:10098C001209517517021208DE851B171208DE8535
:10099C001C171208DEE51B251CF51EE43400F51DA2
:1009AC00D3E51E94FFE51D9400400875170112084D
:1009BC00DE8006E4F5171208DEE51B251CF5171280
:0309CC0008DE2220
:1009CF001209517517051208DE7517031208DE7527
:1009DF0017021208DE75173D1208DE75170112088F
:1009EF00DE7517021208DEE4F5171208DE751745DB
:0409FF001208DE22DA
:100A0300E4F51FE51FC3940C501C851F111207C882
:100A1300851F23E4F524120A2A851F127513031276
:070A2300080C051F80DD2215

:100A2AC085230F12000EE4F50BF50C120783750BE4
:100A3A0080750C01120783E4F525E525C3946C50F3
:0F0A4A000D85240B750C01120783052580EC2206
:100A5900E526D394074004D2028002C202E5265457
:100A690007F5273002058569268003856B26E5276A
:100A7900900BD793FFF45226E526D3940040037FC9
:050A890001227F0022A4
:100A8E00E528D394074004D2048002C204E528541A
:100A9E0007F529E529900BD793F52A632AFF200342
:100AAE0003E4F52A30040EE529900BD7935269E53D
:100ABE002A4269800CE529900BD7935268E52A42A6
:020ACE006B2299
:0F0AD0005389F0438901D28C758CB8758AB8228E
:100ADF00E4F52BC3E52B941EA2E730D201B3500BE4
:100AEF007445252BF8E4F6052B80E3E4F53FF56A0D
:090AFF00C20B120B51120B312243
:100B0800752D0A7445252DF88607EF700C74452558
:100B18002DF8A62C752D1E8002052DC3E52D941ED8
:090B2800A2E730D201B340DB2248
:100B3100E4F52EE52EC3941D50127446252EF88639
:100B4100077445252EF8A607052E80E7E4F56222F5
:100B5100752F1DE52FD3940140127444252FF8867B
:100B6100077445252FF8A607152F80E7E4F54522E0
:0A0B7100C3E5309531F532AF3222B2
:100B7B00E53F6533603DC3E533963FA2E730D201D6
:100B8B00B360025033C3E53F9533F534C3E534947A
:100B9B000A300202B2E720E703753409E4F56FE5BA
:100BAB006FC395345012E54E6005120851153F057E
:0B0B8B006F80EC120B31053F7F002221
:030C0000020BE010
:0C0BE000787FE4F6D8FD758172020026D3
:100BEC00EFF88DF0A4FFEDC5F0CEA42EFEEC88F04E
:040BFC00A42EFE2203
:100C0000CD083D082F8E4937012740193700DA3A383

0959430 US 4,471



:100C100093F8740193F5828883E473740293686097
:060C2000EFA3A3A380DF97
:00000001FF

00000001FF

```

// ChildFrm.h : interface of the CChildFrame class
//
///////////////////////////////////////////////////////////////////

class CChildFrame : public CMDIChildWnd
{
    DECLARE_DYNCREATE(CChildFrame)
public:
    CChildFrame();

// Attributes
public:

// Operations
public:

// Overrides
    // ClassWizard generated virtual function overrides
    ///{{AFX_VIRTUAL(CChildFrame)
    virtual BOOL PreCreateWindow(CREATESTRUCT& cs);
    ///}}AFX_VIRTUAL

// Implementation
public:
    virtual ~CChildFrame();
#ifdef _DEBUG
    virtual void AssertValid() const;
    virtual void Dump(CDumpContext& dc) const;
#endif

// Generated message map functions
protected:
    ///{{AFX_MSG(CChildFrame)
    // NOTE - the ClassWizard will add and remove member functions here.
    // DO NOT EDIT what you see in these blocks of generated code!
    ///}}AFX_MSG
    DECLARE_MESSAGE_MAP()
};

/////////////////////////////////////////////////////////////////

```

2001-05-10 09:59:10

```
// Machine generated IDispatch wrapper class(es) created with ClassWizard
////////////////////////////////////

// ICallback wrapper class

class ICallback : public COleDispatchDriver
{
public:
    ICallback() {} // Calls COleDispatchDriver default constructor
    ICallback(LPDISPATCH pDispatch) : COleDispatchDriver(pDispatch) {}
    ICallback(const ICallback& dispatchSrc) : COleDispatchDriver(dispatchSrc) {}

// Attributes
public:

// Operations
public:
/*
    unsigned long AddRef();
    unsigned long Release();
*/
    void Notify(short *cmd);
};
```

```

// CSCommunication.h : header file
//
#ifndef C_S_COMMUNICATION_H
#define C_S_COMMUNICATION_H

#include <afxmt.h>

const cMaxPorts = 4;
////////////////////////////////////
// CCSCCommunication window

class CCSCCommunication : public CWnd
{
// Construction
public:
    CCSCCommunication();

// Attributes
public:
    DWORD m_wBaudRate;
    BYTE m_bByteSize;
    BYTE m_bParity ;
    BYTE m_bStopBits ;

    short m_nComPort;
protected :

private:
    static int m_nInit;
    static CCSCCommunication* m_pCCSC[cMaxPorts];
    static HWND m_hWND[cMaxPorts];
    static CEvent m_eKill[cMaxPorts];

    HANDLE m_hCom;
    CWinThread* m_pReadThread;

// Operations
protected :
    virtual void ReadData(unsigned char * pData, DWORD nLength, HWND hwnd);
private :
    void SerialRead(HANDLE hCom, int num, HWND hwnd);

public:
    static UINT ReadThreadProc(LPVOID pParam);

    void Create(void);
    BOOL OpenPort(short sComPort);

    BOOL ClosePort(void);
    int WriteData(unsigned char* pData, DWORD nLength);

```

```
// Overrides
// ClassWizard generated virtual function overrides
//{{AFX_VIRTUAL(CCSCCommunication)
//}}AFX_VIRTUAL

// Implementation
public:
    virtual ~CCSCCommunication();

// Generated message map functions
protected:

    //{{AFX_MSG(CCSCCommunication)
    // NOTE - the ClassWizard will add and remove member functions here.
    //}}AFX_MSG
    DECLARE_MESSAGE_MAP()
};

#endif
//=====
```

2001/10/25 04:59:00

```

// CLcmBitmap.h : header file
//

////////////////////////////////////
// class for store LCM bitmap information
// See CLcmBitmap.cpp for the implementation of this class
//

#ifndef CLCMBITMAP_H
#define CLCMBITMAP_H

#define LC16LENGTH 64
#define LC24LENGTH 108

class CLcmBitmapData
{
public: // data member
    WORD m_nChecksum;
    CString m_strName;
    BYTE m_cLength;
    BYTE *m_pData;

public: // functions
    CLcmBitmapData();
    CLcmBitmapData(BYTE *data, BYTE length, LPCTSTR name);
    virtual ~CLcmBitmapData();
};

class CLcmBitmap : public CLcmBitmapData
{
private: // data member
    WORD m_wCounter;
public: // functions
    CLcmBitmap();
    CLcmBitmap(BYTE *data, BYTE length, LPCTSTR name);
    ~CLcmBitmap();
    CLcmBitmap *IncreaseCounter();
    CLcmBitmap *DecreaseCounter();
};

#endif

```

```

// LcmCommand.h : header file
//

#ifndef _LCM_COMMAND_H
#define _LCM_COMMAND_H

#include "LcmCommunication.h"
#include "LcmBitmap.h"

/* the const for command
LCM_SET_COLOR
LCM_SET_BITMAP_LC16
LCM_SET_BITMAP_LC24
*/
#define LCM_BITMAP_INDEX 0x10
#define LCM_TEXT 0x11

struct SLcmText
{
    SLcmText(LPCTSTR line1, short flags1,
             short pointSize1, LPCTSTR typeface1,
             LPCTSTR line2, short flags2,
             short pointSize2, LPCTSTR typeface2);

    SLcmText(SLcmText *p);
    ~SLcmText();

    char * m_cLine1;
    char * m_cLine2;
    char * m_cFont1;
    char * m_cFont2;
    short m_cFlag1;
    short m_cFlag2;
    short m_cPointSize1;
    short m_cPointSize2;
};

struct SLcmCommand
{
    SLcmCommand();
    SLcmCommand(SLcmCommand *p);
    ~SLcmCommand();

    void BitmapIndex(unsigned char keyNum, const char *name);
    void Bitmap(unsigned char keyNum, CLcmBitmap *lcmBitmap);
    void Text(unsigned char keyNum, LPCTSTR line1, short flags1 = 0,
             short pointSize1 = 10, LPCTSTR typeface1 = NULL,
             LPCTSTR line2 = NULL, short flags2 = 0,
             short pointSize2 = 0, LPCTSTR typeface2 = NULL);
    void Color(unsigned char keyNum, unsigned char color);

    unsigned char m_cCmd;
    unsigned char m_cKeyNum;

```

```
union
{
    CLcmBitmap FAR *m_pLcmBitmap;
    unsigned char m_cColorNum;
    SLcmText FAR * m_sText;
    char FAR *    m_pBitmapIndex;
} m_uData;

};

#endif
```

FIG. 3 is a block diagram of the system.

```

// LcmManager.h : header file
//

#ifndef _LCMMANAGER_H
#define _LCMMANAGER_H

#include "LcmServerDoc.h"

#include "LcmSCommunication.h"
#include "PictureLibrary.h"
#include "TextConvert.h"
#include "LcmCommand.h"
#include <afxtempl.h>

struct SCallbackStruct
{
    void (*m_Prc)(void *pInstance, BYTE bCmd);
    void *m_pPrcData;
};

class CLcmManager
{
// Construction
public:
    CLcmManager ();

// Attributes
public:
protected :

private:
    CLcmSCommunication m_cCom;
    CPictureLibrary m_cPictureLibrary;
    CTextConvert m_cTextConvert;

    int m_nReady;
    int m_nStyle;
    int m_nAttachCounter;
    CList<SLcmCommand *, SLcmCommand*> m_cCmdQueue;

    CList<SCallbackStruct*, SCallbackStruct*> m_cCallbackList;

// Operations
protected :

private :
    void CallBack(BYTE bCmd, UINT wParam);

public:
    static void CallBackStatic(void *pInstance, BYTE bCmd, UINT wParam);

// interface operations

```

```
        BOOL OpenPort(short nComPort, int style,
            void (*prc)(void *pInstance, BYTE bCmd), void *pData);
        short Attach(int style,
            void (*prc)(void *pInstance, BYTE bCmd), void *pData);

        int ClosePort(void *pData);

        void Reset(BOOL bKeepData = FALSE);
        BOOL GetKeyInfo(DWORD &dw_id, BYTE &c_type);
        short LoadKeys(SLcmCommand *sCmd, int nLength);

// Implementation
public:
    ~CLcmManager();

};

#endif

////////////////////////////////////
```

FIG. 2 is a block diagram of the system.

```

// LcmSCommunication.h : header file
//
#ifndef LCM_S_COMMUNICATION_H
#define LCM_S_COMMUNICATION_H

#include "CSCommunication.h"
#include "LcmBitmap.h"

////////////////////////////////////
// CLcmSCommunication window

#define WM_LCMDATA WM_USER + 7           // user defined symbol

#define LCM_PING           0x01
#define LCM_POWER_ON_RESET 0x02
#define LCM_KEYBOARD_ID   0x03
#define LCM_ACK            0x06
#define LCM_NACK           0x15
#define LCM_KEY_TYPE_LC16 0x81
#define LCM_KEY_TYPE_LC24 0x82
#define LCM_KEY_PRESS      0x11
#define LCM_UNKNOW_CMD     0x10

#define LCM_READY          0x20
#define LCM_NO_ACTIVE      0x21

// define first parameter at LCM_PING
#define LCM_NO_CMD         0x01
#define LCM_TESTMODE       0x02
#define LCM_IDLE           0x04
#define LCM_POSSIBLE_STATE 0x03

// define first param at LCM_UNKNOW_CMD
#define LCM_FRAME_LENGTH_ERR 0x01
#define LCM_CHECKSUM_ERR    0x02
#define LCM_CMD_ERR        0x03
#define LCM_2BYTES_DATA_ERR 0x04
#define LCM_5BYTES_DATA_ERR 0x05
#define LCM_OTHER_ERR      0x06

// define states
#define LCM_STATE_START_CMD 0x01
#define LCM_STATE_FRAME_LENGTH 0x02
#define LCM_STATE_2BYTES_DATA 0x04
#define LCM_STATE_5BYTES_DATA 0x05
#define LCM_STATE_1BYTE_DATA 0x06
#define LCM_STATE_END_DATA 0x07
#define LCM_STATE_DO_CHECKSUM 0x08
#define LCM_STATE_END_CMD 0x09

#define LCM_FIRST_BYTE_FRAME_LENGTH 0xC0

```

```

// define the constant value for write
#define LCM_WRITEHEAD 0x00
#define LCM_LENGTH_START 4
#define LCM_DATA_START 6

// define the command for writing to lcm
#define LCM_RESET 0x01
#define LCM_GET_KEY_TYPE 0x02
#define LCM_GET_KEYBOARD_ID 0x03
#define LCM_SET_COLOR 0x04
#define LCM_SET_BITMAP_LC16 0x05
#define LCM_SET_BITMAP_LC24 0x07
#define HOST_TO_LCM_ACK 0x06
#define LCM_SEND_NEXT_KEYPRESS 0x08
#define LCM_PING_SYNC 0x09

// define the background color
#define LCM_COLOR_OFF 0x00
#define LCM_DARK_GREEN 0x03
#define LCM_BRIGHT_GREEN 0x33
#define LCM_DARK_RED 0x0C
#define LCM_BRIGHT_RED 0xCC
#define LCM_DARK_ORANGE 0x0F
#define LCM_BRIGHT_ORANGE 0xFF
#define LCM_REDDISH_ORANGE 0xCF
#define LCM_GREENISH_ORANGE 0x3F

#define LCM_NO_COLOR 0x01

#define LCM_DATA_TIMER_ID 2
#define LCM_ACTIVE_TIMER_ID 3

#define LCM_MAX_NO_ACTIVE_TIME 3000

////////////////////////////////////
// The define of CProtocolManager Class
// It is a abstract class

// #define PROTOCOL_NOT_IN_PROCESS 0x01
// #define PROTOCOL_READY 0x02
// #define PROTOCOL_RESET 0x03
#define PROTOCOL_REQUIREST_INFO 0x04
#define PROTOCOL_KEY_DATA_MESSAGE 0x05

#define PROTOCOL_NOT_IN_PROCESS 0x10
#define PROTOCOL_SYNC_INIT 0x11
#define PROTOCOL_RESET 0x12
#define PROTOCOL_SYNC_POWER_ON 0x13
#define PROTOCOL_REQUIREST_KEY_TYPE 0x14
#define PROTOCOL_REQUIREST_KEY_ID 0x15
#define PROTOCOL_REMOVE_KEYS 0x16
#define PROTOCOL_SYNC_REMOVE_KEYS 0x17

```

```

#define PROTOCOL_READY      0x20
#define PROTOCOL_WAITING    0x21
#define PROTOCOL_WAITING_BUFFERED 0x22
#define PROTOCOL_SYNC_NORMAL 0x23
#define PROTOCOL_RESET_CMD  0x24
#define PROTOCOL_SYNC_RESET 0x25
#define PROTOCOL_SYNC_RESET_ONE_MORE 0x26

#define PROTOCOL_KEY_PRESSED 0x30
#define PROTOCOL_SYNC_KEY_PRESSED 0x31
struct SLcmCommand;

class CProtocolManager
{
// Construction
public:
    CProtocolManager ();

// Attributes
public:

protected :

private:
    BYTE m_nProtocolState;

    BOOL m_bACKFlag;
    BOOL m_bPonFlag;
    BOOL m_bKeyTypeFlag;
    BOOL m_bKeyboardIDFlag;
    BOOL m_bResetFlag;

//    BOOL m_bPowerOnResetFlag;
//    BOOL m_bKeyTypeDoneFlag;
//    BOOL m_bDataStackFlag;
//    BOOL m_bNormalFlag;

    BOOL m_bActiveFlag;
    BOOL m_bKeyPressed;
    BOOL m_bKeyTranNumberFlag;
    BYTE m_nLastProtocolState;
    int m_nLastKeyPressTransactionNumber;

// Operations
protected :

    void OpenPortEvent(void);
    void ClosePortEvent(void);
    void EventProcess(BYTE bCmd, UINT wParam);

```

```

void TimeOutEvent(void);
void ResetEvent(BOOL bKeepData);
void KeyDataEvent(SLcmCommand *cmd, int len);
BOOL CheckActive(void);

private:
void ACKEvent(void);
void NACKEvent(void);
void PowerOnEvent(void);
void KeyTypeEvent(BYTE bParam);
void KeyboardIDEvent(UINT wParam);
void PingEvent(BYTE bParam);
void KeyPressEvent(BYTE bKey, BYTE bParam);

virtual void SetKeyType(BYTE bParam) = 0;
virtual void SetKeyboardID(UINT wParam) = 0;

virtual void SendACK(void) = 0;
virtual void SendNACK(void) = 0;
virtual void SendHostPing(void) = 0;
virtual void SendReset(void) = 0;
virtual void SendKeyTypeReq(void) = 0;
virtual void SendKeyboardIDReq(void) = 0;
virtual void SendLastKeyPressNumber(BYTE nTransactionNumber) = 0;
virtual BOOL SendAll(void) = 0;
virtual void KeyboardReady(void) = 0;
virtual BOOL SendStackData(void) = 0; // send data in stack
virtual void SendData(SLcmCommand *cmd, int nLength) = 0;
// on stack data
virtual void SaveData(SLcmCommand *cmd, int len) = 0;
// save data to stack
virtual BOOL FillAllStack() = 0; // move all buttons to stack
virtual void KeyPress(BYTE param) = 0; // call the callback
virtual void ClearAll(void) = 0; // clear stack and button data
virtual void ClearTimer(void) = 0;
virtual BOOL CheckConnect(void)=0;

public:

// interface operations

// Overrides
// ClassWizard generated virtual function overrides
//{{AFX_VIRTUAL(CProtocolManager)
//}}AFX_VIRTUAL

// Implementation
public:
virtual ~CProtocolManager();

```

```

// Generated message map functions
protected:

    //{AFX_MSG(CProtocolManager)
    //}AFX_MSG
//    DECLARE_MESSAGE_MAP()
//};

////////////////////////////////////
////////////////////////////////////
#define NUMBER_OF_KEYS    12
#define HALF_SECOND      500
#define ONE_SECOND       1000

class CLcmSCommunication : public CCSCommunication, CProtocolManager
{
// Construction
public:
    CLcmSCommunication ();
    void Create (
        void (*prc)(void *pInstance, BYTE bCmd, UINT wParam),
        void *pData);

// Attributes
public:

protected :

private:

// parameter for read
    BOOL b_OSWindow95;
    int m_nState;

    int m_nhead;
    DWORD m_dwKeyboardID;
    int m_nTypeNum;
    unsigned char m_cChecksum[2];
    int m_nChecksumNum;
    unsigned short m_sMyChecksum;
    unsigned char m_cFramLength[2];
    int m_nFramLengthNum;
    UINT m_wParameter1;
    LONG m_lParameter2;

// main thread -- for write
    unsigned char m_cWriteBuffer[2048];
    DWORD m_dwSaveKeyboardID;
    BYTE m_cSaveKeyType;

```

```

        BOOL m_bReady;
        BOOL m_bConnect;

        UINT m_nTimerID;

        CLcmBitmap *m_pBitmap[NUMBER_OF_KEYS];
        BYTE      m_cColor[NUMBER_OF_KEYS];

        CLcmBitmap *m_pStackBitmap[NUMBER_OF_KEYS]; // Map on the stack

        BYTE      m_cStackColor[NUMBER_OF_KEYS];

        void (*m_Prc)(void *pInstance, BYTE bCmd, UINT wParam);
        void *m_pPrcData;

// Operations
protected :

private :
        void ReadData(unsigned char * pData, DWORD nLength, HWND hwnd);
        void GotoLCM_STATE_START_CMD(void);

// Operation in main thread

        int WriteDataWithTimer(BYTE *buf, WORD length, UINT interval);
        void SendOneByteData(unsigned char c);
        void SendOneByteDataWithTimer(unsigned char c, UINT interval);
        void SendTwoBytesDataWithTimer(unsigned char c, BYTE n, UINT interval);
        unsigned short BuildSetColor(unsigned char *buf,
                                     unsigned short &checksum, unsigned char keyNum,
                                     unsigned char colorNum);
        unsigned short BuildSetBitmapLC16(unsigned char *buf,
                                     unsigned short &checksum, unsigned char keyNum,
                                     CLcmBitmap *bitmap);
        unsigned short BuildSetBitmapLC24(unsigned char *buf,
                                     unsigned short &checksum, unsigned char keyNum,
                                     CLcmBitmap *bitmap);

        void SetKeyType(BYTE bParam);
        void SetKeyboardID(UINT wParam);

        void SendReset(void);
        void SendResetOnly(void);
        void SendACK(void);
        void SendNACK(void);
        void SendKeyTypeReq(void);
        void SendKeyboardIDReq(void);
        void SendLastKeyPressNumber(BYTE nTransactionNumber);
        void SendHostPing(void);

        BOOL SendAll(void);
        void KeyboardReady(void);

```

```

        BOOL SendStackData(void); // send data in stack
        void SendData(SLcmCommand *cmd, int nLength);
            // on stack data
        void SaveData(SLcmCommand *cmd, int nLength);
            // save data to stack
        BOOL FillAllStack(); // move all buttons to stack
        void KeyPress(BYTE param); // call the callback
        void ClearAll(void); // clear stack and button data
        void ClearTimer(void);
        BOOL CheckConnect(void);

public:

    // interface operations
    BOOL OpenPort(short nComPort);
    BOOL ClosePort(void);

    void SendResetToLcm(BOOL bKeepData = FALSE);
    BOOL GetKeyInfo(DWORD &dw_id, BYTE &c_type);
    void SendToLcm(SLcmCommand *sCmd, int nLength);

    ///-----///
    // for me use only
    LONG OnLcmData(UINT wParam, LONG lParam);

    // Overrides
    // ClassWizard generated virtual function overrides
    //{AFX_VIRTUAL(CLcmSCommunication)
    //}AFX_VIRTUAL

    // Implementation
public:
    virtual ~CLcmSCommunication();

    // Generated message map functions
protected:

    //{AFX_MSG(CLcmSCommunication)
    afx_msg void OnTimer(UINT nIDEvent);
    //}AFX_MSG
    DECLARE_MESSAGE_MAP()
};

inline void CLcmSCommunication::GotoLCM_STATE_START_CMD(void)
{
    m_nState = LCM_STATE_START_CMD;

    m_nhead = 0;

```

```

        m_dwKeyboardID = 0x00000000;
        m_nTypeNum = 0;
        m_nChecksumNum = 0;
        m_sMyChecksum = 0x00;
        m_nFramLengthNum = 0;
    }

    inline unsigned short CLcmSCommunication::BuildSetColor(unsigned char *buf,
        unsigned short &checksum, unsigned char keyNum,
        unsigned char colorNum)
    {
        *buf = LCM_SET_COLOR;
        *(buf+1) = keyNum;
        *(buf+2) = colorNum;

        checksum += LCM_SET_COLOR;
        checksum += keyNum;
        checksum += colorNum;
        return 3;
    }

    inline unsigned short CLcmSCommunication::BuildSetBitmapLC16(unsigned char *buf,
        unsigned short &checksum, unsigned char keyNum,
        CLcmBitmap *bitmap)
    {
        *buf = LCM_SET_BITMAP_LC16;
        *(buf+1) = keyNum;
        memcpy((buf+2), bitmap->m_pData, bitmap->m_cLength);

        checksum += LCM_SET_BITMAP_LC16;
        checksum += keyNum;
        checksum += bitmap->m_nChecksum;
        return bitmap->m_cLength + 2;
    }

    inline unsigned short CLcmSCommunication::BuildSetBitmapLC24(unsigned char *buf,
        unsigned short &checksum, unsigned char keyNum,
        CLcmBitmap *bitmap)
    {
        *buf = LCM_SET_BITMAP_LC24;
        *(buf+1) = keyNum;
        memcpy((buf+2), bitmap->m_pData, bitmap->m_cLength);

        checksum += LCM_SET_BITMAP_LC24;
        checksum += keyNum;
        checksum += bitmap->m_nChecksum;
        return bitmap->m_cLength + 2;
    }

    inline BOOL CLcmSCommunication::CheckConnect(void)
    {

```

```

        return m_bConnect;
    }

    inline void CLcmSCommunication::SendResetOnly(void)
    {
        SendOneByteData(LCM_RESET);
    }
#endif

```

////////////////////////////////////

Table 1. Demographic characteristics of the study population	
Age (years)	65.2 ± 1.2
Gender (male/female)	102/108
Education (years)	12.5 ± 0.5
Marital status (married/divorced/widowed)	150/30/20
Occupation (retired/working)	150/30
Income (€ per month)	1,200 ± 100
Comorbidities (hypertension/diabetes/cholesterol)	80/40/60
Medication (antihypertensive/antidiabetic/anticholesterol)	50/30/40
Smoking status (smoker/nonsmoker)	20/180
Alcohol consumption (yes/no)	30/170
Family history of cardiovascular disease (yes/no)	40/160
Previous cardiovascular events (myocardial infarction/stroke)	10/20
Current symptoms (chest pain/shortness of breath)	20/30
Functional status (independent/dependent)	170/30
Quality of life score (SF-36)	45.2 ± 2.1
Healthcare utilization (hospitalizations/visits)	10/20
Health insurance status (public/private)	150/30
Geographical location (urban/rural)	120/80
Season of recruitment (spring/summer/autumn/winter)	40/40/40/40
Study duration (months)	12
Dropouts (n)	10
Final sample size (n)	190

```

// LcmServer.h : main header file for the LCMSEVER application
//

#ifndef __AFXWIN_H__
    #error include 'stdafx.h' before including this file for PCH
#endif

#include "resource.h"    // main symbols

////////////////////
// CLcmServerApp:
// See LcmServer.cpp for the implementation of this class
//

class CLcmServerApp : public CWinApp
{
public:
    CLcmServerApp();

// Overrides
    // ClassWizard generated virtual function overrides
   //{{AFX_VIRTUAL(CLcmServerApp)
public:
    virtual BOOL InitInstance();
   //}}AFX_VIRTUAL

// Implementation
    COleTemplateServer m_server;
        // Server object for document creation

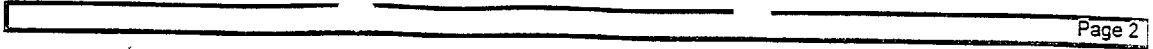
   //{{AFX_MSG(CLcmServerApp)
    afx_msg void OnAppAbout();
        // NOTE - the ClassWizard will add and remove member functions here.
        // DO NOT EDIT what you see in these blocks of generated code !
   //}}AFX_MSG
    DECLARE_MESSAGE_MAP()
};

class CTankCommandLineInfo : public CCommandLineInfo
{
public:
    BOOL m_bShowTestForm;
    BOOL m_bRegServer;
    BOOL m_bUnRegServer;

    CTankCommandLineInfo();
    void ParseParam( LPCTSTR lpszParam, BOOL bFlag, BOOL bLast );
};

inline CTankCommandLineInfo::CTankCommandLineInfo()
{
    m_bShowTestForm = FALSE;
    m_bRegServer = FALSE;

```



```
        m_bUnRegServer = FALSE;  
    }
```

```
////////////////////////////////////
```

US 2001/0033271 A1

```

static void CallBackStatic(void *pInstance, BYTE bCmd);

public:
    virtual ~CLcmServerDoc();
#ifdef _DEBUG
    virtual void AssertValid() const;
    virtual void Dump(CDumpContext& dc) const;
#endif

protected:

// Generated message map functions
protected:
   //{{AFX_MSG(CLcmServerDoc)
    // NOTE - the ClassWizard will add and remove member functions here.
    // DO NOT EDIT what you see in these blocks of generated code !
   //}}AFX_MSG
    DECLARE_MESSAGE_MAP()

// Generated OLE dispatch map functions
//{{AFX_DISPATCH(CLcmServerDoc)
    afx_msg BOOL Disconnect();
    afx_msg void Reset();
    afx_msg BOOL GetKeyInfo(long FAR* k_id, long FAR* k_type);
    afx_msg short Bitmap(short level, short keyNum, LPCTSTR name);
    afx_msg short Color(short level, short keyNum, short color);
    afx_msg short Connect(short ConnectStyle, short PortNumber, LPDISPATCH CbkObject);
    afx_msg short Caption(short level, short keyNum, LPCTSTR line1, short flags1, short pointSize1,
LPCTSTR typeface1, LPCTSTR line2, short flags2, short pointSize2, LPCTSTR typeface2);
    afx_msg void Refresh();
   //}}AFX_DISPATCH
    DECLARE_DISPATCH_MAP()
    DECLARE_INTERFACE_MAP()
};

#endif
//XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

```

20010033271A1

```

// LcmServerView.h : interface of the CLcmServerView class
//
///////////////////////////////////////////////////////////////////

class CLcmServerView : public CFormView
{
protected: // create from serialization only
    CLcmServerView();
    DECLARE_DYNCREATE(CLcmServerView)

public:
    //{{AFX_DATA(CLcmServerView)
    enum { IDD = IDD_LCMSEVER_FORM };
    int         m_nKeySelected;
    int         m_nColorSelected;
    CString     m_sBitmapName;
    CString     m_sLine1;
    CString     m_sLine2;
    CString     m_sPressInfo;
    CString     m_sConnectInfo;
    //}}AFX_DATA

    // Attributes
public:
    CLcmServerDoc* GetDocument();
private:
    CString m_sFaceName1, m_sFaceName2;
    short m_nFlag1, m_nFlag2, m_nPointSize1, m_nPointSize2;

    LOGFONT m_Font1, m_Font2;

    // Operations
public:
    void Notify(BYTE cmd);

    // Overrides
    // ClassWizard generated virtual function overrides
    //{{AFX_VIRTUAL(CLcmServerView)
public:
    virtual BOOL PreCreateWindow(CREATESTRUCT& cs);
protected:
    virtual void DoDataExchange(CDataExchange* pDX); // DDX/DDV support
    //}}AFX_VIRTUAL

    // Implementation
public:
    virtual ~CLcmServerView();
#ifdef _DEBUG
    virtual void AssertValid() const;
    virtual void Dump(CDumpContext& dc) const;
#endif

protected:

```

```

// LcmServerDoc.h : interface of the CLcmServerDoc class
//
///////////////////////////////////////////////////////////////////
#ifndef LCM_SERVER_DOC_H
#define LCM_SERVER_DOC_H

#define SINGLE_LEVEL_SINGLE_USER 1
#define SINGLE_LEVEL_MULTI_USER 2

#define MAX_NUMBER_OF_PORTS 10
#define ONE_LEVEL_LCM_MANAGER 1

struct SManagers
{
    int m_nPortNumber;
    int m_nManagerType;
    void *m_pManger;
};

class ICallback;
class CLcmServerDoc : public CDocument
{
    friend class CLcmServerView;

protected: // create from serialization only
    CLcmServerDoc();
    DECLARE_DYNCREATE(CLcmServerDoc)

// Attributes
public:

private :
    static SManagers m_sManagers[MAX_NUMBER_OF_PORTS];
    static int m_sManagers_len;
    SManagers *m_pMyMgr;
    ICallback *m_pClient;

// Operations
public:

// Overrides
    // ClassWizard generated virtual function overrides
    //{AFX_VIRTUAL(CLcmServerDoc)
    public:
    virtual BOOL OnNewDocument();
    virtual void Serialize(CArchive& ar);
    //}AFX_VIRTUAL

// Implementation
private :
    void CallBack(BYTE bCmd);
public:

```

```
// Generated message map functions
protected:
   //{{AFX_MSG(CLcmServerView)
    afx_msg void OnConnect2();
    afx_msg void OnDisconnect();
    afx_msg void OnLoadBitmap();
    afx_msg void OnLoadColor();
    afx_msg void OnLine1Font();
    afx_msg void OnOneLineText();
    afx_msg void OnTwoLineText();
    afx_msg void OnLine2Font();
    afx_msg void OnBUTTONLoop100();
    afx_msg void OnReset();
    afx_msg void OnRefresh();
    afx_msg void OnBrowseBitmap();
   //}}AFX_MSG
    DECLARE_MESSAGE_MAP()
};

#ifdef _DEBUG // debug version in LcmServerView.cpp
inline CLcmServerDoc* CLcmServerView::GetDocument()
{ return (CLcmServerDoc*)m_pDocument; }
#endif

////////////////////////////////////
```

```

// MainFrm.h : interface of the CMainFrame class
//
///////////////////////////////////////////////////////////////////

class CMainFrame : public CMDIFrameWnd
{
    DECLARE_DYNAMIC(CMainFrame)
public:
    CMainFrame();

// Attributes
public:

// Operations
public:

// Overrides
    // ClassWizard generated virtual function overrides
   //{{AFX_VIRTUAL(CMainFrame)
    virtual BOOL PreCreateWindow(CREATESTRUCT& cs);
   //}}AFX_VIRTUAL

// Implementation
public:
    virtual ~CMainFrame();
#ifdef _DEBUG
    virtual void AssertValid() const;
    virtual void Dump(CDumpContext& dc) const;
#endif

// Generated message map functions
protected:
   //{{AFX_MSG(CMainFrame)
        // NOTE - the ClassWizard will add and remove member functions here.
        // DO NOT EDIT what you see in these blocks of generated code!
   //}}AFX_MSG
    DECLARE_MESSAGE_MAP()
};

/////////////////////////////////////////////////////////////////

```

```
// PictureLibrary.h : header file
//

#include "LcmBitmap.h"
#include "LcmSCommunication.h"

class CPictureLibrary
{
private: // data members
    BOOL m_bCreateFlag;

    CMapStringToPtr m_Bitmaps;

public: // functions
    CPictureLibrary();
    ~CPictureLibrary();
    CLcmBitmap *CreateBitmapByBMP(const char *filename);
    void Create(BYTE key_type = LCM_KEY_TYPE_LC24);
    void Release();

    CLcmBitmap *GetBitmap(CString name);
};

////////////////////////////////////
```

FIG. 33

```

//{{NO_DEPENDENCIES}}
// Microsoft Developer Studio generated include file.
// Used by LcmServer.rc
//
#define IDD_ABOUTBOX 100
#define IDP_OLE_INIT_FAILED 100
#define IDD_LCMSEVER_FORM 101
#define IDR_MAINFRAME 128
#define IDR_LCMSERTYPE 129
#define ArrowDown 130
#define ArrowLeft 132
#define ArrowRight 133
#define ArrowUp 134
#define Blank 135
#define Hand 136
#define Note 137
#define Phone 138
#define Phone1 139
#define Phone2 140
#define Plane 141
#define Plane1 142
#define Plane2 143
#define Speaker 144
#define Time 145
#define TTSI 146
#define Earth1 148
#define Earth2 150
#define Scissors 153
#define Paste 154
#define IDC_CONNECT2 1000
#define IDC_DISCONNECT 1001
#define IDC_KEY_COMBO 1006
#define IDC_BITMAP_COMBO 1007
#define IDC_LOAD_BITMAP 1008
#define IDC_LOAD_COLOR 1009
#define IDC_COLOR_COMBO 1010
#define IDC_LINE1_TEXT 1011
#define IDC_LINE2_TEXT 1012
#define IDC_LINE1_FONT 1013
#define IDC_LINE2_FONT 1014
#define IDC_ONE_LINE_TEXT 1015
#define IDC_TWO_LINE_TEXT 1016
#define IDC_KEY_NUMBER 1018
#define IDC_BUTTONLoop100 1019
#define IDC_PRESS_INFO 1021
#define IDC_CONNECT_INFO 1022
#define IDC_RESET 1023
#define IDC_REFRESH 1025
#define IDC_BROWSE_BITMAP 1026
#define IDC_CUSTOM1 1027

// Next default values for new objects
//

```

```
#ifndef APSTUDIO_INVOKED
#ifndef APSTUDIO_READONLY_SYMBOLS
#define _APS_3D_CONTROLS 1
#define _APS_NEXT_RESOURCE_VALUE 155
#define _APS_NEXT_COMMAND_VALUE 32771
#define _APS_NEXT_CONTROL_VALUE 1028
#define _APS_NEXT_SYMED_VALUE 101
#endif
#endif
```

[illegible]

```
// stdafx.h : include file for standard system include files,
// or project specific include files that are used frequently, but
// are changed infrequently
//

#define VC_EXTRALEAN           // Exclude rarely-used stuff from Windows headers

#include <afxwin.h>             // MFC core and standard components
#include <afxext.h>             // MFC extensions
#include <afxdisp.h>           // MFC OLE automation classes
#ifndef _AFX_NO_AFXCMN_SUPPORT
#include <afxcmn.h>             // MFC support for Windows Common Controls
#endif // _AFX_NO_AFXCMN_SUPPORT
```

0055119-001504

TankServer API For CTI interface

TTSi 2/9/1998

The following are TankServer public methods/functions:

Connect(ConnectStyle As Integer, ComPort As Integer,
LpdCallback As Object) As Integer
Disconnect() As Boolean
GetKeyInfo(KeyId As Long, KeyType As Long) As Boolean
Reset()
Refresh()
Bitmap(Level As Integer, KeyNum As Integer,
Name As String) As Integer
Caption(Level As Integer, KeyNum As Integer,
Line1 As String, Flags1 As Integer,
PointSize1 As Integer, FontName1 As String,
Line2 As String, Flags2 As Integer,
PointSize2 As Integer, FontName2 As String) As Integer
Color(Level As Integer, KeyNum As Integer, Color As Integer) As Integer

The functionality and parameters of each method/function

Connect(ConnectStyle As Integer, ComPort As Integer,
LpdCallback As Object) As Integer
connects to TankServer (automation server) which then establish the
keyboard connection with the ComPort specified.
A CallBack class object with a public method named "Notify" must be
included in the parameter list. The public method Notify(Cmd As Integer)
receives events generated from the keyboard(TankServer).

return values:

- 0 Connection is successful
the comm port is available, and TankServer is connected ok
- 1 Keyboard is ready
Tank Automation Server gets the connection to keyboard
- Eise Connection Fail
reasons could be:
 - 1. the TankServer is previously created in Single User Mode
 - 2. comm port is not available

ConnectStyle values:

- 1 SINGLE_LEVEL_SINGLE_USER, only one app can connect to T2 keyboard .
- 2 SINGLE_LEVEL_MULTI_USER, T2 keyboard can be shared by multiple apps

ComPort values:

- 1 com1
- 2 com2
- 3 com3
- 4 com4
- etc...

Cmd values: — the public method Notify (Cmd As Integer) in CallBack class object

- 32 keyboard status changed:Keyboard Ready

```

33 keyboard status changed:No Keyboard Connection
0 to 11 Key Press: map to Key0 .. Key11
20 to 31 Key Release: map to key0 .. key11

Disconnect() As Boolean
Disconnects the LcmServer.Auto.
return values:
TRUE Disconnection Successful
FALSE Disconnection Fail

GetKeyInfo(KeyId As Long, KeyType As Long) As Boolean
Gets the keyboard ID and keyboard type(passed in as parameters).
return values:
TRUE Call is success
FALSE Call is Fail
KeyId:
This function will ask the Tank to return a KeyboardID.
Each Tank keyboard carries a unique keyboard ID. This ID
can be used for product security, inventory control or
any other maintenance reason

KeyType values:
130: key type 32X24
129: others
This is to determine whether the LCD key type is 32X24.
The parameter returns the value 130 if the keyboard has
32X24 LCD keys; otherwise returns 129.

Reset()
Resets the TankServer and clear all keys.

Refresh()
Refresh all 12 keys

Bitmap(Level As Integer, KeyNum As Integer,
Name As String) as Integer

Loads the key by bitmap name or file path. The Level is not implemented in this version.
- Return value: 0 success, -1 no connection, -2 incorrect key number, -3 bad bitmap format.
- Level: gives the ability to present keys in multi-dimensional (level) view.
- KeyNum: 0..11
- Name: full path of the bitmap file, or the server internal(built in) bitmap name
It is currently not available at this release, and should be defaulted to 0.
Built in Bitmap Name Values:
ArrowDown
ArrowLeft
ArrowRight
ArrowUp
Blank
Hand
Note
Phone

```

0050410, 051504

```

Phone1
Phone2
Plane
Plane1
Plane2
Speaker
Time
TTSi

Caption(Level As Integer, KeyNum As Integer,
  Line1 As String, Style1 As Integer,
  Size1 As Integer, FontName1 As String,
  Line2 As String, Style2 As Integer,
  Size2 As Integer, FontName2 As String) as Integer
Write 1 or 2 lines of text on the keys
- Return value: 0 success, -1 no connection, -2 incorrect key number.
- Level: gives the ability to present keys in multi-dimensional (level) view.
  It is currently not available in this release, and should be defaulted to 0.
- KeyNum: 0..11
- Line1: Line 1 text
- Style1: Line 1 style (in Hex number)

&H1&      Left Justified
&H2&      Bold
&H4&      Italic
&H8&      Underline
&H10&     Strikethrough

Example: If you want the text to be left justified and underlined,
  Style1 = &H1& Or &H8&

- Size1: Line 1 font size
- FontName1: Line 1 font name
- Line2: Line 2 text
- Style2: Line 2 style (see above)
- Size2: Line 2 font size
- FontName2: Line 2 font name

** note: if you want only 1 line to be displayed at the center of the key,
  just simply default the parameters for line 2 to null string (""")
  or 0 value depending on their type.

Example: Caption(2, "Hello", &H1& or &H2&, 10, "Arial", 0, "", 0, 0, "")
This call would produce a text "Hello" at the center of the LCD key

Color(Level As Integer, KeyNum As Integer, Color As Integer) as Integer
Sets the key color.
- Return value: 0 success, -1 no connection, -2 incorrect key number, -3 bad bitmap format.
- Level: gives the ability to present keys in multi-dimensional (level) view.
  It is currently not available at this release, and should be defaulted to 0.
- KeyNum: 0..11

```

```

    - Color:
    0 Dark Green
    1 Green
    2 Bright Green
    3 Dark Red
    4 Bright Red
    5 Dark Orange
    6 Bright Orange
    7 Red Orange
    8 Green Orange

```

Option Explicit

```

'13 public variables
Public dialString As String
Public callID As String

```

```

Enum ConnectStyleEnum
    SINGLE_USER = 1
    MULTI_USERS = 2
End Enum

```

```

Enum CommPortEnum
    Comm1 = 1
    Comm2 = 2
    Comm3 = 3
    Comm4 = 4
End Enum
Dim ts As Object

```

```

Dim cb As New CallBack
'cb is used to receive events sent back from the server

```

```

Private Sub Command1_Click()
    Dim rtn As Integer
    rtn = ts.Disconnect()
    Unload TankAPITemplate
End

```

```

End Sub
Public Function ParseForStatus(ddestring As String)

```

```

' this function parses the string returned from the Interaction client
' to get the current status

```

```

    Dim pos1, pos2, len1 As Integer

```

```

    pos1 = InStr(1, ddestring, "status", vbTextCompare)
    pos1 = pos1 + 7

```

```

pos2 = InStr(pos1, ddestring, "|", vbTextCompare)
len1 = pos2 - pos1

ParseForStatus = Mid(ddestring, pos1, len1)
End Function

' This subroutine is used to determine when a user changes
' their status by watching DDEPhone.Text as it changes

Private Sub DDEPhone_Change()

Dim rtn As Integer

If DDEPhone.Text = "Available" Then
    rtn = ts.color(0, 8, 2)
    rtn = ts.color(0, 9, 8)
    rtn = ts.color(0, 10, 8)
    rtn = ts.color(0, 11, 8)
Elseif DDEPhone.Text = "Available, No ACD" Then
    rtn = ts.color(0, 8, 8)
    rtn = ts.color(0, 9, 2)
    rtn = ts.color(0, 10, 8)
    rtn = ts.color(0, 11, 8)
Elseif DDEPhone.Text = "Gone Home" Then
    rtn = ts.color(0, 8, 8)
    rtn = ts.color(0, 9, 8)
    rtn = ts.color(0, 10, 2)
    rtn = ts.color(0, 11, 8)
Elseif DDEPhone.Text = "Away From Desk" Then
    rtn = ts.color(0, 8, 8)
    rtn = ts.color(0, 9, 8)
    rtn = ts.color(0, 10, 8)
    rtn = ts.color(0, 11, 2)
Elseif DDEPhone.Text = "Clear" Then
    rtn = ts.color(0, 8, 8)
    rtn = ts.color(0, 9, 8)
    rtn = ts.color(0, 10, 8)
    rtn = ts.color(0, 11, 8)
End If
End Sub

Private Sub ctrlDialString_Change()
    dialString = ctrlDialString.Text
End Sub

Private Sub Form_LinkExecute(CmdStr As String, Cancel As Integer)

' Look for "Place" in the left of the string to determine if
' the call is an outgoing call

If Left(CmdStr, 5) = "Place" Then
    ' must be an outbound call
    callID = Mid(CmdStr, 11, 8)

```

```

Else
    ' must be an inbound call
    callID = CmdStr
End If
DDEPhone.Text = callID
End Sub

Private Sub Form_Load()
    "*****"
    'this is a standalone ActiveX EXE program started up in Sub Main()
    "*****"

    Dim rtn As Integer
    'instantiate TankServer object
    Set ts = CreateObject("TankServer")

    'once the object is created, all the public methods are available
    'connect to keyboard with comm port 1
    rtn = ts.Connect(MULTI_USERS, Comm1, cb)

    'example of using some of the methods
    'display an internal(built-in) bitmap named "TTSI" on keym 5
    'rtn = ts.Bitmap(0, 5, "TTSI")

    'display 2-line text "Hello World" on key 3
    'Line1 text is Bold, size 10pt, font type "Arial"
    'Line2 text is Italic, size 8, font type "Times News Roman"
    'rtn = ts.Caption(0, 3, "Hello", &H2&, 10, "Arial", "World", &H4&, 8, "Times News Roman")

    ' set default key text
    rtn = ts.Caption(0, 0, "Dial", &H2&, 10, "Arial", "", 0, 0, "")
    rtn = ts.Caption(0, 1, "Pickup", &H2&, 10, "Arial", "", 0, 0, "")
    rtn = ts.Caption(0, 2, "Discont", &H2&, 10, "Arial", "", 0, 0, "")
    rtn = ts.Caption(0, 3, "Hold", &H2&, 10, "Arial", "", 0, 0, "")
    rtn = ts.Caption(0, 4, "Transfr", &H2&, 10, "Arial", "", 0, 0, "")
    rtn = ts.Caption(0, 5, "Listen", &H2&, 10, "Arial", "", 0, 0, "")
    rtn = ts.Caption(0, 6, "Record", &H2&, 10, "Arial", "", 0, 0, "")
    rtn = ts.Caption(0, 7, "Mute", &H2&, 10, "Arial", "", 0, 0, "")
    rtn = ts.Caption(0, 8, "Avail", &H2&, 10, "Arial", "", 0, 0, "")
    rtn = ts.Caption(0, 9, "No ACD", &H2&, 10, "Arial", "", 0, 0, "")
    rtn = ts.Caption(0, 10, "Home", &H2&, 10, "Arial", "", 0, 0, "")
    rtn = ts.Caption(0, 11, "Away", &H2&, 10, "Arial", "", 0, 0, "")

    ' set default key colors
    rtn = ts.color(0, 0, 4)
    rtn = ts.color(0, 1, 4)
    rtn = ts.color(0, 2, 4)
    rtn = ts.color(0, 3, 4)
    rtn = ts.color(0, 4, 4)
    rtn = ts.color(0, 5, 4)
    rtn = ts.color(0, 6, 4)
    rtn = ts.color(0, 7, 4)
    rtn = ts.color(0, 8, 8)

```

```

rtn = ts.color(0, 9, 8)
rtn = ts.color(0, 10, 8)
rtn = ts.color(0, 11, 8)

```

End Sub

```
' this is a public procdure used to change key colors from other
' modules
```

Public Sub ChangeKeyColor(key, color As Integer)

Dim rtn As Integer

```

rtn = ts.color(0, key, color)

```

End Sub

Table 1. Demographic characteristics of the study population	
Age (years)	65.2 (SD 8.5)
Gender	
Male	52.1%
Female	47.9%
Education (years)	12.5 (SD 2.1)
Marital status	
Married	68.3%
Widowed	21.5%
Divorced	10.2%
Single	0.0%
Income (USD/month)	1,250 (SD 350)
Health status	
Good	75.4%
Fair	18.7%
Poor	5.9%
Functional status	
Independent	82.1%
Dependent	17.9%
Comorbidities	
Hypertension	45.3%
Diabetes	32.1%
Heart disease	28.7%
Stroke	15.2%
Arthritis	38.9%
Chronic lung disease	12.5%
Depression	10.8%
Alcohol use	
Regular	5.1%
Occasional	12.3%
Never	82.6%
Smoking status	
Current	8.7%
Former	23.4%
Never	67.9%

```
'CallBack Class Module for T2 Events (Key Presses for Telephone Functions)
```

```
Public uname As String
Public unamelen, unameret As Long
```

```
' declare windows function that will get back windows user id
Private Declare Function GetUserName Lib "ADVAPI32.DLL" Alias "GetUserNameA" (ByVal lpBuffer As
String, nSize As Long) As Long
```

```
Public Sub Notify(cmd As Integer)
'cmd's are events generated by the keyboard
```

```
    If cmd = 32 Then
'keyboard status changed: keyboard ready
'put your codes for this event here
```

```
    ElseIf cmd = 33 Then
'keyboard status changed: no keyboard connection
'put your codes for this event here
```

```
    ElseIf cmd >= 0 And cmd <= 11 Then
'key-down(key press) event: cmd 0-11 maps to key 0 to 11
'put your codes for this event here
```

```
    Select Case cmd
```

```
        Case 0 ' Dial
            dialString = TankAPITemplate.ctrlDialString
            TankAPITemplate.SetFocus
            TankAPITemplate.DDEPhone.LinkMode = 0
            TankAPITemplate.DDEPhone.LinkTopic = "I3 Client APP DDE Server|Call"
            TankAPITemplate.DDEPhone.LinkMode = vbLinkManual
            phoneCmd = "Dial(" & dialString & "," & vb5 & ", callid, PlaceCall)"
            On Error Resume Next
            TankAPITemplate.DDEPhone.LinkExecute phoneCmd
```

```
        Case 1 'Pickup
            TankAPITemplate.DDEPhone.LinkMode = 0
            TankAPITemplate.DDEPhone.LinkTopic = "I3 Client APP DDE Server|Call"
            TankAPITemplate.DDEPhone.LinkMode = vbLinkManual
            phoneCmd = "Pickup(" & TankAPITemplate.callID & ")"
            On Error Resume Next
            TankAPITemplate.DDEPhone.LinkExecute phoneCmd
            TankAPITemplate.DDEPhone.Text = "Pickup"
```

```
        Case 2 ' Disconnect
            TankAPITemplate.DDEPhone.LinkMode = 0
            TankAPITemplate.DDEPhone.LinkTopic = "I3 Client APP DDE Server|Call"
            TankAPITemplate.DDEPhone.LinkMode = vbLinkManual
```

```

phoneCmd = "Disconnect(" & TankAPITemplate.callID & ")
On Error Resume Next
TankAPITemplate.DDEPhone.LinkExecute phoneCmd
TankAPITemplate.DDEPhone.Text = "Disconnect"

```

```

Case 3 ' Hold
TankAPITemplate.DDEPhone.LinkMode = 0
TankAPITemplate.DDEPhone.LinkTopic = "I3 Client APP DDE Server|Call"
TankAPITemplate.DDEPhone.LinkMode = vbLinkManual
phoneCmd = "Hold(" & TankAPITemplate.callID & ")
On Error Resume Next
TankAPITemplate.DDEPhone.LinkExecute phoneCmd
TankAPITemplate.DDEPhone.Text = "Hold"

```

```

Case 4 ' Transfer
dialString = TankAPITemplate.ctrlDialString
TankAPITemplate.DDEPhone.LinkMode = 0
TankAPITemplate.DDEPhone.LinkTopic = "I3 Client APP DDE Server|Call"
TankAPITemplate.DDEPhone.LinkMode = vbLinkManual
phoneCmd = "Transfer(" & TankAPITemplate.callID & "," & dialString & ")
On Error Resume Next
TankAPITemplate.DDEPhone.LinkExecute phoneCmd
TankAPITemplate.DDEPhone.Text = "Transferring to Extension" & dialString

```

VOICE MAIL DOES NOT WORK WITH DDE

```

Case 5 ' Listen
TankAPITemplate.DDEPhone.LinkMode = 0
TankAPITemplate.DDEPhone.LinkTopic = "I3 Client APP DDE Server|Call"
TankAPITemplate.DDEPhone.LinkMode = vbLinkManual
phoneCmd = "Listen(" & TankAPITemplate.callID & ")
On Error Resume Next
TankAPITemplate.DDEPhone.LinkExecute phoneCmd

TankAPITemplate.DDEPhone.Text = "Listen"

```

```

Case 6 ' Record
TankAPITemplate.DDEPhone.LinkMode = 0
TankAPITemplate.DDEPhone.LinkTopic = "I3 Client APP DDE Server|Call"
TankAPITemplate.DDEPhone.LinkMode = vbLinkManual
phoneCmd = "Record(" & TankAPITemplate.callID & ")
On Error Resume Next
TankAPITemplate.DDEPhone.LinkExecute phoneCmd
TankAPITemplate.DDEPhone.Text = "Record"

```

```

Case 7 ' Mute

' Query the client status, if to see if it is Aval, Aval NO ACD
' Away, or Home
Dim statusVar As String
Dim statusAll As String
uname = String(255, vbNullChar)
unamelen = Len(uname)

```

```

unameret = GetUserName(uname, unamelen)
uname = Left(uname, (unamelen - 1))

TankAPITemplate.DDEPhone.LinkMode = 0
TankAPITemplate.DDEPhone.LinkTopic = "I3 Client APP DDE Server|User"
TankAPITemplate.DDEPhone.LinkItem = uname & ":[ALL]"
TankAPITemplate.DDEPhone.LinkMode = vbLinkManual
TankAPITemplate.DDEPhone.LinkRequest
statusAll = TankAPITemplate.DDEPhone.Text
statusVar = TankAPITemplate.ParseForStatus(statusAll)
If (statusVar <> "Available") And (statusVar <> "Available, No ACD") And (statusVar <> "Gone
Home") And (statusVar <> "Away From Desk") Then
    TankAPITemplate.DDEPhone.Text = "Clear"
End If
TankAPITemplate.DDEPhone.LinkMode = 0
TankAPITemplate.DDEPhone.LinkTopic = "I3 Client APP DDE Server|Call"
TankAPITemplate.DDEPhone.LinkMode = vbLinkManual
phoneCmd = "Mute(" & TankAPITemplate.callID & ")"
On Error Resume Next
TankAPITemplate.DDEPhone.LinkExecute phoneCmd
TankAPITemplate.DDEPhone.Text = "Mute"

Case 8 ' Set Client Status to Available
    TankAPITemplate.DDEPhone.LinkMode = 0
    TankAPITemplate.DDEPhone.LinkTopic = "I3 Client APP DDE Server|User"
    TankAPITemplate.DDEPhone.LinkMode = vbLinkManual
    statusCmd = "Status(Available)"
    On Error Resume Next
    TankAPITemplate.DDEPhone.LinkExecute statusCmd
    TankAPITemplate.DDEPhone.Text = "Available"

Case 9 ' Set Client status to Available - No ACD
    TankAPITemplate.DDEPhone.LinkMode = 0
    TankAPITemplate.DDEPhone.LinkTopic = "I3 Client APP DDE Server|User"
    TankAPITemplate.DDEPhone.LinkMode = vbLinkManual
    statusCmd = "Status(Available, No ACD)"
    On Error Resume Next
    TankAPITemplate.DDEPhone.LinkExecute statusCmd
    TankAPITemplate.DDEPhone.Text = "Available, No ACD"

Case 10 ' Set Client Status to Gone Home
    TankAPITemplate.DDEPhone.LinkMode = 0
    TankAPITemplate.DDEPhone.LinkTopic = "I3 Client APP DDE Server|User"
    TankAPITemplate.DDEPhone.LinkMode = vbLinkManual
    statusCmd = "Status(Gone Home)"
    On Error Resume Next
    TankAPITemplate.DDEPhone.LinkExecute statusCmd
    TankAPITemplate.DDEPhone.Text = "Gone Home"

Case 11 ' Set Client Status to Away From Desk
    TankAPITemplate.DDEPhone.LinkMode = 0
    TankAPITemplate.DDEPhone.LinkTopic = "I3 Client APP DDE Server|User"
    TankAPITemplate.DDEPhone.LinkMode = vbLinkManual

```

```
statusCmd = "Status(Away From Desk)"  
On Error Resume Next
```

```
TankAPITemplate.DDEPhone.LinkExecute statusCmd  
TankAPITemplate.DDEPhone.Text = "Away From Desk"
```

```
End Select
```

```
Elseif cmd >= 20 And cmd <= 31 Then  
'key-up (key release) event: cmd 20-31 maps to key 0 to 11  
'put your codes for this event here
```

```
End If
```

```
End Sub
```

64

' Entry Point for CTI App Start up

Public Sub Main()
TankAPITemplate.Show

End Sub

FIG. 5B

What is claimed is:

1. A computer-telephony integrated user console for manipulating calls in a telephone system and manipulating data in a data processing system, the user console comprising:

- a computer system, the computer system including the data processing system therein;
- a telephone line interface, the telephone line interface being electrically interposed between the computer system and the telephone system; and
- a keyboard electrically coupled to the computer system, the keyboard including at least one programmable display key, the at least one programmable display key including a switch electrically coupled to the computer system for providing input thereto and a programmable electrical display, the programmable electrical display providing a visual indication of at least one of text and graphics in response to signals received from the computer system.

2. A computer-telephony integrated user console as defined by claim 1, wherein the computer is at least partially controlled by a software automation server, the software automation server comprising:

- a software input/output layer, the software input/output layer providing a standard interface between at least one software application and the software automation server;
- a keyboard communications layer, the keyboard communications layer controlling digital communications between the computer and the keyboard; and
- a keyboard control layer, the keyboard control layer being responsive to commands received from the at least one software application and generating keyboard control commands for the at least one display key, the keyboard control layer being further responsive to activation of the at least one switch and generating keyboard event commands in response thereto, the keyboard event commands being presented to the at least one software application through the software input/output layer.

3. A computer-telephony integrated user console as defined by claim 2, wherein the at least one software application includes a computer-telephony integration applications module, the computer-telephony integration applications module controlling commands and communication between the telephone line interface and the computer.

4. A computer-telephony integrated user console as defined by claim 3, wherein the computer is operating a Windows® software platform and wherein the standard interface of the software input/output layer is Object Linking and Embedding compliant.

5. A computer-telephony integrated user console as defined by claim 4, wherein the at least one programmable display key includes twelve (12) programmable display keys.

6. A computer-telephony integrated user console as defined by claim 5, wherein the keyboard control layer establishes assignments for the programmable display keys among the at least one software application.

7. A computer-telephony integrated user console as defined by claim 6, wherein each programmable electrical display of the programmable display keys comprises a liquid crystal display.

8. A computer-telephony integrated user console as defined by claim 7, wherein the keyboard further includes a QWERTY alphanumeric keypad.

9. A method of operating a computer-telephony user console, the console including a keyboard with at least one display key, each at least one display key including a switch and a display, the method comprising the steps:

- a) assigning a first function to the switch of the at least one display key;
- b) assigning at least one of text and graphics to the display of the at least one display key; and
- c) reassigning the function and display of the at least one display key in response to a detected event.

10. A method of operating a computer-telephony user console, the console including a keyboard with a plurality of display keys, each display key including a switch and a display, the method comprising the steps:

- a) assigning a portion of the plurality of display keys to a first software application;
- b) assigning a first function and a first display to the display keys assigned to the first software application;
- c) assigning at least a portion of the plurality of display keys not assigned to the first software application to at least a second software application;
- d) assigning a first function and a first display to the display keys assigned to the at least second software application;
- e) detecting an event and correlating the event to one of the first software application or at least a second software application; and
- f) reassigning the functions and displays of the display keys assigned to the software application correlated to the detected event.

* * * * *