# Keyboard Emulation for Access to IBM-PC-Compatible Computers by People with Motor Impairments

Heidi M. Horstmann M.S. , Simon P. Levine Ph.D. & Lincoln A. Jaros B.S.

*Assist Technol* 1989;1:63–70
© 1989 Demos Publications

# Keyboard Emulation for Access to IBM-PC-Compatible Computers by People with Motor Impairments

Heidi M. Horstmann, M.S., Simon P. Levine, Ph.D., and Lincoln A. Jaros, B.S.

*Rehabilitation Engineering Program, Department of Physical Medicine and Rehabilitation, University of Michigan, Ann Arbor, Michigan*

**The goal of this paper is to aid both clinicians and developers in understanding the issues associated with alternative input systems that permit full access to the IBM-PC family of computers. The first part of the paper discusses the concept of keyboard emulation in general and reviews a variety of keyboard-emulation systems that are currently available. The capabilities of a system called ALTKEY, developed in our laboratory, are described. The second part of the paper discusses the implementation of a keyboard-emulation system in more detail, using ALTKEY as a specific example. Technical design issues are discussed, and successful approaches to these design challenges are presented.**

**Key Words: Keyboard emulation—Alternative inputs—Computer access.**

The personal computer is an important tool offering unprecedented productivity in areas such as written communication, financial analysis, and engineering design. For many disabled individuals with impaired motor functions, computer use is especially important. However, many of these people cannot use a standard computer keyboard. In response to this need, a number of techniques, generally termed alternative input systems, have been developed to provide full access to standard software without requiring use of the keyboard.

This paper presents the advantages, compromises, and shortcomings of various methods for alternative inputs to personal computers. The goal of this work is to aid both service providers and developers in understanding the issues associated with alternative input approaches permitting access to standard personal computer application programs.

This paper is divided into two parts. The first part begins with a brief review of terminology and concepts followed by an overview of the range of techniques available for alternative input. It concludes with a description of the capabilities of a software-based alternative input system for the IBM-PC and IBM-PC-compatible computers (PCs) called ALTKEY (1,2), developed in our laboratory. The second part presents the technical aspects of ALTKEY and compares it to other software-based systems in order to address a number of issues involved with this type of approach to computer access.

## BACKGROUND

A primary component of most alternative input systems is the emulation of keyboard input through hardware and software. A keyboard emulator is a system composed of hardware, software, or both, which passes user input to an application program (such as a word processor or spreadsheet) in a transparent fashion. "Transparent" implies that the computer cannot distinguish between input coming from the standard keyboard and input coming from the alternative input system. Common keyboard emulation systems include expanded keyboards, head pointers, speech recognition, and special switches, each de-

This work was presented in part at the 8th Annual IEEE Conference on Engineering in Medicine and Biology, Fort Worth, 1986, and the 10th Annual Conference on Rehabilitation Technology, San Jose, 1987.

Address correspondence and reprint requests to Dr. S. P. Levine at Rehabilitation Engineering Program, 1C335 University Hospital, Ann Arbor, MI 48109-0032, U.S.A.

signed to provide an alternative computer access method for individuals with motor impairment.

This paper focuses on keyboard-emulation systems that use a memory-resident program for alternative input. Memory-resident programs are a special class of computer software. They remain in memory once they are loaded and allow any standard software program to run while they are still present. Most of the time the application program runs normally, and the resident program remains dormant. However, the resident program can be activated at any time by a specific condition (such as a switch closure). When activated, the memory-resident program interrupts the application program for an instant, executes its function (such as placing a user selection on the screen), then deactivates itself so that the application can continue running.

## OVERVIEW OF ALTERNATIVE INPUT APPROACHES

Approaches to implementing alternative inputs can be classified into two broad categories: hardware and software. Solutions that are primarily hardware-based involve adding some active device or circuitry to the computer system to perform keyboard emulation. For example, the standard keyboard can be replaced with a custom switch array plugged into the keyboard socket. A hardware-based system may require specialized software as well. A headpointing system, for instance, uses hardware to sense the user's head position and software to translate that position into the desired "keypress." Most hardware-based systems have the disadvantage that the system can only be installed on one computer at a time and may be inconvenient to transport between different computers.

An alternative input system will be considered to be software-based if it uses a computer program to alter the way the existing computer system functions and only a minimal amount of hardware such as a switch or joystick. With a software-based approach, the alternative input system can be stored on magnetic disk like any other program, making it easy to transport and install. Also, because there is no external circuitry required, the footprint of the system is minimized, and there are no additional power requirements. These can be significant advantages for portable, battery-operated systems.

The first alternative access systems were developed for the Apple II computer family. Most alternative input solutions for these microcomputers are hardware-based systems. Software-based systems are very difficult to develop on the Apple II because of its internal hardware structure. In order for any alternative input system to work, it must be able to simulate keyboard entry to a given application program. Most Apple II programs obtain keyboard information by reading the contents of a specific hardware location called the keyboard register. This register stores an appropriate key code each time a key is pressed. The contents of this hardware register cannot be modified via software, so a software-based alternative input system cannot pass user selections to an application program through the standard keyboard register. In addition, the power of the Apple II is severely limited by the small internal memory size. Most programs (e.g., word processors and spreadsheets) monopolize this storage space. This makes it awkward to write special input software that will reliably coexist in memory with application programs from a variety of publishers.

One of the most successful special input systems for the Apple II is a hardware-based system called the Adaptive Firmware Card (AFC)[1] (3). This interface is a plug-in card that may be programmed to allow the user to enter information using a variety of methods including one- and two-switch scanning and Morse code.

The IBM-PC and PCs from other companies have become standard equipment in business, science, and higher education, largely because of their processing power and memory capacity. This, combined with a steady decrease in system cost, has made them an excellent choice for the handicapped user. A number of alternative input systems have been developed that provide transparent access to PCs for users with various degrees of motor impairment.

An example of a hardware-based special input system for PCs is the MOD Keyboard,[2] developed in 1983 (4). This system uses an additional computer, the Commodore VIC-20,[3] to present scanning selections to the user and a keyboard emulator box to pass the user's selections to the PC as if they were typed directly on the PC keyboard. This is a hardware-intensive approach that has the disadvantages of using a large physical space and requiring the purchase of a second computer. It has the advantage of reserving one full computer screen (the VIC-20's) for the presentation of scanning selections, so they do not interfere with video information on the PC screen. This dual-screen approach accommodates large characters or special color combinations for visually impaired users.

[1]AFC, Don Johnston Developmental Equipment, Inc., P.O. Box 639, 1000 N. Rand Road, Building 115, Wauconda, IL 60084.

[2]MOD Keyboard, Tash, Inc., 70 Gibson Drive, Unit 12, Markham, Ontario, Canada L3R 4C2.

[3]Commodore Vic-20, Tash, Inc., 70 Gibson Drive, Unit 12, Markham, Ontario, Canada L3R 4C2.

A second example of a hardware-based approach is replacement of the PC keyboard with a dedicated communication aid, connected to the PC through a keyboard emulator. The communication aid provides the special input methods, and the keyboard emulator ensures that the PC treats the user's selections as if they came from the keyboard. This can be very effective but also expensive and bulky. In addition, many communication aids have a single, fixed display. They lack the flexibility of a video screen to tailor the input selections to individual user needs.

Software solutions to the special input problems are easier to implement on the IBM-PC than on the Apple II for a number of reasons. First, the memory capacity of the IBM-PC is much larger than that of the Apple II. Unlike the Apple II, the IBM-PC was designed to allow more than one program to be present in memory at one time. Thus, a memory-resident program serving a keyboard emulation function can be stored in memory at the same time as an application program. Finally, while Apple II programs obtain keyboard entry directly from hardware, IBM-PC application programs obtain it from the operating system, using keyboard control software routines.[4] These routines can be altered or replaced by memory-resident software designed for alternative input.

Several computer-access systems have been developed that are memory-resident and software-based. One example is the PC AID.[5] The PC AID supports a number of different input methods, including expanded keyboard, one- and two-switch Morse code, and one- and two-switch scanning (5). It requires the use of some additional hardware to support the input switches, in addition to the memory-resident software that provides keyboard emulation. The "Virtual Interface Module" (VIM) first described in 1988 is entirely software-based (6). The VIM supports a number of different input methods, including one- to four-switch scanning, and allows flexible customization of user-selection options. Finally, several systems have been developed by Words+, Inc.,[6] including EZKeys for direct selection, Scanning WSKE for scanning input, and Morse WSKE for Morse code. These combine alternative access with features that are designed to enhance input rate, such as word prediction.

The main focus of this article, ALTKEY, was developed in 1986 at the University of Michigan (1). It is a memory-resident alternative input system for the IBM-PC that supports single- and dual-switch scanning and one- or two-switch Morse code input. Scanning selections, as well as Morse code sequences, are programmable and can be tailored to the needs of the user.

[4]The keyboard control routines are located in the BIOS ROM, which consists of built-in software routines that control input and output devices in the IBM-PC.

[5]PC AID, DADA, Inc., 1024 Dupont Street, Toronto, Ontario, Canada M6H 2A2.

[6]Words+, Inc., P.O. Box 1229, Lancaster, CA 93534.

## ALTKEY CAPABILITIES

ALTKEY supports four basic input modes: row/column, step, and inverse scanning, and Morse code. In row/column scanning, the scan highlight advances automatically until the user selects a row and then a column. In step scanning, the highlight advances one position for each switch hit, and in inverse scanning, the highlight advances automatically while the switch is held down. Step scanning, inverse scanning, and Morse code can be used with either one or two switches. When used for scanning, ALTKEY presents the user with a series of input choices listed across one line of the video screen. These choices are highlighted one at a time in a repeating pattern. The user makes a specific selection by activating a switch when the desired entry is highlighted. The momentary switch used for input can be connected to either a game port card, a serial port hand-shaking line, or to a modified keyboard as a "hotwired" shift key (i.e., Control, Alt, or Shift key). Each selection can perform one or more of the following: (a) send a group of characters to the application program, (b) display a new set of selection choices, and (c) execute a wide variety of commands such as generating speech output or changing the scan rate.

The ability to call a new menu of selection choices from a current one leads to a branched structure for selection choices. The entire list of menus is organized into a "scan tree," which has "branches" to schematically represent the various paths between selection menus. The scan tree is defined in an ASCII text file called a scan tree definition file. This file contains all of the menu selections and is structured by a fixed set of syntactical rules. Scan tree definition files can be created and edited, providing customization to meet the specific needs of a user. Additionally, separate scan tree definition files can be created for different applications and automatically loaded from the active scan tree.

## CLINICAL APPLICATIONS OF ALTKEY

The flexibility of the ALTKEY system allows it to meet a variety of needs for a wide range of users including those with spinal cord injury, neuromuscular disease, head injury, and cerebral palsy. For example, ALTKEY has been used as the basis for an integrated augmentative communication and computer-access system for nonvocal head-injured individuals who can use traditional orthography. Since the scan selections can be tailored to the needs of the user, the system capabilities can be increased as the user's cognitive abilities and/or familiarity with the system improve. For individuals who have difficulty seeing the screen, auditory scanning can be employed.

Another common application of ALTKEY uses two-switch (typically sip/puff) Morse code keyboard emulation for high-level quadriplegics. In some cases, the ability to mix scanning with Morse code can help to improve efficiency. This combination allows frequently used items that have long Morse code sequences, such as the cursor keys, to be selected via scanning.

## TECHNICAL ASPECTS OF ALTKEY

The following section describes a number of important technical aspects of the ALTKEY system. This description is intended to illustrate important issues in the design of memory-resident keyboard emulators and serve as a basis for discussing the advantages and shortcomings of memory-resident alternative input systems for PCs.

The ALTKEY system, which was developed in assembler code and the "C" programming language, is composed of two programs. The first, called AKSCAN, is a memory-resident program executed once each time the computer is booted. It contains the code that implements the scanner and causes the user's selections to be treated as actual keyboard input. Since it is memory-resident, it remains in memory even while other nonresident and memory-resident programs are loaded and can be activated under appropriate conditions.

The second part of the system is named ALTKEY. This is a nonresident program that reads and verifies the scan tree definition file and loads the condensed representation of the scan tree into the internal memory buffer maintained by AKSCAN. It is executed each time the user wishes to load a new scan tree.

### ALTKEY's Memory-Resident Component: AKSCAN

When AKSCAN is executed, it installs itself in memory and allocates a storage buffer for the scan tree. The program and the scan-tree buffer combined take up 40 kB of memory. AKSCAN also inserts new interrupt service routines into three of the BIOS read-only memory (ROM) interrupts. An interrupt service routine is a short program that is activated by some internal or external condition. As the name implies, it is present to perform some service in response to an interrupt generated by the operating system.

The primary function of AKSCAN is to perform keyboard emulation. The most important design criterion is a high degree of compatibility with applications software as well as other memory-resident software. The best way to ensure this is to use the standard BIOS ROM routines wherever possible. The standard keyboard buffer could be used for highest compatibility. However, this buffer only has the capacity for 15 characters, which places an undesirable restriction on the length of scan selections and Morse code equivalents. Therefore, AKSCAN has an optional expanded keyboard buffer with a capacity of 128 characters. This means that scan selections can be several sentences long if desired. A high degree of compatibility is preserved because the expanded keyboard buffer is addressed by standard BIOS buffer pointers. Every "well-behaved" application program accesses the keyboard buffer only through these pointers. The rare program that accesses the buffer directly may not be compatible with the expanded buffer, but the standard 15-character buffer can be used for compatibility with these programs.

When a selection is made either from the scan tree or with Morse code, AKSCAN places it in the keyboard buffer. The standard keyboard continues to be active, so keyboard entry is also placed in the buffer via the standard BIOS keyboard service interrupt routine. When an application program is ready to get a character from the buffer, it generates a keyboard-request interrupt and the standard BIOS routine passes a character to the application program.

In order to perform other functions, AKSCAN intercepts the video, keyboard request, and timer tick interrupts. Whenever any of these interrupts is generated by DOS or an application program, AKSCAN becomes active and performs a variety of operations as detailed below.

#### *Video Interrupt*

Programs use the video interrupt to send commands to the hardware that controls the video screen. AKSCAN intercepts the video interrupt, not to change the standard function of the interrupt, but to signal AKSCAN that the video screen is about to change. This gives AKSCAN a chance to remove its own scan line temporarily and restore the original line before the interrupt changes occur. Once the video interrupt has finished, AKSCAN can record the updated video line (for future screen restoration) and then return its scan line to the video display.

*Keyboard Request Interrupt*

As discussed above, AKSCAN has no need to alter the standard function of the keyboard request interrupt, which is to retrieve characters from the keyboard buffer. However, it does have two other reasons to intercept this interrupt. The first concerns video compatibility. To improve speed, many application programs do not use the video interrupt when writing to the screen; this prevents AKSCAN from intercepting the video information via the video interrupt. A likely indicator of a program writing to the screen is its retrieval of a keyboard entry, which is initiated by the sending of a keyboard request. ALTKEY intercepts the keyboard request interrupt in order to anticipate screen updates that occur when keyboard buffer characters are passed to a program. When this occurs, ALTKEY removes the scan line for a brief moment to allow accurate update of the video display. The second reason for intercepting the keyboard request interrupt is to provide a means of communication between AKSCAN and the nonresident program, ALTKEY, as described below.

*Timer Tick Interrupt*

AKSCAN uses the timer tick interrupt (which "ticks" 18.2 times each second) to change the highlighted scan selection on the video screen based on the selected scan rate. It also checks the status of each input source for a switch activation (the keyboard, a game port, or the serial port) once each tick.

## USER/CLINICIAN PROGRAMMABILITY: ALTKEY SCAN TREES

ALTKEY is a nonresident program that loads scan trees into the memory buffer and activates the scanner. To load a new scan tree, ALTKEY is executed with the scan-tree definition file specified on the DOS command line. Any errors discovered in the syntax of the scan file are reported to the user. If there are none, ALTKEY establishes communication with AKSCAN through the keyboard-request interrupt and loads the completed scan tree into the scan-tree storage buffer previously allocated by AKSCAN.

### Scan-Tree Definition File

Much of the power (as well as complexity) of the ALTKEY system derives from the flexibility in scan-tree design afforded by the scan-tree definition file structure. This file is composed of a list of items. There are six distinct types of items: *group, entry, initial, hold, file,* and *comment.*

The majority of the file is usually composed of *group* and *entry* items. In a tree analogy, a *group* item corresponds to a branching node, and an *entry* item corresponds to a terminal node. Two information fields are associated with the *group* and *entry* items: the screen display field and the action sequence. The screen display field holds the text that will appear in the scanner menu for that item. The action sequence represents the actions that result when the *group* or *entry* item is selected by the user. This sequence can be a mixture of simple text and special action codes. The simple text represents the "keystrokes" that will be placed in the keyboard buffer. The action codes (depicted by <action code>) provide for special functions including nonalphanumeric keyboard actions (cursor control, character shifting, *Del, Reset,* etc.), scanning and Morse code functions, and control of input and output.

After the action sequence has been executed, scanning begins again from a location on the scan tree determined by the type of item that was selected. The default starting location after an *entry* is the top of the tree, but it can be programmed to any location via special action codes. After a *group* item is selected, scanning continues at the next scan level.

The three remaining types of items are *initial, hold,* and *file.* An *initial* item provides a method to set starting conditions including scanning time parameters, Morse code timing parameters, operation mode, location of scan display, attachment of input switch, auditory feedback (tone or click), or spoken selections via voice synthesis (for visually impaired users).

Three timing parameters are used to define the characteristics of the scanning operation mode. First, the scan rate for regular and inverse scanning can be specified. A selection delay time can also be specified allowing a time delay between switch activation and item selection. Finally, a second delay can be programmed between item selection and the resumption of scanning. This can be very helpful to the user in locating the next desired item.

Three timing parameters are required to specify the timing characteristics for two-switch Morse code (6). ALTKEY permits independent specification of all three: (a) mcwait, the pause time used to signify the end of a character; (b) mcrepeat, the time length of repeating dots and dashes (also the hold time required for automatic repeat of dots or dashes); and (c) mcpause, the time length of the pause between repeating dots and dashes. For one-switch Morse code there is no repeat; the two applicable parameters are mcwait and mcdot, which defines the length of a dot.

The *hold* item allows the user to select and execute an action sequence by holding the selection key down for a prescribed minimum time. For example, the ALTKEY code, hold:5:<reset>, allows the user to reboot the computer (via the <reset> action code) by holding down the selection switch for 5 seconds. Another possible use for the *hold* item is to switch between Morse code and scanning operation modes.

The *file* item inserts the text of the named file into the text of the main scan definition file at the point where the file item occurs. The effect is the same as a macro-expansion in assembly language and allows the scan-tree programmer to place subtrees in separate files and then reuse them frequently without having to include the complete text of the file each time.

## DISCUSSION

A keyboard emulator must be invisible to the application programs it will be used with. On the input side, ALTKEY's keyboard-emulation technique is highly compatible with other programs because it uses standard BIOS routines to move characters out of the keyboard buffer. At the same time, ALTKEY enhances input capability by expanding the standard buffer to accept a lengthy text string from a single scan selection or Morse code sequence.

The precise emulation techniques used by other computer-access aids such as the PC AID, VIM, and Words+ systems are not published. The PC AID and the Words+ Scanning/Morse WSKE systems, while compatible with a number of programs, have demonstrated incompatibilites with programs that bypass the keyboard request interrupt such as Microsoft Word.[7] The VIM system establishes its own keyboard buffer and intercepts the keyboard request interrupt to maintain compatibility (7). However, this method may prove to be incompatible with programs that bypass the keyboard-request interrupt and additionally does not allow standard keyboard access while it is active (7).

There are definite tradeoffs in designing the format and size of a display for an alternative input system. ALTKEY was specifically designed to provide a highly programmable and flexible system that would only use one line of the screen. Other systems use larger display areas to increase input options, but they reduce the intelligibility of the application program display. The VIM system, for example, uses a display window of seven lines. An alternative approach is to use a full-screen display for the alternative input system and multiplex it with the application program video-display screen (8).

[7]Word, Microsoft Corp., 16011 NE 36th Way, Redmond, WA 98073.

To guarantee compatibility of display outputs, all output to the screen should be routed through the standard system video-interrupt routine. Many programs, including the operating system (either MS-DOS or PC-DOS), actually do use this method of screen access. ALTKEY intercepts each of these calls and updates the screen accordingly. Unfortunately, a growing number of programs access video memory directly, completely bypassing the video interrupt in order to improve the speed of the output display. As described above, ALTKEY indirectly monitors for this condition through the keyboard request interrupt to facilitate accurate screen update. This indirect approach fails on rare occasions with the result that one or two lines of the screen may become slightly jumbled. Even then, it is a temporary condition that rarely impinges seriously on functionality.

The PC AID ensures video compatibility by removing its scan line from the screen after each scan cycle. However, the user must hit the switch to return the scan line to the screen, which adds an extra switch hit to each selection and can significantly reduce user efficiency. The Words+ EZKeys and Morse WSKE systems are compatible with the displays of many applications, but they do not remove their displays from the screen to allow accurate screen update with applications that bypass the video interrupt. The Scanning WSKE does have an option that will remove the scan display from the screen after a selection, which helps reduce video-incompatibility problems.

In addition to transparency, a useful system must give access to as large a percentage of the available keyboard keys and key combinations as possible. ALTKEY can imitate all these choices through key codes, action codes for a specific function (e.g., <reset>), or action codes for a shifted key that produce a composite that will apply that shift key to future key codes (e.g., <altnext> or <altlock>).

The implementation of a tree-structured scan-definition file, which can be reloaded whenever needed, has an important impact on power and efficiency for the user. The scan-tree file provides full customization of the input modes, the scan menu presentation, and system output.

Several ALTKEY timing parameters can be individually adjusted within the scan-tree definition file. Most other computer-access systems do not provide comparable control. It is common to have "input rate" parameters automatically translate to more than one timing interval, such as both the scan rate and the selection delay; or all three Morse code timing parameters. Such implicit definition of timing functions based on fixed relationships between two or more parameters can restrict system optimization for individual users and reduce input rates.

The flexibility of the scan-tree construct also allows the scan menus to be tailored to maximize usability and efficiency. For example, a scan tree can be designed to follow the input modes and choices expected by an application program. This can be very effective for menu-driven programs in which at each step the user is presented with scan choices specific to that menu. Another useful aspect of the scan-tree construct is that new trees may be loaded into the scanner at any time, making it practical to have a special scan-tree definition file for each application. An application program can be automatically loaded together with the appropriate scan tree using batch files.

The scan tree can also be used to decrease the number of scan steps necessary to accomplish a task, especially one that involves making the same choice several times in succession. Moving the screen cursor down 10 lines on the video display using the down arrow key is a good example of this type of task. One solution is to use a scan selection that would send the "down arrow" multiple times. But a more elegant solution is possible by starting with the display,

*Right Left Up Down None*

and then programming the scan tree so that when a selection such as *Down* is made, it rotates to the first position in the scan line as shown:

*Down Up Right Left None.*

The user can then repeatedly choose *Down* very quickly as many times as desired. Once the user has executed all the *Down* moves, another direction (e.g., *Right*) may be selected with the new direction rotated to the front of the scan line:

*Right Left Up Down None.*

These rotating menus are achieved by using the appropriate action code in the scan-tree definition file. Selection of *None* can be programmed to return the user to any chosen menu.

Other scanning computer-access systems do not provide this degree of flexibility in the design of the scanning presentation menus. For example, PC AID allows 10 scan lines of 40 characters each, in which all items in a scan line are of the same length. This limits the number of branches in the scan tree and restricts the contents of each scan line. In addition, scan selections cannot be programmed in the Scanning WSKE system. However, this system does provide integrated abbreviation expansion and word-prediction capabilities, which can be tailored for an individual user.

ALTKEY's scan-tree definition file provides great flexibility for system output. As mentioned above, the existence of a general-action sequence field in the scan-tree file allows ALTKEY to perform special functions, in addition to its keyboard-emulation function. The general-action sequence overcomes a limitation found in other computer-access systems such as the PC AID and AFC. In these systems, a selection results in either text output, function execution, or a branch to a new scan line; combinations of these outputs are not allowed. Any number of functions can be executed with ALTKEY for a given selection, and these functions can be combined with text output and/or a branch to a new scan line. The special functions include the capability for generic serial or parallel port output for speech synthesis, generation of environmental control unit control signals, modem dialing, etc.

One difficulty with all memory-resident software is compatibility with nonresident application programs that are simultaneously stored in memory. ALTKEY has been successfully tested with a number of commercial programs. It works properly with most nonresident software, since it uses standard BIOS routines.

The potential for compatibility problems is compounded when more than one memory-resident program is loaded simultaneously; loading order of memory-resident programs can be an important consideration in maintaining compatibility. ALTKEY works properly with a variety of memory-resident software including keyboard-enhancement programs such as Prokey,[8] OneFinger,[9] and Quickey.[10]

## FUTURE WORK

One group of programs in which ALTKEY currently fails is those that use the video-display screen in graphics mode. The current version of ALTKEY supports character-display mode only. Thus, the scan line is not displayed properly when ALTKEY is run with graphics programs. As more and more application programs are written to support graphics mode, it will be important to remedy this shortcoming with graphics support.

ALTKEY is the basis for an efficient system to control a PC from one or two switches. The full power of the system can only be realized with well-de-

[8]Prokey, Rosesoft, 4710 University Way NE, Seattle, WA 98105.

[9]One Finger, Trace Research and Development Center, 1500 Highland Avenue, Madison, WI 53706.

[10]Quickey, Trace Research and Development Center, 1500 Highland Avenue, Madison, WI 53706.

signed scan-tree definition files that take advantage of the available features. However, creating these files can be a relatively complex and time-consuming task. One solution to this problem is the development of scan-tree libraries, containing files designed for general purposes, such as text entry or cursor movement. It is a much simpler task to customize existing files than to create entirely new ones. In using ALTKEY with patients at the University of Michigan, a number of useful scan-tree files have been developed. In the future, we hope to expand this library of scan trees to make the system as efficient and simple as possible.

**Acknowledgment:** ALTKEY was developed and written by Lincoln A. Jaros from the Rehabilitation Engineering Program, Department of Physical Medicine and Rehabilitation, University of Michigan.

## REFERENCES

1. Jaros LA, Levine SP. A single switch keyboard emulator for the IBM-PC. In: *Proceedings of the 8th Annual Conference, IEEE Engineering in Medicine and Biology.* IEEE, 1986:1823–5.
2. Jaros LA, Levine SP. ALTKEY: a special inputs program for the IBM-PC. In: *Proceedings of the 10th Annual Conference on Rehabilitation Technology.* Washington, DC: RESNA, 1987:642–4.
3. Schwejda P, Vanderheiden G. Adaptive-firmware card for the Apple II. *Byte* 1982;7:276–314.
4. Nelson PJ, Korba L, Park G, Crabtree D. The MOD keyboard. *IEEE Micro* 1983;3:7–17.
5. Dolman L, Meeks S. Alternative access methods for the IBM family of personal computers and true compatibles. In: *Proceedings of the 10th Annual Conference on Rehabilitation Technology.* Washington, DC: RESNA, 1987:678–9.
6. Levine SP, Gauger JG, Bowers LD, Khan KJ. Comparison of communication rates for Morse code and mouthstick keyboard entry. *Augment Altern Commun* 1986;2:51–5.
7. Rowley BA, Davis CA. Interfacing the disabled to computer software through virtual interfaces. In: *Proceedings of the International Conference on Rehabilitation Technology.* Washington, DC: RESNA, 1988:380–1.
8. Gunderson J, Vanderheiden G. One screen multiplexed keyboard for transparent access to standard IBM-PC software. In: *Proceedings of the International Conference on Rehabilitation Technology.* Washington, DC: RESNA, 1988:378–9.