PCT

WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification ^{5} **:** **G06F 3/023, B41J 5/10**	**A1**	**(11) International Publication Number:** **WO 90/07149** **(43) International Publication Date:** 28 June 1990 (28.06.90)

(21) International Application Number: PCT/US89/05745

(22) International Filing Date: 21 December 1989 (21.12.89)

(30) Priority data:

288,011	21 December 1988 (21.12.88)	US
392,727	11 August 1989 (11.08.89)	US
440,994	22 November 1989 (22.11.89)	US

(71)(72) Applicant and Inventor: FREEMAN, Alfred, B. [US/US]; Tibbetts Hill Road, RFD 3, Goffstown, NH 03045 (US).

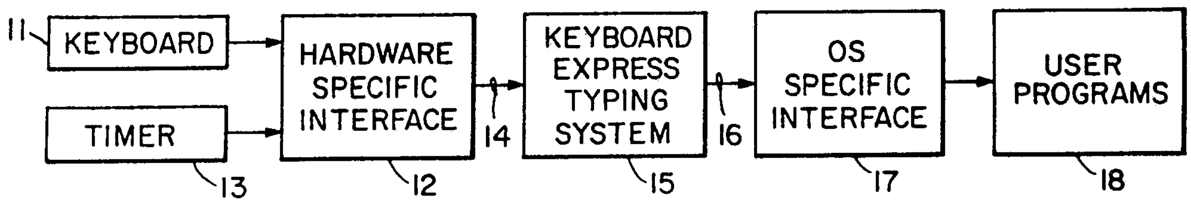
(74) Agents: SMITH, James, M. et al.; Hamilton, Brook, Smith & Reynolds, Two Militia Drive, Lexington, MA 02173 (US).

(81) Designated States: AT (European patent), AU, BE (European patent), CH (European patent), DE (European patent), ES (European patent), FR (European patent), GB (European patent), IT (European patent), JP, LU (European patent), NL (European patent), SE (European patent), SU.

Published

With international search report.
Before the expiration of the time limit for amending the claims and to be republished in the event of the receipt of amendments.

(54) Title: KEYBOARD EXPRESS TYPING SYSTEM



(57) Abstract

A keyboard input system (15) uses key combinations, or chords, in a number of ways to input information faster. Keyboards (11) range from augmented standards to one with ten keys, and chords range from ones very easy for typists to ones of all row key combinations by each hand. Chords of one stroke type the most frequent words along with regular typing or other chording of less frequent words. Other chording handles words as stems, or roots, and inflections which change words to different grammatical forms. Stem coding minimizes inputs for stems and data accompanying stems reduces inputs for inflections. A software program (17) to "Terminate but Stay Resident" with MS DOS on IBM Personal Computers and compatibles implements the preferred embodiment of the system for English.

FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AT	Austria
AU	Australia
BB	Barbados
BE	Belgium
BF	Burkina Fasso
BG	Bulgaria
BJ	Benin
BR	Brazil
CA	Canada
CF	Central African Republic
CG	Congo
CH	Switzerland
CM	Cameroon
DE	Germany, Federal Republic of
DK	Denmark
ES	Spain
FI	Finland
FR	France
GA	Gabon
GB	United Kingdom
HU	Hungary
IT	Italy
JP	Japan
KP	Democratic People's Republic of Korea
KR	Republic of Korea
LI	Liechtenstein
LK	Sri Lanka
LU	Luxembourg
MC	Monaco
MG	Madagascar
ML	Mali
MR	Mauritania
MW	Malawi
NL	Netherlands
NO	Norway
RO	Romania
SD	Sudan
SE	Sweden
SN	Senegal
SU	Soviet Union
TD	Chad
TG	Togo
US	United States of America

KEYBOARD EXPRESS TYPING SYSTEM

Background of the Invention

This application is a continuation-in-part of pending U.S. Patent Application Serial No. 07/288,011 filed December 21, 1988 and pending continuation-in-part U.S. Patent Application Serial No. 07/392,727 filed August 11, 1989, both parents being included herein by reference.

Field of the Invention

The present invention is directed generally to improvements in keyboard systems and particularly to such systems which use key combinations, or chords, to input text and other information faster.

Discussion of the Prior Art

The standard typewriter keyboard with the QWERTY letter key arrangement, despite its shortcomings, is the most widely used means for writing documents and inputting data to computers. August Dvorak et al., in U.S. Patent No. 2,040,248, sought to improve on the QWERTY arrangement but the Dvorak arrangement is not widely used. "The Dvorak Simplified Keyboard: Forty Years of Frustration", by Robert Patterson in Computers and Automation, November 1972, pp. 18-25 report on the failure of the inventors to get the Dvorak keyboard widely adopted. The Dvorak arrangement, however, is a currently available option for some computer keyboards. Various computer programs provide for assigning other characters and functions to keys, including long strings of commands, and for picking items from a menu by typing the first letter of the item.

The Stenograph is a machine for mechanically taking shorthand which produces an output for later transcription to text. The left and right hand fingers input beginning and ending consonant sounds, or letters, respectively by operating combinations of keys in two adjacent rows. The thumbs input vowel sounds, or letters, between the consonants by operating combinations of other keys. Some sounds, or letters, are assigned to keys and others to key combinations for operation by a finger or a thumb so each finger and thumb position, whether operating one or a combination of keys, corresponds to a sound, or letter. The transcriber sorts out the sounds, or letters, from the marks made by the keys on a paper tape and deciphers other spelling shortcuts to determine the text intended. Highly skilled operators reach rates of 200, or more, words per minute and many are employed for recording and transcribing Court and other public proceedings.

In U.S. Patent No. 4,042,777, Bequaert et al. disclose a one handed chord keyboard producing text output directly. As on the Stenograph, the fingers can each operate one, two, or four, keys in adjacent rows and the thumb one, or two, adjacent keys. Each finger position corresponds to a letter. The writing rates reported are far below those of the Stenograph, as might be expected for one hand and the additional complexity for text. Some chords produce word outputs and others strings of two or more letters. The strings generally have letters in the order in which they appear across the keyboard and with a thumb key able to reverse the order. Some chord string outputs are useful for text and others not. The typist can type words with fewer strokes by parsing them into the available strings. Parsing is not a simple task and may require considerable learning for high speed operation.

The article "Multilingual Word Processing", Joseph D. Becker, in the Scientific American of July 1984, discusses some of the difficulties for typing the large number of characters of ideographic languages and further describes a phonetic conversion system presently in use to type Japanese. In this system, the user types words in a phonetic alphabet to computer apparatus which searches a dictionary to find the words having the input sound strings. The sound strings are frequently matched by more than one word and the user then makes additional inputs to select the word wanted from those matched. The system efficiency is not as high as might be expected because of the considerable sound redundancy found in ideographic languages. One syllable, for example, may be used for as many as 64 different ideographs.

The following are of miscellaneous interest. Hilborn et al., in U.S. Patent No. 3,833,765, disclose a response to two successive key strokes for selection of characters, or selection of character strings, or even selection of whole messages. In U.S. Patent No. 4,333,097, Burie et al. disclose a programmable two handed keyboard having only ten keys. U.S. Patent No. 3,633,724 discloses a keyboard having eight keys, each moveable axially and pivotally to select a respective one of five characters out of a total of 40. The aforementioned U.S. Patent No, 4,042,777 includes reference Proceedings AFIPS 1968, Fall Joint Computer Conference, Pages 395-410, "A Research Center for Augmenting Human Intellect" by D.C. Englebart and W.K. English said to describe the selection of characters by use of 31 combinations of a five key keyboard.

As resported in "The Skill of Typing" by Timothy A. Salthouse in the Scientific American of February 1984, a century of study has not revealed how skilled typists type as fast as they do. Tests show that even skilled typists capable of 100 words per minute are limited to speeds of about 20 words per minute when forced to view letter by letter. The speed improves as the number of letters previewed is increased up to 19 or so, even for random text. All of the tests reported presented material as visual text, similar to typing from a text copy, an operation reduced in importance by presenting word processors. The tests did not deal with other inputs, such as self-originated material, dictation, audible record playbacks, shorthand, or stenotyping. All of the foregoing require spelling recall to precede letter translation to key actuations, mental operations which may require more time than is available for fast typing.

The mechanics of text writing involve a lot of learning starting with children practicing the forming of letters or other characters with pencil, pen, or brush. For alphabetic languages, the next step is learning to spell which continues over the years. The dictionary provides help for those who have enough knowledge to locate desired words in alphabetic listings. The typist can do away with the need to form letters or characters at the price of learning to strike keys for letters. But the spelling of words still has to be recalled and the letters converted to key actions. While basically simple and faster than handwriting, long periods of practice is necessary to acquire higher speeds. Speeds of 100 words per minute, which skilled typists can attain as pointed out in the afore-mentioned article by Salthouse, are approximately half the rate required of professional stenographers. Both the time required to acquire typing speed and the limit on the speeds attainable leave substantial room for improvement.

Summary of the Invention

The overall objection of the instant invention is a better and faster way to type any language.

An objective of the instant invention is a serial chording system which is compatible with regular typing and conventional keyboards, uses existing typing skills, and enables typists to rapidly increase typing speed.

Further objectives of the instant invention are optional chording features for the foregoing serial chording system to attain further increases in speed and, in combination, to enable some users to reach stenographic rates.

Still other objectives of the instant invention are chording systems which offer beginners and low skilled typists the easiest and fastest road to high speed text typing.

Yet other objectives of the instant invention are chording systems which provide more efficient control of auxiliary programs along with text or other typing.

Yet another objective of the instant invention is easy to learn one stroke chord typing of the words occurring most frequently in text along with regular typing or serial chording of the less frequent words.

Yet another objective of the instant invention is a chording system which enables children, as beginning students, to learn to write easily, and thereby have a capability to interact with computers, using natural language inputs, and so expedite other learning.

Still further objectives of the instant invention will become apparent from the following description and claims.

Bequaert et al. stated in U.S. Patent No. 4,042,777 that "a word could be assigned to each chord, but this would be hard to learn". While this statement may be true for the system of the aforementioned patent, it is not generally true. An important discovery of the instant invention is that single chord typing of words is inherently the easiest kind of high speed typing. While anyone who spells can immediately "Hunt" and "Peck" on labelled keys, typists achieve high speeds only after extensive drill. To put words on paper, or into a computer, regular typists select, or recognize, words to be types; recall, or read, the spelling of each word; then convert each letter to a key action. Rapid typing required physical skills and the mental processing of large amounts of data. Drilling somehow imparts this ability, perhaps by imposing a memory of the typing actions for at least frequent words and letter strings, on the "neural muscular" system, or the "fingers". Many typists, for example, turn to the keyboard when asked for a word spelling, the answers presumably being more easily accessible from the "fingers".

Single chord rapid typing of words is inherently easier for at least three reasons. One, the chord actions involve less data so the "fingers" have less to learn for semi-automatic high speed typing. (For a simplified analysis, a reasonable assumption for Applicant's chording system is that the average number of keys for typing words by chords is the same as for sequential typing. Chord typing required identification of sets of keys; regular typing additionally requires specification of key sequences; i.e., set versus set plus set sequence.) Two, while chord and regular typists both select, or recognize, words, only the regular typist has to recall and process the spellings. Three, chords require little physical skill and attain almost instant speed. Regular typing requires substantial physical skill for speeds well below those possible with chord typing. In summary, single chord word typing involves less data for memory by the "fingers", bypasses spelling recall and letter-key by letter-key processing, and more rapidly achieves higher speeds with less physical skill.

To exploit these inherent advantages, the instant invention systems provide several kinds of single chord typing for the most frequently used words of the language. Models embodying the invention obtain from 35% to more than 70% of the words found in text documents from such chords. Whatever the percentage, other means are necessary for typing the remaining words. The present invention systems employ various techniques to speed up typing of the remaining words. One technique distinguishes chording from regular typing, thus any word can always be typed regularly. Another technique codes words to access them from a vocabulary. (The codes are mnemonic and require less information to be inputted, whether by single chords or other means.) Breaking words into Stems and Inflections, or endings, is another important technique. (There are fewer Stems than words to be codes, separate inputs for Inflections get more words from Stems, two small sets working in combination reduce total inputs.)

Codes may consist of any set of elements relating to the Stems. Codes vary in size, the smallest ones being assigned to the most frequent Stems and larger ones to the less frequent Stems. For alphabetic languages, the codes consist of the starting letter strings, or SLSs. For non-alphabetic languages, phonetic sounds can be substituted for letters as stem code elements, the SLSs becoming starting phonetic sound representation strings. Since the sound representations, whether letters or other symbols, are not text, the system supplies all of the Stem text, not just the remainders as for SLSs. Other elements, such as parts of speech, may be useful and ideographic languages may supplement, or even replace, phonetic sounds as elements. Possible ideographic character related elements include Radicals (components) and numbers of strokes (parameters) to write characters.

Another variation of coding uses combinations of elements of first and second syllables and/or of first and second ideographic characters. Consonants, letters or sounds, contain more information than vowels. Syllable or ideographic character combinations are very discriminative of ideographic words.

Except for Chinese, all languages known to Applicant are inflected languages which apply Inflections to Stems to form more words. Some Inflections append suffixes to Stems, others change Stem spelling, and still others add auxiliary words. The proper appending of Inflection suffixes may require changes to the spelling of some suffixes and of many Stem endings. The invention systems use built-in rules, Stem ending Inflections arrays, and Stem Inflection codes to make the changes automatically. Users do not have to recall, or even know, the many rules and spelling variations. Inflections that change Stem spelling run into trouble if the letters to change are code elements already outputted. (One solution to this problem is to delay outputs until Inflection changes are made. Another solution is to single chord type the Inflected forms as words.) Auxiliary word Inflections may be treated as words or as inflections, depending upon the model and the language. The present invention systems include data in the vocabulary to specify small Inflection sets for each Stem that encompass most of the

Inflections found in text documents. These small sets reduce the inputs required to select Inflections, allowing them to be included in chords for other inputs.

The instant invention systems implement models for typists with regular keyboards. Basic features use only "key and Space Bar" chords and regular typing skills to enable typists to more than double typing speed. The "key and Space Bar" chords type 35% of the words found in text documents with one stroke, and serially types SLSs and, with the Space Bar held, add Inflections and Punctuation Strings. Enhancement features using chords still easy for typists put stenographic rates within reach: typing two words per stroke, typing many more words with one stroke, and combining Inflections and Punctuation Strings with Stem Code ending elements. Invention hand chords, or HCs, are single-hand operated combinations of keys of one row at a time. While each single-hand row location has 26 possible HCs, only the easiest, such as those of pairs of adjacent keys, suffice for important functions. In two-hand chords for SLSs, HC assignments supplement QWERTY keys to get consonants and consonant clusters with the left hand and vowels and prefixes to precede, and vowels to follow, the consonants with the right hand.

The present invention systems performance can be improved somewhat by modifying regular keyboards. One modification replaces the QWERTY with assignment patterns for more SLSs from two-hand chords. A second substitutes briding key caps for the index fingers, for easy operation of all 26 HCs. A third adds keys for thumb operation by splitting the Space Bar or replacing it with more keys. The thumb keys provide an additional chord section: for Inflections and Punctuation Strings, or for another code element, or for selecting sets of assignments for other keys. Thumb key sections may be especially useful for Japanese post positions along with other Inflections, and for handling multiple sets of Chinese Radicals. Modifications can also go the other way, reducing the number of keys. From the next paragraph's discussion of a Ten Key Keyboard, it is obvious that one row of keys can handle text typing. One-row text typing on a multi-row keyboard opens up some interesting possibilities for interfacing users to other programs. Another row of keys might control a word processor or other concurrently-used program. Or multiple rows might match up with multiple windows.

A Ten Key Keyboard, with eight keys in a row for finger operation and two keys for thumb operation, has enough chords to be competitive with other keyboards for the instant invention typing systems. Ten Key keyboards may, in fact, be the fastest possible for chord typing, offering a maximum stroke rate based on minimum movement and fewer choices. The Ten Key systems still have most of the chording performance features previously discussed. With one thumb key combination, or TKC, on word starts, each hand gets the same MFWs as "key and Space Bar" chords on regular keyboards and two-hand chords can get the same two-word sequences with one stroke. Also in two-hand chords for typing SLSs, one hand gets fifteen frequent consonants while the other hand gets ten vowels and prefixes to precede, and five vowels to follow, the consonants. With fill-ins for otherwise unused chords, Ten Key models can get 765 one stroke words and 254 first stroke starts without the Two-words per stroke feature. AFter first strokes, the TKCs select assignments for typing one or more letters or for typing last letters with Inflections or Punctuation Strings.

Other possibilities for beginners and low skilled typists include a Twelve Key Keyboard. The two extra finger keys add index finger reaches and increase the numbers by a factor of four. The over four thousand chords are good for over three thousand one stroke words and over one thousand first stroke starts. Just as an important contribution is that the 31 combinations per hand get additional consonant clusters for one hand and additional prefixes for the other, reducing the need for Fill-Ins. The alternative to minimum movement keyboards for beginners are regular keyboards with thumb keys added and improved assignment patters for two-hand chord typing of SLSs. Such models would require only the easier HCs and would spread the assignments spacially for reader comprehension. As the "fingers" prepare to operate the next key combinations while moving between rows, the impact of between-row movements on speed is less than might be expected. But beginner's "fingers" would have to learn the spacial locations of the keys, as for regular typing.

The Ten Key Keyboard has an important advantage for young students: small hands can cover the keys easily. Single chord typing of words is also advantageous for beginning students: by responding to graphical presentations of the keys for chords, they can learn to write before they learn to spell. (Just hear the word pronounced, see the presentation, operate the chord keys, see the word text.) The text displays are not only a reward for operating the right keys but also an aid for developing word recognition. The association of writing and reading should reinforce both. Children can first learn to type and to read their verbal vocabularies, which are extensive. Their early ability to communicate by keyboard with computers raises many possibilities for educational improvements. Small keyboards can fit on school desks, perhaps as part of terminals for classroom networks or as student computers also portable for homework assignments.

Compatibility with regular typing is an important feature of most embodiments of the instant invention. The present invention systems can distinguish inputs for chords from those for regular typing with different types of keyboards. Logic for keyboards which signal both key actuations and key releases is more involved but can tolerate a wide range of typing techniques. A range of choices are available for applications and include user adjustments. Logic for keyboards which signal only key actuations is simpler but a little more demanding of regular typing techniques. It is, of course, required for host systems without key release signalling keyboards. With some exceptions regular typing outputs of chord have to be delayed to see if an operated key will become part of a chord by operation of more keys. Key and chord output actions on key releases get the necessary delays with some degree of control left to the user. Fixed delays reset by each key actuation set limits only for successive actuations. Regular typing techniques vary widely with individuals; therefore, adjustments made manually or dynamically by the system are helpful.

Description of the Drawings

The foregoing and other objects, features, and advantages of the invention will be apparent from the following more particular description of a preferred embodiment of the invention, as illustrated in the accompanying drawings in which like referenced characters refer to the same parts throughout the different views. The drawings are not necessarily to scale, emphasis instead being placed upon illustrating the principles of the invention.

Figure 1 is an overall block diagram of embodiments of the instant invention systems.

Figure 2 is a functional block diagram of the main parts of embodiments of the instant invention systems.

Figure 3 shows arrays for English language modifications to Stem ends and Inflections.

Figure 4 is a block diagram of vocabulary packets for words or stems.

Figure 5 is a diagram of the packet pattern of a fragmentary section.

DESCRIPTION OF THE PREFERRED EMBODIMENTS

Figure 1 is a general block diagram of present invention system embodiments. The embodiments associate with host computers and use available components of the host. Nearly all host computers include a suitable keyboard 11, some type of hardware specific interface 12, a timer 13, some type of OS specific interface 17, and user programs 18. Keyboard 11, in response to key strokes, outputs key event codes to hardware specific interface 12. The key events always include key actuations and, in some apparatus, key releases as well. Timer 13 supplies timing signals to hardware specific interface 12. Hardware specific interface 12, which will have different forms in different host apparatus, outputs the key event codes and the time codes on line 14 to keyboard express typing system 15. Hardware specific interface 12 or keyboard express typing system 15 converts timer 13 inputs to time codes which represent the intervals since preceeding key events. Hardware specific interface 12 consists of software for the necessary functions, perhaps a modified version of the software previously handling outputs of keyboard 11 and timer 13.

Keyboard express typing system 15 uses the key event and time code inputs on line 14 from hardware specific interface 12 to produce outputs responsive to the operation of key combinations, or chords, and/or single keys. The outputs of keyboard express typing system 15 go via line 16 to OS interface 17. The OS specific interface 17, uses software routines of the host operating system, answers requests for keyboard services from user programs 18. OS specific interface 17 first looks to keyboard express typing system 15 for outputs to answer the requests, later turning to host apparatus if necessary. Keyboard express typing system 15 or OS specific interface 17 will include a buffer to hold data pending requests for services. Keyboard express typing system 15 either processes all the keyboard data or passes some key codes along to OS specific interface 17 for handling by host apparatus. The user programs 18 are those of the host apparatus requiring keyboard services and include those supplying text data to displays, printers, etc., for rendering or further processing.

Most keyboards 11 include a microprocessor to scan keys and output key event codes and can have a variety of characteristics as shown in the Keyboard Table following. As previously mentioned, some keyboards 11 produce both key actuation and key release code outputs and other key actuation codes only. Regular keyboards 11 can include key caps to enable index fingers to bridge more than one key in a row and can split the Space Bar to provide two keys for thumb operation without impairing regular typing performance. More substantial possible changes include adding sets of more than two keys for the thumbs and new configurations for row bridging. Still more revolutionary keyboards 11 have as few as ten keys to minimize the finger movements required to input data.

Keyboard Table: Summary List of Possible Variations for Keyboard 11 Output Signals

Key Actuation Signals Only
Key Actuation and Key Release Signals
Conventional Configuration
With Bridging Key Caps for Index Fingers (and Other?)
With Space Bar Split into Two Keys for Operation by Thumbs
New Configurations
Multiple Rows of Keys for Row Bridging
With Set of Keys for Thumb Operation

New Minimum Movement Keyboards

Ten Keys--Key per Finger and Thumb

Twelve Key Keyboard--Ten Key Plus Two for Index Fingers

The invention keyboard express typing system 15 is preferably implemented by software which can run on the host computer. The software could also be run by a microprocessor made a hardware part of system 15 or even by the microprocessor normally a part of keyboard 11. Hardware could be on plug-in cards or built into keyboard 11. OS specific interface 17 otherwise has to provide some way to run the system 15 program on the host CPU. In the case of MS DOS running on the IBM Personal Computer, the Terminate but Stay Resident, or TSR, service is the way. Those skilled in the art are aware of other ways to run the system 15 software on other host systems. The main difference aside from the savings in hardware, between using the host CPU and a microprocessor is the times at which the system 15 program would run. On the host CPU, the system 15 program might only run in response to requests for keyboard services to minimize CPU time used. On a microprocessor, the system 15 program could instead run in response to each keystroke, keeping keyboard data always up to date.

In Figure 2, which shows a functional block diagram of keyboard express typing system 15, next key 20 receives the key event and time codes on line

14 from hardware specific interface 12. Next key 20 starts the processing of each key event by determining whether the key involved is a chord key or a non-chord key. Chord keys are those which may become part of a chord when operated in combination with other chord keys. Systems with a conventional keyboard 11 generally use letter, punctuation, and digit keys as chord keys but other keys could be included also. In some systems, all keys may be chord keys. Non-chord key actions can be immediate but chord key actions have to be delayed to see if more keys will be added for chords. Next key 20 outputs non-chord key codes to non-chord keys 21 and selectively routes chord key codes to chord key actuate 22, chord key release 23, or delay time 24. The actuation codes obviously go to chord key actuate 22, the release codes to chord key release 23, and the time codes not accompanied by key codes, to delay time 24.

Non-chord keys 21 passes the other key scan codes received to key code translator 25. Depending on the embodiment, non-chord keys 21 may receive all non-chord key events or only those used locally. Key code translator 25 responds appropriately to the non-chord keys, setting shift bits, checking for special key combinations, etc. Key code translator 25 also translates chord key codes, and non-chord key codes if appropriate, to ASCII, or other character codes. Chord key actuate 22 puts the actuated chord keys in chord data register 26 and supplies outputs for the Key Repeat function to key code translator 25. If, as in the IBM Personal Computer, keyboard 11 repetitively supplies actuation codes for Repeat actions, chord key actuate 22 checks chord data register 26 and produces outputs if the key is already registered. Chord data register 26 not only stores the keys but also registers other data, such as totals for different types and categories.

In embodiments in which keyboard 11 supplies key release codes, chord key release 23 responds with outputs to chord logic 27. In other embodiments in which keyboard 11 only signals chord key actuation events, delay time 24, instead of release 23, signals chord logic 27. Delay time 24 produces a trigger to chord logic 27 whenever the interval since the last actuation exceeds a prescribed value. The Actuation Logic Table shows the key and chord logic for Actuations Only embodiments to be very simple. Chord logic 27 responds to the triggers from delay time 24 by acting on the keys currently in the chord data register 26 and then clearing the register. Spaced apart actuations thus produce single key outputs while actuations within short intervals of each other produce chord outputs. Chord logic 27 also includes means to reject some combinations of keys as illegal and to clear them without other action. The prescribed delay interval is long enough to catch key actuations for chords and short enough to handle regular typing strokes individually. A disadvantage of this logic is that there is no way to detect held over keys.

Actuation Logic Table: Key & Chord Logic for Actuations Only Systems

CHORD KEY EVENTS	CONDITIONS	ACTIONS
Actuation	If key not in register	Register key
	Else Repeat On & only one in register	Output Key
Delayed Trigger	If only one key	Output key Clear register
	Else if > one key & key combination legal	Output chord Clear register
	Else if illegal chord	Clear register

The key and chord logic for embodiments using both key actuations and key releases is somewhat more complex as shown by the Release Logic Table following. Chord key actuate 22 still puts the actuated chord keys in chord data register 26 and handles Repeats as previously discussed. In these embodiments, chord key actuate 22 also sets a chord condition flag for successive actuations within a prescribed interval. Chord logic 27 gets the released key time and scan codes from chord key release 23. If only one key is in chord data register 26, chord logic 27 outputs the released key code to single key action 28 for a single key action. If there are more than two keys, chord logic 27 outputs the data for a chord to chord action 29. Chord logic 27 always removes the released keys from chord data register 26 and removes all of the keys involved in chord actions.

When two keys are operated, chord logic 27 used several conditions to distinguish regular typing overlaps from two key chords. The two keys are a chord if they have been operated together longer than a prescribed interval. The two may also be a chord if they were actuated within a prescribed interval of each other or if the key released first was the last actuated. Chord logic 27 uses different length prescribed intervals for different inputs or situations. The intervals are chosen to best distinguish easy chording from regular typing by various types of users. Chord logic 27 also considers some key combinations as illegal chords and ignores the keys or outputs them one at a time for single key action.

Release Logic Table: Key & Chord Logic Table for Key Release Systems

CHORD KEY EVENTS	CONDITIONS	ACTIONS
Actuation	If key not in register	Register key
	Else Repeat On & only one in register	Output Key
	If sucessive actuations within window	Set act. condition
Release	If only one key	Output key Clear key
	Else if key not registered	No action Clear key
	Else if > two keys & key combination legal	Output chord Clear register
	Else if two keys legal &	
	If time > chord_time	Output chord Clear register
	Else	Output key Clear key
	If Act. condition or If released key was actuated last	Output chord Clear register
	Else if illegal chord	Clear register

Note: In some embodiments, non-released keys are inhibited instead of being cleared from the register. Chord key actuate 22 reactivates any Inhibited keys still being held when an actuation occurs after a prescribed interval following the last release.

Chord Logic 27 calls single key 28 to handle regular typing strokes and chord action 29 to handle chord strokes. Single key 28 outputs the key code inputs to key code translator 25 which converts them to ASCII, or other, character codes for output via output queue 30 to user services 31. Shift status and other outputs of translator 21 go directly to user services 31 which outputs all on line 16 to user programs 18 via OS specific interface 17. Single key 28, for embodiments retaining a regular typing capability, can also be used where single keys are considered chords or all functions can be taken over by chord action 29. Chord action 29 analyzes the contents of chord data register 26, received directly or via chord logic 27, to determine the type of chord action required. Chord action 29 also keeps track of the current word state, changing the states responsive to successive inputs. Word State Table shows word states along with outputs and next word states. In Word Start State, inputs produce word or word start outputs. In Word Started State, inputs continue or end words and may add word endings. In Word Ended State, some inputs add word endings while others advance the state to the Word Start State and then get words or word starts.

Word State Table: Word States, Outputs, and Resulting States

WORD STATE	OUTPUTS	RESULTING STATE
Word Start	One Stroke Words	Word Start
	Starts, Text or Code	Word Started
Word Started	Continue Text or Code	Word Started
	End Word	Word Ended
	Endings for Word	Word Ended
	Complete Word	Word Start
Word Ended	Endings for Word	Word Ended
	Complete Word	Word Start
	Words or Starts	Word Start

Before a further description of chord action 29 functions, it is necessary to first discuss an input coding feature of the instant invention, herein labeled Stem Coding. This Stem Coding feature speeds up typing and lightens the burden on typists by reducing the input data required to produce text. Stems are basic words and/or roots from which other words are formed with the application of Inflections. Inflections include suffixes which append to the Stems, internal Stem text changes, and accompanying auxiliary words. In Indo-European and other Inflected languages including Japanese, a relatively small set of Inflections expand word Stems into more words. Separating words into Stems and Inflections and using Stem Codes reduces the input data required. Stem Code inputs obtain text for assigned Stems from a writing vocabulary, as will later be explained, while separate inputs apply Inflections. Embodiments for different languages employ different Stem Code elements in different types of Stem Coding as shown by the following Stem Code Table.

Stem Code Table: Possible Stem Code Elements

ELEMENTS	CODES
Alphabetic	
Letters	Starting Letter Strings
Consonants	Starting Letters of Syllable Strings
Non Alphabetic	
Phonetic Sounds	Starting Phonetic Sound Strings
Consonant Sounds	Starting Sounds of Syllable Strings
Ideographic	
Character Radicals	Radical Strings With Phonetic Sounds
Number of Strokes	Number Strings With Phonetic Sounds
Miscellaneous	
Parts of Speech	
Order of Element Types	

Alphabetic language embodiments generally use letters as Stem Code elements and starting letter strings, or SLSs, as Stem Codes. Users already know the starting letters and so only have to learn SLS lengths. A consonant letters only alphabetic Stem Code would be more efficient but would be harder to learn. Non-alphabetic language Stem Codes can be similar to SLSs, merely replacing Stem text letters with Stem phonetic sound representations consisting of letters or other symbols. As phonetic sounds are used more redundantly in Ideographic languages than in Indo-European languages, Stem Codes having other elements to supplement, or replace, phonetic sounds may be advisable. Other possible Stem Code elements for Ideographic languages include Ideographic character Radicals and the "Number of Strokes" to form characters. Chinese use the former and Japanese the latter to index dictionary listings. Parts of speech and the order of input of types of elements are other possibilities. All types of Stem Codes assign the shortest, or smallest, codes to the most frequently used Stems and the longer, or larger, ones to less frequent Stems.

Chord action 29, in response to the contents of chord data register 26, transferred by chord logic 28, and to the Word State, manages chord actions by calls to start/continue chords 35, word chords 37, and/or endings chords 39 and by control signals to search 32 and user service 31. Start/continue chords 35 transfers Stem Code elements from element data 36 either to output queue 30 or to code buffer 34. Output queue 30 receives the outputs in embodiments using Stem Code elements that also serve as Stem text characters. Code buffer 34 gets the outputs in other embodiments. In the Word Start State, the start/continue chords 35 outputs start Stem text and/or Stem Codes. In the Word Started State, the start/continue chords 35 extend, or continue, Stem text and/or Stem Codes. Word chords 37, which receives calls only in the Word Start State, transfers Stem Codes for words from element 36, depending upon the embodiment, either to output queue 30 or to code buffer 34. The transfers are the same as those by start/continue chords 35 except that they are used to produce Stem, or word, outputs as will be explained next.

When chord action 29 receives an input marking the end of inputs for a Stem Code, it signals search 32 to look for the Stem in vocabulary 33a assigned to the Stem Code just inputted to either output queue 30 or code buffer 34. In embodiments in which word chords 32 transfers elements from element data 36, inputs for calls to word chords 32 also mark Stem Code ends. If it finds the Stem, search 32 transfers the Stem text, or the remainder of the Stem text, to output queue 30. In embodiments using SLSs as Stem Codes, letter character codes in output queue 30 resulting from single key 28 outputs to key code translator 25 are also Stem Code elements. In other embodiments, single key 28 transfers elements from element data to code buffer 34. Word chords 37 can also respond to inputs from chord action 29 with transfer of Stem text of whatever type from text data 38 to output queue 30 without involving Stem Coding and search 32. Such text transfers are helpful for words not using Stem Codes, such as short words and high frequency groups of words with the same starting strings or other element sets. This handling of text for one stroke words is further useful in embodiments not employing Stem Coding.

In most embodiments retaining regular typing, user services 31 outputs the letters from output queue 30 as soon as possible. Output queue 30 then has to continue to store the letters after output or search 32 has to get the SLS letters from output queue 30 before they are outputted. In either alternative, search 32 is able to respond to the Stem Code end triggers from chord action 29 with a search and transfer of the remainder of the Stem Text from vocabulary 33a to output queue 30. In other embodiments in which SLS letters may have to be changed by Inflections, as will later be explained, user services 31 does not output the contents of output queue 30 until it receives a signal from chord action 29 that the word is ended. Output queue 30 or search 32 then do not have to make other provisions for SLSs, search 32 merely uses the contents of output queue 30 in searching. Search 32 similarly uses the contents of code buffer 34 for searches in embodiments with non-text Stem Codes but vocabulary 33a will have a different arrangement and will supply all of the Stem Text.

Chord action 29 routes inputs for word endings to endings chords 39. The word endings handled by endings chords 39 stretch the literal meaning by including changes to Stem text and auxiliary words along with Inflection suffixes, Punctuation strings, and "Space" characters as shown in the Word Endings Table.

Word Endings Table:
- Inflections:
  - Text Suffixes
  - Changes to Stem Text
  - Auxiliary Words
- Punctuation Strings: Punctuations marks, "Space" characters, and automatic capitalization of next sentence starts.

Other:
- Non-Inflection Suffixes and Words for Compounding:
- Word Ending Characters:
- "Space" Character
- Other Non-Letter Characters

Endings data 40 stores word ending data for the language of each application. The data ranges from suffix text to codes and rules for Stem changes. For English, for example, endings data 40 would include the data in the following English endings Table and Rules to handle as many as possible of the Spelling Change Examples of the following list.

English Endings Table: English Inflections and Punctuation Strings

1	2	3	4	5	6	7	8	9	0
(!)	age	ate	bility	ist	ism	ish	(')	(;)	(:)
q	w	e	r	t	y	u	i	o	p
ical	ic	ous	ful	less	y	ity	ness	ce	cy
a	s	d	f	g	h	j	k	l	;
al	PL/3S	Past	ing	er/or	ive	ion	ble	ly	Past_P
z	x	c	v	b	n	m	,	.	/
ize	nt	nce	ncy	bly	est	ment	(,)	(.)	(?)

List of English Inflection Spelling Change Examples:

1. Word Endings Are Altered For Appending Certain Inflections:
   - i) final "e" drops:
     "believe--believable", "trade--trading"
   - ii) "y" changes to "i":
     "carry--carried", "happy--happiest".
   - iii) final consonant doubles:
     "put--putting", "prefer--preferred".
     "put-putting", "prefer--preferred".
2. Word Endings Determine Leading Vowel for Some Inflection Suffixes:
   - i) "ble" and "bly" are preceeded by "a" or "i": "manageable, measurably; reproducible, reversibly".
   - ii) "nt", "nce", and "ncy" are preceeded by "a" or "e": "informant, attendance, conservancy"; "superintendent, insistence, tendency".
3. Irregular Bridges "at" and "it" Are Inserted Before "ion" and "ive": "form--formation, formative; add--addition, additive".
4. More Elaborate Changes Are Made to Both Word and Suffix:
   "explain--explanation, acquire--acquisition, assume--assumption, justify--justification, apology--apologetic", etc.

Items 2, 3, and 4 of the list are better handled by append data 41 than by Rules. Append data 41 stores combination of ends of Stems and beginnings of Inflection suffixes together with the changes to the text of either or both for appending. Figure 3 shows some examples for English in the form of arrays from C source for a particular embodiment. Each of the arrays are for an Inflection and have several units with the "/" character between two text strings. The first of the two strings is the end of the stem with spelling reversed for convenience in searching. The second string is the replacement for the first, spelled in text order, to precede the standard Inflection suffix. The se_es_plurals array, for example, includes the stem ends which require "es" in place of "s" for the plural. Most of the units do not change the stem end and are included only to identify those which take "es". Others change "fe" and "f" to "v" to set up for "es". The se_ion_ive array, for another example, has a very large number of units changing the spelling of stem ends, the changes being the same for "ion" and "ive".

In some cases, endings chords 39 only transfers suffix or auxiliary word text from endings data 40 to output queue 30 to follow the Stem text. In other cases, endings chords 39 first uses append data 41 to modify either or both the Stem text in output queue 30 and the Inflection suffix text from endings data 40. For additional help, vocabulary 33a includes Stem Inflection Codes, or SICs, with the Stems. Search 32 supplies the SICs to endings chords 39 upon the successful completion of searches. The SICs help endings chords 39 produce Inflected forms for their Stems that are not covered by the usual rules or other data in endings data 40 or append data 41. The English Stem Inflection Code Table following uses SICs starting with 128, the high bit being set as a marker for other purposes.

English Stem Inflection Code, or SIC, Table:

NO_DOUBLE	0x01	/* don't double final consonant */
OR_NOT_ER	0x02	/* use "or" rather than "er" */

CODE_MIN	128
IVE_ION	132
NT_NCE	136
NT_NCY	140
MENT	144
NO_PAST	148
NO_WILD_CARD	152
AL_ALLY	156
OUS_OUSLY	158
IC_ICAL	160
FUL_LESS	162
ER_EST	164
ISM_IST	166
ITY	168
NESS	170
CE	172
CY	174
AGE	176
ATE	178
ESS	180
IZE	182
DOM	184
HOOD	186
SHIP	188
FOREIGN_PLURAL	190
BUILD_BUILT	192
LEAD_LED	196
KEEP_KEPT	200
BRING_BROUGHT	204
TEACH_TAUGHT	208
COME_CAME	212
HOLD_HELD	216
WIN_WON	220
FIND_FOUND	224
HANG_HUNG	228
BEGIN 232	
WRITE_RISE	236
EC_KNOW	240
ADD_N_NE	241
GET_GOT	244
SPEAK_SPOKE	248
BIT_BITE	252

The first line shows the bit to change SICs to indicate that the final consonant is not to be doubled (even though the stem ends with a "vowel-consonant-vowel" the suffix starts with a vowel). The second line shows the bit for "or" instead of "er". The listed SICs for regular verbs, nouns, and adjectives show different possible "To Noun" and "To Adjective" Inflection or other "Wild Cards" for the respective stems. The listed SICs for irregular verbs indicate the types of changes for Past Tense and sometimes for Past Participle also. The particular irregular verbs with each SIC are only by way of example, the SICs also being with other verbs having the same type of change.

The regular verbs have three different sets of "To Noun" and "To Adjective" Inflections: "ion & ive", "a/ence/y & a/ent", "ment & mental". Noun Stems have the "To Adjective" Inflections: "al", "ful", "less", "ic", "ical", and "ous". Adjectives have the "To Noun" Inflections: "ce", "cy", "ity", and "ness". These are all considered to be "Wild Cards" as different Stems picks one of several for the same job. Other types of "Wild Cards" are listed for SICs 176-188. While each Inflection can be assigned, as in the English Endings Table, picking the Stem limits the possible Inflections to a few. Two keys, or two chords can get the "Wild Cards" determined by the SICs. Any Stem uses only a few of the possible Inflections of the language, some only one or two "Wild Cards". A number of the most frequent Inflections, or MFIs, including the "Wild Cards", make up small sets which include a majority of the Inflections used. The advantage of having the SICs identify small sets is that a small set on inputs can make all the selections. Selection from the small sets can be made by parts of start/continue chords 35 for last Stem Code elements and, especially in embodiments with keyboards having a set of thumb keys added, even by parts of word chords 37. Word endings would otherwise require additional strokes for inputs to endings chords 39.

Vocabulary 33b is shown to indicate possible auxiliary sections for applications needing such sections for any of a number of possible uses. Vocabulary 33b can hold words or strings for: a part of speech; a technical field; a computer language; control sequences for other programs; any of a variety of customized uses. Chord actions 29 supplies data, either as part of Stem Codes or directly to search 32, which directs search 32 to use vocabulary 33b instead of vocabulary 33a for its search. Special chords or shift keys can provide the inputs to distinguish vocabularies 33a and 33b. Vocabularies 33a and 33b can use any internal arrangement that allows search 32 to locate selected Stems in the time available. Preferred arrangements, however, also enable easy location of Stems assigned the next longer Stem Code from any Stem code input. Chord part inputs, like those described for MFIs, can then get "Next Stems" with one less stroke.

Selecting MFI members and "Next Stems" with parts of chords inputting last Stem Code elements has the virtue of saving strokes but requires more learning. Look ahead display action 42 gets data for Help Displays to assist both learning and using. Display action 42 receives chord key actuate inputs from next key 20 and, starting with the second element for words, first saves the data currently in output queue 30 and then generates outputs to chord action 29 for calls to word chords 37, search 32, and endings chords 39 to put display data in output queue 30. User service 31 transfers the display data in output queue 30 to line 16 in response to requests for same by user programs 18. (User programs 18 have to have the capability to display the data.) The next output from chord key release 23 or delay time 24 triggers display action 42 to put the original data back in output queue 30. The display data consists of the text which could be obtained by including other keys along with the one triggering display action 42. Such text includes Stems obtainable with inclusion of an end of Stem Code indication, "Next Stems" obtainable with inclusion of more keys, and forms of the Stem Inflected by the MFIs also obtainable with inclusion of more keys.

Vocabularies 33a and 33b can use Stem Packets, such as the one shown in Figure 4, arranged in patterns such as the one shown in Fig. 5.
The Stem packet 50 of Fig. 4 has a Jump Next Branch byte, a Next Code element Byte, one or more text data bytes, and a Stem Inflection Code byte. In alphabetic languages using SLS Stem Codes, the letters of the Stem Code are outputted so the text data section only has to supply the remaining letters of each Stem. For Stem Codes not useful as Stem text, the text data section has data for the complete text of each Stem. The Jump Next Branch byte holds the value for a jump to the next search location if the Next Code Element does not match the input. The Next Code Element is the last element of the Stem Code for the particular Stem packet 50, preceding elements being located in other packets 50. The Stem Inflection Code byte marks the ending of the Stem packet 50 and supplies Stem Inflection data to assist endings chords 39 and identify MFI sets for the Stem as has been explained.

Fig. 5 is a partial block diagram of an arrangment of Stem packets 50 for vocabulary 33a. Lines 51a go to Stem packets 50a having a one element Stem Code. In some systems, search 32 applies the first element input to lines 51a to find the one of Stem packets 50a having a matching element. In other systems, search 32 uses the first element to locate a line 51b to a section of vocabulary 33a for the first Stem Code element. In the first case, search 32 looks for a match of the first element in Stem packets 50a and in the second case for a match of the second element in Stem packets 50b. Search 32 uses the Jump Next Bytes to move to successive branches of Stem packets 50a or 50b while searching for matches. If a match is found and more elements are inputted, search 32 moves through the matching Stem packet 50a, 50b, 50c, etc. and searches the next level Stem packets for a match with the next element. Search 32 outputs the one of Stem packets 50a, 50b, 50c, etc. matching the last Stem Code element inputted when it receives an indication of the end of a Stem Code.

The SLS Stem Code Table shows some words all starting with "ab" and having their SLS and Stem remainders separated by '-'s. The relative frequency of use of words having the same starting letter strings determine the length of SLSs assigned, the more frequent getting the shorter SLSs. Words having the same length SLSs are in the same column. The columns for the lengths two, three, four, etc. correspond to the subscripts b, c, d, etc. of packets 50 in Fig. 5. The respective packets 50 for the words include only the last letters of the SLSs along with the Stem remainders. The starting letter "a" identifies an "a" section similar to the one shown in Fig. 5. Packet 50b of the section has "b" as its Next Code Element and "out" as its Stem remainder. The following packets 50c would have "o", "l", "i", and "s" as their Next

Code Elements. The particular order is based on the relative frequencies of the respective words to set up the most frequent for "Next Stems" and also to minimize overall search time. Search 32, in response to input of the letters "abi", for example, would locate the section for "a", then find the packet 50b for "b", then the packet 50c for "i", then transfer "lity" to output queue 30. "Next Stems" inputs following input of "b" would get "above", "able", "ability", and "absent".

SLS Stem Code Table: Stem Code SLSs Starting with "ab".

Two	Three	Four	Five	Six Letters
ab-out	abo-ve	aboa-rd		
		abol-ish		
		abor-t	abori-gine	
		abod-e		
		abom-inate	abomi-nable	
		abou-nd		
	abl-e	ablu-tion		
	abi-lity	abid-e		
	abs-ent	abst-ract	absta-in	
			abste-ntion	abstem-ious
			absti-nent	
			abstr-use	
		abso-lute	absor-b	

KEY & SPACE BAR CHORDS:

Chords consisting of key and Space Bar combinations are easy for typists and others to learn and in the present invention system are sufficient to substantially increase typing speed. As Word Start State inputs, these chords can one stroke type a set of the most frequent words, or MFWs, of any language. For this action, chord logic

27 recognizes the combinations in chord data register 26 as chord, chord action 29 produces outputs to word chords 37 to transfer corresponding word text from text data 38 to output queue 30. If these are not to be the only one stroke words, then the words might be the most frequent which are a best match for the respective keys. The MFW Table, arranged as a keyboard map, shows a set for English by way of example. (This particular pattern provides for a key per hand two word sequence feature to be described later.) These words comprise approximately 35% of those found in English text documents and so the key and Space Bar Start State chords can contribute substantially to speed.

MFW Table: English MFWs for key & Space Bar QWERTY Keyboard Chords

1	2	3	4	5	6	7	8	9	0
one	two	three	four	five	six	seven	eight	nine	zero
q	w	e	r	t	y	u	i	o	p
with	was	on	in	to	you	all	I	they	this
a	s	d	f	g	h	j	k	l	;
as	is	and	for	of	he	the	a	it	that
z	x	c	v	b	n	m	,	.	/
are	at	had	have	by	not	she	we	which	his

"Key and Space Bar" chords in the Word Started State are used in what is herein called Serial Chording to mark the ends of Stem Code inputs. For Serial Chording, the letters of the SLSs of alphabetic languages, or the Stem Code elements of other languages are typed serially to the end where the Space Bar is included with the key for the last letter or element. Single key 28 places serially typed letters in output queue 30 or serially typed non-text Stem Code elements in code buffer 34. Chord logic 27 detects the "Key & Space Bar" chords, has single key 28 handle the last element and calls search 32 to handle the Stem Code and transfer text from vocabulary 33a to output queue 30. As previously discussed, the text is either for the complete Stem or the Stem remainder. Drill on sets of words all using the same number of elements, two, three, four, etc. assist the "fingers" to learn the new way of typing the words.

The "Key & Space Bar" chords of the Word Started State also lead to the Word Ended State. Releasing the Space Bar gets a "Space" character output to the end of the word. Holding the Space Bar and striking another key results in chord action 29 sending the key code to word endings 39. Word endings 39 responds by applying the ending assigned to the key to the text in output queue 30. As previously discussed, endings include text suffixes, changes to Stem text, Punctuation Strings, and auxiliary words. The English Endings Table, as an example, shows a set assigned to the keys of a QWERTY keyboard. Most of the Inflections shown are suffixes which word endings 39 appends to the Stem text in output queue 30 with any necessary changes to either or both Stem and Inflection suffix text spelling. Word endings 39 uses endings data 40 and append data 41 to make the spelling changes automatically, as previously discussed, so users do not have to recall, or even know, the many complex rules involved. Some examples are listed in the English Endings Table.

The Capitalized entries in the English Endings Table represent a range of possible Inflections; PL/3S for various Plurals, including Foreign Plurals, and Third Singular verbs, Past for verb past tenses, and Past_P for verb past participles. Punctuation Strings consist of punctuation marks, one or more "spaces", and those ending sentences include automatic Capitalization of next sentence starts. Past and Past_P for some Stems are irregular Inflections that change spelling Stem text, substituting different vowels, etc. as specified by the SIC of the Stem.

English Endings Table: English Inflections and Functuation Strings

1	2	3	4	5	6	7	8	9	0
(!)	age	ate	bility	ist	ism	ish	(')	(;)	(:)
q	w	e	r	t	y	u	i	o	p
ical	ic	ous	ful	less	y	ity	ness	ce	cy
a	s	d	f	g	h	j	k	l	;
al	PL/3S	Past	ing	er/or	ive	ion	ble	ly	Past_P
z	x	c	v	b	n	m	,	.	/
ize	nt	nce	ncy	bly	est	ment	(,)	(.)	(?)

List of English Inflection Spelling Change Examples:

1. Word Endings Are Altered For Appending Certain Inflections:
   i) final "e" drops:
      "believe--believable", "trade--trading".
   ii) "y" changes to "i":
      "carry-carried", "happy--happiest".
   iii) final consonant doubles:
      "put--putting", "prefer--preferred".
2. Word Endings Determine Leading Vowel For Some Inflection Suffixes:
   i) "ble" and "bly" are preceded by "a" or "i":
      "manageable, measurably; reproducible, reversibly".
   ii) "nt", "nce", and "ncy" are preceded by "a" or "e":
      "informant, attendance, conservancy";
      "superintendent, insistence, tendency".
3. Irregular Bridges "at" and "it" Are Inserted Before "ion" and "ive"
   "form--formation, formative; add--addition, additive".
4. More Elaborate Changes Are Made to Both Word and Suffix:
      "explain--explanation, acquire--acquisition, assume--assumption
      justify--justification, apology-apologetic", etc.

EXPANSIONS WITH MORE EASY CHORDS:

The More Easy Chords Table lists additional chords that use little more than existing typing skills to further increase typing capabilities. The main use of these chords is in the Word Start State for the single stroke typing of more of the next most frequent words. The "Key & Space Bar" chords have already been discussed. The two adjacent key hand chords, or HCs, consist of pairs of adjacent keys in one row for operation by one hand, with/without the Space Bar. These chords are easily executed by two fingers acting together as one, or alternatively, by one finger on keyboards 11 with bridging key caps. The next two types of easy chords make use of the typist skill in typing a letter with one hand while operating a Shift with the other. The first type uses the '/' and 'z' keys as special shift keys. The second type uses the HC composed of the pair of '/' and '.' keys and the HC of the pair of 'z' and 'x' keys as special shift HCs. Both of these types combine these special shifts with both keys and HCs. (This type of one stroke word typing could be further extended by using additional keys and/or HCs as special shifts.)

More Easy Chords Table:

CHORD TYPES	WORD START	WORD STARTED
Key & Space Bar	MFW	End Stem Code
Two Adjacent Key Hand Chords, or HCs	VHFW	2nd Type Element
& Space Bar	VHFW	2nd Type Element & End Stem Code
SX Key Shift & (Key or HC)	HFWs	Continue Stem Code
& Space Bar	HFWs	Continue & End Code
SX HC Shift & (Key or HC)	HF Verb Past Tense	---
& Space Bar	HF Verb Stem	---

HC Table: VHFWs for Two Adjacent Key HCs, W/Wo Space Bar

(1-2)	(2-3)	(3-4)	(4-5)	(6-7)	(7-8)	(8-9)	(9-0)
January	February	March	April	May	June	July	August
Monday	Tuesday	Wednesday	Thursday	Friday	Saturday	Sunday	Holiday
(q-w)	(w-e)	(e-r)	(r-t)	(y-u)	(u-i)	(i-o)	(o-p)
will	were	who	why	when	my	their	thing
would	well	where	whole	what	me	them	things
(a-s)	(s-d)	(d-f)	(f-g)	(h-j)	(j-k)	(k-l)	(l-;)
also	so	but	or	him	there	an	its
another	some	if	from	himself	then	any	itself
(z-x)	(x-c)	(c-v)	(v-b)	(n-m)	(m-,)	(,-.)	(.-/)
may	can	has	be	no	more	her	our
might	could	having	been	now	most	here	us

SX Key Table: HFWs for SX Key & Key or HC chords, W/Wo Space Bar

1	2	3	4	5	6	7	8	9	0
first	second	third	fourth	fifth	sixth	seventh	eighth	ninth	zeroth
will	would	or	from	but	more	when	an	who	there
q	w	e	r	t	y	u	i	o	p
without	way	even	right	time	year	up	into	out	people
within	world	each	room	times	years	upon	interest	only	place
a	s	d	f	g	h	j	k	l	:
about	same	day	few	good	how	just	kind	long	than
after	since	days	far	great	however	job	knowledge	life	through
z	x	c	v	b	n	m	,	.	/
again	example	course	very	back	new	man	many	while	---
against	experience	city	voice	both	never	men	members	white	---
(q-w)	(w-e)	(e-r)	(r-t)		(y-u)	(u-i)	(i-o)	(o-p)	
quite	work	every	enough		under	other	once	part	
question	war	end	eyes		until	over	off	public	
(a-s)	(s-d)	(d-f)	(f-g)		(h-j)	(j-k)	(k-l)	(l-:)	
almost	development	face	general		high	left	last	something	
always	different	fact	government		hand	less	later	system	
(z-x)	(x-c)	(c-v)	(v-b)		(n-m)	(m-,)	(,-.)	(.-/)	
among	school	child	better		next	mind	set	point	
area	schools	childern	best		night	means	side	program	

SUBSTITUTE SHEET

SX HC Table: HF Verbs for SX HC & Key or HC chords, W/Wo Space Bar

1	2	3	4	5	6	7	8	9/0	
eleven	twelve	thirteen	fourteen	fifteen	sixteen	seventeen /eighteen/nineteen/twenty			
ten	twenty	thirty	forty	fifty	sixty	seventy	eighty	ninety	hundred
q	w	e	r	t	y	u	i	o	p
work	write	expect	run	take	tell	use	think	show	pay
worked	wrote	expected	ran	took	told	used	thought	shown	paid
a	s	d	f	g	h	j	k	l	;
give	say	do	find	get	hear	go	know	leave	look
gave	said	did	found	got	heard	went	knew	left	looked
z	x	c	v	b	n	m	,	.	/
choose	call	come	become	begin	need	make	meet	stand	send
chose	called	came	became	began	needed	made	met	stood	sent
(q-w)	(w-e)	(e-r)	(r-t)		(y-u)	(u-i)	(i-o)	(o-p)	
wear	win	ride	teach		turn	understand	increase	speak	
wore	won	rode	taught		turned	understood	increased	spoke	
(a-s)	(s-d)	(d-f)	(f-g)		(h-j)	(j-k)	(k-l)	(l-;)	
ask	see	drive	feel		hold	keep	lead	sell	
asked	saw	drove	felt		held	kept	led	sold	
(z-x)	(x-c)	(c-v)	(v-b)		(n-m)	(m-,)	(,-.)	(.-/)	
SX HC	break	bring	build		move	start	steal	SX HC	
SX HC	broke	brought	built		moved	started	stole	SX HC	

English examples of very high frequency words, or VHFWs, for HCs are listed in the HC Table, examples of high frequency words, or HFWs, in the SX Key Table, and of high frequency Verb Stems and Past Tenses in the SX HC Table. (The VHFW and HFW categories would perhaps better be designated as VHFW I and VHFW II, or HFW I and HFW II, as the relative-frequencies overlap are not as important as other factors in the division.) These words, and corresponding words of any language, could all be stored as text in text data 38. Some or all could also be assigned Stem Codes, obtainable from element data 36, and be available from vocabulary 33a or 33b. In either case, word chords 37 responds to inputs from chord action 29 to put the word text in output queue 30, using search 32 in the Stem Code case as previously described. These words and the previously discussed MFWs comprise approximately 70% of those found in English text documents.

In the Word Started State, start/continue chords 35 uses HC inputs from chord action 29 to get frequent combinations of elements from element data 36. For SLS Stem Codes, the HC assignments would consist of frequently used letter combinations, such as consonant clusters, prefixes, consonant vowel syllables, vowel consonant syllables, etc. For non-alphabetic languages, the HC assignments can be a similar set of phonetic sound combinations or instead be a set of different type elements, such as the "Number of Strokes" or the "Radicals " of Ideographic characters. In the latter case, the HCs would type the different type elements assigned while the keys typed the phonetic sounds. The order of input of element types could further distinguish

Stem Codes, minimizing the number of other elements used. As previously mentioned, combinations of phonetic sounds and other types of Stem Code elements might make more efficient Stem Codes for Japanese and other Ideographic languages.

The HCs can further type more Inflections and other word endings or replace keys for typing endings. The latter use reduces the number of multiple assignments for keys and offers an alternative serial logic for word endings. HCs are better used for word endings, when included in Both Hand Chords to type MFIs to be discussed next. In this use, one hand uses the keys to type the last Stem Code element while the other hand uses HCs to select a member of the set of most frequent Inflections, or MFIs, for the Stem. The MFIs, as mentioned previously, comprise a majority of the Inflections used so typing them with last elements provides a substantial saving. Stem Inflection Codes, included with the Stems in vocabulary 33a, specify the MFIs to endings chords 39. As shown by the English MFI & Next Stems HC Table following, the HCs of either hand can cover MFIs, including "To Noun" and "To Adj" wild cards to be discussed later, along with some of the previously discussed "Next Stems".

English MFIs and Next Stems HC Table:

LEFT HAND				RIGHT HAND			
(q-w)	(w-e)	(e-r)	(r-t)	(y-u)	(u-i)	(i-o)	(o-p)
a/ible	"ing"	PL/3rd	Next #3	Next #3	"ing"	PL/3rd	a/ible
(a-s)	(s-d)	(d-f)	(f-g)	(h-j)	(j-k)	(k-l)	(l-;)
a/ibly	"ly"	Past	Next #2	Next #2	Past	"ly"	a/ibly
(z-x)	(x-c)	(c-v)	(v-b)	(n-m)	(m-,)	(,-.)	(.-/)
SX	To Adj.	To Noun	Next #1	Next #1	To Noun	To Adj.	SX

BOTH HAND CHORDS:

The previously discussed SX Key and SX HC chords employed both hands but one hand handled "Shifts" only. The Both Hand Chords to be discussed here use each hand to make selections. The Both Hand Chord Table lists several possible types starting with "Key/Hand" chords. The "Key/Hand" chords are shown to obtain two word sequences in the Word Start State and two elements, or an element and a Punctuation String, in the Word Started State. Some may prefer to substitute two elements for the two word sequences in the Word Start State. The second group listed consists of "HC/Hand" and "Key with HC" chords, all distinguishing from the "Key/Hand" chords. The third group listed is for any combination of keys and HCs with the Space Bar included. The Space Bar distinguishes from other "Key/Hand" chords. Inclusion of the Space Bar to indicate the end of inputs for Stem Codes was discussed for other Serial Chording. The punctuation keys get Punctuation Strings, and HCs get MFIs and "Next Stems" along with last Stem Code elements.

Both Hand Chord Table:

CHORD TYPES	WORD START	WORD STARTED
Key/Hand	Two MFW Sequence	Two Elements Element & Punct. Str.
Key & HC or HC/Hand	Stem Code Start	Continue Stem Code & MFI or Next Stem
(Key or HC)/Hand & Space Bar	Stem Code Word	End Stem Code & MFI or Next Stem
(Punct & Key)	MFW & Funct. Str.	End Stem Code & Punct. Str.
(Punct & HC)	VHFW & Funct Str	End Stem Code & Punct. Str.

The MFW Sequence Table shows the MFW Table assignments with alternate words added to the digit keys and to some other keys. The alternate words may come into play when the respective keys are included in Key/Hand chords for sequences of two MFWs. The system includes logical rules, responsive to particular key combinations, to determine the order of output of words from each pair of keys and to determine when alternate words are to be substituted. For English, pronouns more frequently precede verbs but follow prepositions, articles only follow verbs, objects of prepositions are a genitive case, pronoun number determines verb form, etc.

MFW Sequence Table:

1	2	3	4	5	6	7	8	9	0
one	two	three	four	five	six	seven	eight	nine	zero
/will	/would	/or	/from	/but	/more	/when	/an	/who	/there
q	w	e	r	t	y	u	i	o	p
with	was	on	in	to	you	all	I	they	this
	/were				/your		/my	/their	
a	s	d	f	g	h	j	k	l	;
as	is	and	for	of	he	the	a	it	that
	/are/am				/his			/its	
z	x	c	v	b	n	m	,	.	/
are	at	had	have	by	not	she	we	which	his
			/has			/her	/our		

As previously mentioned, HCs can have Stem Code elements and element combinations assigned for the Word Started State. The Both Hand HC QWERTY Table shows assignments that also apply for the Word Start State. These assignments supplement the QWERTY key assignments so Both Hand Chords can obtain a very large number of useful letter combinations for SLSs or SLS Starts. Start/continue chords 35 gets the combinations from element data 36 for Stem Code Starts when the Space Bar is not included. When the Space Bar is included, word chords 37 gets the same combinations from element data 36 but calls search 32 to output the assigned Stems or words. As QWERTY patterns have most of the consonant letters assigned to the left hand, the left hand HCs include the right hand consonants along with frequently used consonant clusters. The right hand HCs, along with right hand keys, get vowels to follow, and vowels and prefixes to precede, left hand consonants. The possible combinations thus include most Consonant Vowel, Vowel Consonant, Consonant Cluster Vowel, and Prefix Consonant combinations along with others.

Both Hand HC QWERTY Table: Two Adjacent Key HC QWERTY Supplement

1	2	3	4	5	6	7	8	9	0
(1-2)	(2-3)	(3-4)	(4-5)		(6-7)	(7-8)	(8-9)	(9-0)	
co	com	dis	per		un	de	re	pro	
q	w	e	r	t	y	u	i	o	p
(q-w)	(w-e)	(e-r)	(r-t)		(y-u)	(u-i)	(i-o)	(o-p)	
pr	wh	th	p		be	al	in	-e	
a	s	d	f	g	h	j	k	l	;
(a-s)	(s-d)	(d-f)	(f-g)		(h-j)	(j-k)	(k-l)	(l-;)	
sh	st	l	h		a	e	o	-a	
z	x	c	v	b	n	m	,	.	/
(z-x)	(x-c)	(c-v)	(v-b)		(n-m)	(m-,)	(,-.)	(.-/)	
pl	ch	m	n		mo	so	to	-r	

The Dvorak keyboard is better than the QWERTY for combining with HC assignments as the vowels are on one hand keys while the consonants are on other hand keys. The Dvorak keyboard, however, is not widely used so there is little reason to adopt its particular pattern of key assignments. Keyboards for beginners can start over to optimize use of keys and HCs. The Key-HC Element Table indicates some possible patterns. The objective, like that just described for the QWERTY, is to have a keyboard in which the key and HC assignments for one hand be those which most frequently combine with those of the other. The general alphabetic pattern of the Key-HC puts consonants on the right hand instead of the left, as for the QWERTY, so the order, except for suffix vowels, is left to right. Without Key-HC duplication, more letter combinations are possible. For Japanese, the left hand can handle some syllables along with all vowel and consonant syllable starting sounds. The right hand has room for "Number of Strokes" and all the syllable ending sounds.

Key-HC Element Table:

LEFT HAND	RIGHT HAND
General Alphabetic:	
Vowels as Prefix	Consonants
Vowels as Suffix	Consonant Clusters
Prefixes to Consonants	Prefixes to Vowels
Japanese:	
Number of Strokes, 1st Char.	Number of Strokes, 2nd Cha
Syllables	Number of Strokes
Syllable Starts	Syllable Ends

As previously discussed, the HCs can get MFIs and Next Stems on the same stroke with last Stem Code elements. The English Stem MFI Set Table shows some additional details. Verb Stems have three Inflections common to nearly all verbs, some Verb Stems also have "To Noun" and "To Adjective" Inflections, and others different Past Participles. Noun Stems have two common Inflections and some nouns have one or two "To Adjective" Inflections. Adjective Stems have one common Inflection, some have "To Noun" Inflections, and others comparative and superlative Inflections. The "To Adjective"s may cascade with "ly" and the "To Noun"s with plurals. Some words serve both as verb and noun or as more than one part of speech in other combinations. The "To Noun" and "To Adjective" Inflections, specific to particular Stems, are shown to range over a variety of forms.

English Stem MFI Set Table:

Verbs	Past Tense, Third Singular, Present Participle	
	Some Stem Additions:	
		Past Participle
		To Noun, "ion", "ment", "a/ence/y"
		To Adjective, "ive", "a/ent", "a/ible/y"
Nouns	Plural, Possessive	
	Some Stem Additions:	
		To Adjective, "al", "ful", "ic", "ical", "less", "ous"
		Misc., "dom", "hood", "ship"
Adjective	To Adverb, "ly"	
	Some Stem Additions:	
		To Noun, "ce", "cy", "ity", "ness"
		Comparative, Superlative, "er", "est"

ADDITION OF MORE HCs:

The two adjacent key HCs are only four of the 26 possible for each hand in each row. The 22 remaining are available for further expansion. Of the 22, six use only two keys and another six consist of the three adjacent, four adjacent, and five adjacent combinations. Either of the two sets of six HCs more than double the two adjacent HCs previously covered. With bridging key caps for the index fingers to operate adjacent keys, all 22 HCs are still relatively easy to execute. Applications thus have a wide range of possible HC sets from which to choose. While there are no limits on the number which might be used for more one stroke words, few, if any, more would be useful for MFIs and Next Stems or for more suffixes. More one stroke words would only require the expansion of chord action 29 and text data 38 or element data 36 for the additional Stems. These include expansion of Both Hand Chords for Stem Code element combinations. The Both Hand Chords are promising as they are more mnemonic and so desirable for large numbers. Further, the same chords, except for with/without the Space Bar, get of one stroke Stems and first stroke Starts. With practice, the "fingers" learn the combinations and users only have to think about the words they want to type.

Both Hand Chord expansions have many possibilities, some of which are shown in the following Expanded Both Hand Chord Table. Just a few more HCs could get all of the consonant clusters, vowel clusters, prefixes, etc. useful for alphabetic languages. The relatively limited number of syllables of Japanese, and perhaps of other languages, could be covered by the additional HCs of either hand so Both Hand Chords could get two syllable combinations. The 26 HCs and five keys per row of one hand could get 93, or 124, of the 214 Radicals of Ideographic characters. The other hand could get "Number of Strokes" along with switching the set of Radicals so all 214 would be covered. "Number of Strokes" of one Ideographic character and the Radical of the same, or another, character is thus possible from Both Hand Chords.

Expanded Key-HC Elements for Both Hand Chord Table:

LEFT HAND	RIGHT HAND
Alphabetic:	
All Vowels, Vowel Clusters Syllables	All Consonants, Consonant Clusters, Syllables
All Prefixes to Consonants	All Prefixes to Vowels
Japanese:	
1st Syllable	2nd Syllables
Japanese and Other Ideographic:	
1st Char. Number of Strokes	2nd Char. Number of Strokes.
Radicals in Two or More Sets	Number of Strokes & Select Set of Radicals for Left Hand
* Radicals in Two or More Sets	Radicals in Two or More Sets

* Set of Thumb Keys or other means to select all set combinations.

ADDITION OF THUMB OPERATED KEY SETS:

Added sets of keys for thumb operation are another way to add capabilities. Two thumb keys, or

TKs, have four possible combinations, or TKCs; four TKs have 16 TKCs. First, the TKCs can get MFIs and Next Stems, both with chords for last Stem Code elements and with chords for one stroke words. Second, the TKCs can input additional Stem Code elements along with those from Both Hand Chords. Third, the TKCs can select the set of inputs for the hand, or hands. Japanese, as in Inflected languages with either Inflections or post positions on nearly every word, could best utilize TKs for MFIs and post positions. Chinese, not an Inflected language, could instead use TKs to select the different sets of Radicals for the two hands. Both Hand Chords could then get all first and second character Radical combinations. The TKCs can also select multiple sets of one stroke words, the words of the sets being related in various ways.

A TEN KEY KEYBOARD EMBODIMENT:

Ten Key Keyboards 11, with a key for each finger and thumb, require minimum movement for typing and so offer a maximum stroke rate. As will be realized from the following description, the per stroke capability compares favorably with that of the conventional keyboards 11. In addition to maximum stroke rate, minimum movement also requires less physical skill of beginners and those unskilled typists whose "fingers" have not yet learned the spacial locations of the conventional keyboard keys. The user positions fingers and thumbs on the keys and only has to move them enough to operate the keys. The typing presents a minimum of information for the "fingers" to learn. The "fingers" are restricted to the part of the muscular neural system for the key operations while "fingers" for conventional keyboards 11 include the muscular neural system for spacial movements as well.

The following Word Start State Ten Key One Hand Sets Table shows four sets of assignments for each hand, one for each TKC. Each set is spread over three rows, the first for single keys, the second for two key HCs, and the third for the three and four key HCs. This pattern affords a spacial dimension for each type that hopefully makes the 15 HCs of each hand easier to learn and associate with the assignments. It will be noted that the TKC I Set compares with the previously described set for "Key & Space Bar" chords on three rows of keys of a conventional keyboard 11. (Both include the same 30 English MFWs arranged for possible two word sequences from Both Hand Chords.) TKC II and TKC III sets consist of VHFWs relating to the MFWs for the same HCs. The TKC IV set includes letters, digits, some consonant clusters, and Special Shifts for typing any string of characters or starting anything possible with regular typing.

Word Start State Ten Key Keyboard One Hand Sets Table:

LEFT HAND	RIGHT HAND
TKC I SET;	
he that a the	of and to in
they not this I it which	is was for with as on
there she all you one	at by had but have
TKC II SET;	
his than an then	if or two into
their no think my its when	are were from will about only
through her also your other	may be can so has
TKC III SET;	
him those any these	off more too important
them now thought me itself who	am well first would after once
though here always years out	might been could said having
TKC IV SET;	
a e i o	q x y
u 1 2 3 4 5	z qu ch br sh Special
6 7 8 9 0	pr st th wh Shift

The Word Start State Both Hand Chord Sets Table following includes TKC I set of MFWs with alternatives for word sequences. This set corresponds to the previously described MFW Sequence Table for conventional keyboards 11. Except for the digit key assignments, TKC I HCs/Hand operates similarly and gets most of the two word sequences previously described for the MFW Sequence Table. The set of one and more letters for SLSs is enabled for all three TKCs, II, III, and IV. The left hand vowels marked with an '*' follow, while other vowels and other prefixes precede, the right hand consonants. Both Hand Chords output left and right hand combinations as SLSs. TKC IV takes the SLSs and first stroke Starts on longer SLSs. TKC II gets a first set of words for th SLSs while TKC III gets a second set of words related to those of TKC II. Instead of being used for two word sequences, TKC I could be used for a third set of SLS related words.

Word Start State Both Hand Chord Sets Table:

LEFT HAND						RIGHT HAND					
TKC I SET;											
he	that	a	the			of	and	to	in		
/his											
they	not	this	I	it	which	is	was	for	with	as	on
/their			/my	/its		/are	/were				
						/am					
there	she	all	you	one		at	by	had	but	have	
/her			/your							/has	
TKC II, III SET for SLS Stems, Words;											
*a	*e	*i	*o			b	c	d	f		
*u	s	a	e	o	al	g	h	k	l	m	n
be	co	de	in	re		p	r	s	t	w	
TKC IV SET CONTINUE;											
*a	*e	*i	*o			b	c	d	f		
*u	s	a	e	o	al	g	h	k	l	m	n
be	co	de	in	re		p	r	s	t	w	

*Vowels follow Right Hand consonants in Both Hand Chord combinations.

Chord action 29, start/continue chords 35, and word chords 37 would handle the Word Start State chords with the assignments in element data 36 and word text 38 much as previously described. The one hand chords get 90 of the most frequent words with one stroke and 30 starts. The Both Hand chords can get up to 225 Starts and up to 450 more one stroke words and two word sequences or up to 675 more one stroke words. Not all of the two hand letter combinations are useful, however, so some combinations instead get Fill-In SLSs or words. The Fill-Ins impose a somewhat heavier load than do the hand assignment combinations but are probably necessary for the totals to be comprised of the most frequent words. The 540 or 765 one stroke words, perhaps as many as users of conventional keyboards 11 will wish to learn, make a substantial contribution to typing speed. The 225 SLS Starts will serve nearly all of the longer SLSs.

The following Word Started State One Hand Chord Sets Table shows TKC I Set to consist of all the letters along with three prefixes not likely to be used. The TKC II Set has the Inflections and Punctuation Strings of the rows of keys of the English Endings Table previously discussed. The TKC III Set is shown as having More Endings to indicate that it is available if more are needed. The TKC IV Set repeats the letters of the TKC I Set for continuations of SLSs or words.

The following Word Started State Both Hand Chord Sets Table shows TKC I Set as repeating the TKC I Set of one hand chords. (Left hand letters precede right hand letters in two hand combinations.) The TKC IV Set repeats this set for continuations of SLSs or words while the TKC II Set has the same left hand and TKC III the same right hand. The right hand of the TKC II Set and the left hand of the TKC III Set each have a set of MFIs, '.' and ',' Punctuation Strings, and a set of Next Stems. Both Hand Chords can thus get many two letter combinations for endings and for continuations and last letters with the MFIs or most frequent Punctuation Strings or Next Stems. The one stroke words previously described get a very high percent of the words of English text documents. Second strokes can obtain a large number of next most frequent words further pushing the percentage. Only a small percent will thus require more than two strokes.

Word Started State One Hand Chord Sets Table:

LEFT HAND						RIGHT HAND					
TKC I SET;											
a	e	i	o			b	c	d	f		
u	s	j	q	v	x	g	h	k	l	m	n
y	z	de	in	re		p	r	s	t	w	
TKC II SET;											
To_A	PL/3S	Past	ing			er/or	ble	ly	To_N		
al	ous	less	ful	ic	ive	y	bly	cy	ce	ity	ness
ize	nt	ical	nce		ncy	est	ment	(,)	(.)	(?)	
TKC III SET;											
More Endings						More Endings					
TKC IV SET CONTINUE;											
a	e	i	o			b	c	d	f		
u	s	j	q	v	x	g	h	k	l	m	n
y	z	de	in	re		p	r	s	t	w	

MINIMUM MOVEMENT VERSUS CONVENTIONAL KEYBOARD CHORDING:

On conventional keyboards 11, the HC preparation time for successive strokes can pipeline with movements between rows. The losses in time for between row movements may thus be reduced by the HC preparation time. Executing the same HCs in different rows will take little more learning than in one row. The row reaches will require additional learning for those who are not already typists. The spacial distribution of multiple row chording with its use of additional muscles and neural networks could have countering advantages even for beginners. It is possible, however, that beginners will find that minimum movement chording or either a Ten Key or a Twelve Key keyboard 11 is both easier to learn and faster. A Twelve Key keyboard 11 adds index finger reaches and generally increases total numbers by a factor of four. A Twelve Key keyboard 11 could also get along without Fill-Ins.

Word Started State Both Hand Chord Sets Table:

LEFT HAND							RIGHT HAND					
TKC I SET;												
a	e	i	o				b	c	d	f		
u	s	j	q	v	x		g	h	k	l	m	n
y	z	de	in	re			p	r	s	t	w	
TKC II SET;												
a	e	i	o.				ly	ing	Past	PL/3S		
u	s	j	q	v	x	(,)	(.	C)	a/ibly	a/ible	To_N	To_A
y	z	de	in	re			Nxt5	Nxt4	Nxt3	Nxt2	Nxt1	
TKC III SET;												
PL/3S	Past	ing	ly				b	c	d	f		
To_A	To_N	a/ible	a/ibly	(.	C)	(,)	g	h	k	l	m	n
Nxt1	Nxt2	Nxt3	Nxt4	Nxt5			p	r	s	t	w	
TKC IV SET CONTINUE;												
a	e	i	o				b	c	d	f		
u	s	j	q	v	x		g	h	k	l	m	n
y	z	de	in	re			p	r	s	t	w	

While the invention has been particularly shown and described with reference to preferred embodiments thereof, it will be understood by those skilled in the art that various changes in form and details may be made without departing from the spirit and scope of the present invention as defined by the appended claims.

CLAIMS

1. In a keyboard system to output data to a data handling system which includes a keyboard having a plurality of keys and means for sensing operations of keys, the improvement comprising:

   means responsive to said sensing means for registering actuated keys;

   a set of data packets assigned to sets of one or more of said keys;

   means responsive to the pattern of successive operations of said keys for enabling the output of data packets assigned to keys registered by said registering means;

   means responsive to said enabling means for outputting enabled data packets; and

   means for limiting the outputs from keys registered by said registering means to one.

2. A keyboard system as claimed in Claim 1 wherein said enabling means is responsive to the intervals following actuations of said keys exceeding a predetermined length.

3. A keyboard system as claimed in Claim 1 wherein:

means responsive to the pattern of successive operations of said keys for selecting either the set comprised of said released key or the set of said registered keys; and

means responsive to key releases and said selecting means for outputting the data packet assigned to said selected set.

4. A keyboard system as claimed in Claim 3 wherein said selecting means is responsive to the sets registered by said registering means.

5. A keyboard system as claimed in Claim 3 wherein said selecting means includes means responsive to successive operations of said key for timing the intervals between said operations.

6. A keyboard system as claimed in Claim 5 wherein said selecting means is responsive to the intervals preceeding release of said keys.

7. A keyboard system as claimed in Claim 6 wherein said selecting means has different responses to said preceeding intervals for different sets of keys registered by said registering means.

8. A keyboard system as claimed in Claim 5 wherein said selecting means is responsive to predetermined intervals between successive actuations of said keys.

9. A keyboard system as claimed in Claim 1 wherein:

said keys include a space bar and are arranged in multiple rows suitable for conventional typing; and

said data packets for sets of single keys comprise sets of characters for regular typing.

10. A keyboard system as claimed in Claim 9 wherein others of said data packets assigned to sets of more than one of said keys consist of very frequently used words.

11. A keyboard system as claimed in Claim 1 wherein some of said data packets are for very frequently used words.

12. A keyboard system as claimed in Claim 1 wherein:

some of said data packets are for word stems; and

others of said data packets are for inflections for word stems.

13. A keyboard system as claimed in Claim 1 wherein some of said data packets are for word endings.

14. A keyboard system as claimed in Claim 13 wherein said word endings include punctuation strings.

15. A keyboard system as claimed in Claim 1 wherein:

said keys are divided into more than one set of sets; and

said enabling means enables the output of data packets assigned to each of said set of sets.

16. A keyboard system as claimed in Claim 15 wherein:

the data packets assigned to a first of said set of sets consist of a first set of very frequently used words; and

the data packets assigned to a second of said set of sets consist of a second set of very frequently used words.

17. A keyboard system as claimed in Claim 15 wherein:

the data packets assigned to a first of said set of sets consist of a first set of very frequently used word stems; and

the data packets assigned to a second of said set of sets consist of a set of very frequently used inflections.

18. In a keyboard system to output data to a data handling system which includes a keyboard having a plurality of keys and means for sensing key operations, the improvement comprising:

a set of word stems;

a set of stem codes each assigned to a respective word stem and consisting of a set of code elements relating to the respective word stem, said sets of elements being smallest for the most frequently used word stems and larger as necessary to be unique for less frequently used word stems;

means responsive to operations of said keys for producing said stem codes; and

means responsive to said producing means for outputting to said data handling system the text for the respective word stems.

19. A keyboard system as claimed in Claim 18 including:

a set of inflections; and

means responsive to said sensing means for applying inflections to the text from said outputting means.

20. A keyboard system as claimed in Claim 18 wherein said producing means is responsive to special sets of said keys operated together for marking the end of inputs for said stem codes.

21. A keyboard system as claimed in Claim 20 wherein:

said keys include a space bar and are arranged in multiple rows for conventional typing; and

said special sets are those including said space bar.

22. A keyboard system as claimed in Claim 21 wherein:

said sets of elements comprising said stem codes consist of lengths of the starting strings of characters of the respective stems and said characters are assigned to said keys; and

said producing means is responsive to a sequence of operations of said keys for the respective characters ended by a set including said space bar.

23. A keyboard system as claimed in Claim 22 wherein:

said characters consist of the letters of the starting letter string of the respective stems.

24. A keyboard system as claimed in Claim 22 wherein said characters represent the phonetic sounds of the starting phonetic sound string of the respective stems.

25. A keyboard system as claimed in Claim 18 wherein:

said keys are divided into more than one set of sets;

sets of stem code elements assigned to said sets of sets; and

said producing means is responsive to combinations of said stem code elements for keys of more than one of said set of sets operated together.

26. A keyboard system as claimed in Claim 25 wherein:

said stem code elements assigned to a first one of said sets consist of stem code elements relating to a first syllable; and

said stem code elements assigned to a second one of said sets consist of stem code elements relating to a second syllable.

27. A keyboard system as claimed in Claim 18 wherein:

the text of the word stems consist of more than one ideographic character; and

said stem codes include elements relating to said ideographic characters.

28. A keyboard system as claimed in Claim 27 wherein said stem code elements include the number of strokes for forming one of said ideographic characters.

29. A keyboard system as claimed in Claim 27 wherein said stem code elements include radicals identifying a part of one of said ideographic characters.

30. A keyboard system as claimed in Claim 18 wherein:

said keys are divided into more than one set of sets;

a set of stem code elements assigned to said sets of sets;

a set of frequently used inflections assigned another of said set of sets;

said producing means is responsive to the stem code elements assigned to the keys of the respective one of said set of sets operated together; and

including means responsive to keys of said another set of sets operated together for applying the assigned inflection to the text from said outputting means.

31. In a keyboard system to output data to a data handling system which includes a keyboard having a plurality of keys and means for sensing key operations, the improvement comprising:

a set of stems;

a set of inflections of said stems;

first means responsive to operations of said keys for selecting stems;

second means responsive to other operations of said keys for selecting inflections;

means responsive to said first and second selecting means for combining said selected inflections to said selected stems; and

means for outputting the combinations of said selected stems and said selected inflections to said data handling means.

32. A keyboard system as claimed in Claim 31 wherein:

said first selecting means also selects a set of the most frequently used inflections for said selected stem; and

said second selecting means selects from said set of most frequently used inflections.

33. A keyboard system as claimed in Claim 32 wherein said other operations of said keys for said second selection means occurs together with operations of said keys for said first selecting means.

34. A keyboard system as claimed in Claim 31 wherein said other operations of said keys for said second selection means occurs after operations of said keys for said first selecting means.

35. A keyboard system as claimed in Claim 34 wherein:

said keys include a space bar and are arranged in multiple rows suitable for conventional typing; and

said operations for said first selecting means end with operation of said space bar and said space bar is held operated to enable said second selecting means.

36. A keyboard system as claimed in Claim 31 wherein:

said inflections consist of suffixes; and

said combining means appends said suffixes to said stems.

37. A keyboard system as claimed in Claim 36 including means responsive to selected inflections and selected stems for modifying either as necessary for the particular combination.

38. A method of inserting text in a data handling system having processor means and a keyboard input device with a multiplicity of element keys arranged in a plurality of rows on a keyboard, each key providing a different text element in response to user operation thereof, the method comprising the steps of:

operating element keys to form codes uniquely corresponding to desired text, the codes including a series of text elements; and

processing through the processor means the formed codes such that the corresponding desired text is inserted and displayed on the display means.

39. A method of inserting text as claimed in Claim 38 wherein the step of operating element keys to form codes includes:

serially operating element keys to provide a series of text elements, the element key corresponding to the last text element of the series being operated substantially simultaneously with an auxiliary key to indicate that the series forms a stem code which corresponds to certain desired text, upon releasing the element key and auxiliary key the processing through the processor means inserting the certain desired text and displaying the text on the display means.

40. A method of inserting text as claimed in Claim 38 wherein the step of operating element keys to form codes includes:

serially operating element keys to provide a series of text elements, the series forming a stem code corresponding to desired text;

operating the element key corresponding to the last text element of the series substantially simultaneously with an auxiliary key, releasing the element key after operation but maintaining operation of the auxiliary key; and

while maintaining operation of the auxiliary key, operating an element key corresponding to a desired inflection to be appended to the desired text in proper form by the processor means.

41. A method of inserting text as claimed in Claim 38 wherein the step of operating element keys includes:

operating a combination of an auxiliary key and an element key to form a processor signal code for selecting most frequently used text, different element keys providing different very frequently used text.

42. A method of inserting text as claimed in Claim 38 wherein the element keys include punctuation keys for providing different punctuation to text, and

the step of operating element keys includes operating the combination of an auxiliary key, a punctuation key corresponding to desired punctuation, and an element key providing a last text element in a series of text elements forming a stem code for certain desired text, said operation forming a processor code for punctuating the certain desired text with the desired punctuation.

43. A method of inserting text as claimed in Claim 38 wherein the step of operating element keys includes:

chording at least two adjacent element keys in a same row to form a processor code for providing preassigned text, different adjacent element keys corresponding to different preassigned text.

44. A method of inserting text as claimed in Claim 38 wherein the step of operating element keys includes:

operating at least one element key of one side of the keyboard while substantially simultaneously operating adjacent element keys in a same row as each other on an opposite side of the keyboard to form a processor code for providing different preassigned text as a function of row in which the adjacent element keys are operated.

45. A method of inserting text as claimed in Claim 44 wherein the step of operating at least one element key further comprises the step of operating an auxiliary key in combination with the element key and chording to provide other preassigned text as a function of row in which the adjacent element keys are chorded.

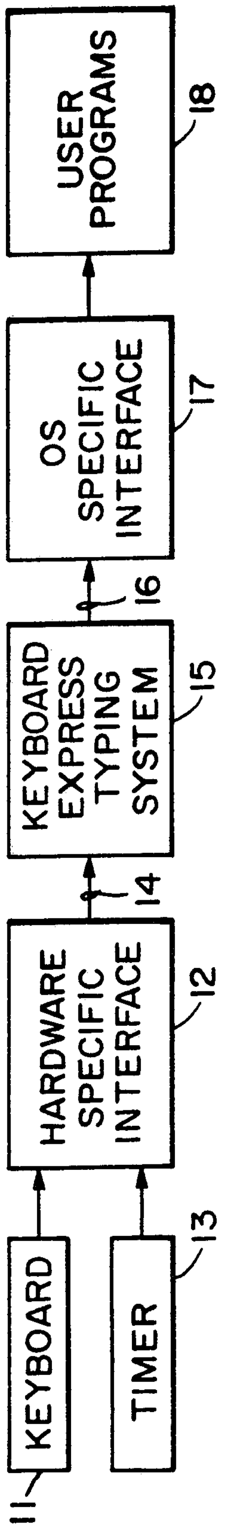


Fig. 1

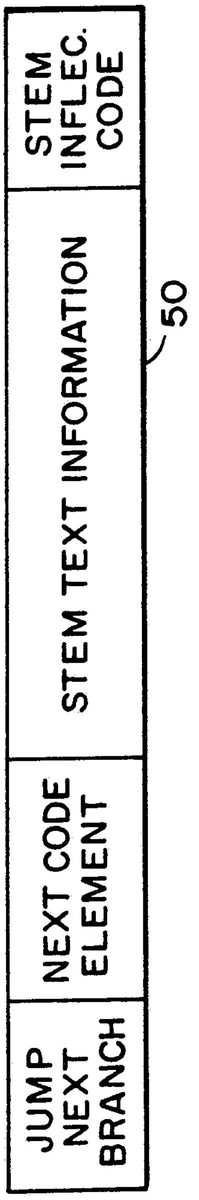


Fig. 4

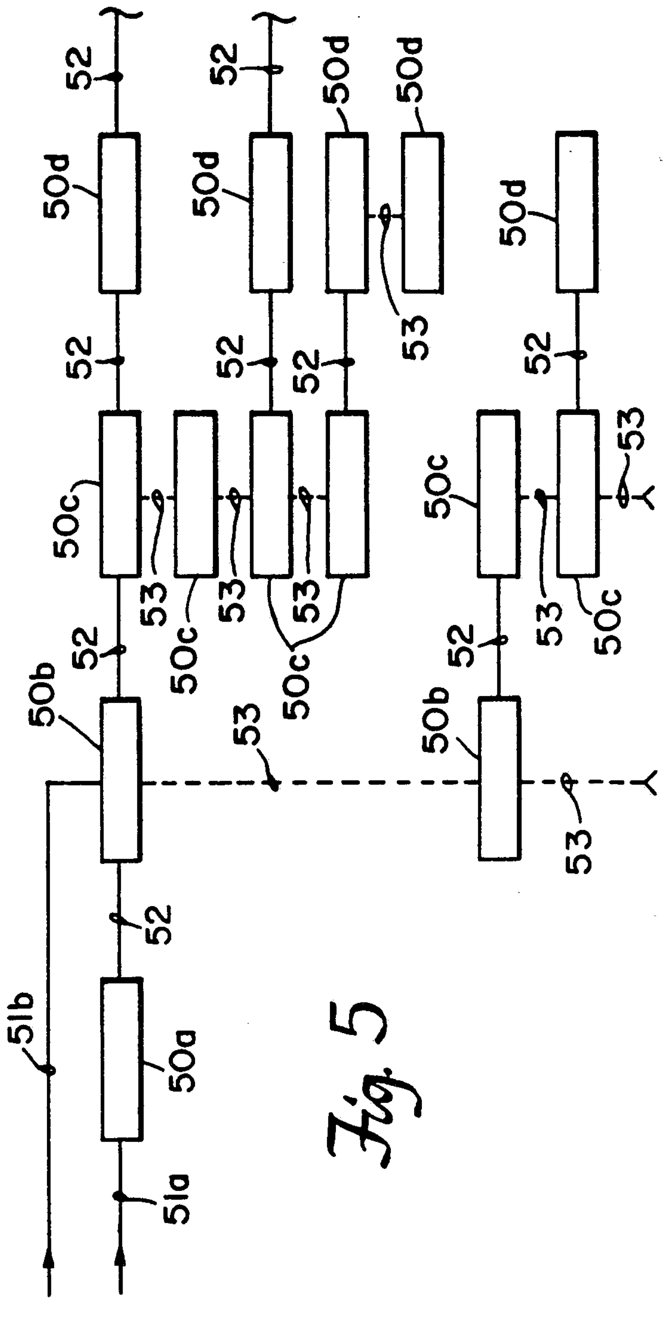


Fig. 5

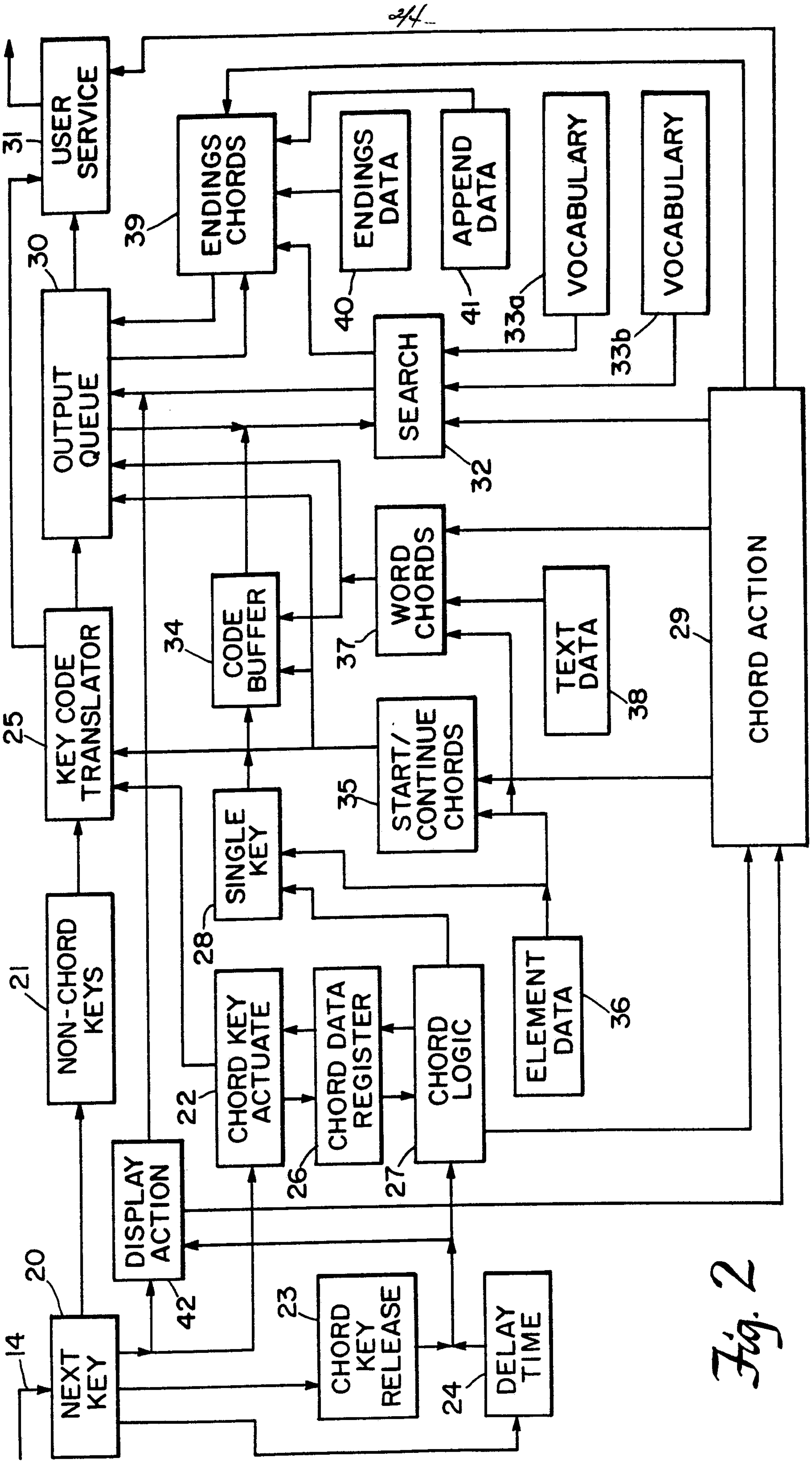

14
NEXT KEY
20
NON-CHORD KEYS
21
KEY CODE TRANSLATOR
25
OUTPUT QUEUE
30
USER SERVICE
31
DISPLAY ACTION
42
CHORD KEY ACTUATE
22
SINGLE KEY
28
CODE BUFFER
34
ENDINGS CHORDS
39
CHORD KEY RELEASE
23
CHORD DATA REGISTER
26
START/ CONTINUE CHORDS
35
WORD CHORDS
37
SEARCH
32
ENDINGS DATA
40
APPEND DATA
41
DELAY TIME
24
CHORD LOGIC
27
ELEMENT DATA
36
TEXT DATA
38
VOCABULARY
33a
VOCABULARY
33b
CHORD ACTION
29

Fig. 2

```
static char *se_es_plurals[] = ("efi/iv", "fa/av", "fl/lv", "fr/rv", "hc/ch",
"hs/sh", "o/o", "s/s", "x/x", "z/z", " ");

static char *foreign_plurals[] = ("a/ae", "mu/a", "no/a", "is/es", "su/i", " ");

static char *se_ion_ive[] = ("br/rpt", "dd/ddit", "dne/ent", "dn/ndat",
"di/idat", "d/s", "eb/pt", "ecu/uct", "ede/ess", "edi/is", "edu/us", "ed/pt",
"eko/ocat", "em/mpt", "eni/inat", "eno/onat", "era/arat", "eriu/uisit", "eri/irat",
"eru/urat", "ero/orat", "eso/osit", "esr/rsat", "ete/etit", "etuc/cut", "etut/tut",
"etu/utat", "eu/uat", "evie/ept", "evir/rivat", "evl/lut", "evr/rvat", "ez/zat",
"hsi/it", "lae/elat", "leb/bell", "le/uls", "ll/llat", "mr/rmat", "ngi/ignat",
"nia/enat", "red/derat", "ret/trat", "te/etat", "tib/bit", "tid/dit", "tim/miss",
"tio/oitat", "tnev/vent", "tn/ntat", "tr/rtat", "tsef/festat", "xa/axat", "yf/ficat",
"ylp/plicat", "yp/pat", " ");

static char *se_ate[] = ("a/", "reb/br", "reg/gr", "ylp/plic", " ");

static char *se_ce_cy[] = ("et/", "t/", " ");

static char *se_ly[] = ("y/i", "eu/u", "el/", "ll/l", " ");

static char *se_ous[] = ("a/e", "dn/nti", "ec/ci", "ega/age", "el/ul", "f/v",
"lev/vel", "th/hte", "tr/rte", "yn/ni", "yr/ri", "yt/te", " ");
```

Fig. 3 SHEET I of 2

```
static char *se_al[] = ("ecn/nti", "ec/ci", "el/ular", "esn/nsu", "eta/",
"hc/chi", "ne/in", "reg/geri", "ret/tr", "rot/tori", "tc/ctu", "te/etu",
"ti/itu", "tnedis/sidenti", "tp/ptu", "x/xu", " ");

static char *se_ic[] = ("a/at", "el/olic", "esp/pt", "esy/yt", "eta/act",
"me/emat", "sis/t", "xa/act", "yg/get", "yhc/ch", "yhp/ph", "yh/het", "yr/r",
"yx/ct", " ");

static char *se_ment[] = ("egd/dg", "eu/u", "mr/rma", "nru/urna",
"rep/pera", "tse/esta", " ");

static char *se_ant_ent[] = ("dne/ende", "di/ida", "dr/rda", "ecs/sce",
"ediu/uida", "edi/ide", "eg/ge", "el/la", "ero/ora", "eru/ura", "eu/ua",
"evr/rva", "fe/efia", "f/va", "mr/rma", "nia/enfa", "rae/eara", "reb/bra",
"red/dra", "ref/fere", "reh/hera", "rett/ttera", "ret/tra", "revi/ivera",
"rev/vere", "ru/urre", "ti/ita", "tp/pta", "tr/rta", "ts/ste", "wo/owa",
"yfi/ifica", "yl/lia", "ylp/plica", " ");

static char *se_ble_bly[] = ("cu/uci", "ed/si", "ecr/rci", "eca/acea",
"esno/onsi", "esr/rsi", "ss/ssi", "tp/pti", "ts/sti", "xa/axa", "xe/exi",
"yre/era", "yro/ora", " ");

static char *se_ity[] = ("el/il", " ");
```

Fig. 3 SHEET 2 of 2

INTERNATIONAL SEARCH REPORT

International Application No PCT/US 89/05745

I. CLASSIFICATION OF SUBJECT MATTER (if several classification symbols apply, indicate all) [6]

According to International Patent Classification (IPC) or to both National Classification and IPC

IPC^5: G 06 F 3/023, B 41 J 5/10

II. FIELDS SEARCHED

Minimum Documentation Searched [7]

Classification System	Classification Symbols
IPC^5	G 06 F 3, B 41 J 5

Documentation Searched other than Minimum Documentation to the Extent that such Documents are Included in the Fields Searched [8]

III. DOCUMENTS CONSIDERED TO BE RELEVANT [9]

Category [*]	Citation of Document, [11] with indication, where appropriate, of the relevant passages [12]	Relevant to Claim No. [13]
X	EP, A, 0151009 (QUIXATE CORP.) 7 August 1985 see figures 1-4; page 3, line 16 - page 4, line 8; page 5, lines 9-26; page 6, lines 12-20; page 8, lines 1-28; page 9, line 30 - page 10, line 22; page 12, lines 18-25	1-11
Y		12,15,17-19, 25,26,30,32, 38
A		16
	--	
Y	US, A, 4765764 (LEFLER) 23 August 1988 see the whole document	12,15,17-19, 25,26,30-32, 38
A		1,3,9,10,13, 26-29,34,36, 37,44
	--	
	./.	

* Special categories of cited documents: [10]

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

IV. CERTIFICATION

Date of the Actual Completion of the International Search 4th May 1990	Date of Mailing of this International Search Report 08.06.90
International Searching Authority EUROPEAN PATENT OFFICE	Signature of Authorized Officer Mme N. KUIPER

III. DOCUMENTS CONSIDERED TO BE RELEVANT (CONTINUED FROM THE SECOND SHEET)

Category *	Citation of Document, with indication, where appropriate, of the relevant passages	Relevant to Claim No
A	US, A, 4791408 (HEUSINKVELD) 13 December 1988 see figures 1-4,8(a-c), 13; column 1, lines 54-63; column 2, lines 36-43; column 4, line 61 - column 6, line 3; column 7, line 65 - column 8, line 2	1,9,10,18, 20,25,27,28, 31,38

ANNEX TO THE INTERNATIONAL SEARCH REPORT ON INTERNATIONAL PATENT APPLICATION NO. US 8905745

SA 33250

This annex lists the patent family members relating to the patent documents cited in the above-mentioned international search report. The members are as contained in the European Patent Office EDP file on 31/05/90
The European Patent Office is in no way liable for these particulars which are merely given for the purpose of information.

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
EP-A- 0151009	07-08-85	US-A- 4638306	20-01-87
		CA-A- 1220871	21-04-87
		JP-A- 60179817	13-09-85
		US-A- 4694280	15-09-87
US-A- 4765764	23-08-88	None	
US-A- 4791408	13-12-88	None	

For more details about this annex : see Official Journal of the European Patent Office, No. 12/82