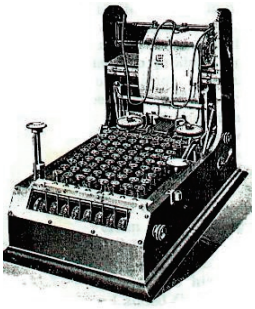
LANGUAGE AND NUMERICAL STRUCTURES

By James Redin

"Numbers are free creations of the human mind that serve as a medium for the easier and clearer understanding of the diversity of thought."

Julius Wilhelm Richard, German mathematician (1831-1916)

Sundstrand's ten-key pad (1914)



The Comptograph (1889)

Human beings have been entering numbers in computing machines through 10 digits since Dorr Eugene Felt invented the first successful keyboard operated adding machine, the Comptometer, back in 1885. The original mechanical keyboards had a matrix of 10 rows where each row corresponded to a digit, and each column corresponded to the position of that digit in the decimal number. A significant improvement occurred 90 years ago in 1914, when Oscar J. Sundstrand of Rockford, Illinois, introduced the mechanical ten-key pad based on a 3x3 matrix of digits plus a zero key located under the matrix.

TI-1795 SV (2004)

In the late 1960's, Ted Hoff, Stan Mazor and Federico Faggin, from Intel co-invented the microprocessor, and the world was never the same. Electronic calculators, micro-computers and every kind of electronic device have reshaped the world in ways that not even the most talented visionaries of the science were able to predict. Yet, nothing has changed in the way we enter numbers, and Sunstrand's ten-key pad design is still used today.

Numbers have always been entered as decimal numbers represented by a sequence of digits. However, our mind does not conceive a number as a sequence of digits; instead it conceives a number as an object composed of small quantities (such as thirteen or twenty) supported by names given to some powers of 10 (such as hundred, thousand, or million) that work as numerical structures. For example, we are living in the year "two thousand four," not in the year "two zero zero four."

The object of this document is to present an analysis of the mathematical rules that define the relationship between the verbal expression of a number and its decimal representation, and show an alternative way to represent numbers by using combinations of digits and decimal structure symbols, named verbal numerals. Finally, this document will show the advantages of verbal numerals and how they can be used to enhance the ergonomics of the number-input operations applied to electronic devices such as calculators, computers, and digitizer devices.

Numbers and Language Rules

When a number is expressed orally, several rules, which are language dependent, must be applied in order to determine the proper way to express the number.

The following analysis will define the common rules that determine the verbal expression of numbers in different languages, and find an alternative way to represent this decimal number that is different from its conventional decimal representation.

Small Numbers

In most languages, special names and/or special naming conventions are applied to numbers smaller than 100. For the purpose of this analysis these numbers will be referred to as *"small numbers."* In general, the smaller the number, the more specific is its name. In English, for example, numbers ranging from 0 to 12 have single names which do not follow any rule at all; each name is unique and shows no relationship with the others. Numbers ranging from 13 to 19 also have single names, but this time the name is formed by combining a root taken from the names assigned to numbers 3 to 9 with a suffix "teen." A similar approach is used to name the remaining multiples of ten, 20 to 90, by using the suffix "ty." Numbers starting with 21 up to 99, not included in the previous set, have a composite name made up from the name of the immediate lower multiple of 10 plus the unique name assigned to the number that corresponds to the remaining number of units; as an example, the number 37 is expressed as "Thirty-seven."

In Spanish a similar scheme is applied to numbers from 0 to 15, and multiples of 10 from 20 to 90, while every other number in the range has a composite name constructed as described above for English numbers larger than 20. As an example, the number 17 is expressed as "Diecisiete," which is a concatenation of "Diez (ten) y siete (seven)."

It is interesting to notice the way some small numbers are constructed in French. For example, the numbers 70 and 80, instead of being assigned single names as they are in other languages, are expressed with the composite names "Soixant (sixty) Dix (ten)" and "Quatre (four) Vingts (twenty)", which translated literally into English would mean "Sixty Ten" and "Four Twenties."

In Japanese , the number 10, is named "juu" and the names of the multiples of ten from 20 to 90, instead of having special names as in the Western languages, are a combination of the initial digit and the word "juu": "ni (two) juu (ten)" for twenty, "san (three) juu (ten)" for thirty, "yon (four) juu (ten)" for forty, and so on. Notice that twenty-four is named "ni (two) juu (ten) yon (four)." In several Asian languages, the small numbers are expressed in a more consistent way than in Western languages.

It follows from the discussion above, that except for some Asian languages, small numbers have no general naming conventions, and the way they are expressed greatly depends on the language applied.

Numerical structures

In every language, special non-composite names have been assigned to certain powers of ten that can be used for structuring or building up the names of larger numbers. For the purpose of this analysis, these powers of ten will be named "numerical structures." Notice that not all powers of ten can be considered numerical structures because their name, as in the case of "hundred thousand," has been derived from the names of other powers of ten that qualify as numerical structures.

In Western languages, the most common numerical structures are 100, 1,000 and 1,000,000. For example, in English, these numerical structures are named "Hundred," "Thousand" and "Million;" in Spanish "Cien," "Mil," and "Millón;" and "Cent," "Mille," and "Million" in French. These are the only numerical structures that remain consistent across major Western languages.

Larger powers of ten have also been assigned single names, but they do not always have consistent meanings. The most typical case is the numerical structure "Billion" which in the American system of numeration (originally invented by the French and also used in Canada) means one thousand millions (1,000,000,000), while in the British system of numeration (used in most Germanic and Romance languages) it means one million millions (1,000,000,000,000). By the same token, the numerical structure "Trillion" in United States represents a

unit followed by twelve zeroes, while in England it represents a unit followed by eighteen zeroes. In general, the names "Billion," "Trillion", "Quadrillion," "Quintillion," "Sextillion," "Septillion" and "Octillion" have been defined, both in the American system and in the British system, but have different meanings. In the American system each denomination is a thousand times the preceding, while in the British system each denomination is a million times the preceding. There are also names assigned for larger structures, but the rule is still the same. The largest numerical structure assigned a name in these systems is the "Centillon" that represents 10^{303} and 10^{600} in the American system and in the British system, respectively.

According to the Japanese JIS Standards, the following powers of ten have been defined as number components and therefore qualify as numerical structures: "juu" for 10, "hyaku" for 100, "sen" for 1,000, "man" for 10,000, "oku" for 10^8, "chou" for 10^{12}, "kei" for 10^{16}, "gai" for 10^{20}, "jo" for 10^{24}, "jou" for 10^{28}, "kou" for 10^{32} "kan" for 10^{36}, "sei" for 10^{40}, "sai" for 10^{44} and "goku" for 10^{48}. Equivalent names are used in other Asian languages such as Chinese and Korean for some of these powers of ten. Notice that in this numeration system, for numerical structures larger than 10,000, each denomination is ten thousand times the preceding.
Table 1 shows the different names assigned in several languages to the main numerical structures.

We can see that major languages have defined a set of single names for some powers of ten (numerical structures), a set of single names for the ten digits, and in some cases, a set of single names for some small numbers other than digits. The remaining numbers are a combination of these single names according to certain structural rules.

Number Name Parameters

For any number, the largest power of ten identified as a numerical structure, that is smaller than the number will be defined as the *"Order"* of the number, unless the number is smaller than the smallest numerical structure available, in which case the Order will be considered to be 1. The Order of the number is therefore the value 1 or a numerical structure that can be used to build up the number by using the following arithmetic expression:

number = int(number/Order) x Order + rem(number/Order)

where *int(number/Order)* represents the result of applying an integer division of the number by its Order, and *rem(number/Order*) represents the remainder of the same operation. For the purpose of this analysis, these values will be defined as the *"Factor"* of and the *"Module"* of the number, respectively. Therefore, above expression can be written as follows:

number = Factor x Order + Module

Notice that when the Order is 1 the Module is always zero.

Numerical Structure	English US	English UK	Spanish	French	Japanese
10^1	-	-	-	-	Juu
10^2	Hundred	Hundred	Cien	Cent	Hyaku
10^3	Thousand	Thousand	Mil	Mille	Sen
10^4	-	-	-	-	Man
10^6	Million	Million	Millon	Million	-
10^8	-	-	-	-	Oku
10^9	Billion	-	-	Milliard	-
10^{12}	Trillion	Billion	Billon	Billion	Chou
10^{15}	Quadrillion	Billion	-		-
10^{16}	-	-	-	-	Kei
10^{18}	Quintillion	Trillion	Trillion	Trillion	-
10^{20}	-	-	-	-	Gai

TABLE 1. Names of Numerical Structures in Several Languages

It will be shown later that the values of the Order, Factor and Module are closely related with the verbal expression of the number; for this reason, they will be called the *"number name parameters."*

Table 2- shows some illustration examples of above defined concepts when the numerical structures 100, 1,000 and 1,000,000 are used.

Table 3- shows the same examples when numerical structures 10, 100, 1,000 and 10,000 are used (as in the Japanese language.)

Verbal Numerical Expressions.

As shown in the examples of the previous section, in some cases the Factor and/or the Module themselves can be large numbers. In these cases, the original expression can be expanded recursively until all the Factors and Modules of the expression are small numbers as described by the following algorithm:

(1) Find the Order, Factor and Module of the number.

(2) If the Factor is a large number, apply recursively steps **(1)** to **(5)** to obtain the Factor expression and then enclose the factor expression within parentheses, otherwise use the Factor as the Factor expression.

(3) If the Module is a large number, apply recursively steps **(1)** to **(5)** to obtain the Module expression, otherwise use the Module as the Module expression.

(4) If the Order is greater than 1, append *"Factor expression x Order"* to the arithmetic expression that represents the number; otherwise, use the Factor as such expression.

(5) If the module is greater than 0, append *"+ Module expression"* to the arithmetic expression that represents the number.

An example, the application of above algorithm to number 457,128 by using numerical structures 100 and 1,000 yields the following arithmetic expressions:

Expand the number:
457 x 1,000 + 128

Expand the Factor:
(4 x 100 + 57) x 1,000 + 128

Expand the Term:
(4 x 100 + 57) x 1,000 + 1 x 100 + 28.

Now, it is interesting to realize that the expressions obtained by the application of the above procedure are consistent with the verbal expression of the number. In fact, notice that the name of the number can be easily obtained just by arranging the names of the numbers and numerical structures of the expression in the same order as they appear in the expression without paying attention to the arithmetic symbols used in the expression. For example, above expression can be used to

Number	Factor	Order	Module
100,000	100	1,000	0
350	3	100	50
99	99	1	0
0	0	1	0
2,457,128	2	1,000,000	457,128
457,128	457	1,000	128
352,457,128	352	1,000,000	457,128

TABLE 2. English Number Parameters

Number	Factor	Order	Module
100,000	10	1,0000	0
350	3	100	50
99	99	1	0
0	0	1	0
2,457,128	245	10,000	7,128
457,128	45	10,000	7,128
352,457,128	3	100,000,000	52,457,128

TABLE 3. Japanese Number Parameters

obtain the name of the number 457,128 by using exclusively the names of the small numbers and numerical structures as follows:

(4 x 100 + 57) x 1,000 + (1 x 100 + 28)
four hundred fifty-seven thousand one hundred twenty-eight.

The same procedure with minor adaptations can be used in languages other than English. For example, In Japanese, the expression becomes:

(4 x 10 + 5) x 10,000 +(7 x 1000 +(1 x 100 +(2 x 10 + 8)))
yon juu go man nana sen hyaku ni juu hachi.

Because of this correlation between the algorithm results and the verbal expression of a number, expressions obtained by applying the algorithm described above will be referred to as *"verbal numerical expressions,"* and the algorithm will be called the *"verbal expression algorithm."*

Notice that all the components of a verbal numerical expression must follow the format: *"Factor x Order + Module;"* if any component of the expression is permuted, the new expression, according with the commutative law of numbers, will still represent the same number, however it will no longer be consistent with its verbal notation. For example, the following expression:

1,000 x (4 x 100 + 57) + 1 x 100 + 28

still represents the number from the example, however, it is no longer consistent with its verbal notation. Notice that *"thousand four hundred fifty-seven one hundred twenty-eight"* is not the name of a valid number.

Properties of Verbal Numerical Expressions

By observing the nature of verbal numerical expressions, the following properties can be found:

- The Module is always smaller than the Order.
- Whenever the Order is smaller than the largest numerical structure available in the set of numerical structures used to construct a verbal numerical expression, the Factor is smaller than the Order.
- Except for the representation of zero, either the Factor or the Module component is always greater than zero.
- Order components are always greater than 1.

The above properties may be used to determine if a given expression is a valid verbal numerical expression.

Verbal Numerical Expressions and Verbal Numerals

In the previous section it was shown that the name of a number is actually the representation of a verbal numerical expression. Another way to represent a verbal numerical expression is by assigning symbols to the numerical structures (i.e.: "H," "T" and "M" for Hundred, Thousand and Million, respectively) and combining them with the small numbers used in the verbal numerical expression in a mode similar to the way the name of the number is constructed orally.
As an illustration example, the application of the verbal expression algorithm to the number 35,178,971 will yield the following intermediate and final verbal numerical expressions:

35 x 1,000,000 + 178,971
35 x 1,000,000 + 178 x 1,000 + 971
35 x 1,000,000 + (1 x 100 + 78) x 1,000 + 971
35 x 1,000,000 + (1 x 100 + 78) x 1,000 + 9 x 100 + 71.

By replacing the numerical structures with the corresponding symbols and removing the parentheses and arithmetic operators, these expressions can also be expressed symbolically as follows:

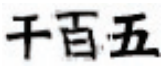
35M178971
35M178T971
35M1H78T971
35M1H78T9H71.

Since a verbal numerical expression represents a number, the symbolic representation of a verbal numerical expression will also represent a number. In general, we use the word numeral to mean the symbolic representation of a number; therefore the symbolic representation of a verbal numerical expression in the way described above is a numeral. For the purpose of this analysis, this symbolic representation of a number will be

called *"verbal numeral."*

Some languages, like Spanish, omit the pronunciation of the Factor when it is equal to 1. As an illustration example, the number *"One thousand one hundred"* (1T1H) is expressed in Spanish as *"Mil cien"* (TH). This is a convenient feature to be included in a verbal numeral because it reduces the number of components required by some verbal numerals, for example, the number illustrated above could also be represented as: 35MH78T971.

It is interesting to notice that in Japanese and Chinese, digits and numerical structures have single Kanji symbols, and a number is already written as a verbal numeral instead of a sequence of digits. For example the number 1,105 with verbal numeral TH5 is pronounced in Japanese as *sen-hyaku-go* (for thousand-hundred-five), and in Kanji characters is written as:



corresponding to "sen," "hyaku" and "go."

Conversion of Numbers into Verbal Numerals

Since a verbal numeral is the symbolic representation of a verbal numerical expression, the procedure used to obtain the verbal numeral of a number should be similar to the verbal expression algorithm.

The following is an algorithm that can be used to convert a number into a verbal numeral:

(1) Find the Order, Factor and Module of the number.

(2) If the Factor is a large number, apply recursively steps **(1)** to **(4)** to obtain the verbal numeral of the Factor, otherwise use the decimal representation of the Factor as the verbal numeral of the Factor.

(3) If the Module is a large number, apply recursively steps **(1)** to **(4)** to obtain the verbal numeral of the Module, otherwise use the decimal representation of the Module as its verbal numeral representation.

(4) Obtain the verbal numeral of the number by appending to the verbal numeral of the Factor the symbol of the Order and the verbal numeral of the Module, in that order.

Due to the similarity of steps **(2)** and **(3)**, the procedure can be simplified to use only one single call to the recursive procedure by subtracting the product *Factor X Order* from the number and then repeating the procedures until the result is zero. A flow-chart showing the simplified version of this algorithm is shown in Fig. 1.

There are many applications where it is desirable to build the full-word name of a number. Examples of these applications are programs and routines used to print the words for dollar amount in a check. A procedure can easily be developed where the number is converted into a verbal numeral by using the algorithm described in this section, and then the verbal numeral can be scanned from left to right replacing the contiguous groups of digits with the name of the small number that they represent, and replacing the numerical structure symbols with their names.

Conversion of Verbal Numerals into Numbers

The verbal expression algorithm can also be used to develop an algorithm to obtain the decimal value of a number that is represented by a verbal numeral. Basically, this procedure is similar to the verbal expression algorithm, except that this time the expression components are actually computed and added to the number, and the Order is extracted directly from the verbal numeral rather than computed from the number. This recursive algorithm may be summarized as follows:

(1) Find the Order symbol of the verbal numeral by locating the symbol of the largest numerical structure contained in the verbal numeral; if found, the Order is the absolute value of the Order symbol; otherwise the Order is 1.

(2) Get the value of the Factor by using the string segment of the verbal numeral located at the left side of the Order symbol; if no Order symbol is found assign the value 1 to the Order and use the value represented by the digits in the string segment as the value of the Factor. If no string segment is available, the value of

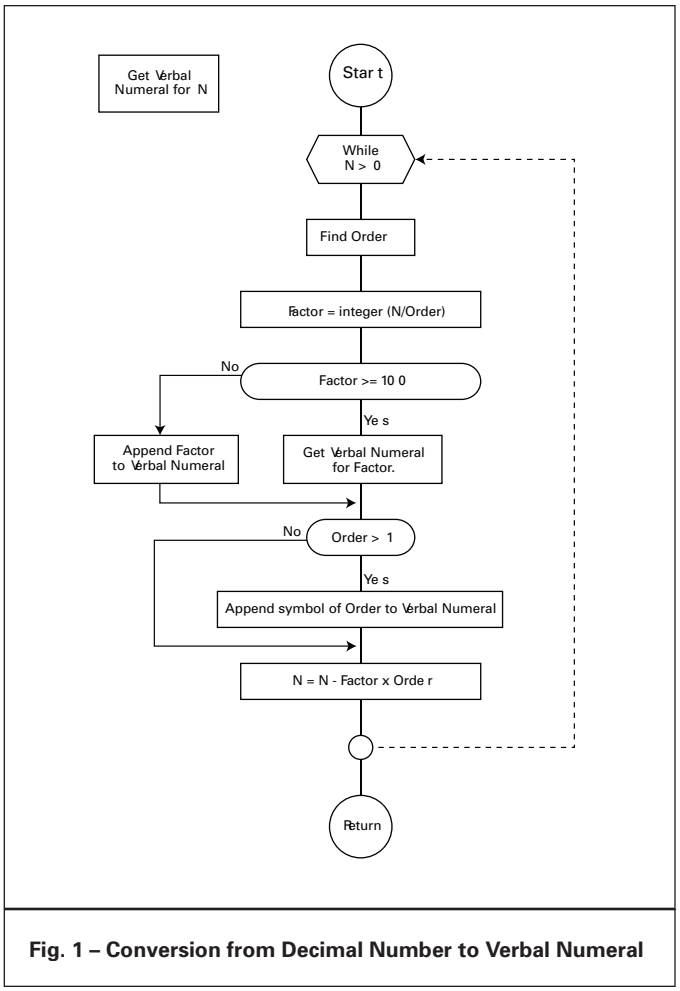


Fig. 1 – Conversion from Decimal Number to Verbal Numeral

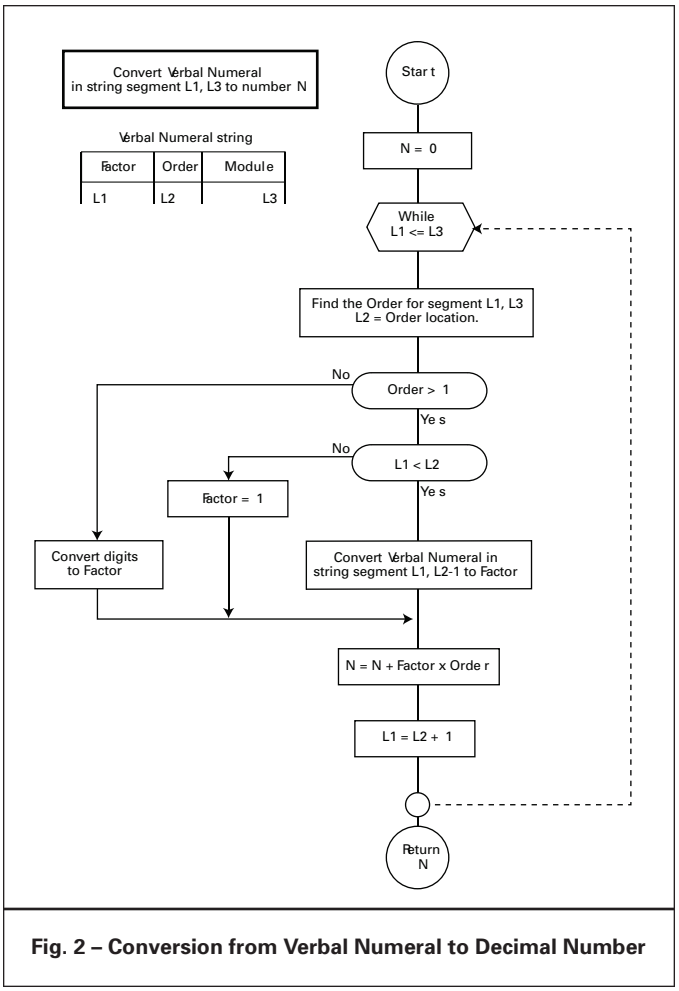


Fig. 2 – Conversion from Verbal Numeral to Decimal Number

the Factor is 1. If the string segment contains at least one numerical structure symbol, apply recursively steps **(1)** to **(4)** on this string segment to obtain the value of the Factor.

(3) Get the value of the Module by using the string segment of the verbal numeral located at the right side of the Order symbol; if no Order symbol is found, use the value represented by the digits in the string segment as the value of the Module; if no string segment is available, assign the number zero to the Module. If the string segment contains at least one numerical structure symbol, apply recursively steps **(1)** to **(4)** on the string segment to obtain the value of the Module.

(4) Determine the value of the number by applying the following expression: *Factor x Order + Module*.

Since the logic to convert the Module is similar to the one used to convert the Factor, it is possible to simplify the logic to use only one single call to the recursive procedure as shown in the flow chart of Fig. 2.

The algorithm shown in this section can be implemented in the logic of a numerical data-entry device such as a calculator, a computer keyboard or a touch sensitive device in order to accept numbers entered as verbal numerals. A similar procedure can also be adapted to enter numbers in voice and gesture recognition systems.

Support for Combined Numerical Structures

As mentioned before, in most Western languages the only common numerical structures are "hundred," "thousand" and "million." This can make it impractical to include keys for

larger numerical structures such as "billion" or "trillion" in a keyboard. In these situations, the algorithms described above can easily be modified to accept or recognize *"combined numerical structures."* For this to work, a combined numerical structure must be defined as a sequence of one or more consecutive numerical structures where the value of each numerical structure is equal to or larger than the value of its preceding numerical structure.
As examples of applying this concept, the following numerical structures may be defined by using the "hundred (H)," "thousand (T)" and "million (M)" numerical structures:

HH Japanese "man" (10,000)
HT Japanese "oku" (10^8)
TM American "billion" or French "milliard" (10^9)
MM British "Billion" or Japanese "chou" (10^12).

As another example, the British version of the number "two billion three hundred million" may be entered with the following verbal numeral: 2MM3HM.

The algorithm should identify these types of sequences within the verbal numeral and consider them as single numerical structures.

Validation Rules

As mentioned before, the oral expression of a number must follow certain rules, for example, the number "Four hundred and thirty seven hundred" does not exist (although the words would be intended to represent the number 43700.) By the same token not every combination of digits and numerical structure symbols should yield a valid verbal numeral. The following are the rules that can be used to validate a verbal numeral:

- The Factor or the Module segment of a verbal numeral cannot have a leading zero.

- The Factor of a verbal numeral cannot represent a number larger than the Order of the verbal numeral.

- The Module of a verbal numeral cannot represent a number equal or larger than the Order of the verbal numeral.

The following are examples on of invalid verbal numerals:

4H37H (Module 37H is larger than Order 100)
437H (Factor 437 is larger than Order 100)
3T025 (Module 025 has a leading zero)
3T1000 (Module 1000 is equal to Order 1000)
4TTHM (Module HM is larger than Order TT).

These rules can easily be implemented as part of the algorithm described in Fig. 1 in order to validate a verbal numeral entered by a user in a device designed to accept verbal numerals as a way to enter numeric quantities.

Advantages of Verbal Numerals

One obvious advantage of verbal numerals is its consistency with the way the human mind conceives and orally expresses the number. This allows for a more natural interface when entering numbers that can be used in many input devices from keyboards to voice recognition systems. Verbal numerals may also reduce the number of data entry errors as the operator would not have to translate an orally expressed number to its decimal sequence.

By using combined numerical structures, verbal numerals can be used to represent very large numbers, even if special symbols have not been assigned for very large numerical structures as in the following example:

245,000,000,072 245MM72

In many instances verbal numerals require fewer number of symbols than decimal numbers. Here are several examples:

3,000,005	3M5
350,000	3H50T or 350T
2,000,305	2M3H5 or 2M305
1,001,000	1M1T or MT
1,000,000,000	1TM or TM
1,000,000,100	1TM1H or TMH

A large number usually may be represented by several alternative verbal numerals; this provides flexibility not available with decimal numbers. For example, following

representations of the number 205,001,048 are equally valid:

2H5M1T48
2H5MT48
205MT48
205M2048

Another advantage of verbal numerals is its capability to grow gradually as the number is being pronounced orally. This property does not exist in the corresponding decimal numbers. For example, in the construction of the number *"Five Million Three Hundred Thousand Six"* the following intermediate numerals are involved:

Number Name	Decimal Number	Verbal Numeral
Five...	5	5
million...	5,000,000	5M
three...	5,000,003	5M3
hundred...	5,000,300	5M3H
thousand...	5,300,000	5M3HT
six	5,300,006	5M3HT6

Notice that while the decimal number changes substantially in each intermediate step of the number, the verbal numeral does not change except for the addition of the new component to the previous numeral.

Verbal numerals can easily be adapted as input means in electronic devices, with the additional advantage that they can coexist with the traditional number-entry input means.

Conclusions

The way we write decimal numbers is not the only way to represent numbers in decimal mode. Numbers can also be represented as verbal numerals that, in addition to having the flexibility to represent the same number in different alternative ways, are consistent with the way numbers are conceived and expressed verbally.

The technological limitations that caused decimal numbers to be entered in mechanical devices as a sequence of digits are a thing of the past. We are no longer forced to convert a number into this cold sequence of digits before entering it; the device can do that for us. Verbal numerals can now be used to increase the ergonomics of number entry procedures in electronic devices, without having to replace the usage of the traditional decimal numbers.

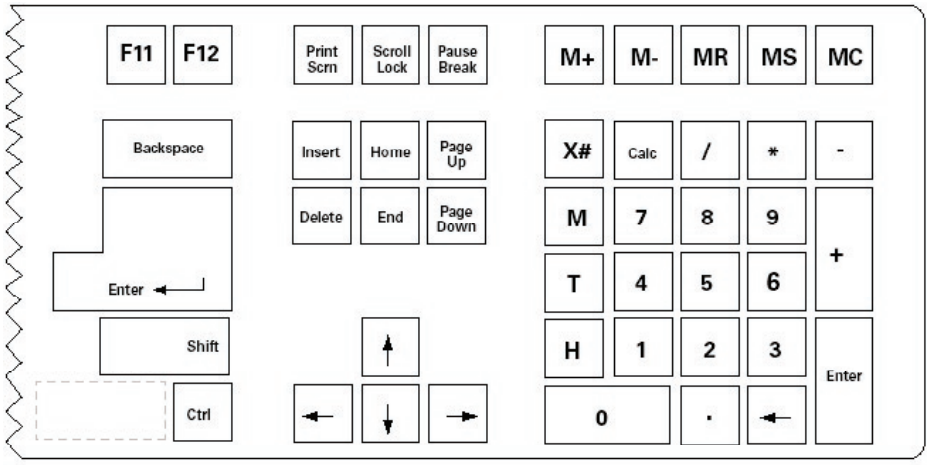


Fig. 3 – Possible implementation of verbal numerals on a keyboard

James Redin is a senior systems analyst at Keane, Inc. In his free time, he likes to write about the history of calculators, and invent new ways to enter numbers in electronic devices. He is also the holder of U.S. Patent 5,623,433 that describes the procedures to convert decimal numerals into verbal numerals and the other way around. Demonstration keyboards and application examples of the concepts provided in this article can be found at **http://www.xnumber.com** James Redin can be contacted at **jredin@xnumber.com**