ON FROFF: A TEXT PROCESSING SYSTEM FOR ENGLISH TEXTS AND FIGURES

Norihiro Abe,* Nobutaka Uemura,*
Masahiro Higashide,** Saburo Tsuji*

* Faculty of Engineering Science
Osaka University

** Nippon Electric Company

In order to meet the needs of a publication of papers in English, many systems to run off texts have been developed. In this paper, we report a system **FROFF** which can make a fair copy of not only texts but also graphs and tables indispensable to our papers. Its selection of fonts, specification of character size are dinamically changeable, and the typing location can be also changed in lateral or longitudinal directions. Each character has its own width and a line length is counted by the sum of each character. By using commands or rules which are defined to facilitate the construction of format expected or some mathematical expressions, elaborate and pretty documments can be successfully obtained.

1. Introduction

With a rapid advancement of scientific exchages, a high qualitative text processing system have been expected. Under these circumstances, many systems which run off texts have been implemented. In our laboratory, a typewiriter-based word processor with a fixed widths of characters (it is called ROFF) was designed. Its output is, however, inferior in its quality to the one printed by phototypesetter or manual operations.

The Flexible Run OFF(FROFF) system is an integrated system that is designed for setting not only ordinary texts but also figures or tables indispensable to our documents, and utilizes a dot-printer as its output device. Here we must make mention of the TROFF designed for setting running text including mathematics, but as this system drives an expensive phototypesetter on the UNIX operating system, member of the almost all standard research institutes can not adopt this system configuration when he comes to think of its cost performance. This FROFF system is easy to realize in a standard computer system with the basic I/O devices including a display, dot-printer. Recently, a laboratory provided such devices incrementally increasing, it can be known for sure to be true that the circumstances under which a system with equivalent equipments to FROFF is implemented has been prepared.

This program is written in mainly FLISP (a descendant of INTERLISP) and partly in Fortran, Assembler. A reason for selection of FLISP as a programming language is an easiness of programming, debugging, and extension of the system.

2. Hardware for FROFF

The FROFF system is implemented on a mini-computer system HP2108A (64Kbytes) with an auxiliary memory (256Kbytes) which is designed to be used as a free-list region of FLISP and an image processing memory (see Fig.1). When an image input/output is required, the auxiliary memory is partitioned into the two parts for list and image regions and a manipulation of image memory is performed by the simple FLISP commands.

The basic I/O devices necessary for a picture processing including keyboad, dot-printer, display, TV-camera, so on are required to input pictures and process them. If no image outputs is expected, only the keyboad and dot-printer are necessary.

3. Main Functions

1) A variety kinds of command provided by the system allows us to produce various kinds of type styles including line-drawing, picture output. An easy application of this system being realized by making use of rules installed into the system, to whom isn't satisfied with a specification of such standard rules, he can define rules corresponding to his own demands. Using rules, Subscripts are automatically printed in an appropriately size with no special intervention.

2) Dynanic change in a font or a size of characters are permitted and its computation of the widths of strings of characters is done automatically.

3) An Automatic conversion of the head of sentences into capital letters enables a text to be input from a computer terminal not equipped with small characters.

4) A Dynamic motions of a text to arbitary horizontal and vertial directions is allowed, being utilized in a rule to print mathematics.

5) A variety of kind of lines can be drawn, table formats and pictures stored in the disc file being printed and a input devices such as TV-camera, joystic make it easy to input those pictures.

We want to add explanations for each of them.

1) There are about 60 commands to fullfill forms processing requirements, by taking maximum advantage of them we are able to specify any text formats and to get an elaborate output document. It is, however, not easy for the users inexperienced to this system to master functions of these commands and to handle them effectively. In regard to this problem, an application of rules is taken into the system as one of its primal functions. By the way, note that there is another merit brought with the rule definition.

Using FROFF, user can utilize any set of rules provided with the system by declearing names of the rules he wants to use. It is needless to say that there are many conventional styles in documents and that FROFF should provide all of facillities that one needs for doing them, but it is also true that there exists various kind of requirements peculiar to him. Those who are not satisfied with such system-installed functions of rules can define suitable rules to their own demands and also these rules are easy to be taken into the system as its library rules. Here note that a primal reason for using rules in specification of forms is that the system should be easy to be extended to have more facilities, since requirements in format will vary ocassionally and it is necessary that the modification be easy to implement at any time. The method taken

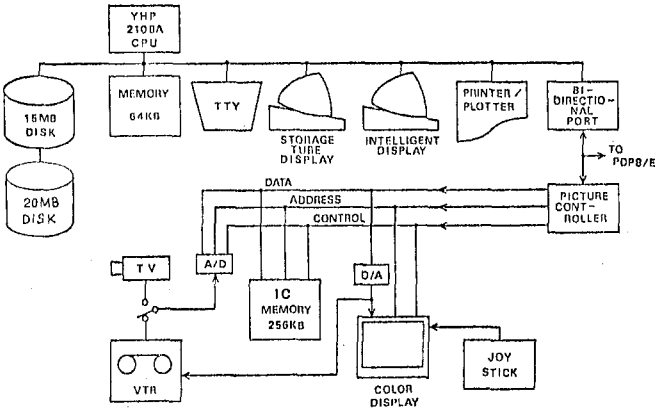

Fig.1. Configuration of FROFF system.

on this system simplifies this improvement of its functions.

2) As each character is given its own width, a well-balanced arrangement is obtained as its consequence. For example, sample pattern for character I and W are shown in Fig.2, whose width are 13x12, 29x32 respectively, where the size is shown in a "dot". As has already been understood, this paper is printed by FROFF, a various kind of fonts including roman, **Bold** etc. are dynamically available and their typeset in *italic* is also done easily. At present fourteen kinds of fonts are prepared,and 60 fonts will be in use near future.

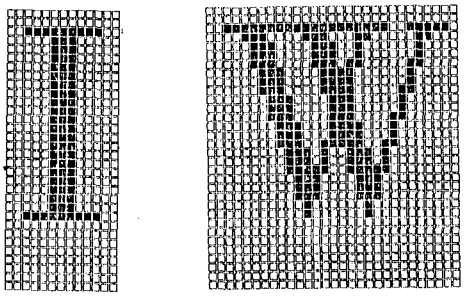

Fig.2. Sample patterns I and W.

Of cource, commands for size specification of characters enable a variety sizes of output to be accomplised. To my regret, characters enlarged too much are not necessarily good with respect to its grace, but there is no inconvenience to our usual purposes (see Fig.3 for example).

3) An input text can be processed at a terminal not equipped with small characters such as the conventional TTY and graphic displays. The decision whether a heading of words must be printed in capital or not is automatically decided by the system using common-sense rules that a head character of the sentence, which usually follows to ".", should be printed in capital and that any words following to "i.e.", "e.g." etc. be in small. As a result of these settlement being assumed, there is a disadvantage in the case where one wants to type all characters in capital like "FLISP SYSTEM".

Fig. 3. An example of available character sizes.

To cover up such a weak point which are occasionally encounterd when there are some mixtures of big and small characteres in a string, sevaral rules to change fonts or sizes of characters or to concatenate them are prepared. (Another way of handling this problem will be stated in 9).

4) We are able to avoid writing a sophisticated procedure such as one needed to produce a mathematical expression enclosed by a box (shown in Fig.4) because some commands permits an output location to be shifted relatively to any directions in respect to the previous output position.

5) It can be thought of as true thing that there is no paper without any figures or tables. Especially for a laboratory investigating picture processings and their related fields, commands for a picture output or rules for a tabulation make it easy to insert computed results into the document. In this case, it is, of cause, possible to add simple operations such as, inversion, partial extraction of pictures to them. Fig. 5(a) and (b) shows examples of picture outputs. Restricted to holizontal or vertical directions, various types of lines can be readily drawn (see Fig.6).

4. A Coustruction of FROFF.

Previous to the detailed explanation of rules and commands, an overall structure of FROFF must be given to guide an easy comprehension of it.

At first, input text is given using the text editor. As shown in Fig.7, when texts are fed into the system, interpretation of rules is conducted in the following way. It scanning the sequence of texts for symbols defined as user specified or system library rules, they are replaced with their definitions when found. Upon completion of this stage, evaluation of commands is executed, which is done in the way where a LISP function corresponding to each command is invoked. A result obtained from this step, as described in a flowchart in Fig.7, is converted to the binary data in order to drive the versatec dot-printer by several programs written in Fortran, Assembler.

5. Commands and their examples.

A source text of FROFF comprises text lines and command lines, which begines with a period, and blanks are regared as delimiters. A string of characters punctuated with a delimiter is called a

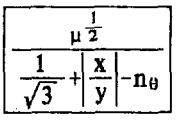

Fig. 4. A mathematical expression enclosed by a box.

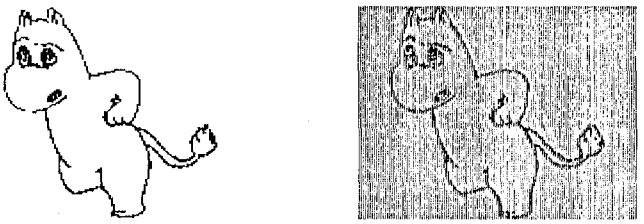

Fig. 5-(a), 5-(b). Picutre output from a disc file.

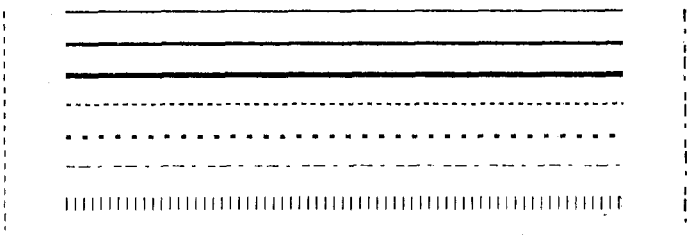

Fig. 6. An output of lines .

"unit", an allocation of texts including justifications and a page segmentation is accompleshed in terms of this "unit". Any combinations of variables and values with units such as "MM", "CM" involving the arithmetical rules are permitted as arguments for these commands (if a unit is abbreviated, a unit is regarded as a "dot", where 1 cm equals to 78.8177 dots). These commands are partitioned into two classes, those for formats and for controling forms. The former is used to specify the type form like an indentation, spacing, font change, centering etc. and the latter serves for dynamic control of such type styles. The general command form is the following

command-name arg-1 arg-2 --- arg-n

where arguments can be default according to the specification of the command. Some examples of the format commands are listed in Table 1.

utilization and Definition of rules

As mentioned earlier, there is no need for being troubled with detailed restrictions on commands if a

people not skilled with them takes maximum advantages of library rules. To apply those rules, the following declaration is required.

.. rule rule-1 rule-2 ---- rule-n (*)

where rule-i must be the system rule. To whom wants to define rules for his own purpose, the command ".." allows him to formulate them in the following way.

..rule-name arg-1 arg-2 ---- arg-n
body of rule
.

where the body of rule is cosisting of a sequence of commands or texts or a declaration of other rules to be used in this rule, that is, if a new rule is defined using rules already defined in the rest of texts, their name must be declared as in (*).

.. rule-name arg-1 arg-2 ---- arg-n
a sequence of commands
.. rule rule-1 rule-2 ---- rule-n
a sequence of commands
.

System parameters and control commands

To fulfill various requirements for format, the needs for chaging a width of page or variety of control are expected. For doing so, in this system the application of rules are provided to users. Yet, proper side-effects by each rule must be resulted from the rule evaluation and the old status of parameters must be recovered as soon as its execution terminates. For example, assume now texts are printed in Roman, and some strings are expected to be in BOLD, where of course this is done by a rule R, and then the specification of font must be tailored to suit a condition that a font specification is free from the old one. For this purpose, the following rules are required.

```
.. R: R has no argument
.ST %A: save a current font
.A 5: set a new font
.IT 4: write in Italic
.: end

..-R: -R kills the effect of R
.RS %A: recover the old font
.: end
```

where %A is one of system variables, the value of which is altered by subjecting to an effect of ".A". To supply flexibilities for controlling formats, almost all parameters used in commands are opened to users. Those variables and their relation to page style is demonstrated in Fig.8. Several samples of control commands and their meaning are listed in Table 2.

6. Figure outputs

Text processing system should provide feasibilities to insert several sizes of figures. Allocations of figures

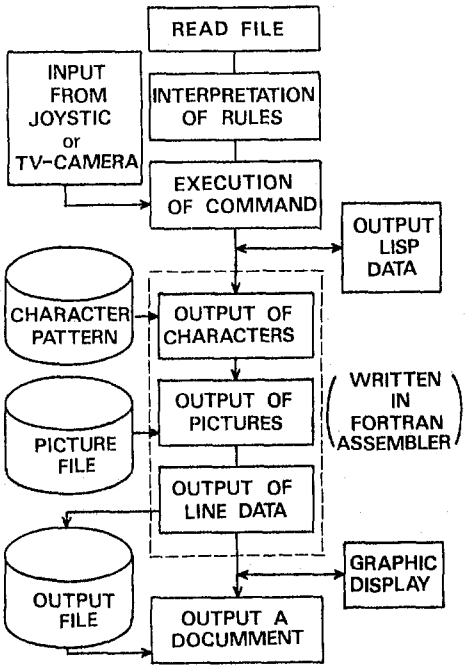

Fig. 7. A system flowchart of FROFF.

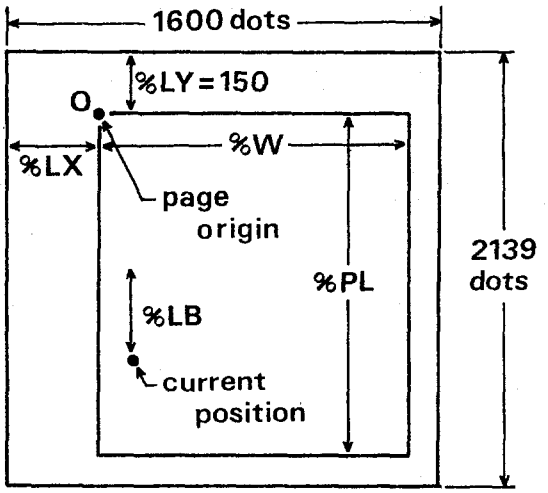

Fig.8. A page format and its relation to parameters.

are known as difficult because the program neccessitates a modification to texts allocation if there doesn't remain enough space for displaying the picture in a current page, and this procedure must be suspended and the remaining space must be filled with the subsequent texts. And this interrupted commands should be resumed when a new page is encountered (This is not restricted to figures, the same is true for tables or mathematics). Figure outputs are conducted with the following commands.

.pic name cr fnum x_0 y_0 m_x m_y s_x s_y e_x e_y type
an explanation statement for this figure

where an object of output is a fnum-th picture in a picture file, whose name is name, stored in a cartridge cr and the real output is enlarged by m_x, m_y after an extraction from the original picture. Parameters such as s_x, e_y indicate this extraction domain. It's needless to say that they are printed in a top location in the next page when sufficient lines can't be found.

To be out of order, we show briefly a mechanism and meaning of this command. When this command is encountered in the course of scanning of texts, program protects its current system parameters described in Fig.8 and by checking arguments of this command it computes a necessary longitudinal length, allocating the explanatory texts by assuming enough lines are remained in the current page. In effect, even if a true end of page was found, this is not suspended. In case all texts are successfully settled, the stored parameters are neglected and the subsequent texts are processed in an ordinary way. Otherwise, on completion of execution of ".pic" command, its result including the retained parameters are removed from the consequence so far acheived, and preserved for a next page. Space not occupied in so doing are used to allocate the rest of texts, and when an end of page is detected, system parameters at that time are stored in a reserved location. On a new page encountered, the above mentioned result is reverted, as if ".pic" command was written immediately after a page changing command, after adding a slight modification to its vertical coordinate values. Then, the stored system parameters are reset to as their old values and the system proceeds with successive procedures.

7. Line drawing and Tabulations

Any types of lines expected should be drawn in relation to the current output location or to the pre-specified coordinates for ready achievment of tabulations and others. For this purpose next commands are prepared.

.line? s_x s_y e_x e_y size n_1 n_2 --- n_i
.line s_x s_y e_x e_y size n_1 n_2 --- n_i

Their relations between arguments and allocations are given in Fig.9, where an origine of ".line?" refers to the previous vertical location and that of ".line" to the location given explicitly by user (regarding their horizontal one, both refer to %LX shown in Fig.8). Especially, the former command brings easy achievement of tabulations or mathematics, like a horizontal line in a fraction or a root symbol. For tabulations, all specifications needed for achieving various types of forms desired are not necessarily implemented in the current version because quite many styles on tabulation being required, it is impossible to supply enough rules to fulfill them.

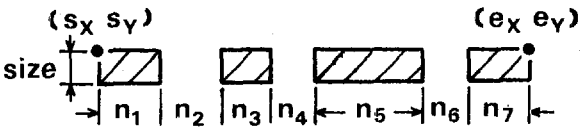

Fig.9. Parameters of a line command.

But from experiences, it seems apparant that tables shown in Table 1,2 are most familiar to us in comparison to any other types, and it is enough for practical uses if rules corresponding to them are prepared. Still yet, this leads to easy achievement of tables and easy learning of such rules.

Let explain a first rule to make a list with equal line widths in the following.

*table X Y Z A B M N

sets a window on a current page as shown in Fig.10, and a rule, *tb text, causes its contents to be enterd in a specific order. The rule, *tbn F, completes the current row, switching from a current row to the next one with a vertical line-drawing. When a completion of table required, rule *endtb cancells its current settlement of window and the old status for formattings is reverted.

On the other hand, it's difficult to give a convenient method for tables with variable line widths as shown in Table 2, only a means are arranged with which lines are drawn to the entries of table after allocation of them in a similar way mentioned above. However to my regret, results created by manual operations seems superior to that generated by rule with respect to its easiness of a construction.

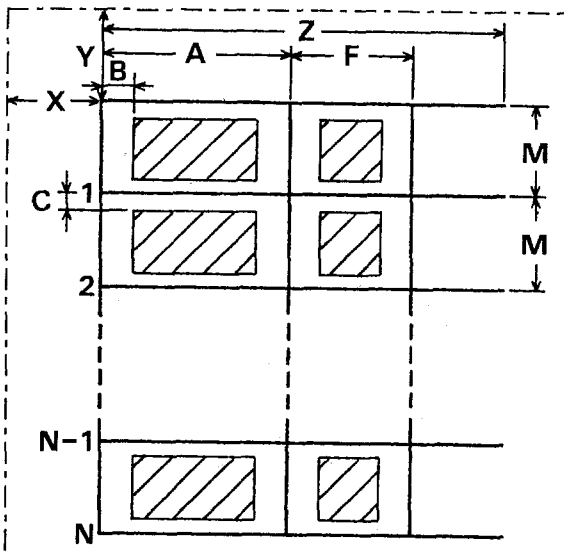

Fig.10. A rule for a typical tabulation.

8. Mathematics

As already stated in [3], this is known as difficult to set in type than any other kind of texts. In TROFF, a language was designed to translate a convenient expression for mathematics into a form readily evaluated with TROFF. This methodology quite suits to a easy extension of system facility and excels in the point that any person not skilled with details of the system are able to use it. But its easiness in extension depends most on existence of a good compiler-compiler which permits the system to be added new facilities. On the contrary, for our system not assumed such tools, another way of simplifying the specification of mathematics is requisite.

As a typical user of this system can be considered to have common sense on technology, mathematical expressions must be described in a Polish notation as descrived below. Of course, standard things in mathematics, like subscripts, superscripts and their locations, are automatically adjusted in an appropriate size. But in this system, there is a weakness in that special commands or rules must be written to concatenate subexpressions as in *fra a b+ *omega, which generates

$$\frac{a}{b+\omega}$$

The reason why such inconvenience occurs should be elucidated. Unlike the TROFF, this system treats mathematics in a similar manner to ordinal texts, which means that any expressions delimitted with a blank is regarded as a "unit" and this fact causes the expressions to be broken up. In this regard, TROFF is superior to our system. But except for this defect, almost all of the facilities that one needs for doing mathematics are defined using rules. As has been discussed, new facilities needed for mathematics are easily added to the system by using rules. Now let us show some examples in the following. First of all, subexpressions are defined and by using defined symbols main expressions are printed.

```
.D *x1 *2 *delta *() t-n : δ(t-n)
.D *x2 *sp e *2 -jn *omega : e^-jnω
.D *x3 *fra *sp *lambda n *2 n ! : λ^n / n!
.D *x4 *sigma3 n=0 *inf *x3 : Σ_{n=0}^{∞} λ^n / n!
.D *sub *2 *sp e *2 - *lambda *x4 : e^-λ Σ_{n=0}^{∞} λ^n / n!
*5 f *() t = *sub *x1
*5 F *() *omega = *sub *x2
```

$$f(t)=e^{-\lambda}\sum_{n=0}^{\infty}\frac{\lambda^n}{n!}\delta(t-n)\ ,\ F(\omega)=e^{-\lambda}\sum_{n=0}^{\infty}\frac{\lambda^n}{n!}e^{-jn\omega}$$

where, symbol *k concatenates k"units".

At last, we show a rule bringing a square root expression. The relation of parameters to the style is shown in Fig.11.

```
*Root (x)
.Ox?
.Ssxy
X
.Gsxy x1 y1 y2
.Set xm / * 23 %x 32
.Sox xm
.Ox -- + x1 xm
.St %a %y
.Set yh / %y 8
.Y + yh y1
.Oy - y2 yh
.A 30
.Bi !
.Rs %a %y
.Line? 0 0 + x1 / %x 6 0 / %y 16
.Oy - yh y2
.Ox + x1 / %x 3
.Sesy -y2 yh + y1 y2
```

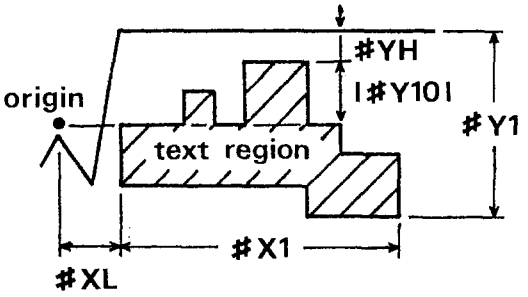

Fig11. An explanation for *Root.

9.Experiences

Problems such as, how well this system satisfies a goal of being easy to use, how easy it was to build or extend beyond its original limitations should be discussed. As to ease of use, from our experinces, FROFF is found to be useful tool for making fair copy of documents except for a few deficiencies. They are inconvenient in doing mathematics,and has a weakpoints in its speed and a lack of small characters, which have been already mentioned in this paper. Though there is obviously a great deal more to be done in order to improve these deficiencies, any other way of supplementing the first problem is not taken at present.

For the rest of those problems, the following method is adopted. That is, we decided to use FROFF as a device to accomplish a complete document, and use the above mentioned ROFF with fixed character sizes as that for a draft or a debugging. As shown in this figure, the text which was typed in accordance with the prospectus of ROFF are translated into that for FROFF. In this case, commands or rules for mathematics and figures are not affected with this translation, only a conversion from mixtures of big or small characters into a description corresponding to the restrictions of FROFF is conducted.

As this ROFF and its translator are written in Assembler, steps needed for obtaining a complete result are decreased to a large extent. This paper is, of course, typeset by the authors using first the ROFF, secondly the translator and at last using FROFF.

Acknowledgements

The authors would like to thank Fumihide Itoh, Itsuya Soga for their contribution in preparation of this paper.

TYPE	SYMBOL	DEFINITION
COMM -AND	.A *n*	Selection of font of characters
	.X *n*	Alter a width of characters into n
	.L *n*	Space out n lines
	.Pl *n*	Set a length of page to n
	.B *n*	Definition of character size : big or small letter etc.
	.Ox *n*	Sift an output location to horizontal direction
	.Joy	Accept a coordinate by the joystic
	.Pic	Output a picture
	.Line	Drow a line
	.Box	Output a box
RULE	*It	Print an expression in italic form
	*Sf a b	Make an expression with a suffix : a_b
	*Sp a b	Make an expression a^b
	*Fra a b	Print a fractional expression $\frac{a}{b}$
	*Sigma3 a b c	Make an expression $\sum_a^b c$
	*Theta	Print a Greek letter θ
	*Phd osaka	Output a string *Ph.D. Thesis, Osaka University,*
	*Joho 5	Print a string *Information Processing of Japan, 5*

Table 1. An Exampl of Commands and Rules.

Reference

1] Jack E. Shermer and Lorinda L. Cherry : The Soft Display Word Processor, Computer, Vol.11, No.12 p.39 (DEC. 1978)

2] David H. Hartke, Warren M. Sterling, and Jack E. Shermer : Design of a Raster Display Processor for Office Applications : IEEE Computers, C-27, No.4, pp.337-348 (APR. 1978)

3] B. W. Kernighan, L. L. Cherry, A System for Typesetting Mathematics, CACM, Vol.18, No.3, p.151 (1975)

4] Tsukuba Computer Center : Tsukuba Run-off System, (1976)

5] M. E. Lesk and B. W. Kernighan : Computer Typesetting of Technical Journals on UNIX, Proc. of NCC, pp.879-888 (1977)

6] K. Hasebe, S. Nomoto and H. Ishida : An On Line Phototypesetter Support System, Computer Center, Univ. of Tokyo, (MARCH 1979)

7] M. Higashide : Roff User's Manual, Osaka University, (APR. 1978)

8] M. Higashide : FLISP System and its Application to Run Off, Master Thesis of Osaka University, (FEB. 1979)

9] N. Abe, M. Higashide and S. Tsuji : An Improvement of FLISP System by Micro Programming, IECE-D, (1979)

10] M. Higashide, K. Konishi, N. Abe and S. Tsuji : The FLISP System for Mini Computer using B-frame Technique and M-frame Techinque, Information Processing of Japan, No.1, pp.8-16, (1979)

Command name	Functions
.St arg-1 --- arg-n	Arg-i is a variable including system parameters, this stacks values of arg-i.
.Rs arg-1 --- arg-n	This resets values of arg-i stacked by .St.
.Ox n	Shifts a next output location to horizontal by n.
.Ox?	This is converted to the command .Ox when a command .Sox is found in the text.
.Gsy arg-1 arg-2 arg-3	As shown in Fig.11, height of text from the origin is computed.
..If pred text1 text2	If pred is true, text1 is regarded as a text, else text2.

Table 2. Examples of control commands.