THE RC8000 TEXT COMPOSING SYSTEM

RCSL No:

Edition: February 1979

Author: Felicity Pors
Henrik Quistgaard
Henning Suwalski
Erik Poulsen

Keywords: RC8000, description, typsetting, Typol

Abstract: The document contains a description of a system for the RC8000 for transformation of an errouneous and unjustified text to a justified with respect to both lines and pages, and written in a format corresponding to one of various numerically controlled typesetting machines.

Contents:

1. INTRODUCTION.

The aim of the system is easy transformation of an erroneus and unjustified text to a correct text with justified lines and page make up, and written in a format corresponding to one of various numerical controlled typesetting machines.
The justification of lines to alignment at the margin is automatically made by calculating the appropriate size of the variable spacing and calling a hyphenation routine if necessary. Underlined and composed characters are correctly treated.
The page make up includes automatic pageshift and pagination and insertion af a running head. Foot-notes, place for figures etc. specified in the input text vill be correctly placed automatically.

The input is formed so that the typographical information is inserted in the unjustified text as simple mnemonic codes, and the input can be supplied on 6-7-8-channel papertape, magnetic tape, disc, typewriter and other media corresponding to the peripheral units on the actual RC8000 configuration
Similary output can be written on all kinds of existing output media, typesetting machines coupled on-line included.
Correction are made with the RC8000 EDITOR program, which is a versalite program for editing and correcting af data in text form, including possibilities for deletion, insertion, substitution, merging of different data etc.
The system is designed so that only a minimum of typographical knowledge is necessary for the users and the typographical units are as far as possible converted to metric units. The operation of the system is made as simple as possible so that the system can be run by people without computer education.
The system is primary designed for working with the ISO character set, composed and underlined characters included, but additional characters can easily be included and other types of codes e.g. Flexowriter-code and TTS-code can be accepted.

2. GENERAL DESCRIPTION.

The system is shown on the flowchart in appendix A and consists of the following stages:

1. Writing of textdata for the input program.

2. Running the input program.

3. Proofreading of the printer output from the input program.

4. Correction of the output from the input program with the EDITOR program.

5. Running the composing program.

6. Proof reading of the printer output from the composing program.

7. Composing.

2.1 Writing of textdata

The text is written on papertape or magnetic tape, with typographical instruction in mnemonic codes inserted. The instructions are in many cases just a formalizing of the comments which the author writes in his manuscript as information to the typesetter. In appendix C the syntax and the semantics of the Typol command set is described. Appendix B1 shows an example of input illustrating most of the Typol commands.

The page layout command *pl 297,40,210,25,10* denotes that the page length is 297 mm, the distance from the upper edge of the page to the first line of the text 40 mm, the distance from the first text line to the last is 210 mm, the distance from the upper edge of the paper to the running head and the page number 25 mm and the distance from the lower edge of the page to a page number placed below 10 mm.

The centering command *ct* has the effect that all lines will be centered with respect to the actual left margin and linewidth until the command *rj* (reestablish justification) is met. *nl* and *np* start a new line and a new paragraph respectively.

2.2 Running the input program.

The input program reads the data mentioned above and a new set af data to be used as input to composing program is produced.

During this process the following is carried out:

a. a check of the typographical commands.
b. a listning of the output including possible error messages.
c. insertion of linenumbers for every 10 lines facilitating a succeeding correction of errors.

In appendix B2 an example of the output listning is shown.

2.3 Proof reading of print-out from input program

The print-out is proof read. If no errors occur the output from the input program is used as input to the composing program otherwise the errors should be corrected with the RC8000 EDITOR program.

2.4 Corrections

Corrections are, as mentioned above, made with the EDITOR program which includes possibilities of corrections made on-line as well as off-line. More detailed information about this program can be found in Ref. 2.

The linenumbers inserted by the input program make it easy to find the lines where the corrections are to be made, because the inserted linenumbers can be used as identifiers in the search command: line./<identifier>/.

If the corrections concern the typographical instructions it is recommended that the input program checks the corrected data before they are used by the composing program.

2.5 Running the composing program.

The composing program reads the corrected data and carries out the justification and the page make up corresponding to the typographical commands. Normaly the first run of the composing program is made with a proof reading file as output. This file will have lineshift and page make up corresponding to the justified test but will be without justified right margin because of the lack of individual spacing on the lineprinter. Appendix B3 contains an example of a proof reading file.

If errors are found during proof reading of the file the input to the composing program is corrected as mentioned in par. 2.4

and a new proof reading file is produced. When the proof reading file is correct an object file is produced containing the justified data in a format and on a medium corresponding to the actual typesetter machine.

Hyphenation information will, during production of a proof reading file, normaly be furnished either by an algorithme in the program or manualy from an RC8000 terminal. If the information is supplied from a terminal it can be stored and, after acceptance of the proof reading file, be used during production of the object file.

Appendix B4 shows the text used in the previous examples set on a Diablo 1620.

3. Program description.

3.1 Input program.

3.1.1. Summary.

The input program carries out a syntax check on data to be used in the composing program. A copy of the data is made with numeric names inserted every tenth line which can be used as pointers if the text is to be corrected. (Subsequent runs of the input program or the composing program ignore the names.) The output is in the correct format for the composing program concerning composed graphics.

A text file is created, if required, showing output file with numeric names, and error messages, if any.

3.2.2. Input

There may be one or more input files, the end of a file is recognised when an EM character is received. The end of the input data is recognised when the command EF is read.

The input data is text and TYPOL commands as defined in the appendix TYPOL COMMAND SET.

3.1.3. Output.

The output file is a copy of the input data - several input files may be used to produce one output file.

Composed characters are arranged so that the sequence of the elements is

<graphic><backspace><graphic><backspace><graphic>...

Numeric names are inserted at the beginning of every tenth line. The names is the line number bracketed by the separator which is used at that time. A line consists of graphic characters terminated by an NL or FF character.

3.1.4. Text.

The text file consist of the output data and error messages, if any, written so that if the output device is printer there will be 45 lines to a page. A left margin is inserted in every line, and the numeric names are placed in the margin so that they stand out in the text.

Tabulating charactersted by the character &. Error messages are output after the line containing the error. The error output is defined in appendix E.

3.1.5. Input table file.

This file required if the user wishes to specify:

a. The conversion of input code to ISO code

   and/or

b. The ISO code values corresponding to the caracters valid on the equipment for which the composing program is providing input.

The standard input table which will be used, if no other is specified, is for ISO code input and Line Printer being used as typesetting equipment.

The format of data in the file is

```
<value of the character on users equipment><delimiter>
<ISO value required for this character><delimiter>
<class of the character><delimiter>  ..............
...............  EM
```

<delimiter> is as defined in 2.0.1 of the Algol specification.
<value of the character on user equipment> has the range 0-511.
<ISO value> has the range 0-127
<class> has the range 0-5.

class 0	blind, the character is skipped.
class 1	shift character.
class 2	simple character.
class 3	line terminator. members of the class NL, FF, EM characters
class 4	special graphic character. member of the class BS,CR characters
class 5	invalide character which must have ISO value 33

A check is made so that characters with ISO value 0,127, 10,12,26,8 or 13 are given their respective classes.

Any values in the range 0-511 which are not supplied are given the ISO value 33 and class 5.

It is recommended that section 9.32 of the algol 6 manual (Ref. 3) is studied, especially with referance to character to be given class 1.

N.B. If the user is to have an input table, he must ensure that there is a character specified which is transformed to the ISO value 25, an EM character.

3.2. Composing program.

3.2.1. Summary.

The aim of this program is to transforme a text in ISO-code containing correct typographical commands (typol) to a justified text written in a format corresponding to the selected typesetting machine.

3.2.2. Input.

To get a well-defined output the input must fulfil the following conditions:

a. the format must be ISO

b. it must contain correct typographical commands(typol)

c. if composed or underlined character are present they must be arranged as defined in the description of the input program.

These conditions are fulfilled if the input is generated as output from the input syntax checking program.

All readable external media are allowed for the input file.

3.2.3. Output.

Output consists primary of two kinds of data:

a. Object file

The text justified with respect to line and pages etc. according to the typol commands, the text is written on paper tape, magnetic tape, or another medium corresponding to kind and format of the input medium of the selected typesetting machine.

b. Proofreading file.
The same information ad above written on a document selected by the user.
The printout is as near as possible to the output from the typesetting machine, but without a right justification margin because of the lack of individual spacing on for instance a lineprinter. This is also the reason you somtimes observe that the position of the same tabulator mark on two lines is not exactly identical. Font shifts and changes in leading which do not exist on normal peripheral devices are marked by *) and **) respectively. A special symbol defined be substitute command is marked by *d, d being the number of the special symbol corresponding to the Special Symbol Table for the selected typesetting machine. ISO-characters which are not available on the actual typesetting machine will be substituted by *.

4. PROGRAM CALL.

This chapter descripes the call of the programs. The general rules for the File Processor are given in Ref. 1.

4.1 Input program

Syntax:

```
                                  *
                 <s><source>
<object> = inp
                 <s><modifier>
                                  1


                  table.<table file>
<modifier> ::=    text.<text file>
                  machine.<machine number>


  <object>
  <source>
  <table file>    ::=  <name>
  <text file>

<machine number>  ::=  <integer>
```

Semantic:

`<source>`	The list of sources specifies the input files to inp. Source may be any RC8000 text file. At least one source must be present.
`<object>`	Object may be any RC8000 text file. If the file does not exist, or cannot be used, a temporary file is created.

<modifier> The list of modifiers is scanned from left to right. Each modifier change a variable that controls inp. When the scan starts, the variables are initialized to the value explained below.

table.<table file> Inp reads the input table from table file, as explained in 3.1.5.
The initial setting is standard input table.

text.<text file> Select the output text file (3.1.4) to <text file>, which may be any RC8000 text file. If the does not exist, or cannot be used, a temporary file is created.
The initial setting is no text file.

machine.<machine number> Select typesetting machine.
1 Line Printer
2 Diablo 1620
The initial setting is Line Printer.

Error messages after program call:

***inp end. no object
Object file missing in program call.
The program is terminated.

***inp end. no source
Source file missing in program call.
The program is terminated.

***inp end. input table data error
Data in the character table file is invalde e.g. outside limits for ISO char. value, or class value.
The program is terminated.

```
***inp end. connect <name>, <result>


                               no ressources
                               malfunction
                <result> ::=   not user, not exist
                               convention error
                               not allowed

                A file cannot be connected by inp.
                The program is terminated.

***inp param <invalide param>
                Invalide parameter in program call.
                The program is terminated.
```

4.2 Composing program

Syntax:

```
                    1                     1                     *
   <object> =         compose  <s><source>    <s><modifier>
                    0                     0                     0


                      proof.<proofreading file>
                                   .c
<modifier>  ::=       hyphen       .c.<hyphen file>
                                   .<hyphen file>
                      machine.<machine number>

   <object>
   <source>
   <proofreading file>   ::=   <name>
   <hyphen file>

<machine number>   ::=   <integer>
```

Semantic:

<object>	Object may be any RC8000 document. If object does not exist, or cannot be used, a temporary file is created.
<source>	The source specify the input to compose. Source may be any RC8000 text file. I no source is specified, compose reads the source from current input.
<modifier>	The list of modifiers is scanned from left to right. Each modifier change a variable that controls compose. When the scan starts, the values are initialized to the values explained below.

proof.<proofreading file> Select the proofreading file to < proofreading file>, which may be any RC8000 text file. If the file does not exist or cannot be used, a temporary file is created.
Initial setting is no proofreading file.

machine.<machine number>   Select typesetting machine.
1 Line Printer.
2 Diablo 1620.
Initial setting is Line printer.

hyphen.c.<hyphen file>   Information for hyphenation is supplied from the terminal (see 4.2.1 for details) and stored in the so-called hyphenation file for possible later use with the same source file after an accepted proofreading. The hyphenation file may be any RC8000 text file. If the file does not exist or cannot be used, a temporary file is created.

hyphen.c   Information is supplied from terminal, but not stored.

hyphen.<hyphen file>   Information is read from hyphenation file.

The initial setting for hyphenation mode is that information is supplied from the hyphenation algorithm alone whitout any communication with the user.

Error messages after program call:

```
***compose end. connect <name>, <result>

                                  no resources
                                  malfunction
              <result>  ::=       not user, not exist
                                  convention error
                                  not allowed
```

A file cannot be connected by compose
The program is terminated

```
***compose param <param>
```

Invalide parameter in program call.
The program is terminated.

4.2.1 Hyphenation information

The program has a built-in algorithm for hyphenation, but has a the possibility for an online check of the result from this algorithm by using hyphen.c modifier in the program call of compose. If this is the case the actual word and a proposal for hyphenation is written on the terminal in the following way:

```
<sting1>/<string2><breakinf><string3>/<string4>
+/-
```

where <string1>,<string2>,<string3> and <string4> are strings of succeeding characters of the word, together containing the whole word. Possible special symbols are printed as stars.

<breakinf> is a new line character if a natural breakpoint is found, as for instance a period or question mark, or it is a hyphen followed by a new line character if a hyphenation point is found.
If the algorithm has been unable to find a breakpoint <breakinf> is empty.
The slashes indicate the limits for the position of the breakpoint to get the line justified with respect to the values for minimum and maximum allowed word space

The last line in the message indicate your possibilities for reply expected from the terminal:

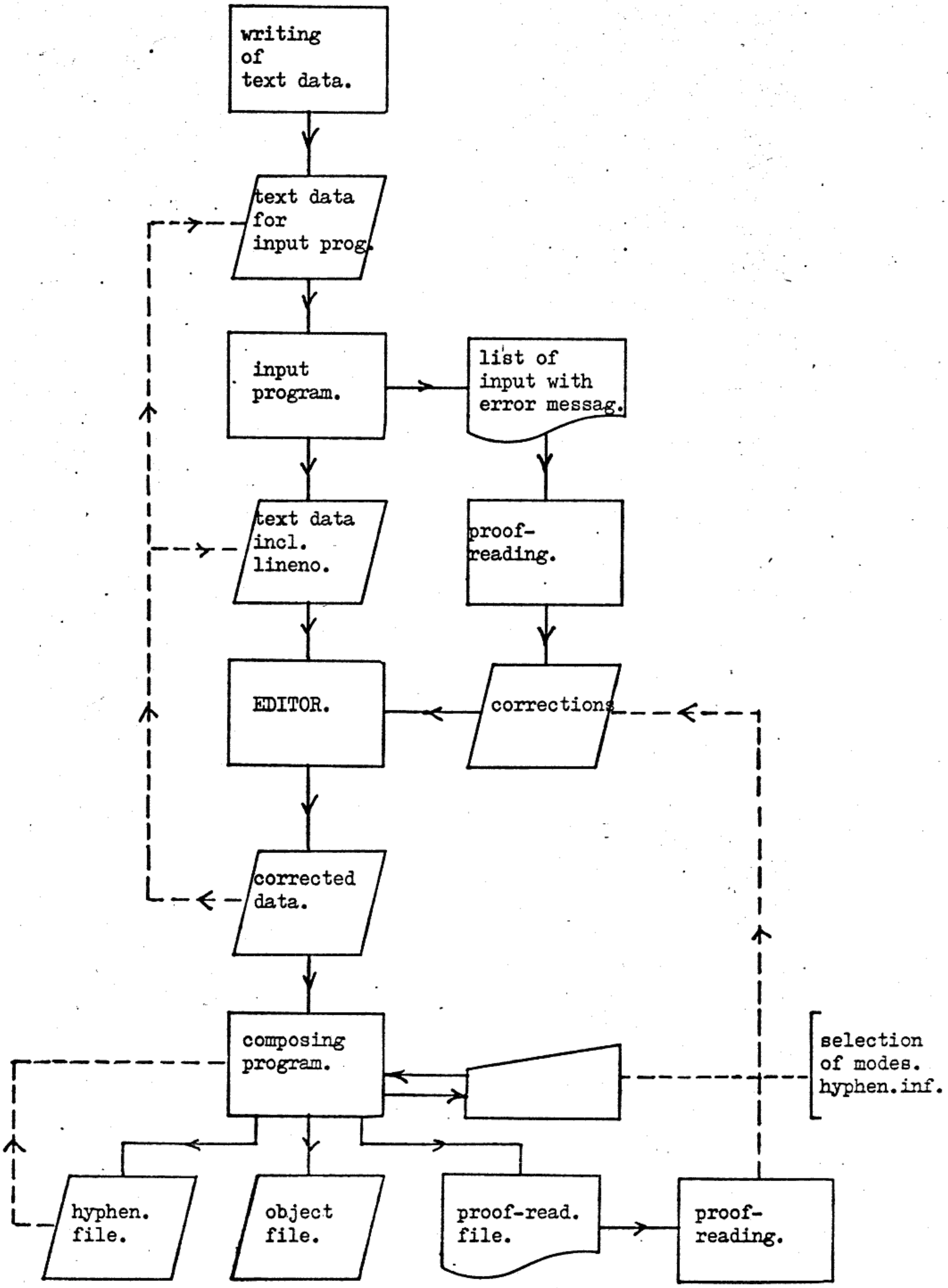
Type a plus sign and a new line character if you accept the proposal.

Type a minus sign and a new line character if you can not accept the proposal. The message:

```
position:
```

will follow and an integer followed by a new line character is now expected, denoting the position relative to first slash of the correct breakpoint. <breakinf> character, if present, are ignored. In the case a hyphen has to be inserted the integer should be positiv else negative .
If no breakpoint exist between the slashes a zero can be typed, and no hyphenation takes place.

THE RC8000 TEXT COMPOSING SYSTEM



```
*pl 297,40,210,25,10*  *pn 5,135*  *lw 135*  *ld15*
*ps 40*  *rh 1,EXECUTION TIMES*
*qr*  Appendix H
*ct*

    Executions times in microseconds
*rj*  *np*
The times given below represent the total physical times for
execution of algorithmic constituents. The total time to exe-
cute a program part is the sum of the

       times for the constituents.
 The times are only valid under
thefolluwning assumpti *ro*  the following  assumptions:
*sj*
   1.   The time for transfer of segments is
        negligible.
   2.   The program is not waiting for peripheral
        devices.
   3.   The time interval is 25.6 ms or more.
   4.   The program is the only internal proces in
        the computer.
*nl**rj*

When the computer is time shared, assumption 4 is not   fulfilled,
but then the time represent the CPU time used by the program.

*nl*  *ta 5,122*
H.1. Operand referances
 Referances to local identifier and constants   0
*cm place the number up to a right edge*
 Referances to non-local identifiers 0-4
*rj*  *lm 10*  *ld 9*  *lw 85*
If a sequence of identifiers from the same non-local block area
referenced without intervening references to other non-local
blocks, the first reference costs 4 microseconds and the later
ones usally 0.
*ld 15*  *lm 0*  *lw 135*  *ta 5,122*
 Referance to name parameter, actual is simple   9
 Reference to name parameter, actual is composite 150

*rj*
H.2 The compiler *np*
The compiler occupies about 13000 instructions divided into 12
passes, either on backing storage or on magnetic tape. In the first
case, it may be used for simultaneous translation in several job
processes. *np*  *sl3*
The 12 passes   of the compiler perform the following tasks: Pass
0 is a common administration routine. Pass 1 to 8 perform the trans-
lation into binary code by means of 8 scans of the source program
*ds !* !u
*fn 1,The program is stored in the place later occupied by the
      binary program*!d.
Pass 9 rearranges the binary program, inserts referances to stan-
dard procedures used in the program. Pass 10 includes the run time
```

```
administrative system (RS). *fn 1,When an external procedure is
translated, pass 9 only rearranges the binary procedure and RS
is not included.* Pass 11 is only active when the branch
test mode is specified.
*ds*

*ns 1,2,H.3 Semantics**lm 33*
*mt 1,bs file*  A file descriptor describing a backing storage
area. It is used as working area for the compilation and the object
code ends here and is described in the file descriptor. If
bs file initially is an empty note, the computer creates a
working area of 100 segments, translates       the program
and cuts the length the segments necessary. *nl*
If translation is successful, bs file will contain a complete
object program or an external procedure ( a standard procedure).
*nl2*

*mt 1,source*      The list of sources specifies the input
files to the compiler (see 2.0.3). If no source is specified
the compiler reads the source from current input.
*nl2*
*mt 1,branch.yes*The algol program is modified so that
the running program prints a list of all the branch points passed
(if then , for do , case , and switch)
A survey of branches not passed is printed when the program exits
through the final end. A set of the line interval to which
the branch test is restricted may be specified.
 Details of the branch-test
ideas area published separately.  *nl*
The initial setting is branch.no.
*nl2*  *fg40*

*mt 1,details.yes*
Intermediate output from all passes of the compiler is printed on
current output. The meaning of the output will be explained in
the Maintenance Manual. The output may be restricted
to an interval of pass numbers and to an interval of line numbers.
*nl* The initial setting is details.no
*nl2*

*mt1,survey.yes*A summery is printed on current output after
the completion of each pass of the translation. The meaning
will be explained in the Maintenance Manual.

*ef*
```

```
      *pl 297,40,210,25,10*  *pn 5,135*  *lw 135*  *ld15*
      *ps 40*  *rh 1,EXECUTION TIMES*
      *gr*  Appendix H
      *ct*

          Executions times in microseconds
      *rj*  *np*
      The times given below represent the total physical times for
      execution of algorithmic constituents. The total time to exe-
      cute a program part is the sum of the
*0010*
*0010*         times for the constituents.
       The times are only valid under  the following  assumptions:
      *sj*
         1.    The time for transfer of segments is
               negligible.
         2.    The program is not waiting for peripheral
               devices.
         3.    The time interval is 25.6 ms or more.
         4.    The program is the only internal proces in
               the computer.
*0020**nl**rj*

      When the computer is time shared, assumption 4 is not   fulfilled,
      but then the time represent the CPU time used by the program.

      *nl*  *ta 5,122*
      H.1. Operand referances
      &Referances to local identifier and constants&  0
      *cm place the number up to a right edge*
      &Referances to non-local identifiers&0-4
      *rj*  *lm 10*  *ld 9*  *lw 85*
      If a sequence of identifiers from the same non-local block area
*0030*referenced without intervening references to other non-local
      blocks, the first reference costs 4 microseconds and the later
```

```
        ones usally 0.
        *ld 15*  *lm 0*  *lw 135*  *ta 5,122*
        &Referance to name parameter, actual is simple&  9
        &Reference to name parameter, actual is composite&150

        *rj*
        H.2 The compiler *np*
        The compiler occupies about 13000 instructions divided into 12
        passes, either on backing storage or on magnetic tape. In the first
*0040*case, it may be used for simultaneous translation in several job
        processes. *np*  *sl3*
        The 12 passes   of the compiler perform the following tasks: Pass
        0 is a common administration routine. Pass 1 to 8 perform the trans-
        lation into binary code by means of 8 scans of the source program
        *ds !* !u
        *fn 1,The program is stored in the place later occupied by the
             binary program*!d.
        Pass 9 rearranges the binary program, inserts referances to stan-
        dard procedures used in the program. Pass 10 includes the run time
*0050*
*0050*administrative system (RS). *fn 1,When an external procedure is
        translated, pass 9 only rearranges the binary procedure and RS
        is not included.* Pass 11 is only active when the branch
        test mode is specified.
        *ds*

        *ns 1,2,H.3 Semantics**lm 33*
        *mt 1,bs file*  A file descriptor describing a backing storage
        area. It is used as working area for the compilation and the object
        code ends here and is described in the file descriptor. If
        bs file initially is an empty note, the computer creates a
*0060*working area of 100 segments, translates      the program
        and cuts the length the segments necessary. *nl*
        If translation is successful, bs file will contain a complete
        object program or an external procedure ( a standard procedure).
```

```
      *nl2*

      *mt 1,source*      The list of sources specifies the input
      files to the compiler (see 2.0.3). If no source is specified
      the compiler reads the source from current input.
      *nl2*
      *mt 1,branch.yes*The algol program is modified so that
*0070*the running program prints a list of all the branch points passed
      (if then , for do , case , and switch)
      A survey of branches not passed is printed when the program exits
      through the final end. A set of the line interval to which
      the branch test is restricted may be specified.
       Details of the branch-test
      ideas area published separately.  *nl*
      The initial setting is branch.no.
      *nl2*  *fg40*

      *mt 1,details.yes*
*0080*Intermediate output from all passes of the compiler is printed on
      current output. The meaning of the output will be explained in
      the Maintenance Manual. The output may be restricted
      to an interval of pass numbers and to an interval of line numbers.
      *nl* The initial setting is details.no
      *nl2*

      *mt1,survey.yes*A summery is printed on current output after
      the completion of each pass of the translation. The meaning
      will be explained in the Maintenance Manual.

      *ef*
```

Appendix H

Executions times in microseconds

The times given below represent the total physical times for execution of algorithmic constituents. The total time to execute a program part is the sum of the times for the constituents. The times are only valid under the following assumptions:

1. The time for transfer of segments is negligible.
2. The program is not waiting for peripheral devices.
3. The time interval is 25.6 ms or more.
4. The program is the only internal proces in the computer.

When the computer is time shared, assumption 4 is not fulfilled, but then the time represent the CPU time used by the program.

H.1. Operand referances

Referances to local identifier and constants	0
Referances to non-local identifiers If a sequence of identifiers from the same non-local block area referenced without inter-vening references to other non-local blocks, the first re-ference costs 4 microseconds and the later ones usally 0.	0-4
Referance to name parameter, actual is simple	9
Reference to name parameter, actual is composite	150

H.2 The compiler

The compiler occupies about 13000 instructions divided into 12 passes, either on backing storage or on magnetic tape. In the first case, it may be used for simultaneous

translation in several job processes.
The 12 passes of the compiler perform the following tasks: Pass 0 is a common administration routine. Pass 1 to 8 perform the translation into binary code by means of 8 scans of the source program !u 1) !d. Pass 9 rearranges the binary program, inserts referances to standard procedures used in the program. Pass 10 includes the run time administrative system (RS). 2) Pass 11 is only active when the branch test mode is specified.

H.3 Semantics

bs file	A file descriptor describing a backing storage area. It is used as working area for the compilation and the object code ends here and is described in the file descriptor. If bs file initially is an empty note, the computer creates a working area of 100 segments, translates the program and cuts the length the segments necessary. If translation is successful, bs file will contain a complete object program or an external proceaure (a standard procedure).
source	The list of sources specifies the input files to the compiler (see 2.0.3). If no source is specified the compiler reads the source from current input.
branch.yes	The algol program is modified so that the running program prints a list of all the branch points passed (if then , for do , case , and switch) A survey of branches

1) The program is stored in the place later occupied by the binary program
2) When an external procedure is translated, pass 9 only rearranges the binary procedure and RS is not included.

	not passed is printed when the program exits through the final end. A set of the line interval to which the branch test is restricted may be specified. Details of the branch-test ideas area published separately. The initial setting is branch.no.
details.yes	Intermediate output from all passes of the compiler is printed on current output. The meaning of the output will be explained in the Maintenance Manual. The output may be restricted to an interval of pass numbers and to an interval of line numbers. The initial setting is details.no
survey.yes	A summery is printed on current output after the completion of each pass of the translation. The meaning will be explained in the Maintenance Manual.

Appendix H

Executions times in microseconds

The times given below represent the total physical times for execution of algorithmic constituents. The total time to execute a program part is the sum of the times for the constituents. The times are only valid under the following assumptions:

1. The time for transfer of segments is negligible.
2. The program is not waiting for peripheral devices..
3. The time interval is 25.6 ms or more.
4. The program is the only internal proces in the computer.

When the computer is time shared, assumption 4 is not fulfilled, but then the time represent the CPU time used by the program.

H.1. Operand referances

Referances to local identifier and constants	0
Referances to non-local identifiers If a sequence of identifiers from the same non-local block area referenced without intervening references to other non-local blocks, the first reference costs 4 microseconds and the later ones usally 0.	0-4
Referance to name parameter, actual is simple	9
Reference to name parameter, actual is composite	150

H.2 The compiler

The compiler occupies about 13000 instructions divided into 12 passes, either on backing storage or on magnetic tape. In the first case, it may be used for simultaneous translation in several job processes.

The 12 passes of the compiler perform the following tasks: Pass 0 is a common administration routine. Pass 1 to 8 perform the translation into binary code by means of 8 scans of the source program 1) . Pass 9 rearranges the binary program, inserts referances to standard procedures used in the program. Pass 10 includes the run time administrative system (RS). 2) Pass 11 is only active when the branch test mode is specified.

H.3 Semantics

bs file	A file descriptor describing a backing storage area. It is used as working area for the compilation and the object code ends here and is described in the file descriptor. If bs file initially is an empty note, the computer creates a working area of 100 segments, translates the program and cuts the length the segments necessary. If translation is successful, bs file will contain a complete object program or an external procedure (a standard procedure).
source	The list of sources specifies the input files to the compiler (see 2.0.3). If no source is specified the compiler reads the source from current input.
branch.yes	The algol program is modified so that the running program prints a list of all the branch points passed (if then , for do , case , and switch) A survey of branches

1) The program is stored in the place later occupied by the binary program

2) When an external procedure is translated, pass 9 only rearranges the binary procedure and RS is not included.

not passed is printed when the program exits through the final end. A set of the line interval to which the branch test is restricted may be specified. Details of the branch-test ideas area published separately.
The initial setting is branch.no.

details.yes	Intermediate output from all passes of the compiler is printed on current output. The meaning of the output will be explained in the Maintenance Manual. The output may be restricted to an interval of pass numbers and to an interval of line numbers. The initial setting is details.no
survey.yes	A summery is printed on current output after the completion of each pass of the translation. The meaning will be explained in the Maintenance Manual.

Appendix C. TYPOL COMMAND SET

The following notation is used in the definition of the commands:

<argument>	: <quantity in mm>
<parameter by name>	: <integer>
<string>	: <textstring without commands>
<separator>	: <visible character defined by the separator command or by standard>
<command>	: <separator><typol><separator>
<diablo separator>	: <visible character defined by the diablo separator command>
<diablo command>*	: <diablo separator followed by one or two characters>

*A <diablo command> has no effect if typesetting machine is not diablo 1620.

The <diablo command> must be of the types stated below:

diablo command	action
<diablo separator>u	move the paper <subscript leading> points up.
<diablo separator>d	move the paper <subscript leading> points down.
<diablo separator>h<char>	The character <char> is written as subscript <subscript leading> below current line.
<diablo separator>l<char>	The character <char> is written as superscript <subscript leading> points above current line.

Typol

Use of small or capital letters in the mnemonic codes is optional. Spaces outside the parameter <string> are blind.
When the composing program is started the parameters are assigned the standard values given in the description of the commands.
The source text is handled in one of five modes, dependent of the last command met, which changes the mode:

1. justifying mode (command RJ)
2. non-justifying mode (command SJ)
3. tabulating mode (command TA)
4. centering mode (command CT)
5. quadding right mode (command QR).

Standard: justifying mode.

Comment command

CM <string>

<string> is interpreted as a comment and is not printed with the remaining text.

Centering command

CT

Transition to the centering mode.
The text on the following lines is centered with respect to the current left margin and linewidth. All characters are significant with the exception of spaces preceding the first visible character on a line.
If the command occurs inside a text line, that is no significant or ordered new line character exists between the last text character and the command, the line is terminated and a new line character is generated.

Diablo Separator

DS <diablo separator>

Specifies the separator signifying start of a diablo command, according to the definitions prefacing this appendix. Underline, overline and hyphen are not allowed as <diablo separator>.
The parameter can be omitted, for example
DS
release a possible character used as <diablo separator>.
Standard: no character is <diablo separator>

End of file command

EF

Indicates end of source file.
The current page is completed and the program terminates in an orderly fashion.

Figure command

FG <argument>

Reserves room for a figure with the height <argument>, defined as the distance from base of current line to base of the line following the command, in the entire width of the page.
If it is impossible to place the figure on the current page the reservation is made on the upper part of the next page.

Footnote command

FN <font>,<string>

Specifies a footnote <string> with font no. <font> to be placed at the bottom of the page. The command is replaced with n) where n is the footnote number on that page. The footnote is handled in the justifying mode with respect to the left hardware margin and the current right margin and placed at the page bottom so that the linefeed between the last text line and the lower page limit is preserved if possible. The footnote is printed with minimum leading, which will be automatically established if necessary (see the leading command, LD).

Font command

FT<font>

Orders a shift to font no. <font>.
Standard: 1

Leading command

LD <leading>

Specifies the distance in points between the lines as <leading>.
(1 point=0.353 mm).
Standard: see appendix for the selected typesetting machine.

Left margin command

LM <argument>

Specifies left margin relative to hardware margin as <argument>. In addition the line width is changed so the current right edge is preserved.
If the command does not occur in connection with a line shift, it is first executed after the current line is finished.
Standard: 0

Line width command

LW <argument>

Specifies line width as <argument>.
If the command does not occur in connection with a line shift, it is first executed after the current line is finished.
Standard: 140

Margin text command

```
MT <font>,<string>
```

Specifies <string> to be placed with font no. <font> in the left margin at the current line, if the command occurs before the first real character else in the left margin at the following line.
If 2 margin texts is to be placed in the margin at the same line, the program terminates with an error message
It is noticed that margin text in connection with new section text is not possible.
<string> must not contain any new line characters.

New line command

```
NL <linefeed>
```

Orders <linefeed> line shifts to be generated. <linefeed> can be omitted and one line shift is generated.
The command is executed in all modes, but remember that the new line character is significant in the non-justifying, tabulating, centering and quadding right mode.

New paragraph command

```
NP <argument>
```

Works like NL 1, but in addition an indentation corresponding to <argument> will be applied to the first word in the following line.
<argument> can be omitted and the standard value 5 is used.

New section command

```
NS <font>,<linefeed>,<string>
```

Generates <linefeed> line shifts, prints <string> with font no. <font> starting from the current left margin and finally generates one line shift.
<string> must not contain any new line characters.

Page layout command

```
PL <arg.1>,<arg.2>,<arg.3>,<arg.4>,<arg.5>
```

Specifies the page layout as following (see figure 1):

<arg.1>	page length.
<arg.2>	distance from upper page limit to the first real text line, that is page number and running head not included.
<arg.3>	distance from first to last real text line.
<arg.4>	distance from upper page limit to the first line in running head.
<arg.5>	distance from upper page limit to a centered page number above or from lower page limit to a page number below depending on which way the page number is to be placed (see command PN).

An argument but not the commas can be omitted and the standard value is used.

Standard:

<arg.1>	297
<arg.2>	30
<arg.3>	235
<arg.4>	18
<arg.5>	10

Page numbering command

```
PN <position>,<page number>
```

Specifies by the parameter <position> the way for placing the page number:

0 no page numbering.
1 below and centered
2 above and centered.
3 below and at left hardware margin if even and at right margin if odd page number.
4 above and at left hardware margin if even and at right margin if odd page number.

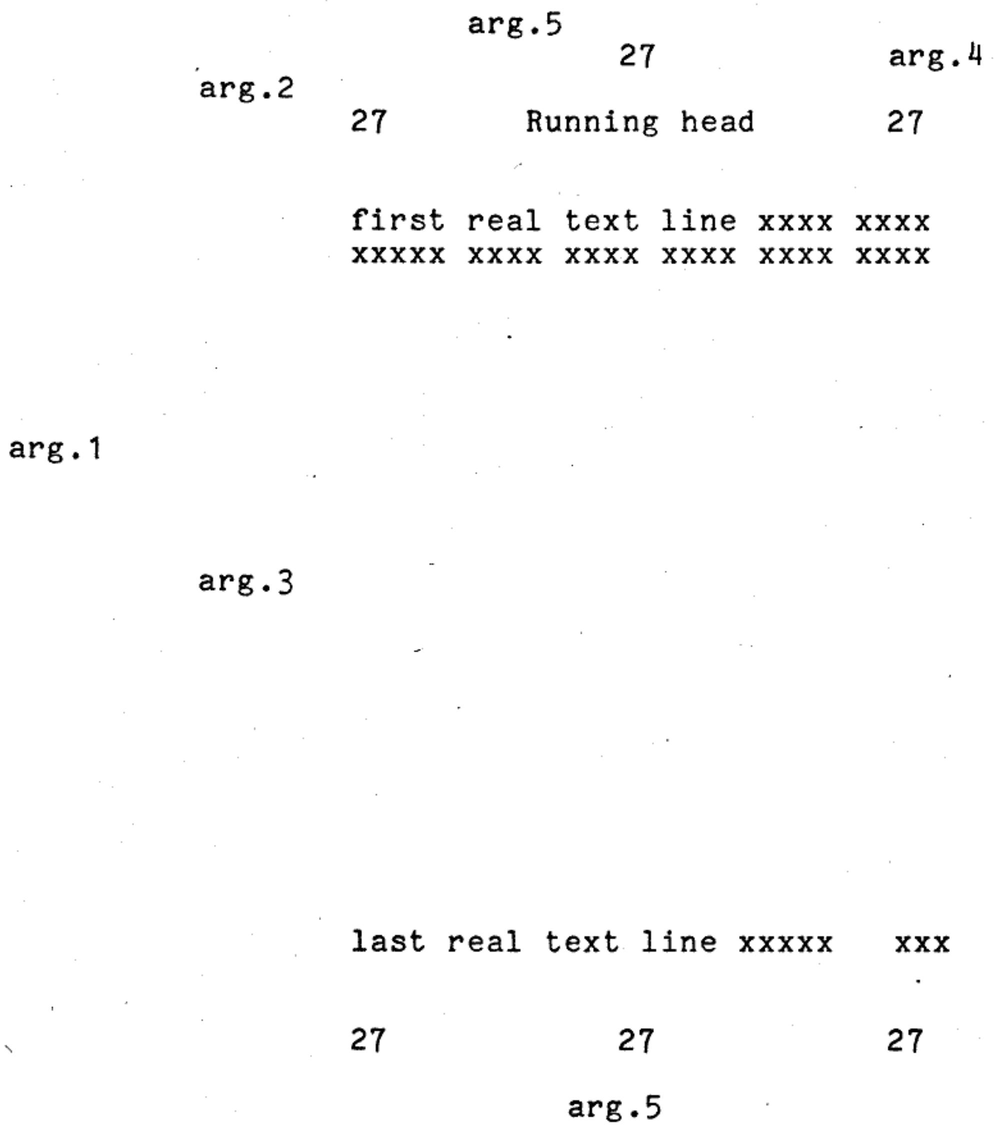

arg.5
27
arg.4
arg.2
27
Running head
27
first real text line xxxx xxxx
xxxxx xxxx xxxx xxxx xxxx xxxx
arg.1
arg.3
last real text line xxxxx xxx
27
27
27
arg.5

Figure 1. Page layout. The arguments refer to the page layout command (PL)

5 above and at right margin.

Right margin is defined as the current left margin plus the current line width.
The parameter <page number> is assigned to the page number.

Standard:

<position>	5
<page number>	1

Page shift command

PS <arg1><arg2>

Orders a page shift with a specified distance from upper page limit to the first real text line. The page shift may be conditional of the distance from current line to last real text line.

<arg1>	distance from upper page limit to the first real text line.
<arg2>	minimum distance left on page before PS is performed

An argument can be omitted, but if <arg1> is present the comma cannot be omitted. The standard values are
<arg1> 50
<arg2> 0

Important:
The input to the composing program always must begin with a command group, which at least contains the page shift command.
It is supposed that the position of the paper in the start of a run is just at page limit.

The command causes the following points (only point e. and f. in the case of the initialisation call mentioned above) to be executed:

a. places possible footnotes.
b. places possible page number.
c. generates linefeed on to page limit and a stopcode for correction of the paper position if this is necessary for the

selected typesetting machine.
d. increases the page number by 1.
e. places possible page number and running head.
f. generates linefeed on to first real text line according to <argument>.

If you want to place a running head or a page number at the top of the first page you consequently have to use the corresponding commands before the first page shift command.

Quadding right command

QR

Transition to the quadding right mode.
The text on the following lines is placed at the current right margin, leaving the left end ragged. All characters are significant.
On certain conditions mentioned under command CT a line shift is generated.

Running head command

RH <font>,<string>

Specifies a running head <string> to be placed on the following pages with font no. <font>.
<string> is centered with respect to the left hardware margin and the current right margin.
<string> must only contain 2 new line character (3 lines).

The parameters can be omitted, for example,
RH
specifying an empty running head.
Standard: no running head.

Restore justification command

RJ

Transition to the justifying mode.
The text is justified with respect to the current left margin and linewidth by means of varying the space room between the words inside certain limits (see appendix for the selected typesetting machine). If this justification is impossible a hyphenation takes place.
A word delimiter consists of one or more of the following characters:

a) space (ISO-value 32)
b) new line (ISO-value 10)
c) form feed (ISO-value 12)
d) underline, if not part of a composite character (ISO-value 95)
e) command.

The calculation of the space width is independent of the composition of the delimiter, for instance 1 space has the same effect on the justification as 3 spaces followed by 2 new line characters.
If an underline is part of the delimiter the whole space will be underlined.
<hyphen><new line> and <hyphen><new line><delimiter> is blind inside a word, that is hyphenation in the source text is allowed.
Under certain conditions mentioned under command CT a lineshift is generated.

Rub-out command

RO

Deletes all characters back to and including the last new line character.
The command is only legal as input to the input program and not to the composing program.

Substitute command

SB <character>,<symbol number>

Substitutes in the following text <character> by the special symbol no. <symbol number> in the socalled special Symbol Table (see appendix for the selected typesetting machine).
All visible ISO-characters, the current separator, + and - excepted, are allowed as <character>.

SB <symbol number>

releases a possible character used as special symbol no. <symbol number>, that is assigns the normal value to the character.
It is not necessary to release a character, when you want to change the special symbol to be substituted by the character, or when the special symbol is used before, that is
SB £,7 *SB $,8* ...text...
SB £,8
will release the plus character and in the following text substitute the minus character by the special symbol no. 8.

Stopcode command

SC

Generates a stopcode according to the selected typesetting machine.

Separator command

SE <separator>

Specifies the separator signifying the start and the end of a command in the following source text according to the definitions prefacing this appendix.
Underline, overline and hyphen are not allowed as <separator>.
Standard: *

Suppress justification command

SJ

Transition to the non-justifying mode.
With respect to the current left margin the following text is copied from the input to the output media with no alteration in the typographical look, that is all characters are significant. Under certain conditions mentioned under command CT a line shift is generated.

Subscript leading

SL <subscript leading>

Specifies the distance in points between current line and sub/superscript. (1 point = 0.353 mm).
Standard: 6

Tabulating command

TA <argument>,<argument>,<argument>....

Transition to the tabulating mode.
Places up to 24 horizontal tabulator marks. The first argument specifies the distance from the current left margin to the first mark, the following arguments specify the distance from the preceding mark.
The text is tabulated in the usual way, that is for every horizontal tabulation character (<HT>, ISO-value 9) met spaces are generated up to the next mark.
All characters are significant.
The width of a space in this mode is assigned to the width of a number, which is useful when you want to place numbers with a different number of digits up to a right edge.
The <HT> character is blind outside the tabulating mode.
Under certain conditions mentioned under command CT a line shift is generated.

<u>Type size command</u>

```
TS <size>
```

Orders a shift to type size no. <size>.
Standard: 1.

Appenxid D. List of Typol commands.

command	parameter	standard
CM ComMent	string	
CT CenTring mode		
DS Diablo Separator	char	
EF End of File		
FG FiGure height	height (mm)	
FN FootNote	font no., string	
FT FonT	font no.	1
LD LeaDing	leading (points)	
LM Left Margin	margin (mm)	0
LW Line Width	width (mm)	140
MT Margin Text	font no., string	
NL New Line	line feed	1
NP New Paragraph	indentation (mm)	5
NS New Section	font no.,linefeed,string	
PL Page Layout	vt1,vt2,...,vt5(mm)	297,30,235,18,10
PN Page Shift	vt2, vt6 (mm)	50,0
QR Quadding Right		
RH Running Head	font no., string	
RJ Restore Justifying		
RO Rub Out		
SB SuBstitute	char, symbol no.	
SC Stop Code		
SE SEparator	char	*
SJ Suppress Justifying		
SL Subscript Leading	leading (points)	6
TA TAbulating mode	ht 1,2, ... ,24 (mm)	
TS Type Size	size	1

vt 1, page length
vt 2, distance from upper page limit to first real text line
vt 3, distance from first to last real text line
vt 4, distance from upper page limit to first line in RH
vt 5, distance from upper page limit to a centered page no. above or
distance from lower page limit to a centered page no. below
depending on position type
vt 6, minimum distance left on page before PS is performed

position = 0, no page numbering
position = 1, page no. below and centered
position = 2, page no. above and centered
position = 3, page no. below at left if even and at right if odd
position = 4, page no. above at left if even and at right if odd
position = 5, page no. above at right margin

symbol no.	Line printer	Diablo
1	%	£
2	?	$
3	"	%
4	å	&
5	Å	@
6	space	space
7		^
8		~

Appendix E. ERROR MESSAGES FROM THE INPUT PROGRAM

The error messages are output to either

a. the text file, where they appear after the line containing the error in the form

   * <error message>

or, if there is no text file,

b. on current output as

   * <line number> <error message>.

xx BS in typol command
: xx = any typol command
  Data for typol commands which is not text, may not contain composed graphics. The composed graphic is ignored when checking the command.

command not first input
: The input data does not start with a command.

PS command not in group
: There is no PS command in the first command group.

illegal character
: One or more characters which are invalid, according to the input table, have been found in the line.
  An illegal character is replaced by ! (ISO value 33, exclamation mark) in the output data.

xx invalid command
: xx = RO
  a. more than one RO command in line.
  b. invalid data in command.
  c. command occurs after 126 characters have been received in the line.
  The command is ignored.

xx invalid data

xx = any typol command
Data which is invalid has been supplied. The typol command specification defines the data types for each command.
The data is ignored.

line too long

A line has occured containing composed characters which may cause that the maximum number of characters transferred at one time is exceeded. Since the preparation of composed graphics is not complete some characters may be erroneous.

no EF command

The input has been exhausted before an EF command has been received.

xx too many HT's in preline

xx = RJ!SJ!CT!QR!TA!PS!EF!FG!empty
In tabulating mode, if a line change character is recognised (NL or FF), (xx empty), or a command is received which generates a line change, a check is made that the number of HT characters since the last line change does not exceed the number specified in the relevant tabulating command.
If the number is exceeded the above error message will appear.

xx too many lines in text

xx = RH!MT!NS
The maximum number of lines in <text> has been exceeded.
For RH the maximum is 2, and for MT and NS it is 1.

xx unknown command

A separator has been recognised but the two following characters, xx, do not constitute a known command.
The following is skipped until the next separator character.

Appendix F. ERROR MESSAGES FROM COMPOSING PROGRAM

All the error messages have the form:

```
***typol <error message> page <page no.> line <line no.>
```

where the page and line number refer to the proofreading file. Current output is used.

no EF command
: The input has been exhausted before an EF command has been received. The run is terminated

SB invalide data
: Data which is invalide has been supplied the SB command. The run is terminated.

to much composite char.
: More than 3-double composite character. The run is terminated.

invalide data
: Diablo separator not followed by u, d, h, or l Or character after <separator>h or <separator>l illegal character class.

char. in footnotes
: No. of characters in the footnotes on the page exceeds 400. The run is terminated.

char. in line
: No. of characters in the line exceeds 300. The run is terminated.

char. in margin text
: No. of characters in margin text exceeds 300. The run is terminated.

char. new section text

No. of characters in new section text exceeds 300.
The run is terminated.

char in running head

No. of characters in running head exceeds 150.
The run is terminated.

char. in word

No. of characters in the word that can be saved from one page to the next exceeds 75.
The run is terminated.

double margin text

It is tried to place a margin text two times in the same line.
The run is terminated.

hyph. file not correct

The hyphenation file does not contain information corresponding to the actual source file.
The run is terminated.

justification impossible

Line width exceeded by the first or the second word in the line and hyph. routine gives up .
The run is terminated.

last tab. mark exceeded

A HT-character is met after the last tabulator mark on the line is passed.
No spaces are generated and program continues.

left margin

The left margin is ordered outside the current right margin
The run is terminated.

linewidth exceeded	The current right margin is exceeded in non-justifying, tabulating, centering or quadding right mode. The whole line is printed and program continues.
number of footnotes	More than 6 footnotes on the page. The run is terminated.
number of words in line	More than 23 words in the line. The run is terminated.
unknown command	Unknown command met. This message does not occur if the source file is an accepted output from the input syntax checking program. The run is terminated.
width of margin text	The width of margin text exceeds the margin. Only the first part of the text is printed and program continues.

Appendix G. Diablo 1620 control program

The aim of this program is to copy a backing storage file to a Diablo 1620 terminal. The file must be an object file from the composing program (see. 3.2.3), and the selected typesetting machine must be Diablo 1620.

The copying may be restricted to a specified number of pages from the object file.

The program must be running in an online process, with the Diablo as current input and current output.

Before copying starts and after each page is copied to the Diablo, the copying is stopped and program waits for operator intervention. If the operator types LF on the Diablo the next page is copied. If a @ is typed, the program is terminated.

Program call:

```
diablo <bs file>  <first>.<last> 1
                                 0

   <first>
              ::=  <integer>
   <last>

<bs file> ::= <name>
```

Semantic:

<bs file>	Specify the file to be copied to the Diablo. The file must be objecy file from the composing program.
<first>.<last>	Specify the first and last page to be copied.

Error Messages:

```
***diablo end. connect
               The object file could not be connected.
               The program is terminated.

***diablo param <invalide parameter>
               Invalide parameter in program call.
               The program is terminated.
```

Appendix H. LINE PRINTER

The Line Printer is in this context a typesetting machine connected to the RC8000 with 1 font, 1 type size and 2 leading values, 12 and 24 point.
The characters all have the same width:
1 unit = 2.54 mm.
The maximum line width is 335 mm.
The following standard values, dependent upon the typesetting machine, are used:

leading :	12 point	(to change by typol)
minimum allowed word space:	1 unit	(to change in program)
maximum allowed word space:	2 unit	(to change in program).

C h a r a c t e r s e t

```
a b c d e f g h i j k l m n o p q r s t u v w x y z æ ø å
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z Æ Ø Å
0 1 2 3 4 5 6 7 8 9
! " % & ' ( ) * + , - . / : ; < = > ? _
```
space and new line.

S p e c i a l S y m b o l T a b l e

parameter used in command SB	Line Printer character
1	%
2	?
3	"
4	å
5	Å
6	space

Appendix I. Diablo 1620

The Diablo 1620 is in this context a typesetting machine connected to the RC8000 with 1 font, 1 type size and leading automatically controlled from 3 to 30 point in 3-point increments.
The characters all have the same width of 6 units.
1 unit = 0.4633 mm
The maximum line width is 325 mm.
The following standard values, dependent upon the typesetting machine, are used

leading :	12 point	(to change by typol)
maximum allowed word space:	1 unit	(to change in program)
maximum allowed word space:	2 unit	(to change in program).

Character set

a b c d e f g h i j k l m n o p q r s t u v w x y z æ ø å
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z Æ Ø Å
0 1 2 3 4 5 6 7 8 9
! "£ $ % & ' ( ) * + , - . / : ; < = > ? @ _ ^ ~
space and new line.

Special Symbol table

parameter used in command SB	Diablo 1620 characters
1	£
2	$
3	%
4	&
5	@
6	space
7	^
8	~

Appendix J.

References:

1. Utility Programs I
by Hans Rischel
RCSL No: 31-D364

2. Utility Programs II
by Tove Ann Aris
RCSL No: 31-D422

3. Algol 6, User's Manual
by Hans Dinsen Hansen
RCSL No: 31-D322