Book 3 Reference

SCRIPT/PC

Productivity Series

IBM

IBM Software
for IBM Personal
Computers

Personal Computer
Productivity Series

SCRIPT/PC

Book 3 Reference

Program by: *Robert N. Seidel*
Charles W. Gainey Jr.

First Edition (January, 1984)

The following paragraph does not apply to the United Kingdom or any country where such provisions are contrary to local law: International Business Machines Corporation provides this manual "as is", without warranty of any kind, either expressed or implied, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this manual at any time and without notice.

This publication contains examples of data markup and text. All names of individuals, companies, brands, and products used in these examples are fictitious, and any similarity to the names, addresses or products used by an actual business is entirely coincidental.

This product could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of this publication.

Products are not stocked at the address below. Requests for copies of this product and for technical information about the system should be made to your authorized IBM Personal Computer dealer.

A Reader's Comment Form is provided at the back of this publication. If the form has been removed, address comments to: IBM Corp; Personal Computer, P.O. Box 1328-C, Boca Raton, Florida 33432. IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligations whatever.

About This Book

How to Use This Book

This book is the detailed *Reference* to SCRIPT/PC Control Words and the Generalized Markup Language (GML) Starter Set of tags. Other books supplied with SCRIPT/PC are *Book 1 How to Use* and *Book 2 Quick Reference.*

The information in this manual is organized in the following manner:

- The first chapter gives you a detailed explanation of each SCRIPT/PC control word, its options, and a detailed description of each. Examples have been added to supplement explanations and to aid you in deciding how to use the control word.

- The second chapter gives you an equivalent detailed explanation of every tag supported by the SCRIPT/PC Starter Set of GML tags. Examples have also been added to assist you.

The first two chapters are the main reference sections for the control words and tags that you will use to markup your document.

- The third chapter contains information on symbols and how they relate to the control words of SCRIPT/PC.

- The appendixes contain:

1. Error messages, their meaning, and any corrective action you have to take.

2. Tables for quick lookup of each control word or Starter Set tag.

3. Comparison information between SCRIPT/PC and SCRIPT/VS. SCRIPT/VS runs on large computer systems (called host systems) under complex operating systems such as VM/SP and OS/MVS.

- A glossary of terms has been added to assist you in learning some of the more common terms associated with text processing.

- An index of terms is also provided.

Related Books

We assume that you are familiar with the operation of the IBM Personal Computer and the Disk Operating System (DOS). DOS is described in the following publications:

- IBM Personal Computer *Guide to Operations*
- IBM Personal Computer *Disk Operating System*

This product is a compatible subset of the IBM Document Composition Facility (DCF); that is, SCRIPT/PC control words and GML tags operate the same way on the IBM Personal Computer or on a larger computer system that uses the Document Composition Facility. For detailed knowledge at this level refer to the following publications:

- IBM *Document Composition Facility: User's Guide;* order number, SH20-9161

- IBM *Document Composition Facility Introduction to the Generalized Markup Language: Using the Starter Set;* order number, SH20-9186

For additional Information on transferring files between personal computers or between a personal computer and a large host system, refer to the IBM Personal Computer *Asynchronous Communications Support* book.

To create a backup copy of this product, go to the IBM Personal Computer *Guide to Operations* book and the chapter describing DOS.

Contents

Chapter 1. Understanding Control Words

Contents

Introduction to Control Words

This chapter describes each descriptive control word in the SCRIPT/PC language. Usage notes and examples have been included for learning assistance.

Control Words

All control words are identified by a two-character name. This name always follows a period (.) located in the first character position of an input line. As an example, the skip-a-line control word name is **.SK** and will be recognized by SCRIPT/PC when it is written as:

```
.SK
```

Most control words accept additional information, called *options* or parameters. In the case of the skip command, if you want to skip five lines, you would add the *value* option and write the command as:

```
.SK 5
```

> **Note:** Options used with tags are known as *attributes.*

Control Word Separator

You can write more than one control word on a line by separating the control words with a semicolon (;). This allows you to conserve space in your input file and also to place a control word within a line of text. If you followed the previous skip control word with the format control word **(.FO ON)**, you have the option of writing them two ways. Either on two lines as:

```
.SK 5
.FO ON
```

or, if you use the control word separator (;), as:

```
.SK 5;.FO ON
```

SCRIPT/PC will interpret both examples the same way.

To use the control word separator within a line of text, our example must be changed. If you are writing more than one control word on an editorial line, the control word separator may be placed next to the control word, without an intervening space.

```
.SK 5;.FO ON
This is an example of a
very;.US ON;important;.US OFF;function.
|     |     |         |       |
|     |     |         |       Text
|     |     |         Underscore OFF
|     |     Text to be underscored
|     Underscore ON
Text
```

The input lines will be read by SCRIPT/PC and the formatted output results will be:

```
This is an example of a very important function.
```

The following is an example of the correct way to write control words in your input file.

```
Column => 1   5   10  15   20
          .XX [value]
          .XX [value];.XX [option]
          text;.XX [value];text
```

Control Word Modifier

You can stop SCRIPT/PC from searching for a control word separator within an input line by use of the control word modifier, which is a single quotation mark (') following the control word period:

```
.'CE Say "Hello"; at least smile."
```

Any control word containing the control word modifier must be the last control word on a line as all control word separator characters are ignored and used as text.

Notes:

1. Control words and tags cannot be mixed on the same input line.

2. If an input line starts with a period (.), the entire line is considered to contain only control words, their options and text. Any colons (:) and associated tags will be treated as text.

3. When some control words are written without an option being selected, a default value is assigned by SCRIPT/PC. In the case of the Space and Skip control words, the value is 1.

How to Read the Format of a Control Word Description

Each control word has a specific way it must be written. The added information (options) that can be attached to the control word also must be written correctly. It is the purpose of this reference section to show you the correct way of writing the control words and their options.

Each control word will be written in a standard format using the following rules, called syntax conventions.

1. To assist in readability, the control words will be shown in UPPERCASE letters along with their options. However, you may use upper- or lowercase characters when you write the control words and their options. Any exceptions to these rules will be pointed out in the notes for an individual control word.

2. Additional options information, for which you must supply the value, are shown in lower case.

3. If a value is inserted by SCRIPT/PC when you do not choose to enter a value, that default value will be brought to your attention in the description of the control word. Sometimes a control word with more than one optional item will not have default values inserted for each item. In this case, you must be very careful to insert values for those items that *do not* have defaults.

4. An optional item (parameter) that *does not* have to be entered at all times is shown in small brackets: **[like this]** . When you see an item in small brackets, you may ignore it if you wish.

5. When a control word will accept one optional item from a list of items, or even allow you to ignore selecting any item from the list of items, the list will enclose those optional items within large brackets. For example, the box control word is formatted as:

```
.BX  [ d1 ... dn   ]
     [ OFF         ]
     [ CHAR cname  ]
```

6. A list of items in which at least one item *must* be entered encloses each item in the list with braces: **{like this}**. For example, the conditional section control word is formatted as:

```
.CS  n    { ON      }
          { OFF     }
          { INCLUDE }
          { IGNORE  }
```

7. Many of the optional items have a vertical, horizontal, or numeric value. They will normally be written as *v, h,* and *n,* where:

```
v    Specifies a vertical value
h    Specifies a horizontal value
n    Specifies a numeric value
```

Format of Control Words

The control words are listed in alphabetical order on the following pages.

And - Conditional .AN Control Word

Purpose: Use the And [.AN] conditional control word with the If [.IF] control word to process SCRIPT/PC input lines conditionally. The result of the test performed is logically And'd to the result of the most recently performed If, And, or Or control word to determine whether the target should be processed.

Format:

```
.AN comparand1 test comparand2 [target]
```

Remarks:

comparand1 Is any string of characters to be used as the first compared value. This comparand may be the value of a set symbol.

comparand2 Is any string of characters to be used as the second compared value. It too may be the value of a set symbol.

test Is a 1- or 2-character code that tells SCRIPT/PC what type of comparison to make between the two comparands:

The list of tests and the formats in which they may be written are as follows:

And - Conditional .AN Control Word

Test	Form 1	Form 2
Equal	EQ	=
Not equal	NE	<>
Greater than	GT	>
Less than	LT	<
Greater than or equal to	GE	>=
Less than or equal to	LE	<=

target Is any valid SCRIPT/PC input line (control word or text). If this condition and the most recently performed If, And, or Or are both true, the target line is processed next (the first non- blank character after the AND is treated as the first position of the subject line. Otherwise, the target line is ignored, and processing continues with the input line that follows the IF control line.

- The And and Or control words, used with the If, Then, and Else control words allow you to construct complex logic statements.

- The And control word itself does not cause a break. However the target control word might if it is processed.

- Each comparand can be up to 255 characters in length. The shorter comparand will be extended to the same length as the longer comparand (using trailing blanks).

And - Conditional .AN Control Word

- Even when symbol substitution is off [.SU OFF] and the And comparand is found, all valid symbols in the comparands will be resolved before the comparison is made.

Examples:

- If the following symbols are defined:

```
.SE A = ''
.SE B = 'Fred'
.SE C = 'Fred'
.SE D = ''
```

  Then, **&A.&C = &B.&D**; however, **&A <> &B** or **&C <> &D.**

- The following results in typing **Big Apple Country** if both results are *true*:

```
.IF &A = 'New York'
.AN &B = 'New Jersey' .TY 'Big Apple Country'
```

Your Notes:

Append .AP Control Word

Purpose: The Append [.AP] control word is used to add an additional SCRIPT/PC file as a continuation of your current file. Files can be *chained together* by having one file append the next. Only the end of the *chain* is the last file.

Format:

```
.AP  [d:]filename[.ext]
```

Remarks:

d: Is the drive specifier (A: B: C: D: E: F: G: H:) if the file you want to add to the original file is not on the DOS selected default drive.

filename Is a 1- to 8-character filename. It is the name of the input file you want to append.

.ext Is the optional file extension. SCRIPT/PC will assume a default extension of **SCT** if this item is omitted.

- When the Append control word is found, your current file is ended, and the file that you named in the Append control word is added as a continuation of your current file. Any text or control words in your current file that happen to be located on lines after the Append control word are ignored.

- SCRIPT/PC will *not* return to the original file that contained the Append control word (unlike

Append .AP Control Word

the Imbed control word [.IM] which resumes formatting your current file after completing the imbedded file).

Examples:

- The following example:

```
.AP ABC
```

  results in the original input file being closed when the Append control word is found and the contents of the file **ABC.SCT** being added immediately (no page eject is forced).

- The following example is located on a drive other than the DOS default drive.

```
.AP B:ABC
```

Your Notes:

Begin Font .BF Control Word

Purpose: The Begin Font [.BF] control word is used to save the current font and then begin to use a new font.

Format:

```
.BF  font-id
```

Remarks:

font-id Specifies the name of a font previously created by the Define Font [.DF] control word.

- The input text that follows the begin font control word is formatted using the new font.
- You can change the font within a single output line when formatting is on (.FO ON or .FO LEFT).
- This control word does not cause a formatting change called a break.
- Use the Previous Font [.PF] control word to return to the font that was active prior to SCRIPT/PC finding the begin font control word.

Examples:

- The following example selects the font-id named "UNDERCAP."

```
.BF UNDERCAP
```

Begin Font .BF Control Word

- To select another font use the Begin Font control word again:

```
.BF UNDERCAP
.
.
.
.BF UPPER
```

Your Notes:

Bottom Margin .BM Control Word

Purpose: The Bottom Margin [.BM] control word is used to set the amount of space that should be reserved at the bottom of output pages.

Format:

```
.BM  v
```

Remarks:

v Is the number of lines to be reserved at the bottom of output pages. This item cannot be omitted by the user.

Initial Setting: 6 Default: 0

- The Bottom Margin is the space that is reserved at the bottom of each page that can contain the running title (set by the Running Title [.RT] control word).
- The value set by the Bottom Margin control word takes effect immediately. It will stay at the new value until another Bottom Margin control word is read.
- The size of the bottom margin is not affected by line spacing, either Single or Double spacing.
- You may reserve as many lines as you wish, as long as the Bottom Margin plus the Top Margin is not greater than the Page Length.

Bottom Margin .BM Control Word

Examples:

- The following creates 1/2 (.5) inch of space at the bottom of the page at 6 lines per inch:

```
.BM 3
```

- To create 1 inch of space at the bottom of the page at 8 lines per inch:

```
.BM 8
```

Your Notes:

Box .BX
Control Word

Purpose: The Box [.BX] control word is used to create box specifications. The box can overlay other text to create tables, charts, or other graphic types of information. The Box control word defines and starts horizontal lines for SCRIPT/PC output. It also defines vertical lines for following output lines.

Format:

```
.BX [ d1 ... dn ]
    [ OFF       ]
    [ CAN       ]
    [ CHAR cname ]
```

Remarks:

d1 ... dn Specifies the start and stop column for a box's horizontal line (—). It also defines any vertical lines (|) within the box.

OFF Specifies the ending (bottom) of the box. All vertical lines are stopped and the box is closed with a horizontal line.

CAN Specifies the ending (bottom) of the box. All vertical lines are stopped and the box is ended *without* a horizontal line.

CHAR Allows you to change the box character if you have a matrix printer. If this item is omitted, it is assumed that you have a graphics printer and the GRAPHICS option is selected.

Box .BX
Control Word

cname Is the name of the box character set SCRIPT/PC is to use for drawing all subsequent boxes. The valid names are:

GRAPHICS Specifies the box character set for a graphics printer.

MATRIX Specifies the alternate box character set for a matrix printer.

Default: GRAPHICS

- If a box is started, the Box control word by itself (without *any* options) can be used to create a horizontal line within the box.
- You can use the space control word [.SP] or input text to continue drawing the vertical lines once a box is started.
- This control word causes a formatting change called a break.

Examples:

These examples show you how to draw a calendar.

1. Let's draw a simple box first:

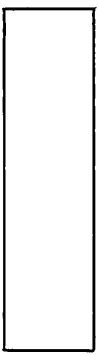
```
.BX 5 40
.SP 2
.BX OFF
```

This results in:

2. Let's rotate the box 90 degrees:

```
.BX 5 10
.SP 10
.BX OFF
```

results in:

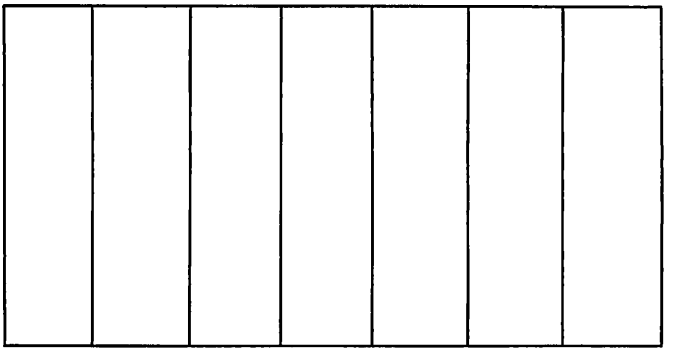


3. Let's add six more boxes to the first one:

```
.BX 5 10 15 25 30 35 40
.SP 10
.BX OFF
```

This results in:

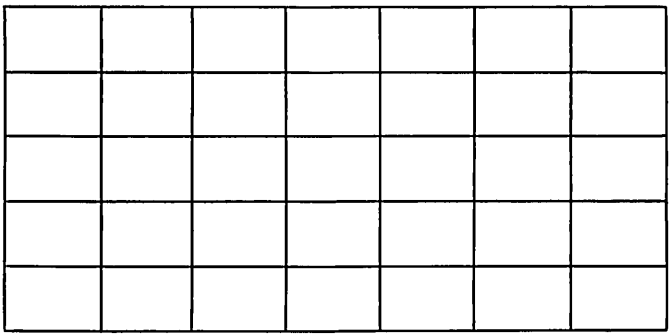
Box .BX
Control Word



4. Now let's make squares within the boxes:

```
.BX 5 10 15 20 25 30 35 40
.SP
.BX
.SP
.BX
.SP
.BX
.SP
.BX
.SP
.BX OFF
```

This results in:

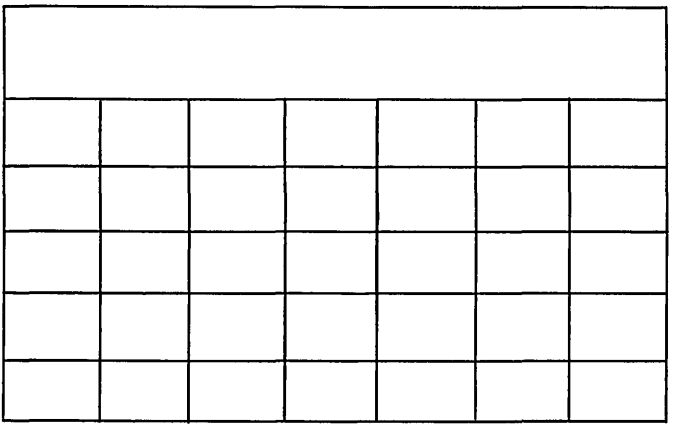


5. Let's go on to place the first box on top of the squares:

```
.BX 5 40
.SP
.BX 5 10 15 20 25 30 35 40
.SP
.BX
.SP
.BX
.SP
.BX
.SP
.BX
.SP
.BX OFF
```

This results in:

Box .BX
Control Word



6. Now that you know how to code the boxes, let's make our example into something more practical – a calendar. All you have to do is insert text in the form of month, year and daily dates into the correct print positions in the file.

   Note: To insert text into the box, you should be careful about counting spaces or you will place your text in the wrong area. The boxes that have been developed are really an overlay for the text. All of the dates have been spaced so that they will fall within the boxes at the correct print position.

```
.BX 5 40
   JUNE                                  1983
  Sun  Mon  Tue  Wed  Thu  Fri  Sat
.BX 5 10 15 20 25 30 35 40
                    1    2    3    4
.BX
    5    6    7    8    9   10   11
.BX
   12   13   14   15   16   17   18
.BX
   19   20   21   22   23   24   25
.BX
   26   27   28   29   30
.BX OFF
```

This results in:

JUNE Sun	Mon	Tue	Wed	Thu	Fri	1983 Sat
			1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30		

7. Another method to insert the text into the box would be to use the Tab Setting control word. An example of this method is contained in *Book 1 How to Use* in Chapter 4.

Box .BX
Control Word

Your Notes:

Break .BR Control Word

Purpose: Use the Break [.BR] control word to force the next input line to start a new output line.

Format:

```
.BR
```

Remarks:

- The Break control word is necessary only when SCRIPT/PC is gathering or concatenating input lines.
- Many other control words also cause breaks. The break control word is not necessary when one of the other control words is used.
- An input line that starts with a blank character also causes a break.

Examples:

- To force a break with formatting on:

```
.FO ON
For example:
.BR
This is a list of states.
```

results in:

```
For example:
This is a list of states.
```

Break .BR Control Word

- If the Break control word [.BR] was not used the lines would format as:

```
For example: This is a list of states.
```

Your Notes:

Center .CE Control Word

Purpose: The Center [.CE] control word is used to center output lines between the current page margins and indentations.

Format:

```
.CE { ON   }
    { OFF  }
    { line }
```

Remarks:

ON Signals that the following text lines are to be centered.

OFF Stops centering if it was ON.

line Signals that this one line of text is to be centered.

Default: line

- Lines are centered between left and right margins. Any indent and offset values are also used in determining centering. No formatting is done to the lines that are being centered.
- This control word causes a formatting change called a break.
- The first word of *line* should not be ON or OFF.
- If your line is longer than the current column or line length, the line is cut off and an error message is created.

Center .CE Control Word

Examples:

- To center one line of text:

```
.CE Important Information:
```

When this input line has been formatted, the output result will be centered between the margins:

```
          Important Information:
```

- To center several lines of text:

```
.CE ON
John Smith
5555 West Atlantic Avenue
Hartford, Connecticut
11822
.CE OFF
```

Each of the 4 lines between .CE ON and .CE OFF is centered on its own line.

```
                John Smith
          5555 West Atlantic Avenue
            Hartford, Connecticut
                   11822
```

Your Notes:

Column Definition .CD Control Word

Purpose: The Column Definition [.CD] control word is used to define the starting character positions for Multicolumn Mode.

Format:

```
.CD  2  p1  p2
```

Remarks:

2 Specifies that two columns are to be created for output. This value cannot be changed or omitted.

p1 Signals the starting column position for the first column.

p2 Signals the starting column position for the second column.

- The Column Definition control word only *sets up* the specifications for future two-column text. To *start to format* in two-columns, you should use the Multicolumn Mode [.MC] control word.

- The *gutter* space between columns is obtained by setting the column width to a value less than the distance between column starting positions. Use the column width [.CL] control word to set up the text width.

Column Definition .CD Control Word

Example:

- To define the start of printing in column 1 and column 35:

```
.CD 2 1 35
```

- To print 30 characters and leave a 5-character gutter between columns, add the column width you want:

```
.CD 2 1 35
.CL 30
```

- To start printing in two-column mode, add the multicolumn control word that starts it:

```
.CD 2 1 35
.CL 30
.MC
```

- To return to a one-column mode of operation, use the single-column control word:

```
.CD 2 1 35
.CL 30
.MC
  . . .
.SC
```

  Column width will return to the previous line length value, and the column definition and column width for multicolumn mode will be saved, in case you want to go back to the same multicolumn mode later on in your document.

Your Notes:

Column Width .CL Control Word

Purpose: The Column Width [.CL] control word sets the width of each column of SCRIPT output in multicolumn (2-column) mode.

Format:

```
.CL   [ h  ]
      [+h  ]
      [-h  ]
```

Remarks:

h Signals the width of each column of formatted text.

Initial Setting: Same as Line Length.

Default: Restores same width as Line Length.

- The Column Width control word should be used with the Column Definition [.CD] control word to define the width of each column of text in a two-column format.
- You may use the **+h** or **-h** to add to or subtract from the current setting of the column width.
- Column width does not affect page margins, top and bottom margins, and running headings and titles.
- If you do not set the column width, it has the same value as the line length.

Column Width .CL Control Word

- This control word will cause a formatting change called a break.

Example:

- To set your column width during multicolumn mode to 25 characters:

```
.CL 25
```

- To change the previous column width from 25 to 30:

```
.CL 25
.
.
.
.CL +5
```

Your Notes:

Comment .CM Control Word

Purpose: Use the Comment [.CM] control word to place comments. The comments will not appear in your output.

Format:

```
.CM  [comments]
```

Remarks:

comments May be any character, text or numeric data. This input line is scanned for a control word separator, the semicolon (;), which is the signal that the comment has ended.

- The Comment control word allows comments to be stored in an input (source) file for future reference. Any comment line is ignored when formatting. However, the comment can be seen whenever the file is edited. It is a very convenient way for you to add reminder notes to an input file.

- An alternative control word whose function is the same as **.CM** is the .* control word. The only difference is that the **.CM** control word is only active until a control word separator is found on the same line. The .* control word treats all characters on the same line as a comment.

Comment .CM Control Word

Example:

```
.FO OFF
When a telephone rings,
.CM This file created on 3/24/83
.CM by John Smith;RING! RING! RING!
.CM *** Run on 8.5 X 11 inch paper ***
It is best to answer it!
```

This results in an output of:

```
When a telephone rings,
RING! RING! RING!
It is best to answer it!
```

You will see the comment lines when you edit your input (source) file. However, you will not see them print on the output device. The comments in this example tell you who originally created the file. It also lists the type of paper to be used. When your input file is run, the words **RING! RING! RING!** are considered to be part of your input text (not comments) and will be printed on your output device.

Your Notes:

Comment .*
Control Word

Purpose: Use the Comment [.*] control word to place comments. The comments will not appear in your output.

Format:

```
.*  [comments]
```

Remarks:

comments May be any character, text or numeric data. This input line is completely ignored by SCRIPT/PC.

- The Comment control word allows comments to be stored in an input (source) file for future reference. Any comment line is ignored when formatting. However, the comment can be seen whenever the file is edited. It is a very convenient way for you to add reminder notes to an input file.

- An alternative command whose function is the same as .* is the **.CM** command. The only difference is that the **.CM** control word is only active until a control word separator is found on the same line. The **.*** control word treats all characters on the same line as a comment.

- If you use comments in your tag (macro) definitions, this type of comment lines will not take any space when the macro library is read into memory. This allows you to write extensive comments in your macro definitions.

Comment .*
Control Word

Example:

```
.FO OFF
When a telephone rings,
.* This file created on 3/24/83
.* by John Smith;RING! RING! RING!
.* *** Run on 8.5 X 11 inch paper ***
It is best to answer it!
```

This results in an output of:

```
When a telephone rings,
It is best to answer it!
```

You will see the comment lines when you edit your input (source) file. However, the comment lines will not be a part of your printed output. In the example given, the comments tell you who originally created the file. It also lists the type of paper to be used. When your input file is run, the words **RING! RING! RING!** are considered to be part of your comment and do not appear as part of your output, unless you use .CM instead.

Your Notes:

Conditional Page Eject .CP Control Word

Purpose: The Conditional Page Eject [.CP] control word causes a page eject to occur if fewer lines than necessary remain on the current page.

Format:

```
.CP  [ v ]
```

Remarks:

v Is the amount of vertical space (lines) that should be left on the page before a page eject occurs. If this value is omitted by the user, then a break and page eject is forced unless the current line is already the top text line of a new page (not a title).

Default: Causes a page eject unless there is no text on the current page.

- The Conditional Page Eject control word can be used to guarantee that enough space is left on the page for your following text that should be kept together or to allow enough lines for a figure to be inserted. For example, **.CP 15.**
- The Conditional Page Eject control word written without a value (**.CP**) causes an unconditional page eject to the top of the next page, if not at the top of the page already.
- This control word always causes a formatting change, called a break.

Conditional Page Eject .CP Control Word

Examples:

- To check for 2 inches of space within a page:

```
.CP 12
```

  If there are less than 12 lines of space (2 inches at 6 lines per inch) remaining, a page eject occurs. If there is at least 2 inches of space remaining, then formatting continues with the next control word.

- The following shows how you might use **.CP** in an example application:

```
.CP 12
The following is a business card format:
.FO OFF
.SP
.BX 1 38
.SP
                              XYZ Corp.
.SP
         Joseph Anderson
         Company Representative
         3579 Success Boulevard
         Anaheim, CA  13579
.SP
.BX OFF
```

  The text and business card are printed either within the page or at the top of the next page, depending on the number of lines still left on the page:

Conditional Page Eject .CP Control Word

The following is a business card format:

```
                                XYZ Corp.

        Joseph Anderson
        Company Representative
        3579 Success Boulevard
        Anaheim, CA  13579
```

Your Notes:

Conditional Section .CS Control Word

Purpose: The Conditional Section [.CS] control word is used to allow you to mark certain sections of your input text so that they may be processed conditionally or totally ignored.

Format:

```
.CS    n  { ON }
          { OFF }
          { INCLUDE }
          { IGNORE }
```

Remarks:

n Is the conditional text section number, which should be a number from 1 through 9.

ON Marks the beginning of the conditional section.

OFF Marks the end of the conditional section.

INCLUDE Tells SCRIPT/PC to process all the input lines between the ON and OFF for the conditional text section number, represented by *n*.

IGNORE Tells SCRIPT/PC not to process the input lines located between ON and OFF for the conditional text section number, represented by *n*.

- The Conditional Section control word allows you to mark certain sections of your input file to be included or ignored when SCRIPT/PC formats

that input file. You may have up to 9 unique numeric names (1-9) to create separate section codes. Each section code can be used to identify many sections, if you wish. The ON or OFF items mark the beginning and end of these sections, where the INCLUDE and IGNORE items tell SCRIPT/PC whether to process the section at all.

- Since the Conditional Section control word does not cause a formatting change or break, you can turn conditional sections ON and OFF within a paragraph or even within a sentence.

- All conditional sections are assumed to be included unless IGNORE is found.

- All input text and control words within an ignored section will not be processed except, of course, the Conditional Section control word that identifies the end of that particular conditional section (For example, .CS *n* OFF).

- You may insert or nest a conditional section within another conditional section. However, the nested section will only be available for processing if the original section is included. If the original section is ignored, all nested conditional sections within the original section will also be ignored.

- Conditional sections lend themselves well to being set up using the Read Variable [.RV] control word that allows you to set up an INCLUDE or IGNORE in response to a prompt from the Type on Terminal [.TY] control word.

Conditional Section .CS Control Word

Example:

- The following example sets up two conditional sections:

```
.CS 1 IGNORE
.CS 2 INCLUDE
 .
 .
Dear Customer,
.SP
I am writing this letter to tell you that
.CS 1 ON
we have not received your
last payment. Please send us the money!
.CS 1 OFF
.CS 2 ON
you are our preferred customer and we are
happy to give you a 10% discount on any
item in our catalog over $100.00.
.CS 2 OFF
.SP;.FO OFF
Respectfully Yours,
.SP 2
James Giant
```

  results in the following letter:

```
Dear Customer,

I am writing this letter to tell you that
you are our preferred customer and we are
happy to give  you  a 10% discount on any
item in our catalog over $100.00.

Respectfully Yours,


James Giant
```

- If you now reverse the conditional sections:

Conditional Section .CS Control Word

```
.CS 1 INCLUDE
.CS 2 IGNORE
  .
```

The results of SCRIPT/PC processing the input file would be:

```
Dear Customer,

I am writing this letter to tell you that
we  have  not received your last payment.
Please send us the money!

Respectfully Yours,


James Giant
```

These simple techniques allow you to use one file for a variety of letters or memos.

Your Notes:

Continued Text .CT Control Word

Purpose: The Continued Text [.CT] control word causes the text that follows the control word to be treated as a continuation of the previous text without the insertion of a space character. In normal SCRIPT/PC formatting, a space will be inserted between the last word on an input text line and the first word on the next text input line. This control word will allow you to remove the space between input text lines.

Format:

```
.CT  [ line ]
```

Remarks:

line Is the line to be considered a continuation of the previous text input line. SCRIPT/PC normally does not allow the spanning of two input lines. It will be allowed if the second line starts with the continued text [.CT] control word.

- The Continued Text control word is ignored when formatting has been turned off (for example, after **.FO OFF**). It only works in **.FO ON** and **.FO LEFT** modes.

- If the option *line* is omitted, continuation is ended.

Continued Text .CT Control Word

Example:

```
.FO ON;.LL 40
There is a very easy way
to under;.US ON;.CT score;.US OFF
a partial word.
```

Results in:

There is a very easy way to underscore a partial word.

Your Notes:

Define Character .DC Control Word

Purpose: Use the Define Character [.DC] control word to define various special characters that SCRIPT/PC will recognize as having special significance.

Format:

```
.DC      { PS  }  [ c   ]
                  [ OFF ]
         { CW  }  [ c ]
         { RB  }  [ c ]
         { GML }  [ c ]
```

Remarks:

c Specifies the character to be recognized. It may be any single character.

OFF Causes the character to be undefined and not used. This only applies to PS.

PS Indicates that this definition is for the page number symbol. It may be any character other than blank. This character is used in headings and titles to be replaced by the current page number. It is also the first character of any symbol (as defined by the set symbol control word). If this item is omitted, SCRIPT/PC assumes the character ampersand (&) for this function.

CW Indicates that this definition is for the control word separator character. This character is used to separate more than one control word and text that may be on one input line. If this

item is omitted by the user, SCRIPT/PC assumes the semicolon (;) as a control word separator.

RB Indicates that this character is for the required blank character. Required blanks are not recognized as interword spaces for formatting, but they are translated to ordinary blanks after formatting has been completed. SCRIPT/PC assumes this character to be an underscore unless defined otherwise.

GML Indicates that this is the character used to identify the Generalized Markup Language (GML) tags. SCRIPT/PC assumes this character to be a colon (:) unless defined otherwise.

Default: Restores the initial setting for the specified character.

- The character item cannot be entered as a two-character hexadecimal format.
- Do *not* redefine the control word separator when you are using declarative tags from a macro library or the results will be unpredictable. The control word separators within the library cannot be redefined without changing the library.

Your Notes:

Define Data File-id .DD Control Word

Purpose: Use the Define Data File-id [.DD] control word to set up the filename and extension of an input file that will be used with the End of File [.EF] control word to create unique documents such as labels and form letters.

Format:

```
.DD  { filename }
     { d:filename.ext }
```

Remarks:

d: Is the drive specifier (A: B: C: D: E: F: G: H:) if the file you want is not on the DOS selected default drive.

filename Is a 1- to 8-character filename. It is the name of the input file that contains blocks of data.

.ext Is the optional file extension. SCRIPT/PC will assume a default extension of **SCT** if this item is omitted.

- Each block entry in the file should be separated from the next entry by the End of File [.EF] control word.
- When SCRIPT/PC reads the file, it only reads the current block of text until it finds the next block separator (.EF). It then stops reading the file and remembers where it left off. When the file is imbedded again, SCRIPT/PC starts

reading the text from the point it stopped on the previous operation and continues to read text until the next block separator (.EF) is found.

- Data files may include control words.
- There is no limit to the number of blocks of text within the data file.
- A Set Symbol control word should be set within the last block so that the calling file will know where to stop.

Example:

- Following is an example of the data within a data file. Each block of data is separated by an End of File [.EF] control word.

```
Mary Jones
1234 Pickford Avenue
Great Barrington, MA  12345
.EF
John Smith
5678 Block Road
Tampa, FL  54321
.EF
George Tell
9876 Broadway
New York, NY  09876
.EF
Margaret Victoria
c/o Jennifer Reilley
5432 Main Street
Seattle, WA  67890
.SE LASTNAME = 1
```

Your Notes:

Define Font .DF Control Word

Purpose: Use the Define Font [.DF] control word to identify a specific font that you want to use. This font is different than the font you normally use and can be tailored to the style that you prefer. The Begin Font [.BF] control word will start the font you have defined. To return to the original font, use the Previous Font [.PF] control word.

Format:

```
.DF font-id [ US ] [ FONT definitions ]
            [ UP ]
            [ UC ]
```

Remarks:

font-id Specifies the name of a set of printer options. Up to 15 names are allowed.

Following is a list of the printer options as they must be written and an explanation of the options:

US Specifies that all characters are to be underlined (underscored).

UP Specifies that all alphabetic characters will be capitalized (uppercase).

UC Specifies that all characters are to be both underlined (underscored) and capitalized.

Define Font .DF Control Word

FONT Specifies that you want to do more than highlight the default font. You can make multiple choices in the definitions list (for example, .DF NEW FONT C2 L8 EM)

definitions Specifies the options available for your selection. You may select one option from each of the following groups if you want to define a combination font-id. Each available definition is listed as it must be written. It is followed by a description of the definition:

Character Sets

C1 This selects the graphics printer Character Set 1.

C2 This selects the graphics printer Character Set 2.

SCRIPT/PC assumes Character Set 1 as the default.

Printing Direction

BD Bidirectional - prints as the print mechanism moves in either direction, left to right or right to left. This speeds printing because there is no waiting for a "carriage return."

LR Unidirectional - prints only as the print mechanism moves from left to right. Does not

print when it returns from right to left. This is similar to a standard typewriter style of printing.

SCRIPT/PC assumes bidirectional printing as the default.

Print Lines Per Inch

L6 Selects 6 lines of print per inch (default).

L8 Selects 8 lines of print per inch.

SCRIPT/PC assumes 6 lines per inch as the default.

Fonts

CM Compressed Characters are selected. This font creates very narrow characters.

DW Double Width Characters are selected. This font is twice as wide as the default print characters. This font can only be used with format off (.FO OFF).

SP Superscript Character Set is selected. This font prints small characters at the top of the print line.

SB Subscript Character Set is selected. This font prints small characters at the bottom of the print line.

Define Font .DF Control Word

Emphasizing Text

DS Double Strike Print - bold

EM Emphasized Print - very bold

Initial Setting: C1 L6 BD

Default: C1 L6 BD

- Specify up to a maximum of 15 font definitions. For example:

```
.DF FONT1 UP
.DF FONT2 UC
```

  results in defining two fonts, one that capitalizes all characters and the other font that both capitalizes and underscores all characters.

- If a font-id name has already been defined, then the new font-id definition replaces the old font-id definition.

- The definition terms may be defined in either upper- or lowercase.

Examples:

- To use the font whose font-id is FONT1:

```
.DF FONT1 UP
.
.
.
.BF FONT1
This is a new font
```

Define Font .DF Control Word

Results in:

```
THIS IS A NEW FONT
```

- To use the font whose font-id is FONT2:

```
.DF FONT2 UC
.
.
.
.BF FONT2
This is a new font
```

Results in:

```
THIS IS A NEW FONT
```

Your Notes:

Define Macro .DM Control Word

Purpose: Use the Define Macro [.DM] control word to establish macro definitions for series of SCRIPT/PC control words or text lines. SCRIPT/PC selects the macro you have created if you start the macroname with a period (.) or colon (:) on an input line. Macros can be included in your input text as if they were SCRIPT/PC control words.

Format:

```
.DM  macroname  [OFF]
```

Remarks:

macroname Is a 1- 10 character symbolic name that you want to assign to the macro.

OFF Specifies that the identified macroname will be turned off and the memory space needed for the macro has been freed for more macros.

- The macro you build can be in-line as part of your input text or it can be inserted into a library of tags. This library may be one provided by SCRIPT/PC (GMLLIB.SCT and HEADDEF.SCT) or it may be a library that you have named.

- This control word starts a macro definition. All input lines after this control word will be used as part of the definition until the Macro Exit [.ME] control word is found.

- The Macro Exit [.ME] control word that terminates a macro should be the only control

word on an input line and it should start in column 1, although other macro exits can also be used within a macro.

- The following defined macro and its option cannot appear *within* a macro:

```
.DM macroname OFF
```

It should be on an input (source) line.

- The special symbol &* has a unique meaning to macros. It is the line of data, control words and/or text, passed to the macro when it is selected. Thus, if the macro defined with:

```
.DM TYPIT
.TY **********************
.TY &*
.TY **********************
.ME
```

is selected with the line:

```
.TYPIT Place new diskette in Drive A!
```

then the symbol **&*** is replaced by the text. When the macro is processed, the results are:

```
******************************
Place new diskette in drive A!
******************************
```

Note: You will note that the Type on Terminal control word [.TY] only allows the output text to be sent to the display screen.

- If a macro has the same name as a control word, it is only selected if its first character is a colon, as in :SP, instead of a period, as in .SP.

Define Macro .DM Control Word

Your Notes:

Double-Space Mode .DS Control Word

Purpose: Use the Double-Space mode [.DS] control word when you want your output to be double spaced.

Format:

```
.DS
```

Remarks:

- This control word does not cause a break.
- The Double-Space Mode control word doubles the line spacing. When Double-Space Mode is in effect, each space [.SP] and skip [.SK] control word value is multiplied by 2.
- A blank line is placed after each output line. However, a page cannot be started with one of these blank lines but instead it will be started with an output line of text. By "throwing away" blank lines at the top of pages during double-space mode, columns will be created that may be shorter than you expected.
- Double spacing is not done during formatting of headings or titles unless it is specifically defined in the Running Heading [.RH] control word.

Double-Space Mode .DS Control Word

Example:

```
.DS
This text will be
double spaced after the file
has been formatted.  The starting point
for the double spacing
will be the .DS
control word.  Single spacing can be started
again by writing the single space [.SS]
control word.
```

results in:

This text will be double spaced after the file has been formatted. The starting point for the double spacing will be the .DS control word. Single spacing can be started again by writing the single space [.SS] control word.

Your Notes:

Else .EL Control Word

Purpose: The Else [.EL] control word can be used in combination with the IF control word to process SCRIPT/PC input lines conditionally. The *target* item of the Else control word is processed only if the most recently performed If, And, or Or control word resulted in a *false* condition.

Format:

```
.EL  [target]
```

Remarks:

target Is any valid SCRIPT/PC input line (control word or text). If the most recently performed IF was *false*, the target line is processed next, with the first non-blank character after the Else control word treated as the first position of the subject line. If the condition was *true*, the target line is ignored and processing continues with the next input line (the one that follows the Else).

- The Then and Else control words, used with the If, And, and Or control words allow you to construct complex logic statements.

- The Then and Else control words do not cause a break, or change the true/false condition. However, the target control word might, if it is processed. For example, the input lines:

```
.IF &A = &B
.EL .IF &C = &D
.TH .TY Yes
```

are equivalent to the line:

```
.IF &A = &B .OR &C = &D .TY Yes
```

- More than one Then and Else control word may follow an If control word. When the If control word's comparison answer is true, the Then control word is processed. When the If control word's comparison answer is false, the Else control word is processed.
- If there is *no* most recently performed comparison, the target will not be processed.

Examples:

- The following input lines:

```
.IF &A = YES .IM B:ACCEPT
.
.
.
.EL .IM B:REJECT
```

Result in:

1. A test of symbol A to see if it is set to YES.
2. If the answer is true, the file-id B:ACCEPT.SCT is inserted into your file.
3. If the answer is false, the file-id B:REJECT.SCT is inserted into your file when the else control word is read.

- The following input lines:

Else .EL Control Word

```
.IF &A NE &B .TY Yes
.IF &A NE &B .TY still
```

are the same as the following lines:

```
.IF &A EQ &B
.EL .TY Yes
.EL .TY still
```

Your Notes:

End of File .EF Control Word

Purpose: The End of File [.EF] control word is used to stop an input file before the physical end of the file has been read by SCRIPT/PC.

Format:

```
.EF  [ CLOSE ]
```

Remarks:

CLOSE Tells SCRIPT/PC not to keep track of the current position in the file, but to start at the top of the file the next time it is imbedded.

- The End of File control word applies only to a file being imbedded which has its name identified in a previous Define Data File-id [.DD] control word.
- If the End of File control word is found in a file that has *not* been defined by a Define Data File-id [.DD] control word, it will end the file (treated as a .EF CLOSE).

Example:

- An example of how to use the End of File control word is included in the description of the Define Data File-id control word.

Your Notes:

Footnote .FN Control Word

Purpose: Use the Footnote [.FN] control word to create footnotes at the bottom of a page.

Format:

```
.FN    { ON  }
       { OFF }
```

Remarks:

ON Indicates the start of the footnote text data.

OFF Indicates the end of the footnote text data.

- Any valid SCRIPT/PC control word (not a macro) can be used in a footnote definition.
- This control word does not cause a formatting change, called a break.
- **.FN ON** starts a footnote. All following input lines are placed in the footnote, until a **.FN OFF** is read.
- Before a footnote can be placed on a page, space should be reserved at the bottom of the page for your footnote(s). Decide how many lines you need for footnotes and then use the following to reserve it:

  ```
  .SE $FN = n
  ```

  Where *n* is the number of lines you want to reserve.

Examples:

- The following creates a footnote:

```
.SE $FN = 4
.
.
.
This is the best cherry pie ever baked.
.FN ON
I realize this cherry pie could be
controversial. However, who am I to
deny my Mother's pie?
.FN OFF
The quality and taste of the cherries
are superb.
```

Results in:

```
This is the best cherry pie ever
baked. The quality and taste of the
cherries are superb.
      . . .
      . . .
      . . .
------------------
I realize this cherry pie could be
controversial. However, who am I to deny
my Mother's pie?
```

- If you want to place a superscript character next to a footnote for identification, you must reserve character space to the left of your text, begin a superscript font, and insert the character that you want to print. The following example shows how to create these requirements.

Footnote .FN Control Word

```
.SE $FN = 4
.
.
.
This is the best cherry pie ever baked.
.FN ON;.OF 2;.BF SUPER;1;.PF
I realize this cherry pie could be
controversial. However, who am I to
deny my Mother's pie?
.FN OFF
The quality and taste of the cherries
are superb.
```

Results in:

```
This is the best cherry pie ever
baked. The quality and taste of the
cherries are superb.
        . . .
        . . .
        . . .
-------------------
1  I realize this cherry pie could be
   controversial. However, who am I to
   deny my Mother's pie?
```

Note: SUPER is one of the two font-ids reserved by SCRIPT/PC. The other fontid is the normal print font. SUPER provides superscript characters until ended by another font-id or the Previous Font control word.

Your Notes:

Format Mode .FO Control Word

Purpose: The Format mode [.FO] control word stops or starts stringing together, concatenation, of input lines and justification (to the right margin) of output lines.

Format:

```
.FO    [ ON   ]
       [ OFF  ]
       [ LEFT ]
```

Remarks:

ON Restarts default SCRIPT/PC formatting, including justification and concatenation of lines. If the format control word is entered without any option, Format ON is assumed.

OFF Stops concatenating input lines and justifying of output lines. Text following the format off control word is printed "as is." No error message is given if an input line is longer than the column width.

LEFT Specifies that input lines are to be concatenated but not justified. The resulting output lines are aligned on the left margin. This style of printing is also called *ragged right*.

Initial Setting: ON Default: ON

- The normal format is right justified. The text output lines give a smooth right edge to your text.

Format Mode .FO Control Word

- This control word causes a formatting change, called a break.

Examples:

- To cause the formatter to stop stringing together input lines:

```
.FO OFF
Mary had a little lamb,
her fleece was white as snow.
Everywhere
that Mary went,
the lamb
was sure to go.
It followed her to school one
day, which was against the rules.
```

Results in output text of:

```
Mary had a little lamb,
her fleece was white as snow.
Everywhere
that Mary went,
the lamb
was sure to go.
It followed her to school one
day, which was against the rules.
```

- To cause the formatter to start stringing together input lines and justifying text for a smooth right margin using the same input text:

```
.FO ON
```

Results in output text of:

```
Mary had  a little lamb, her  fleece  was
white as snow. Everywhere that Mary went,
the lamb was sure to go. It followed  her
to school one day,  which was against the
rules.
```

- To cause the formatter to start stringing together input lines and leaving justification off (ragged right) using the same input text:

  ```
  .FO LEFT
  ```

  Results in output text of:

  ```
  Mary had a little lamb, her fleece was
  white as snow. Everywhere that Mary went,
  the lamb was sure to go. It followed her
  to school one day, which was against the
  rules.
  ```

Your Notes:

Goto .GO
Control Word

Purpose: The Goto [.GO] control word allows you to change the order in which control word lines are executed within a macro.

Format:

```
.GO  label
```

Remarks:

label Is the name of a line within the same macro that is set with the Set Label [...] control word.

- Use the Goto control word to branch or skip to another place in your macro definition (those control words between the Define Macro [.DM] control word and the Macro End [.ME] control word). If the label specified by the Goto control word is *not* found, an error message appears.

- This control word does not cause a formatting change, called a break. The input line preceding the Goto control word and the input line at the target of the Goto control word are processed in sequence as if they were that way in the input file.

- You may use more than one Goto control word within the same macro pointing to the same label.

- The Goto control word is particularly useful when performed conditionally as the target of an IF control word. To understand this better, you might want to review the IF control word.

Examples:

- To test conditionally, let's look at a macro that has been written and added to the library. You can see that it has two variable-items that start with an ampersand (&). The variables are &NAME and &GRADE.

```
.DM ADMIT
.TY ENTER NAME OF STUDENT.
.RV NAME = '
.TY ENTER HIGH SCHOOL AVERAGE.
.RV GRADE = '
.IF &GRADE <= 83 .GO FAIL
.TY ******************************
.TY * &NAME has been accepted,
.TY * send an acceptance letter.
.TY ******************************
.IM B:ACCEPT
.GO END
...FAIL
.TY ******************************
.TY * &NAME has been rejected,
.TY * send a rejection letter.
.TY ******************************
.IM B:REJECT
...END
.ME
```

When the tag is used that calls this macro, the setting of the &GRADE variable determines whether the first or second message will be printed. This tag results in one of two conditions:

1. If the input file is processed by SCRIPT/PC with the following input:

Goto .GO
Control Word

```
.FO OFF
:ADMIT
.
.
.
ENTER NAME OF STUDENT.
Mary Jones
ENTER HIGH SCHOOL AVERAGE.
92
```

Then the output will be:

```
******************************
* Mary Jones has been accepted,
* send an acceptance letter.
******************************
 (Letter of acceptance is now printed.)
```

2. However, if the input file is processed this way:

```
.FO OFF
:ADMIT
.
.
.
ENTER NAME OF STUDENT.
John Smith
ENTER HIGH SCHOOL AVERAGE.
82
```

Then the output will be:

```
******************************
* John Smith has been rejected,
* send a rejection letter.
******************************
 (Letter of rejection is now printed.)
```

Note: Remember that you should precede the tag name by a colon (:) or period (.) to call the tag (macro) from the library.

Goto .GO
Control Word

Your Notes:

Heading 0 - Heading 6 .H0 - .H6 Control Word

Purpose: The control words .H0 through .H6 format topic headings in SCRIPT/PC output. The definition of a particular heading level may also result in an entry in the table of contents for that heading. If any of the control word headings (.H0 through .H6) are used in your document, you *must* identify the library file HEADDEF.SCT to SCRIPT/PC before any of the heading control words are read.

Format:

```
.Hn text
```

Remarks:

n is the number of the heading level from 0 to 6.

text Is the data to be formatted as a subject head. It may also be placed in the table of contents.

- These control words cause formatting changes, also called breaks.
- If the heading level control word causes an entry into the table of contents, the text is inserted into the table of contents as it was entered.

Example:

```
.H1 Pole Vaulting and the Olympics
```

Your Notes:

If .IF Control Word

Purpose: Use the IF [.IF] control word to process SCRIPT/PC input lines conditionally.

Format:

```
.IF  comparand1 test comparand2 [target]
```

Remarks:

comparand1 Is the name of a symbol to be used in the test; this symbol can be a number or a string of characters.

comparand2 Is the second value to be tested. It too may be the value of a set symbol.

test Is a 1- or 2-character code that tells SCRIPT/PC what type of comparison to make between the two comparands. The list of tests and the formats in which they may be written are as follows:

If .IF
Control Word

Test	Form 1	Form 2
Equal	EQ	=
Not equal	NE	<>
Greater than	GT	>
Less than	LT	<
Greater than or equal to	GE	>=
Less than or equal to	LE	<=

target Is any valid SCRIPT/PC input line (control word or text). If the tested condition is TRUE then the target line is processed next (the first non- blank character after the second comparand is treated as the first position of the subject line). Otherwise, the target line is ignored, and processing continues with the input line that follows the IF control line.

- Comparand1 and Comparand2 may be a symbol, number, or a string of characters enclosed in single quotes (') if there are blanks in the string of characters.

- The If control word, used with the Then, Else, And, and Or control words allows you to construct complex logic statements.

- The If control word itself does not cause a break. However the target control word might, if it is processed.

- Each comparand can be up to 255 characters in length. The shorter comparand will be extended to the same length as the longer comparand using trailing blanks.

- Even if substitution is off (**.SU OFF**) when the IF control word is found, all valid symbols in the comparands will be resolved before the comparison is made.

- The only tests on strings of data that can be done are the equal and not equal tests.

Examples:

- To use an IF control word in its simplest form:

```
.IF &GRADE < 65  .TY &NAME Fails
.IF &NAME >= 65  .TY &NAME Passes
```

- The target of the IF control word may be another IF. For example, if it is after noon on the first day of the month, imbed a file called **ABC.SCT** into your input file. To do this, you might use the following:

```
.SE day = &SYSDAYOFM
.SE hour = &SYSHOUR
.
.
.
.IF &hour >= 12 .IF &day = 1 .IM ABC.SCT
     |                  |
     |                  └ Is today the first
     |                    day of the month?
     |
     └ Is hour after noon (greater than or
       equal to 12:00)?
```

- If one of the comparands could be a null (nothing as opposed to space) symbol, use a character in front of the normal comparand you are using. For example, if the symbol **answer,** represented in your input as **&answer,** is a null, then your input:

  ```
  .IF &answer = high .IM abc.sct
  ```

  would be processed incorrectly as:

  ```
  .IF = high .IM abc.sct
  ```

  Using an extra character as follows:

  ```
  .IF X&answer = Xhigh .IM abc.sct
  ```

  causes the statement to be processed correctly as:

  ```
  .IF X = Xhigh .IM abc.sct
  ```

  Note: The symbol is a null only if set by the Set Symbol [.SE] control word.

Your Notes:

Imbed .IM Control Word

Purpose: The Imbed [.IM] control word is used to insert an additional SCRIPT/PC file into the current input file at the point where the Imbed control word is found. When the additional file has been processed and inserted into the output file, processing will resume in the current input file.

Format:

```
.IM  [d:]filename[.ext]
```

Remarks:

d: Is the drive specifier (A: B: C: D: E: F: G: H:) if the file you want to imbed in the current file is not on the DOS selected default drive.

filename Is a 1- to 8-character filename. It is the name of the input file you want to imbed.

.ext Is the optional file extension. SCRIPT/PC will assume a default extension of **SCT** if this item is omitted.

- Any SCRIPT/PC control word or text file may be an imbedded file. Files may be imbedded within files. This is called nesting.
- The Imbed and Append control words operate basically the same way. However, the Imbed control word allows the contents of a second file to be inserted into the processing of an existing file, rather than only adding to the existing file.

Imbed .IM Control Word

Imbedding may be used to insert groups of control words into a file, as well as for many other purposes.

Examples:

- If the input file you want is on the DOS default drive, then you would type:

```
.IM ABC
```

The contents of the file, whose file-id is **ABC.SCT,** are inserted into the processing procedure of the current file being processed. When the end of the file **ABC.SCT** has been reached, processing of the current file is resumed.

- If the input file you want is on a drive other than the DOS default drive, then you would type:

```
.IM B:ABC
```

Your Notes:

Indent .IN Control Word

Purpose: Use the Indent [.IN] control word to change the left margin of SCRIPT/PC output for all following lines.

Format:

```
.IN   [  h  ]  [NOBREAK]
      [ +h  ]
      [ -h  ]
```

Remarks:

h Specifies the amount of space to be indented. If omitted, 0 is assumed, and indentation returns to the left margin.

NOBREAK Indicates that a formatting change, called a break, is not to be done.

Initial Setting: 0 Default: 0

- The Indent control word temporarily resets the current left margin.
- This control word causes a formatting change to occur, unless NOBREAK has been selected.
- If NOBREAK is selected, the control word changes the current output line, not the next line.
- If Indent 0 [.IN 0] is specified, all indentation is canceled and the first character of the output line is printed at the original left column margin.

Indent .IN Control Word

- Indentation control words are absolute values; for example, the following:

```
.IN 0
.IN 10
.IN 5
```

  Results in the final indentation value of 5 characters (the last value entered). In contrast, if you want the indent control word values to add to or subtract from each other, use the relative form of the control word (+ and -). For example, the following:

```
.IN 0
.IN 10
.IN +5
```

  Results in the final indentation value of 15 characters.

Examples:

- To indent 7 character spaces:

```
The following shows how indent works.
.IN 7
.SP
Mary had a little lamb,
it's fleece was white
as snow.
Everywhere that Mary went, the lamb
was sure to go.
```

Results in:

```
The following shows how indent works.

       Mary had  a  little lamb, it's
       fleece  was   white  as  snow.
       Everywhere that Mary went, the
       lamb was sure to go.
```

- To go back to the original indent setting:

```
.IN 0
   or
.IN
```

Results in:

```
The following shows how indent  works.

Mary had  a  little lamb, it's  fleece
was  white  as  snow.  Everywhere that
Mary went, the lamb was sure to go.
```

Your Notes:

Indent Line .IL Control Word

Purpose: Use the Indent Line [.IL] control word to indent the next output line. The following lines will return back to the original setting.

Format:

```
.IL  [ h ]
```

Remarks:

h Specifies the number of character spaces to shift the next output line from the current margin.

Initial Setting: 0 Default: 0

- The Indent Line control word provides a way to indent only the next output line. The line is shifted to the right of the current margin.
- This control word causes a formatting change, called a break.
- Note that the Indent Line [.IL] control word and the Undent [.UN] control word are opposites. They cause the new margin for the next line to be shifted right (Indent Line) and left (Undent).
- You will find the Indent Line control word useful when starting new paragraphs.

Indent Line .IL Control Word

Example:

- To use it in the beginning of a new paragraph:

```
This is the last line of a paragraph.
.SP
.IL 5
This is the first line of a new paragraph.
Note that the first line
of this new paragraph has been indented
5 spaces.
```

results in:

```
This is the last line of a  paragraph.

     This is the first line of  a  new
paragraph. Note that the first line of
this new paragraph has  been  indented
5 spaces.
```

Your Notes:

Indent Right .IR Control Word

Purpose: Use the Indent Right [.IR] control word to change the right margin on the page.

Format:

```
.IR   [  h  ]  [NOBREAK]
      [ +h  ]
      [ -h  ]
```

Remarks:

h Specifies the amount of space to be indented. If omitted, 0 is assumed, and indentation returns to the previous right margin.

NOBREAK Indicates that a formatting change, called a break, is not to be done.

Initial Setting: 0 Default: 0

- The Indent Right control word temporarily resets the current right margin. Any new indent right control word can change the current indentation value. This indentation remains in effect until another Indent Right control word is found. Indent Right with a zero value [.IR 0] cancels the indentation and output continues at the original right margin setting.

- The value of *h* is the number of characters you want to move your right margin to the left.

Indent Right .IR Control Word

- This control word causes a formatting change, called a break, unless NOBREAK has been selected.
- If NOBREAK is selected, the control word changes the current output line, not the next line.

Examples:

- The following example shows an indent right of 5 characters (.5 inches):

```
.IR 5
This paragraph is 5 characters
shorter on the right side
because of the indent right
control word.
```

which results in:

```
This  paragraph  is  5  characters
shorter on the  right  side because
of the indent right control word.
```

- To reset the right margin you can do the following:

```
.IR 0
This paragraph will be 5 characters
longer on the right side
because the indent right
control word has been canceled.
```

results in:

```
This paragraph is  5  characters  longer
on the  right  side  because  the indent
right control word has been canceled.
```

Indent Right .IR Control Word

Your Notes:

Index .IX
Control Word

Purpose: The Index [.IX] control word causes an index, created from the entries specified by the Put Index [.PI] control word, to be formatted.

Format:

```
.IX
```

Remarks:

- The Index control word *must* be the last control word in your document.
- The formatting specifications that are in effect when the index control word is read will be the formatting specifications for the index. You may want to switch to multicolumn mode before you write the index control word. Most indexes look better when they are in multicolumn mode.
- This control word causes a formatting change, called a break.
- The format of the index is fixed and cannot be altered after it has been started.

Your Notes:

Line Length .LL Control Word

Purpose: The Line Length [.LL] control word sets up the width of running titles and running headings. It also controls the width of lines of text, if those lines of text had *not* been previously set by the Column Width [.CL] control word.

Format:

```
.LL    [  h  ]
       [ +h  ]
       [ -h  ]
```

Remarks:

h Signals the number of characters you want on a line. If no value is given, a default value will be inserted.

Initial Setting: 72

Default: Restores the initial setting.

- This control word takes effect immediately after it has been read.
- This control word causes a formatting change, called a break.
- All indentation is calculated from the setting of the Line Length control word.

- All indentation is canceled temporarily when a running title or running heading is being created. It is restored before the next normal input line is read.

Examples:

- To set the line length of a page at 75 characters:

```
.LL 75
```

- To set the line length of a page at 40 characters

```
.LL 40
```

Your Notes:

Macro Exit .ME Control Word

Purpose: The Macro Exit [.ME] control word is used to cause SCRIPT/PC to stop processing a macro (a grouped set of control words). You may use this control word to either exit a macro or terminate a macro.

Format:

```
.ME
```

Remarks:

- The Macro Exit control word will only be implemented if the current input is in a macro.
- All macros that are called must be ended with a Macro Exit control word, just as they must start with a Define Macro [.DM] control word.
- If the current source input is not a macro and a Macro exit control word is found, an error message appears.
- The Macro Exit control word can be anywhere within a macro. There can be more than one Macro Exit control word within a macro definition, but only one can start in column 1. When the control word is read, the macro is exited and input to SCRIPT/PC continues on the input line following the input line that originally called the macro or the next command on the same line.
- To create a normal macro end condition, the Macro Exit control word must start in column 1 and it must be the only control word on the input line.

Examples:

- The following shows that the last item in a macro must be a Macro Exit:

```
.DM SKIPTWO
.SK 2;.IN 5
.ME
```

  Note: To call the macro listed above, insert a colon (:) or period (.) preceding the tag name, for example, **.skiptwo.** The colon should be used when you have developed a name that is the same as a control word.

- This example shows more than one Macro Exit in a macro. It also shows how to exit a macro as the result of a conditional statement:

```
.DM CHEM101
.IF &GRADE = F .ME
.TY &NAME has passed.
.ME
```

  Note: To call the macro listed above, insert a colon (:) or a period (.) preceding the macro name. For example:

```
.SE GRADE = B
.SE NAME = 'Mary Jones'
.CHEM101
```

Your Notes:

Multicolumn Mode .MC Control Word

Purpose: The Multicolumn Mode [.MC] control word starts two-column processing. It must be preceded by a Column Definition [.CD] control word that sets up the starting columns. Multicolumn Mode can be stopped by the Single-Column Mode [.SC] control word.

Format:

```
.MC
```

Remarks:

- The Multicolumn control word stops the single-column mode that was in effect either by default or because it was started by the Single-Column Mode [.SC] control word.
- The Multicolumn control word restores the multicolumn column definition. When the single-column mode is ended, it keeps the current column definition for later use, should you decide to go back to single-column mode.
- The Column Definition [.CD] control word starts a new column definition, and is independent of the **.SC** or **.MC** control words that are in effect.
- Multicolumn mode is not displayed on the display screen when SCRIPT/PC is formatting your file. It is displayed as a single column.

Multicolumn Mode .MC Control Word

Example:

- To start multicolumn mode, the following input:

```
.CD 2 1 35
.
.
.
.CL 30
.MC
```

  results in two columns starting in character positions 1 and 35. Each column is 30 characters wide. The columns are not started until the Multicolumn mode control word is read, even in the middle of a page.

Your Notes:

Offset .OF Control Word

Purpose: Use the Offset [.OF] control word to indent all except the first line of output text following the control word.

Format:

```
.OF  [ h ]
```

Remarks:

h Specifies the number of characters you wish to move text to the right. If you omit this value, 0 is assumed.

Initial Setting: 0 Default: 0

- The indentation is delayed until after the next text line is processed. The Offset control word stays in effect until another Offset, Undent or Indent control word is read from your input file.
- This control word causes a formatting change, called a break.

Examples:

- The following shows how an offset is started:

```
.OF 3
The output line that is generated next
remains at the current left margin.
However,
all following lines are offset (indented).
```

Results in:

```
The output line that is generated next
remains at the current left margin.
However, all following lines are offset
(indented).
```

- The following shows how an offset is stopped:

```
.OF
The output line that is generated next
moves back to the original setting
of the current left margin. All
following lines will also be in line with
the first output line.
```

Results in:

```
The output line that is generated next
moves back to the original setting of the
current left margin. All following lines
will also be in line with the first output
line.
```

Your Notes:

Or - Conditional .OR Control Word

Purpose: Use the Or [.OR] conditional control word with the IF control word to process SCRIPT/PC input lines conditionally. The result of the test performed is logically OR'd to the result of the most recently performed If, And, or Or control word to determine whether the target should be processed.

Format:

```
.OR comparand1 test comparand2 target
```

Remarks:

comparand1 Is any string to be used as the first comparand. This comparand may be the value of a set symbol.

comparand2 Is any string to be used as the second comparand. It too may be the value of a set symbol.

test Is a 1- or 2-character code that tells SCRIPT/PC what type of comparison to make between the two comparands:

The list of tests and the formats in which they may be written are as follows:

Or - Conditional .OR Control Word

Test	Form 1	Form 2
Equal	EQ	=
Not equal	NE	<>
Greater than	GT	>
Less than	LT	<
Greater than or equal to	GE	>=
Less than or equal to	LE	<=

target Is any valid SCRIPT/PC input line (control word or text). If this condition or the result of the most recently performed If, And, or Or is true, then the target line is processed next (the first non-blank character after comparand2 is treated as the first position of the subject line). Otherwise, the target line is ignored, and processing continues with the input line that follows the OR control line.

- The And and Or control words used with the IF, Then, and Else control words allow you to construct complex logic statements.

- The Or control word itself does not cause a break. However the target control word might, if it is processed.

- Each comparand can be up to 255 characters in length. The shorter comparand will be extended to the same length as the longer comparand using trailing blanks.

Or - Conditional .OR Control Word

- Even if substitution is off when the OR control word is found, all valid symbols in the comparands will be resolved before the comparison is made.

Example:

The following input line:

```
.IF &A = &B .OR &C = &D .TY Yes.
```

Is equivalent to the input lines:

```
.IF &A = &B
.EL .IF &C = &D
.TH .TY Yes.
```

Your Notes:

Page Eject .PA Control Word

Purpose: Use the Page Eject [.PA] control word to force all following text to the top of a new output page.

Format:

```
.PA  [ n ]
```

Remarks:

n Specifies the page number you want on the next page.

Initial Setting: 1

Default: Next sequential page number

- This control word causes a formatting change, called a break.
- When a Page Eject control word is found, the rest of the current page is skipped, and the next page is started.

Examples:

- To skip to the top of the next page:

```
.PA
```

- To skip to the top of the next page and put your own page number on the page:

```
.PA 25
```

Page Eject .PA
Control Word

Your Notes:

Page Length .PL Control Word

Purpose: The Page Length [.PL] control word decides the correct vertical length (depth) of output pages. All top and bottom margins are *subtracted* from the value to which the page length was set.

Format:

```
.PL  v
```

Remarks:

v Specifies the vertical length of the paper you are using. When you compute this value, you must know your total lines-per-page.

Initial Setting: 66

- The Page Length control word allows you to use various paper sizes for your output paper. The page length may be changed anywhere in a file. Do not use this control word in the middle of a page as the results will be unpredictable.
- This control word does not cause a formatting change, called a break.
- To compute page length, you must know your printer's lines-per-inch setting. If you are working with 11-inch paper at 6 lines-per-inch, then your page length should be set to 66 (length times lines-per-inch). If your printer is set to 8 lines-per-inch and you have the same paper, then your page length should be set to 88.
- Page length includes the body of your page plus your top and bottom margin.

Page Length .PL Control Word

Examples:

- To set up a page length for 11-inch paper when your printer is set to 6 lines-per-inch:

  ```
  .PL 66
  ```

- To set up a page length for 11-inch paper when your printer is set to 8 lines-per-inch:

  ```
  .PL 88
  ```

- To set up a page length for a 3-inch form when your printer is set to 6 lines-per-inch (as for creating 3 X 5 inch cards from continuous paper stock).

  ```
  .PL 18
  ```

Your Notes:

Page Margins .PM Control Word

Purpose: Use the Page Margin [.PM] control word to set up the left margin on a page. This establishes the starting left margin for the text that occurs in the running heading, running title, and body of a page.

Format:

```
.PM  h
```

Remarks:

h Specifies the amount of horizontal space you want before your text starts, measured in characters. If you omit this value, the page margin is set to 0.

Initial Setting: 0 Default: 0

- Use the Page Margin control word to shift all output lines to the right. For example, to allow for punching ring binder holes in the left margin or to allow space for a revision code marker.
- The Page Margin control word value is calculated in characters per inch (10). Therefore, if you want to set your page margin at 1.5 inches, you would need to know that .1 inch equals 1 character. Then your value would be 15.
- This control word does not cause a formatting change, called a break.
- Note that, depending upon how the paper tractor is set up in your printer, the printer may not actually start printing at the left edge of the paper.

Page Margins .PM Control Word

Examples:

- Without physically shifting the paper in your printer, to create 1/2 (.5) inch of space for your left page margin:

  ```
  .PM 5
  ```

- This changes your page margin to 1 inch:

  ```
  .PM 10
  ```

Your Notes:

Page Numbering Mode .PN Control Word

Purpose: Use this Page Numbering mode [.PN] control word to change the normal page numbering.

Format:

```
.PN     { n }
        { ON }
        { OFF }
```

Remarks:

n Specifies the number of the next page. When the next page eject occurs, either naturally because of the page getting full, or as a result of a Page Eject control word, the new page will have the value defined in *n*.

ON Turns page numbering on and sets it to the current page number.

OFF Turns page numbering off and causes the page number set symbol to be set to nothing (null).

Initial Setting: ON Default: 1

- This control word takes effect as soon as it is read. If there are top titles or headings that require a page number, issue this control word just before the start of that page.
- This control word does not cause a formatting change, called a break.
- While page numbering is off (.PN OFF), page numbers are still being counted internally within

Page Numbering Mode .PN Control Word

SCRIPT/PC. When page numbering is restarted (.PN ON), the number of the correct page is printed.

Example:

- To change your next page number to page 17, which cancels the original numbers:

```
.PN 17
```

Your Notes:

Previous Font .PF Control Word

Purpose: Use the Previous Font [.PF] control word to resume the font that was in use prior to the last Begin Font [.BF] control word.

Format:

```
.PF
```

Remarks:

- The Previous Font control word is used to go back to a font that was in use before the last Begin Font control word was found.
- There can be up to 15 fonts described on SCRIPT/PC. They are started by using the Begin Font control word along with a font-id (font name). To go backward through the list of started fonts, you can use the Previous Font control word. You do not have to remember the names of the fonts.
- This control word does not cause a formatting change, called a break.

Example:

- To select a new font:

```
.DF NEWFONT US
.
.
.
.BF NEWFONT
Example of a New Font
.PF
```

Previous Font .PF Control Word

Results in a new font being selected. When the original print font is desired, the Previous Font control word is used:

```
Example of a New Font
```

Your Notes:

Put Index .PI Control Word

Purpose: The Put Index [.PI] control word saves the specified lines for use in building an index. The Index [.IX] control word causes this index to be inserted into the document.

Format:

```
.PI    [ REF   ]  /text[/ref]/
       [ START ]
       [ END   ]
```

Remarks:

REF Specifies that this entry should be an index reference. A reference entry does not have a page number, but instead refers the entry *text* to the entry *ref*.

START Indicates that this is not a single page entry, but the start of a series of pages to bear this entry name.

END Specifies that this is the end of a series of pages that have the same entry name.

text Is the text of the index entry.

ref Is a secondary reference field, to be used only with REF .

/ Is the delimiter character. It must be in your control word; however, it will not appear in any text.

Put Index .PI Control Word

- This control word does not cause a formatting change, called a break.
- Both the *text* and *ref* fields can have a maximum length of 20 characters; more may be entered but they will be *cut off*.

Your Notes:

Put Table of Contents .PT Control Word

Purpose: Use the Put Table of Contents [.PT] control word to add lines or control words to the special file that is generated when a Table of Contents control word is found in the input file.

Format:

```
.PT       {line  }
          {  line}
```

Remarks:

line Is any text or control word line that you want in the table of contents. This line may be preceded by one or more leading blanks.

When the line starts with one or more leading blanks, then the line is assumed to be text, even if it starts with a period. The extra leading blanks are removed and a .SX control word is created, using the first non-blank character as the start of data.

- For text lines, the Put Table of Contents control word generates a Split Text [.SX] control word to be written into the table of contents temporary file. The control word is in the form:

  ```
  .SX F /text line/ ./PP/
  ```

  where the page number used (PP) is the actual page number where the Put Table of Contents control word is found.

- This control word is useful for defining head levels with the Define Macro [.DM] control word.

Put Table of Contents .PT Control Word

Examples:

- The following line places the Page Eject [.PA] control word in the table of contents temporary file so that when the table of contents is finally formatted, a page eject occurs at this point. You would do this if you want to place different content sections on different pages.

  ```
  .PT .PA
  ```

- This example shows that if more than one space is left after the Put Table of Contents control word, the extra space(s) are then removed and a Split Text control word is built. Care should be used when writing the Put Table of Contents control word.

  ```
  .PT      .PA
  ```

  results in:

  ```
  .SX F /.PA/ ./33/
  ```

Your Notes:

Quick Quit .QQ Control Word

Purpose: The Quick Quit [.QQ] control word causes SCRIPT/PC processing to stop immediately, without a page eject.

Format:

```
.QQ
```

Remarks:

- Since the Quick Quit control word causes SCRIPT/PC to end without a final page eject, some output that has been formatted may not be seen.
- The Quick Quit control word can be entered in response to a Terminal Input [.TE] control word, which allows you to enter it from the keyboard and quickly end processing.

Your Notes:

Quit .QU
Control Word

Purpose: The Quit [.QU] control word causes SCRIPT/PC processing to stop with a final page eject.

Format:

```
.QU
```

Remarks:

- The Quit control word causes a final page eject so that the last partial page of formatted text may be seen.
- The Quit control word causes SCRIPT/PC to end no matter where or when it is read, including in imbedded files.

Your Notes:

Read Variable .RV Control Word

Purpose: The Read Variable [.RV] control word is similar to the Set Symbol [.SE] control word, except that the value of the symbol is read from the keyboard of the computer on which you are working.

Format:

```
.RV       symbolname  =  [']
```

Remarks:

symbolname Is the name of the symbol that you want to define. It can be any name that is allowed on the left side of the equal sign in the set symbol [.SE] control word.

= ['] Indicates that the value set into the named symbol is to be treated as a quoted string of characters. If you do not specify the single quotation mark ('), SCRIPT/PC processes whatever character string it reads according to the rules for the value on the right hand side of the equal sign in the Set Symbol [.SE] control word.

- When the Read Variable control word is found, an input line is read directly from your computer's keyboard. This input line is used as the text to the right of the equal sign (=) and sets a value for the symbol named in the Read Variable control word.
- This control word does not cause a formatting change, called a break.

Read Variable .RV Control Word

- Since no message is displayed before the keyboard is unlocked on your computer, you might want to prompt yourself by issuing a Type on Terminal [.TY] control word just prior to issuing the Read Variable control word.

Examples:

- To enter a name into the terminal:

```
.TY Enter name of student!
.RV NAME = '
.TY Enter English 101 Grade!
.RV GRADE =
```

results in your being prompted for a name and then a grade. Upon completion of these steps, the symbols &NAME and &GRADE have a value which you just entered from the keyboard.

```
Enter name of student!
John Adams
Enter English 101 Grade!
B
```

The symbols and their values after the preceding entry are:

```
&NAME = John Adams
&GRADE = B
```

- To use this control word to set up the If conditional control word:

```
.TY Answer YES when you have placed the
.TY ADD diskette in Drive A!
.RV A =
.IF &A = YES .AP B:ALPHA
.EL .TY SCRIPT/PC is ending!;.QU
```

Read Variable .RV Control Word

When the file is being formatted, the messages are displayed on the screen as:

```
Answer YES when you have placed the
ADD diskette in Drive A!
=>
```

If you answer YES, SCRIPT/PC will append the file whose drive specifier, filename and extension = B:ALPHA.SCT. If your typed reply was something else, the message **SCRIPT/PC is ending!** is issued and SCRIPT/PC will be ended by the Quit control word (.QU).

Your Notes:

Revision Code .RC Control Word

Purpose: Use the Revision Code [.RC] control word to allow a revision code to be printed at the left margin of your document. You would use these marks to mark all lines that have changed from a previous edition of a book.

Format:

```
.RC     n     [ c   ]
              [ ON  ]
              [ OFF ]
      [ ADJUST h ]
```

Remarks:

n Specifies the revision code number. Revisions can fall in the range of 1 to 9.

c Specifies the revision code character to be printed along the left margin. It may be any single character. If no character is selected, a blank is assigned.

ON Starts the series of lines that are to be marked with the revision code.

OFF Ends marking of the lines with the revision code.

ADJUST Specifies the amount of horizontal space to be used to contain the revision code character and the space between the character and the left column boundary.

h Is the amount of horizontal space. If *h* is omitted, the horizontal value of 2 is assumed. If a horizontal value of 0 is given, or if the amount specified is greater than the available horizontal space, no revision codes will be printed.

- The Revision Code control word has three functions:

  1. To define a revision code symbol,
  2. To activate or deactivate the revision code, and
  3. To set the revision code adjust.

- You may create up to 9 revision codes; each revision code a different character and individually controlled. The revision codes are started or stopped by the ON and OFF parts of the control word. You should define a revision code first (for example, .RC 3 |) and then start it for the revision code to be inserted (For example, .RC 3 ON). When you want to stop it, you should enter the control word again (For example, .RC 3 OFF).

- By assigning different characters for each revision code, you can show various levels of changes in one document.

- This control word does not cause a formatting change, called a break. This allows you to mark a single line within a paragraph with a revision code.

- There must be space reserved by the Page Margin (.PM) control word before the revision code can be printed.

Revision Code .RC Control Word

- Revision codes are not applied to running titles or headings, but they are applied to lines that have been spaced or skipped over.

Examples:

- To set up only one revision code:

```
This is a normal line of text.
.RC 1 |
.SP 2
.RC 1 ON
This is a revision code example.
.SP
These lines have been changed from
the original text.
.SP
.RC 1 OFF
This is back to a normal line of print.
```

Results in:

```
   This is a normal line of text.


|  This is a revision code example.
|
|  These lines  have  been changed  from the
|  original text.

   This is back to  a normal line of  print.
```

- To use more than one revision code:

Revision Code .RC Control Word

```
This is a normal line of text.
.RC 1 |
.RC 2 +
.SP 2
.RC 1 ON
This is a revision code example.
.SP
These lines have been changed from
the original text.
.RC 1 OFF
.RC 2 ON
These lines are the second change
from the original text.
.SP
.RC 2 OFF
This is back to a normal line of print.
```

Results in:

```
   This is a normal line of text.


|  This is a revision code example.
|
|  These lines have been changed from
|  the original text.
+  These lines are the second change
+  from the original text.
+
    This is back to a normal line of print.
```

Your Notes:

Right Adjust .RI Control Word

Purpose: Use the Right Adjust [.RI] control word to position an output line flush with the right margin.

Format:

```
.RI    [ ON   ]
       [ OFF  ]
       [ line ]
```

Remarks:

ON Specifies the start of a series of lines that are to be right adjusted.

OFF Tells SCRIPT/PC to stop right adjusting the output text.

line Specifies a single data line to be right adjusted.

- When right adjustment is in effect, formatting is stopped.
- This control word causes a formatting change, called a break.
- If the input line is longer than the current column width, the input line is cut off and an error message appears.
- The Center [.CE] control word is similar to the Right Adjust control word. If either control word is found, the other (if in effect) is canceled.

Right Adjust .RI Control Word

Example:

- The following:

```
This line is normally left-justified.
.RI ON
These three lines are
right-adjusted,
as you can see.
.RI OFF
Now you are back to the original.
```

Results in:

```
This  line is  normally  left-justified.
                    These three lines are
                          right-adjusted,
                          as you can see.
Now you are back to the original.
```

Your Notes:

Running Heading .RH Control Word

Purpose: Use the Running Heading [.RH] control word to specify that the following input lines are to be saved and printed at the top of every page.

Format:

```
.RH    [ ON     ]
       [ OFF    ]
       [ CANCEL ]
       [ SUP    ]
       [ RES    ]
```

Remarks:

ON Specifies that the following input lines are a running heading and are to be saved and printed at the top of each succeeding page.

OFF Stops the running heading that was started by .RH ON.

CANCEL Indicates that the previously defined running heading is to be cancelled.

SUP Indicates that the previously defined running heading is to be suppressed or stopped for awhile.

RES Indicates that the running heading that was suppressed can now be started again.

Default: None

Running Heading .RH Control Word

- The running heading will be placed on the page immediately below the space that is defined by the Top Margin [.TM] control word. It is formatted under control of the initial set of formatting values, and if page numbers are used, all page number symbols will be replaced by the current page number. Any control words (not tags or macros) can be inserted on the input lines within the running heading.

- You can indent in the heading information. However, SCRIPT/PC will return to the normal text indentation when the heading is completed.

- The running heading defined by the Running Heading control word will take effect on the page following the control word.

- The value CANCEL will be ignored if the running heading has not been created first.

- No macros are allowed within the running heading.

Example:

- This is a simple heading to be printed at the top of a page:

```
.RH ON
.CE ON
Final Theme
.SP
English 101        &SYSDATE.
.CE OFF
.RH OFF
```

Running Heading .RH Control Word

results in printing at the top of every page:

```
                Final Theme

English 101              12-15-1983
```

Your Notes:

Running Title .RT Control Word

Purpose: Use the Running Title [.RT] control word to save a specific line of text to be printed at the top or bottom of each output page, within the top and bottom margin areas.

Format:

```
.RT { TOP }  n /part1/part2/part3/
    { BOTTOM }
```

Remarks:

TOP Indicates that the line being defined should be placed in the top-margin area.

BOTTOM Indicates that the line being defined should be placed in the bottom-margin area.

n Is the number of the line in the top or bottom margin area that is being defined. This line number should be a number from 1 to 10. Lines in the top margin are numbered from top to bottom. Lines in the bottom margin are numbered from bottom to top.

/ Is used to separate the information in the three parts of the title line. This is called a delimiter character and is determined by the first non-blank character after the line number (n).

part1 Is the portion of the title that is to be left justified.

Running Title .RT Control Word

part2 Is the portion of the title that is to be centered between the left and right margins.

part3 Is the portion of the title that is to be right justified.

- Every occurrence of the page number symbol (&.) in the running title is replaced with the current page number.
- Symbol replacement is done when the running title is defined, instead of on a page-by-page basis, as is the case with most symbols.
- Any running title can be defined with new information at any time.
- This control word takes effect as soon as it is read by SCRIPT/PC.

Example:

- The following example:

```
.RT BOTTOM 4 /Term Paper/Nuclear Power/Page &./
```

Results in:

```
Term Paper     Nuclear Power     Page 1
```

Your Notes:

Set Label ... Control Word

Purpose: The Set Label [...] control word marks a line inside a macro description. The line is marked so that it can become the target of a Goto [.GO] control word.

Format:

```
...label[ line]
```

Remarks:

label Is a unique name. It can be any number of characters.

line Is an optional input line that is read when a Goto control word points to this label. This item begins with the first non- blank character after the label. This line can not start with a blank character.

- The Set Label control word can only be the target of a Goto [.GO] control word that is in the *same* macro.
- In labels, uppercase and lowercase characters are different. You should be careful to enter labels exactly as they are entered in the Set Label control word. For example, TEST1, test1 and Test1 would be considered to be three different labels.

Example:

- The following example shows you how to use a label within a macro:

Set Label ...
Control Word

```
.IF &A = 1 .GO ABC
.
.
.
...ABC .SK;.CE Important Information
```

results in a one-line skip and then a centering operation:

This occurs when a Goto [.GO] control word points to the target **ABC** or when this line is read by SCRIPT/PC as the next sequential line.

Your Notes:

Set Symbol .SE Control Word

Purpose: The Set Symbol [.SE] control word allows you to define and assign values to symbols. When you use the Set Symbol control word, you can give a symbolic name to a page number, a word, or even a string of SCRIPT/PC control words. The Set Symbol control word itself, or any of its parameters, can be a symbol.

Format:

```
.SE symname = 'string'
              &
              value
              SUBSTR string start length
              INDEX string1 string2

              OFF
```

Remarks:

symname Is the name to which you wish to assign a symbolic value. This symbolic value will be replaced when SCRIPT/PC finds it. The name can contain a maximum of 10 characters that may be uppercase and lowercase alphabetic, numeric, and the characters @, #, and $.

= The equal sign is required, except for use of OFF.

'string' Can be any string preceded by a single quote ('). The string may be ended by another single quote. You can put quotes within the string. Quotes are necessary when your string contains a blank

character. If you do not have a blank within your string, you may omit the quotes. The quotes are also not necessary if you are not using special characters.

& Assigns the symbol name to a value equal to the current page number string.

value Is a numeric value, or a series of numbers preceded by the +, -, * or / sign (These are called arithmetic operators).

SUBSTR Gets the specified characters (substring) from a given character string and assigns them to the symbol you are defining.

string This is the character string from which the substring is to be taken.

start This is a numeric value that signals the starting position of the substring within the string you have identified.

length Specifies the number of characters that are to be taken from the string, in other words, the length of the substring.

INDEX Searches string1 to see if it contains the character string identified in string2. If it does find the string, the symbol value is set to the position of the starting character of string2 within string1. If it does not find a match, the symbol value is set to 0.

string1 Is the character string that is to be searched for the contents of string2.

string2 Is the character string that is to be searched for in string1.

OFF Indicates that the symbol identified is to be removed from the symbol list.

- In symbol names, uppercase and lowercase characters are different. You should be careful to enter symbol names exactly as they are entered in the Set Symbol control word. For example, TEST1, test1 and Test1 would be considered to be three different symbol names.

- If the symbol value is omitted, the symbol's value is set to a null (nothing) character string with a length of 0.

- Separate your symbolname from following text by using a period(.).

- The only arithmetic operators that are active are the plus (+), minus (-), multiply (*) and divide (/) operators.

- Care should be exercised when using arithmetic set statements. For example, if spaces are accidentally omitted, the following set symbols:

```
.SE A = 0-3
.SE B = 5+&A
```

  would result in an arithmetic statement that is wrong when symbol substitution takes place

```
.SE B = 5+0-3
```

- The value on the right side of the equal sign can be another symbol.

Set Symbol .SE Control Word

Example:

- The following example:

```
.SE NAME = 'John Adams'
.SE GRADE = B
.
.
.
The student, &NAME., passed the course
with a &GRADE grade.
```

Results in:

```
The student, John Adams, passed the
course with a B grade.
```

Your Notes:

Single-Column Mode .SC Control Word

Purpose: The Single-Column Mode [.SC] control word starts a single-column format.

Format:

```
.SC
```

Remarks:

- The Single-Column Mode control word starts formatting in a single column. Column width defaults to the value of the current line length.
- The Column Definition [.CD] control word starts an entirely new column definition.
- If there are any pending output lines for two-column mode, they will be balanced and printed.

Your Notes:

Single-Space Mode .SS Control Word

Purpose: Use the Single-Space Mode [.SS] control word to cancel a previous Double-Space Mode [.DS] control word.

Format:

```
.SS
```

Remarks:

- This control word does not cause a break.
- The output that follows the Single-Space control word is single-spaced. Since single-space is the normal output format, it is needed only to cancel a previous Double-Space Mode [.DS] control word.

Your Notes:

Skip .SK Control Word

Purpose: Use the Skip [.SK] control word to generate blank vertical lines before the next text output line, except at the top of a column or page.

Format:

```
.SK  [ v ]
```

Remarks:

v Is the number of lines to be inserted in the output. If the vertical value is omitted, 1 is assumed.

Default: 1

- No blank space is generated by the Skip control word if it is at the top of a column. Columns can appear to be short if the intended space has been discarded.
- If the blank space you want to skip is not at the top of the column, using the Skip control word is the same as using the Space [.SP] control word.
- If the Skip control word value is greater than the remaining lines in the column, then only the remaining lines are skipped and the text processing continues at the very top of the next page.
- If double-spacing is in effect, the number of lines skipped is multiplied by the line spacing amount.
- This control word causes a formatting change, called a break.

Skip .SK Control Word

Examples:

- To create one line of vertical space:

  ```
  .SK
  ```

- To create 1.5 inches of space when you are printing at 6 lines-per-inch (1.5 X 6 = 9):

  ```
  .SK 9
  ```

- To create 1.5 inches of space when you are printing at 8 lines-per-inch (1.5 X 8 = 12):

  ```
  .SK 12
  ```

Your Notes:

Space .SP Control Word

Purpose: Use the Space [.SP] control word to generate blank vertical lines (space) before the next text output line.

Format:

```
.SP    [ v ]
```

Remarks:

v Is the number of lines to be inserted in the output. If the vertical value is omitted, 1 is assumed.

Default: 1

- If double-spacing is in effect, the number of spaces generated is multiplied by 2.
- This control word causes a formatting change, called a break.

Examples:

- To create one line of vertical space:

```
.SP
```

- To create 1.5 inches of space when you are printing at 6 lines-per-inch (1.5 X 6 = 9):

```
.SP 9
```

- To create 1.5 inches of space when you are printing at 8 lines-per-inch (1.5 X 8 = 12):

```
.SP 12
```

Space .SP
Control Word

Your Notes:

Split Text .SX Control Word

Purpose: The Split Text [.SX] control word is used to split a string of text between the left- and right-column margins, with a filler in between the two.

Format:

```
.SX    { F }  /lpart/fill/rpart/
       { C }
```

Remarks:

F Specifies that the middle part of the string is to be repeated until the middle part has been filled up.

C Specifies that the middle part of the string is to be centered, not repeated.

/ Is any delimiter character. The first non-blank character will be used as a delimiter.

lpart Is the string to be placed against the current left margin.

fill Is a string of characters to be used to fill the space between lpart and rpart.

If the **C** option was specified, then the fill text is to be centered in the character space assigned to ***fill.***

rpart Is the string to be placed against the current right margin.

Split Text .SX Control Word

- The delimiter character between the strings may be any unique character that does not occur within the strings themselves.
- Any of the three parts of the line may be null.
- This control word causes a formatting change, called a break.
- The Put Table of Contents [.PT] control word writes Split Text control words into the table of contents file to be processed when the table of contents is formatted.

Examples:

- Split text with a null fill string.

```
.SX F  /Left side//Right side/
```

Results in:

```
Left side                          Right side
```

- Split text with null left and right parts.

```
.SX F //Hello //
```

Results in:

```
Hello Hello Hello Hello Hello Hello
```

- Split text with centered characters.

```
.SX C /Left side/Hello /Right Side/
```

Split Text .SX Control Word

Results in:

```
Left side       Hello          Right side
```

Your Notes:

Substitute Symbol .SU Control Word

Purpose: Use the Substitute Symbol [.SU] control word to cause SCRIPT/PC to stop replacement (substitution) of defined set symbols or to restore symbol replacement (substitution).

Format:

```
.SU     [ ON  ]
        [ OFF ]
```

Remarks:

ON Indicates that substitution is to start on all following input lines.

OFF Indicates that substitution has been stopped on all the following lines.

Initial Setting: ON

- This control word temporarily stops symbol substitution.
- The Substitute Symbol control word that was last read stays in effect until another Substitute Symbol control word is read.
- When an input line is substituted, each symbol may go through more than one stage of substitution. Any symbol name that has no definition is left on the input line as text.
- If the formatter is turned off (**.FO OFF**), the substitution of set symbols may increase or

decrease the length of your original input line. Output lines, therefore, may be longer or shorter than you expect.

Your Notes:

Tab Setting .TB Control Word

Purpose: Use the Tab Setting [.TB] control word to set tabs for column information. Character spacing between tab settings and to the right of input text will be set to blank characters.

Format:

```
.TB   [ h ... h ] ...
```

Remarks:

h Specifies the horizontal displacement of the tab stops. SCRIPT/PC will move to the next tab setting by padding with blanks.

Initial Setting: 5 10 15 20 25 30 35 40 45 50

Default: 5 10 15 20 25 30 35 40 45 50

- If no tabs are specified with the Tab Setting control word and options, all tabs are set to the default values.
- You may set a maximum of 10 tab stops with the Tab Setting control word.
- This control word works correctly only with the formatter turned off (**.FO OFF**).

Example:

- The following sets the tab settings for five columns:

```
.FO OFF
.TB 10 20 30 40 50
```

Tab Setting .TB Control Word

SCRIPT/PC will now set the tab settings to character positions (columns) 10, 20, 30, 40, and 50.

Your Notes:

Table of Contents .TC Control Word

Purpose: The Table of Contents [.TC] control word causes a table of contents to be created, imbedded, and printed. Entries are placed in the table of contents under control of heading levels, or directly with the Put Table Of Contents [.PT] control word.

Format:

```
.TC  [ n ]  [ name    ]
            [ control ]
            [ /       ]
```

Remarks:

n	Is the number of pages to be reserved for the table of contents. If you omit this item, 1 is assumed. When you use TWOPASS mode, ***n*** should be correct for the size of the table of contents or there will be page numbering errors.
name	Is an optional item to be used as the title of the table of contents. If this item is omitted, CONTENTS is assumed and printed.
control	Is a control word or macro, starting with a period (.). If ***control*** is used, it is placed in the TOC "as-is," without the normal Heading 1.
/	Indicates that no Heading 1 or no control word is to be used for the TOC.

Table of Contents .TC Control Word

Default: 1 Page

- When a Table of Contents control word is found a page eject is done.
- Formatting of the table of contents is controlled by the line and page values in effect when the Table of Contents control word is found.
- This control word causes a formatting change, called a break.

Example:

- See the table of contents of this book for an example of an automatically created table of contents.

Your Notes:

Terminal Input .TE Control Word

Purpose: Use the Terminal Input [.TE] control word when you want to enter text or a control line during the processing of your input file.

Format:

```
.TE  [ n ]
```

Remarks:

n Specifies the number of lines that will be accepted from your keyboard. If this item is omitted, 1 is assumed.

Default: 1

- When the Terminal Input control word is found, your terminal keyboard is unlocked to accept an input line. The input line you type in may be text or control words. This line will then be processed as if it had been read from an imbedded file. In other words, the Terminal Input control word operates very similar to the Imbed [.IM] control word. Upon completion of reading your terminal's input, SCRIPT will continue processing your input file at the next input line after the Terminal Input control word.

- If you want to set up a prompting sequence, use the Type on Terminal [.TY] control word to set up the prompting messages.

- Once you are in Terminal Input, you may enter one line or the amount of lines designated by *n*.

To stop Terminal Input mode before you run out of designated lines, you should press the ↵ key twice in succession, causing a null entry.

- You will know the keyboard has been activated when you see the same prompt you see in practice mode (=>).

Example:

- To set up the information for entry into your input file; type:

```
The following student,
.TY Enter name of student!
.TE
.CT , has passed English 101 with a
.TY Enter grade student received!
.TE
grade.
```

Results in the display screen prompt/response sequence:

```
Enter name of Student!
John Adams
Enter grade student received!
B
```

and a printed output of:

```
The following student, John Adams, has
passed English 101 with a B grade.
```

Note: The typed entries are inserted into the input text and are *not* saved for future use.

Your Notes:

Then .TH Control Word

Purpose: The Then [.TH] control word can be used in conjunction with the IF control word to process SCRIPT/PC input lines conditionally. The **target** line is processed only if the most recently performed If, And, or Or control word resulted in a true condition.

Format:

```
.TH     [target]
```

Remarks:

target Is any valid SCRIPT/PC input line, control word or text. If the most recently performed If [.IF] control word was true, the target line is processed next, with the first non-blank character after the Then control word treated as the first position of the subject line. If the condition was false, the target line is ignored and processing continues with the input line that follows the Then control line.

- The Then and Else control words, used with If, And, and Or control words allow you to construct complex logic statements.

- The Then and Else control words do not cause a break or change the true/false condition. However, the target control word might, if it is processed. For example, the input lines:

```
.IF &A = &B
.EL .IF &C = &D
.TH .TY Yes.
```

are equivalent to the line:

```
.IF &A = &B .OR &C = &D .TY YES
```

- More than one Then and Else control word may follow an If control word. When the If control word's comparison answer is true, the Then control word is processed. When the If control word's comparison answer is false, the Else control word is processed.

Example:

- The following input lines:

```
.IF &A = &B .TY Yes.
.IF &A NE &B .TY still
```

are the same as these lines:

```
.IF &A = &B
.TH .TY Yes,
.EL .TY still.
```

Your Notes:

Top Margin .TM Control Word

Purpose: The Top Margin [.TM] control word is used to set the amount of vertical space that should be reserved above the text and running heading at the top of output pages. Information may be printed in the top margin area by using the Running Title [.RT] control word and the TOP option.

Format:

```
.TM          [v]
```

Remarks:

v Signals the amount of vertical lines to be reserved at the top of output pages. If no value is given for v, then the initial value is restored.

Initial Setting: 6 Default: 0

- The value set by the Top Margin control word applies to the page after the Top Margin control word has been read. It will stay at the new value until another .TM is read.
- The size of the top margin is not affected by either the Single-Space [.SS] or Double Space [.DS] control words.
- This control word does not cause a formatting change, called a break.
- You may reserve as many lines as you wish, as long as the top margin plus the bottom margin is not greater than the page length.

Top Margin .TM Control Word

Your Notes:

Translate Character .TR Control Word

Purpose: Use the Translate Character [.TR] control word to change the output representation of any character in the input text. This is useful when you cannot enter a particular character in the input text that you want to see as part of your output text.

Format:

```
.TR    [ s  t ] ...
```

Remarks:

s Is the input (source) character to be translated. It must be a single character.

t Is the desired output character. It may be a single character, a 2-digit hexadecimal number, or a 3-digit decimal number.

Default: Restores the original input characters

- A decimal number must be preceded by a **D.** For example, **.TR A D127.**
- You can define up to five pairs of translate characters with one Translate Character [.TR] control word.
- Use the Translate Character control word when the final output device uses a different character set than the device that was used to create the original input (source) file.

Translate Character .TR Control Word

- Translate characters remain in effect until you change them. For example, to change back to your original character set, write **.TR** in your input file.

- A Translate Character control word with no additional information will cancel all previously translated characters and the character set will be reset to SCRIPT/PC normal values.

- The Translate Character control word does not cause a text break.

Example:

- To allow the printing of periods in column 1, normally the start of a control word, and to change the - (hyphen) to print as a = (equal sign):

```
.SP
The following control words control
Printer vertical paper movement:
.TR / . - =
.SP
/PA - Page Eject
/CP - Conditional Page Eject
/SK - Skip Line(s)
/SP - Space Line(s)
.SP
/CP can also test for remaining
lines on a page.
.TR
.SP
Default SCRIPT/PC printing is at
6 lines-per-inch.
```

- The previous example would display and print as follows:

Translate Character .TR Control Word

```
The following control words control
Printer vertical paper movement:

.PA = Page Eject
.CP = Conditional Page Eject
.SK = Skip Line(s)
.SP = Space Line(s)

.CP can also test for remaining
lines on a page.

Default SCRIPT/PC printing is at
6 lines-per-inch.
```

Your Notes:

Translate Input .TI Control Word

Purpose: Use the Translate Input [.TI] control word to translate the input text from one input character to another character.

Format:

```
.TI [ s  t ] ...
```

Remarks:

s Is the input (source) character to be translated. It may be a single character, a 2-digit hexadecimal number, or a 3-digit decimal number.

t Is the desired output character. It may be a single character, a 2-digit hexadecimal number, or a 3-digit decimal number.

Default: Restores the original input characters.

- You can define up to five pairs of translate input characters with one Translate Input [.TI] control word.
- Translate input remains in effect until you change the input again. For example, to change back to your original character set, write **.TI** in your input file.
- A Translate Input control word with no additional information will reinitialize the normal translation table and will cancel all previously translated characters.

Translate Input .TI Control Word

- The Translate Input control word does not cause a text break.

Example:

- To translate the greater than symbol (>) from your keyboard to be interpreted within your computer as the tab setting character (D009), type the following:

```
.TI > D009
```

  This tells SCRIPT/PC to go to the next tab setting every time it finds the greater than (>) character.

Your Notes:

Type on Terminal .TY Control Word

Purpose: The Type on Terminal [.TY] control word allows you to send a one-line message to your computer's display screen, no matter where the SCRIPT/PC formatted output is being sent.

Format:

```
.TY  [ text [=] ]
```

Remarks:

text Is the line to be displayed on your display screen. It is used only for this message and does not become part of your output document.

= Specifies that no return to a new line is to be done. Your cursor stays at the end of the same line that contains your message.

- When the Type on Terminal control word is found by SCRIPT/PC, the text portion of the control word format is displayed (typed) on the display screen. This line does not become part of your permanent output file, and is not formatted in any way. What you enter on the text line is exactly how it is displayed back to you during SCRIPT/PC processing.

- In general, the Type on Terminal control word should be used to create messages when the formatted output is being sent to a printer or disk.

- This control word can be used to give directions to the person who is SCRIPTing a file. For

Type on Terminal .TY Control Word

example, use this control word when you want to inform the user to put a certain diskette into the IBM Personal Computer.

Example:

```
.TY Remove diskette from Drive A!
.TY Put TEST diskette in Drive A!
```

When the file is being formatted, the messages are displayed on the screen as:

```
Remove diskette from Drive A!
Put TEST diskette in Drive A!
```

Your Notes:

Undent .UN Control Word

Purpose: The Undent [.UN] control word is used to shift only one line (the next line) left from the previous starting column (set up by the Indent control word).

Format:

```
.UN  [ h ]
```

Remarks:

h Signals the number of characters the beginning of the next output line is to be shifted left. This is a temporary change for only one line.

Initial Setting: 0 Default: 0

- The value used in the .UN control word is subtracted from the current indent value to find the starting column for the next line of output text.
- This control word causes a formatting change, called a break.
- The undent operation occurs on the next text, skip, or space line, after the .UN control word is read.

Undent .UN Control Word

Example:

```
.IN 5
This is the best way to look at this
and to understand it.
.UN 5
However, the results may look strange
if it is not used correctly.
```

Results in:

```
     This is the best way to look at this
     and to understand it.
However, the  results  may  look  strange
     if it is not used correctly.
```

Your Notes:

Underscore .US Control Word

Purpose: The Underscore [.US] control word automatically underscores one or more input lines.

Format:

```
.US  [ ON   ]
     [ OFF  ]
     [ line ]
```

Remarks:

ON Signals that the following text lines are to be underscored.

OFF Stops underscoring if it was on.

line Signals that this one line of text is to be underscored.

Initial Setting: OFF Default: *line*

- Use the Underscore control word whenever you have a line of text to be underlined.
- The Underscore control word does not cause an automatic break; single words within a sentence may be underscored in **.FO ON** and **.FO LEFT**

Underscore .US Control Word

Examples:

- Underscoring a single word:

```
This sentence has
.US one
underscored word.
```

Results in:

This sentence has <u>one</u> underscored word.

- Underscoring more than one line:

```
.FO ON
.UD OFF
.US ON
This sentence will be totally underscored.
This one will also be underscored.
.US OFF
However, this sentence will not be underscored.
```

Results in:

<u>This sentence will be totally underscored. This one will also be underscored</u>. However, this sentence will not be underscored.

Your Notes:

Underscore and Capitalize .UC Control Word

Purpose: The Underscore and Capitalize [.UC] control word automatically underscores and capitalizes one or more input lines.

Format:

```
.UC [ ON   ]
    [ OFF  ]
    [ line ]
```

Remarks:

ON Indicates the start of a series of lines that are to be underscored and capitalized.

OFF Indicates that underscoring and capitalizing are to stop.

line Is a single text line to be underscored and capitalized.

Initial Setting: OFF Default: *line*

- Use the Underscore and Capitalize control word whenever you have a line of data that is to be formatted in capital letters and underlined. This control word combines the function of the Underscore [.US] and Uppercase [.UP] control words.
- The Underscore and Capitalize control word does not cause a break in text; single words in a sentence can be underscored and capitalized.

Underscore and Capitalize .UC Control Word

Example:

- Underscoring and capitalizing a single word.

```
This sentence has
.UC one
word processed by the control word.
```

Results in:

This sentence has ONE word processed by the control word.

Your Notes:

Underscore Definition .UD Control Word

Purpose: Use the Underscore Definition [.UD] control word to specify whether blank characters should be underscored whenever underscoring is being done by the use of the Underscore [.US] or Underscore and Capitalize [.UC] control words. It is also in effect when a print font that calls for underscoring is being used.

Format:

```
.UD [ ON  ]
    [ OFF ]
```

Remarks:

ON Specifies that blanks are to be underscored.

OFF Specifies that blanks are not to be underscored.

Default: ON

- This control word does not cause a formatting change, called a break.
- This control word is operative only when **.FO ON** is in effect. If **.FO OFF** is in effect, all character positions, including blanks, will be underscored.

Underscore Definition .UD Control Word

Example:

- To turn on underscoring blanks:

```
.UD ON
.
.
.UC chemistry 202
```

Your Notes:

Uppercase .UP Control Word

Purpose: The Uppercase [.UP] control word automatically capitalizes one or more input lines.

Format:

```
.UP [ ON   ]
    [ OFF  ]
    [ line ]
```

Remarks:

ON Signals that the following text lines are to be capitalized.

OFF Stops capitalizing if it was on.

line Signals that this one line of text is to be capitalized.

Initial Setting: OFF Default: *line*

- Use the Uppercase control word whenever you have a line of text to be formatted in capital letters.
- The Uppercase control word does not cause an automatic break; single words within a sentence may be capitalized.

Uppercase .UP Control Word

Examples:

- Capitalizing a single word:

```
This sentence has
.UP one
capitalized word.
```

Results in:

This sentence has ONE capitalized word.

Your Notes:

Diagnostic Mode .ZZ Control Word

Purpose: The Diagnostic Mode [.ZZ] control word is used for tracing input lines and viewing symbol substitution.

Format:

```
.ZZ  { ON }
     { OFF }
```

Remarks:

ON Signals that all input lines are to be traced and all symbol substitution is to be displayed.

OFF Stops diagnostic mode if it was on.

Initial Setting: OFF

Your Notes:

Chapter 2. Understanding Tags

Contents

Introduction to Tags

SCRIPT/PC provides a Starter Set of markup tags for assisting in text markup. These tags, also known as macros, are made up of many SCRIPT/PC control words. In place of entering the same string of control words every time you need the same function, you can select a tag to do it for you. In other words, the Starter Set tags that are part of the Generalized Markup Language provide you with an additional, and in many cases, easier, and shorter method to mark up a document for text formatting.

The Starter Set tags provided by SCRIPT/PC will assist you in learning an efficient way to format your documents. The tags are of great assistance in marking up a formal book or booklet, and are set up to automatically do a large part of your organizing of a document into front matter, body, appendix, and back matter.

The tags are kept in a special disk library file where each tag is known as a macro. The tag name and macro name are always the same. The library containing your Starter Set Tags is located on your main diskette under the filename of GMLLIB.SCT. Each of the Starter Set tags contains many control words. However, each tag is filed under its special name in the library, somewhat as you would file a book in a real library. When SCRIPT/PC reads a tag in your input file it:

- Searches the specified library until it finds the correct name
- Reads the control words in the macro (remember, the tag is known as a macro when it is inside the library)
- Follows each control word in sequence until it is told to end
- Returns to the input file to read the next input line

The tags provided by SCRIPT/PC are a compatible subset of the set in the Starter Set of Generalized Markup Language (GML) Tags. The Starter Set of tags are provided by the System/370 Document Composition Facility (DCF) product, also known as SCRIPT/VS.

The tags (macros) provide a "shorthand" method to do repetitive functions frequently used in text processing and text formatting applications. Instead of entering long strings of SCRIPT/PC control words over and over again, you can enter the "shorthand" tags that allow you to do make shorter entries and create fewer mistakes. In addition to the tags provided in GMLLIB.SCT, you may wish to write your own tags. You may put them in the file GMLLIB.SCT or under your own filename. To see the format in which tags are written, you may wish to print the file GMLLIB.SCT.

> **Note:** If you have DOS 2.00, you can get a printed copy of the tag library by using the DOS PRINT command.

Each tag is actually a means of expressing a series of SCRIPT/PC control words to the SCRIPT/PC processing program. The control words and other controlling information for a specific tag is kept in an area called a macro library (GMLLIB.SCT or HEADDEFS.SCT). Thus, the macro library contains areas for each tag containing specific control word sequences for each tag. The specific area for a particular tag is composed of three parts. These parts together form a macro, which is addressed by a tag name:

Head — Starts with the define macro [.DM] control word, followed by the tag name. For example, :DM ADDRESS.

Body — Is composed of SCRIPT/PC control words and text (if necessary). There are additional controls that use symbols very extensively.

Tail Is the macro exit [.ME] control word which notifies SCRIPT/PC that this is the end of the tag.

An example of a simple tag is:

```
.DM skiptwo
.SK 2
.IN 2
.ME
```

Where:

.DM Defines the tag beginning control word

skiptwo Specifies the tag name

.SK 2 Is the skip control word that says to skip two lines before beginning the next text line

.IN 2 Is the indent control word that says to indent two character spaces on the next text line

.ME Is the macro exit control word that says to end the tag and return control to the input text.

This example is only meant to show you how a series of control words can be strung together to create a tag. This tag is not in the tag library provided with SCRIPT/PC.

If you look at GMLLIB.SCT, where the Starter Set of tags are kept, you will see that they use the same format. Of course, there is additional memory space for more tags you may want to develop. The use of the Starter Set tags is entirely optional, which allows even more memory space for your own tags, should you choose not to use the Starter Set.

A tag must be defined in a separate file similar to the Starter Set or it can be written *inline* before it is used in an input (source) file that is to be formatted. If the tag

you defined is in a separate file, then the separate file-id must be imbedded into your input (source) file prior to the use of the first tag. Other than this difference, any defined tag may be used in the same way as that of a SCRIPT/PC control word.

To review what was just presented, if you want to use a tag library, make sure the desired library name is entered on the Options Profile you are going to use or imbed it into your input file with an Imbed control word **before** you use the first tag name. In case of duplicate names, the last tag entered will be the one that is used.

The following example imbeds the library (GMLLIB.SCT) into the input file. The tags then build a simple list (:SL) as directed, and insert items into the list (:LI), in this case, the name of states. You will not need to imbed the tag library if you specify GMLLIB on your Options Profile:

```
.IM B:GMLLIB
The following list identifies the
states:
:SL compact
:LI.ALABAMA
:LI.ALASKA
:LI.ARKANSAS
:ESL
```

Results in:

```
The following list identifies the states:

   ALABAMA
   ALASKA
   ARKANSAS
```

The GML Starter Set Library File

The GML Starter Set library (GMLLIB.SCT) file can be located on the DOS default diskette drive or on whichever disk drive you choose on the Options Profile. If you do not select the GML Starter Set library with the Options Profile and you are going to use Starter Set tags, you should imbed the Starter Set library file into your input text. The tag definitions it contains are available for use when:

```
.IM B:GMLLIB
```

is found by SCRIPT/PC. Insertion at the beginning of your input file is suggested because it must be selected or imbedded prior to use of any tag.

Please note that other tag definition files may be imbedded in the same way. Any tags you define can be included to further personalize your text processing. Remember that in the case of duplicate names, the last file that was imbedded contains the tag that will be used.

Tags

The following sections describe each of the tags supplied with the SCRIPT/PC product.

Each tag follows a standard syntax (language rules) which is free form in nature. A tag may even be imbedded within text. This is especially useful when highlighting text and creating certain types of lists. A non-GML tag starts with a period (.) instead of a colon (:) and will not be treated the same as the GML type tag.

The GML Starter Set tags:

- Must start with a colon (:) and may appear anywhere within an input line. You may type more than one GML tag within an input line. However, you may *not* mix tags and control words on the same input line because the results are unpredictable.

  - If an input line starts with a colon (:) or leading blanks followed by a colon (:) and :GDOC has preceded the input line, the entire line is considered to be a tag line. All control words and semicolons on the line will be treated as text. The semicolon (;) will not be recognized as a control word separator until a new line is read.

  - If an input line starts with text, the first tag or control word separator read will determine how the remaining data in the input line is to be treated, as tags and text, or control words and text.

- Must have a blank character position between the tag and any attribute.

- May also be ended by a period, which is usually followed by the text the tag is to modify.

  Note: In this chapter every tag is shown in capital letters. However, you may enter tags in either upper- or lowercase.

The following is an example of the correct way to write Starter Set tags in your input file:

```
:XXX [attribute]
:XXX[.text for tag]
:XXX [attribute][.text for tag]
text :HP1.highlighted text:EHP1. text
:DT.term:DD.definition text
```

The non-GML tags that start with a period are more like control words and must follow control word syntax by always starting in column 1. They must be separated by control word separators (;), if you write

more than one control word on a line. Non-GML tags can be placed into your own file or can be added to GMLLIB.

The following is an example of the correct way to write non-GML tags (control words) in your input file:

```
Column => 1    5    10   15    20
          .XX [value]
          .XX [value];.XX [option]
          text;.XX [value];text
```

Changing the Title Page Format

SCRIPT/PC left justifies all elements of the title page. Another title page format is to center each line of the title page. If you prefer this format, each of the tags described in this section may be altered by editing and changing the uppercase Break control word (**.BR**) as it occurs in each tag to a Center control word (**.CE**) which will produce a "centered" title page format, or a Right Adjust control word (**.RI**) which will produce a right justified page format. Remember that the tags are located in the file **GMLLIB.SCT**

This modification should be done only on the following tags:

- ABSTRACT
- ALINE
- AUTHOR
- DATE
- DOCNUM
- PREFACE

- TITLE
- H0
- H1

Tag Error Processing

As we have already seen, a tag is really an organized collection of SCRIPT/PC control words. As such, the error checking that is done is left to SCRIPT/PC to check each individual control word. There is no particular syntax or relational checking to ensure that a tag you build will work correctly or that tags which are dependent upon each other are specified in the proper order. That will be your job as a part of any tag development you may choose to do. The Diagnostic control word (.ZZ) can assist you in tag development as it will allow you to see how each line of your tag is working under alternative conditions.

It is possible by using set symbols, conditional, type (**.TY , .TE**) and quit (**.QU**) control words to do some error checking and thereby end SCRIPT/PC due to a tag-detected problem.

Tag Note [:MNOTE]

This tag is used internally within some of the other tags described in this document. It is used to display a message, suspend execution so that the message can be seen, and then to end SCRIPT/PC if necessary.

```
:MNOTE.message text
```

Format of Starter Set Tags

The following Starter Set tags are arranged in alphabetical order:

Abstract :ABSTRACT Tag

Purpose: The Abstract tag identifies the summary of a document.

Format:

```
:ABSTRACT
```

Remarks:

Attributes: None

Content: No immediate text

Ended by: :PREFACE, :TOC, or :BODY

- The abstract can only occur within the front matter, after the title page.
- This tag is optional.

Abstract :ABSTRACT Tag

Example:

```
:GDOC
  :FRONTM
    :TITLEP
      :TITLE.The History of
      :TITLE.American Agriculture
      :DOCNUM.987654321
      :DATE
      :AUTHOR.John Doe
      :ADDRESS
        :ALINE.7896 Highlight Court
        :ALINE.Miami, FL  78901
      :EADDRESS
    :ETITLEP
=>  :ABSTRACT
      :p.This book describes the details
      of the SCRIPT/PC control words and
      the Generalized Markup Language (GML).
    :PREFACE
    :TOC
    :FIGLIST
       . . .
       . . .
```

Your Notes:

Address :ADDRESS Tag

Purpose: Use the Address tag to identify a street or mailing address. It is located within the title page. It usually is the address of an author or publisher.

Format:

```
:ADDRESS
```

Remarks:

Attributes: None

Content: No immediate text

Ended by: :EADDRESS

- The address can be placed on the title page and can be repeated.
- It is an optional tag.
- The address can contain more than one address line.
- The Address Line tag [:ALINE] actually places the address. The Address and End Address actually define the beginning and end of the address.

Address :ADDRESS Tag

Example:

```
:GDOC
  :FRONTM
    :TITLEP
      :TITLE.The History of
      :TITLE.American Agriculture
      :DOCNUM.987654321
      :DATE
      :AUTHOR.John Doe
=>    :ADDRESS
        :ALINE.7896 Highlight Court
        :ALINE.Miami, FL  78901
=>    :EADDRESS
    :ETITLEP
    :ABSTRACT
    :PREFACE
    :TOC
    :FIGLIST
      .
```

Your Notes:

Address Line :ALINE Tag

Purpose: The Aline tag identifies the line of an address.

Format:

```
:ALINE.text of address
```

Remarks:

Attributes: None

Content: Text on same line (no other tags)

Ended by: End of text line

- The address line can only be used after the Address tag.
- You may insert more than one address line.

Address Line :ALINE Tag

Example:

```
:GDOC
  :FRONTM
    :TITLEP
      :TITLE.The History of
      :TITLE.American Agriculture
      :DOCNUM.987654321
      :DATE
      :AUTHOR.John Doe
      :ADDRESS
=>      :ALINE.7896 Highlight Court
=>      :ALINE.Miami, FL  78901
      :EADDRESS
    :ETITLEP
    :ABSTRACT
    :PREFACE
    :TOC
    :FIGLIST
       . . .
       . . .
```

Results in address output lines of:

```
7896 Highlight Court
Miami, FL  78901
```

Your Notes:

Appendix Section :APPENDIX Tag

Purpose: The appendix tag identifies that portion of a document that contains explanatory and illustrative material helpful to a reader, but not required by the main text.

Format:

```
:APPENDIX
```

Remarks:

Attributes: None

Content: No immediate text

Ended by: :BACKM or :EGDOC

- The appendix section normally occurs immediately after the body.
- Each appendix, A, B, or C, is correctly page number prefixed starting at A-n, B-n, C-n and so on. The Heading 1 creates a new Appendix and changes the alphabetic character prefix.
- This is an optional tag.

Appendix Section :APPENDIX Tag

Example:

```
:GDOC SEC='Company Restricted'
   :FRONTM
        . . .
        . . .
   :BODY
        . . .
        . . .
        . . .
=> :APPENDIX
      :H1.Executive Telephone List
           . . .
   :BACKM
        . . .
:EGDOC
```

Your Notes:

Author Name :AUTHOR Tag

Purpose: The Author tag identifies the writer of the document.

Format:

```
:AUTHOR.text
```

Remarks:

Attributes: None

Content: Text on same line (no other tags)

Ended by: End of text line

- The author name can only be on the title page.
- It is an optional tag.
- The author tag may be repeated for those documents that have more than one author.

Author Name :AUTHOR Tag

Example:

```
:GDOC
  :FRONTM
    :TITLEP
      :TITLE.The History of
      :TITLE.American Agriculture
      :DOCNUM.987654321
      :DATE
=>    :AUTHOR.John Doe
      :ADDRESS
        :ALINE.7896 Highlight Court
        :ALINE.Miami, FL  78901
      :EADDRESS
    :ETITLEP
    :ABSTRACT
    :PREFACE
    :TOC
    :FIGLIST
 :BODY
```

Your Notes:

Back Matter :BACKM Tag

Purpose: The Back Matter tag identifies the part of a document that includes such reference material as a glossary, bibliography, and an index.

Format:

```
:BACKM
```

Remarks:

Attributes: None

Content: No immediate text

Ended by: :EGDOC

- The back matter normally occurs at the end of the document, immediately after the appendix section, if you have one.
- The Back Matter tag is optional.

Back Matter :BACKM Tag

Example:

```
:GDOC SEC='Company Restricted'
   :FRONTM
        . . .
        . . .
   :BODY
        . . .
        . . .
        . . .
   :APPENDIX
        . . .
=> :BACKM
      :H1.Glossary
           . . .
      :H1.Bibliography
           . . .
      :INDEX
           . . .
:EGDOC
```

Your Notes:

Body :BODY Tag

Purpose: The Body tag identifies that part of a document that contains the main text.

Format:

```
:BODY
```

Remarks:

Attributes: None

Content: No immediate text

Ended by: :APPENDIX, :BACKM, or :EGDOC

- The body normally occurs directly after the front matter.
- It can contain either a series of parts (:H0) or a series of chapters (:H1).
- It generates a conditional new page.

Body :BODY Tag

Example:

```
:GDOC
  :FRONTM
    :TITLEP
       . . .
    :ETITLEP
    :ABSTRACT
    :PREFACE
    :TOC
    :FIGLIST
=>:BODY
    :H1.Chapter 1. Native Trees
      :P.There are approximately .....
       . . .
    :H2.Chapter 2. Native Grasses
```

Your Notes:

Citation :CIT
Tag

Purpose: The Citation tag is used to define an underscored citation from a quoted source.

Format:

```
:CIT.text of citation
```

Remarks:

Attributes: None

Content: Cited text (no other tags)

Ended by: :ECIT

- This tag forces underscoring.

Your Notes:

Document Date :DATE Tag

Purpose: The Document Date tag identifies a date you want to place on the document, such as the date of creation, publication, or revision.

Format:

```
:DATE.text
```

Remarks:

Attributes: None

Content: Text on same line (no other tags)

Ended by: End of line

- The document date can only be placed within the title page.
- If a date is written after the tag, then that date text is inserted into the output text. If no date is written after the tag, the date known to your computer system is used (the same as the symbol &SYSDATE).

Document Date :DATE Tag

Example:

```
  :GDOC
    :FRONTM
       :TITLEP
          :TITLE.The History of
          :TITLE.American Agriculture
          :DOCNUM.987654321
=>        :DATE
          :AUTHOR.John Doe
          :ADDRESS
             :ALINE.7896 Highlight Court
             :ALINE.Miami, FL  34567
          :EADDRESS
       :ETITLEP
       :ABSTRACT
       :PREFACE
       :TOC
       :FIGLIST
        . . .
```

Your Notes:

Definition Description :DD Tag

Purpose: The Definition Description tag identifies a definition, description, or explanation of a word or phrase in a definition list.

Format:

```
:DD.text
```

Remarks:

Attributes: None

Content: Implied paragraph structure

Ended by: :DT, :LP, or :EDL

- The definition description can only occur within a definition list [:DL], immediately following its definition term [:DT.]
- There must be a :DD for every :DT.
- It contains an implied paragraph, and may also contain basic document parts.

Definition Description :DD Tag

Example:

```
:P.This paragraph comes before
a definition list.
:DL TSIZE=7 TERMHI=0
  :DT.TERM1
=>  :DD.First definition description.
    This starts an implied paragraph, a
    paragraph without a paragraph tag.
      :P.This paragraph has a tag.
  :DT.TERM2
=>  :DD.Second definition description.
      :LP.This list part introduces
      the following terms and descriptions.
      :P.A list part may comment about
      preceding terms and descriptions.
  :DT.TERM3
=>  :DD.Third definition description.
    It has two topics:
      :OL
        :LI.This is the first topic.
        :LI.This is the second topic.
      :EOL
  :DT.TERM4
=>  :DD.Fourth definition description.
:EDL
:P.This paragraph follows the list.
```

Definition Description :DD Tag

Results in:

This paragraph comes before a definition list.

TERM1 First definition description. This starts the implied paragraph, a paragraph without a a paragraph start tag.

This paragraph has a tag.

TERM2 Second definition description.

This list part introduces the following terms and descriptions.

A list part may comment about preceding terms and descriptions.

TERM3 Third definition description. It has two topics:

1. This is the first topic.

2. This is the second topic.

TERM4 Fourth definition description.

This paragraph follows the list.

Your Notes:

Definition List :DL
Tag

Purpose: Use the Definition List tag to identify a list of words and phrases and their corresponding definitions, descriptions, or explanations.

Format:

```
:DL [COMPACT] [TSIZE=ss] [TERMHI=h]
```

Remarks:

Attributes: COMPACT , TERMHI= , TSIZE=

h Specifies TERMHI highlighting level.

ss Specifies TSIZE term size in characters.

Content: No immediate text

Ended by: :EDL

- The COMPACT attribute stops the line spacing between list entries.
- The TERMHI attribute defines the type of highlighting for the definition term [:DT.] Its value is a number, 0 or 2, which identifies the type of emphasis associated with the definition terms on this list:

  0 = No Highlighting
  2 = Bold only

Definition List :DL Tag

- The TSIZE attribute defines the maximum length of a definition term [:DT.] Its value is a number of characters. The number is at least one greater than the longest definition term that you want to enter on the list, it may be more.

- The definition list can occur wherever a basic document part can occur. It contains a series of definition list items, which can be mixed with individual parts known as *list parts* (:LP).

- Each Definition List item is actually a word or phrase; the Definition Term (:DT), and its corresponding definition or explanation, the Definition Description (:DD).

Definition List :DL Tag

Example:

```
:P.This paragraph comes before
a definition list.
=>:DL TSIZE=7 TERMHI=0
  :DT.TERM1
    :DD.First definition description.
    This starts an implied paragraph, a
    paragraph without a paragraph tag.
      :P.This paragraph has a tag.
    :DT.TERM2
    :DD.Second definition description.
      :LP.This list part introduces
      the following terms and descriptions.
      :P.A list part may comment about
      preceding terms and descriptions.
    :DT.TERM3
    :DD.Third definition description.
    It has two topics:
      :OL
        :LI.This is the first topic.
        :LI.This is the second topic.
      :EOL
    :DT.TERM4
    :DD.Fourth definition description.
  :EDL
:P.This paragraph follows the list.
```

Definition List :DL Tag

Results in:

This paragraph comes before a definition list.

TERM1 First definition description. This starts the implied paragraph, a paragraph without a a paragraph start tag.

This paragraph has a tag.

TERM2 Second definition description.

This list part introduces the following terms and descriptions.

A list part may comment about preceding terms and descriptions.

TERM3 Third definition description. It has two topics:

1. This is the first topic.

2. This is the second topic.

TERM4 Fourth definition description.

This paragraph follows the list.

Your Notes:

Definition Term :DT Tag

Purpose: The Definition Term tag identifies a word or phrase which is defined, described, or explained by an item of a definition list.

Format:

```
:DT.text of term
```

Remarks:

Attributes: None

Content: Text on same line

Ended by: End of text line

- The definition term can only occur within a definition list [:DL.]
- There can be more than one definition term.
- The definition term's maximum length is defined by the TSIZE=attribute of the definition list.
- It must be followed immediately by its definition description [:DD] item.

Definition Term :DT Tag

Example:

```
:P.This paragraph comes before
a definition list.
:DL TSIZE=7 TERMHI=0
=>:DT.TERM1
    :DD.First definition description.
    This starts an implied paragraph, a
    paragraph without a paragraph tag.
      :P.This paragraph has a tag.
=>:DT.TERM2
    :DD.Second definition description.
      :LP.This list part introduces
      the following terms and descriptions.
      :P.A list part may comment about
      preceding terms and descriptions.
=>:DT.TERM3
    :DD.Third definition description.
    It has two topics:
      :OL
        :LI.This is the first topic.
        :LI.This is the second topic.
      :EOL
=>:DT.TERM4
    :DD.Fourth definition description.
:EDL
:P.This paragraph follows the list.
```

Definition Term :DT Tag

Results in:

```
This paragraph comes before a
definition list.

TERM1  First definition description.
       This starts the implied
       paragraph, a paragraph without a
       a paragraph start tag.

       This paragraph has a tag.

TERM2  Second definition description.

This list part introduces the following
terms and descriptions.

A list part may comment about preceding
terms and descriptions.

TERM3  Third definition description.
       It has two topics:

       1.  This is the first topic.

       2.  This is the second topic.

TERM4  Fourth definition description.

This paragraph follows the list.
```

Your Notes:

Document Number :DOCNUM Tag

Purpose: The Document Number tag identifies a number associated with the document, such as a form, serial, or order number.

Format:

```
:DOCNUM.text of number
```

Remarks:

Attributes: None

Content: Text on same line (no other tags)

Ended by: End of line

- The document number can only occur on the title page.
- This tag is optional.

Document Number :DOCNUM Tag

Example:

```
:GDOC
  :FRONTM
     :TITLEP
        :TITLE.The History of
        :TITLE.American Agriculture
=>      :DOCNUM.987654321
        :DATE
        :AUTHOR.John Doe
        :ADDRESS
           :ALINE.7896 Highlight Court
           :ALINE.Miami, FL  34567
        :EADDRESS
     :ETITLEP
     :ABSTRACT
     :PREFACE
     :TOC
     :FIGLIST
  .BODY
```

Your Notes:

Example :XMP Tag

Purpose: The Example tag identifies the start of an example, such as a diagram, table, or other illustration. It does not have a caption, and it can not be referred to from any other place in your document.

Format:

```
:XMP  [DEPTH=dd]
```

Remarks:

Attributes: DEPTH=

Content: No immediate text

Ended by: :EXMP

- The DEPTH attribute is optional. Its value is the amount of space you need on the page for your example. If there is not enough vertical space left on the page, a page eject is forced. The value "dd" must be a number from 1 through 99.
- A line skip is forced, formatting is turned off, and your example is placed in your output, line for line, as you typed the input.

Example:

```
:XMP
   25 x 25 = 625
   40 x 10 = 400
   15 x 15 = 165
:EXMP
```

Example :XMP Tag

Your Notes:

Figure :FIG Tag

Purpose: The Figure tag identifies a diagram, table, or other illustration.

Format:

```
:FIG  [ID=figure-id]
      [FRAME=RULE  ]
      [      BOX   ]
      [      NONE  ]
      [PLACE=INLINE]
      [      TOP   ]
      [WIDTH=PAGE  ]
      [      ww    ]
     [DEPTH=dd]
```

Remarks:

Attributes: ID=, PLACE=, FRAME=, DEPTH=, WIDTH=

Content: No immediate text

Ended by: :EFIG

- This tag identifies the start of a figure.
- The ID attribute allows you to reference from other parts of your document. It allows you to always refer to this figure by the same unique identifier.

Figure :FIG Tag

- The FRAME attribute allows you to decide what type of highlighting you want surrounding the figure, so that it is easily set apart from the nearby text:

  FRAME=RULE Is assumed if the FRAME attribute is ignored. The rule means that a page width line (rule) will be drawn before and after the figure.

  FRAME=BOX Will cause a box to be placed around the figure.

  FRAME=NONE Will not put any rule or box around the figure.

- The DEPTH attribute is used to make sure that the figure will fit in the space that remains on the page if PLACE=INLINE is used. If you are not going to enter a figure, but only want to reserve the space for later drop-in art, a space (.SP) or skip (.SK) control word should be used.

- The WIDTH attribute can be specified to page width (WIDTH=PAGE) or to a specific number of columns (WIDTH=ww).

- A figure contains a figure body, and can also contain a figure caption [:FIGCAP] and a figure description [:FIGDESC].

- When the figure is referred to by a figure reference [:FIGREF], then ID attribute and the figure caption must be used.

- If the figure is to be listed automatically in the list of illustrations, the figure caption must be used.

- The figure body can contain basic document parts (except figures) and line parts. A line part is an input line that is processed as one output line.
- Formatting off (**.FO OFF**) is the default for the body of the figure, but it can be overridden to formatting on (.FO ON) if you wish.
- You cannot attempt to draw a box, using the "box" character set, within a figure or box if you are using the IBM Graphics Printer.

Example:

```
=>:FIG ID=sx
      This
           is
              the
                  figure
                        space
    :FIGCAP.Map of Cold States
    :FIGDESC.States below 0 are colored.
 :EFIG
      . . .
```

Your Notes:

Figure Caption :FIGCAP Tag

Purpose: Use the Figure Caption tag to identify the title or designation of a figure.

Format:

```
:FIGCAP.caption text
```

Remarks:

Attributes: None

Content: Text on same line (no other tags)

Ended by: End of line

- The figure caption can occur only within a figure.
- It is used to provide caption text to the figure. A caption is usually a word or short phrase which helps identify a figure's purpose.
- The caption phrase appears next to the figure number within the figure.
- This tag must be written between the figure [:FIG] and end figure [:EFIG] tags.
- For an example of this tag, see the description of the figure [.FIG] tag.
- If a reference is made to a figure by the figure reference [:FIGREF] tag or if it is to be listed in the list of illustrations [:FIGLIST], then the figure caption must be used. However, the figure caption tag can be entered without any text.

Figure Caption :FIGCAP Tag

- The figure caption text should be entered with initial capital letters.

Example:

```
 :FIG ID=sx
      This
           is
              the
                  figure
                        space
=>   :FIGCAP.Map of Cold States
     :FIGDESC.States below 0 are colored.
 :EFIG
      . . .
```

Your Notes:

Figure Description :FIGDESC Tag

Purpose: The Figure Description tag identifies an extended comment or description of a figure.

Format:

```
:FIGDESC.description text
```

Remarks:

Attributes: None

Content: Line of text

Ended by: End of Line

- The figure description can occur only within a figure.
- The text is located immediately after the figure caption text. However, it does not appear in the list of illustrations.
- This is an optional tag.

Example:

```
 :FIG ID=sx
      This
           is
              the
                  figure
                        space
   :FIGCAP.Map of Cold States
=> :FIGDESC.States below 0 are colored.
 :EFIG
      . . .
```

Figure Description :FIGDESC Tag

Your Notes:

List of Illustrations :FIGLIST Tag

Purpose: The List of Illustrations tag identifies a listing of the figures in the document and the page numbers on which they occur.

Format:

```
:FIGLIST
```

Remarks:

Attributes: None

Content: Generated by SCRIPT/PC

Ended by: End of markup

- A table of illustrations is similar to a table of contents.
- The twopass option must be used if you want the list of illustrations to appear at the beginning of your document.
- This tag is optional.
- If your list of illustrations is larger than 2 pages, the symbol &FIGLST must be changed in the same manner as the &TOC symbol is changed for the table of contents. For example, if you need 4 pages for your list of illustrations:

```
.SE FIGLST = 4
:TOC
```

List of Illustrations :FIGLIST Tag

Example:

```
:FRONTM
   :TITLEP
      . . .
   :ETITLEP
   :ABSTRACT
      . . .
   :PREFACE
      . . .
=> :FIGLIST
:BODY
```

Your Notes:

Figure Reference :FIGREF Tag

Purpose: The Figure Reference tag identifies a reference to a figure, this is sometimes identified as a "figure call-out".

Format:

```
:FIGREF [REFID=reference-header-id]
        [PAGE=YES ]
        [     NO  ]
```

Remarks:

Attributes: REFID=, PAGE=

Content: Generated by SCRIPT/PC

Ended by: End of markup

- This tag is used in input text to refer to a figure somewhere else in the document.
- The REFID attribute identifies the mandatory reference header ID. Its value must match the ID of the figure being referenced.
- For all references that precede the input line that references them, twopass mode must be set as a SCRIPT/PC processing option.

Figure Reference :FIGREF Tag

Example:

```
:P.:FIGREF REFID=sx.Illustrates
that the states that have
the coldest weather have a large
commitment to agriculture.
```

Your Notes:

Footnote :FN Tag

Purpose: Use this Footnote tag to identify a note of reference, explanation, or comment to be placed below the text on the output page.

Format:

```
:FN [ID=reference id].footnote text
```

Remarks:

Attributes: None

Content: Text on line

Ended by: :EFN

- Each footnote has a unique identifying number that starts with 1 and increments by 1.
- The text surrounding the footnote reference appears as it is written in the input text with the additional inclusion of the footnote reference number if the :FNREF tag is used.
- The space at the bottom of the output page *must be reserved* for the placing of the footnote. It is reserved by using the special system variable $FN on a page prior to where you need the footnote.

Example:

If your footnote(s) will take up three lines, you must reserve four lines, three for your footnote and one for the footnote identification line. The reserving of the four lines must be done prior to the page on

which it will be used. The set symbol [.SE] control word is used to set the special system variable $FN. The amount of lines reserved is dependant upon the value of this special system variable. It is shown as follows:

```
.SE $FN = 4
.PA
This is the best cherry pie ever baked.
=> :FN.I realize this cherry pie could
   be controversial. However, who am I to
   deny my Mother's pie?
   :EFN
      . . .
.SE $FN = 0
The quality and taste of the cherries
are superb.
```

Results in:

```
This is the best cherry pie ever
baked. The quality and taste of the
cherries are superb.
      . . .
      . . .
      . . .
------------------
1  I realize this cherry pie could be
   controversial. However, who am I to
   deny my Mother's pie?
```

Your Notes:

Footnote Reference :FNREF Tag

Purpose: Use this Footnote Reference tag to place a superscript callout next to the most current footnote.

Format:

```
:FNREF REFID=reference id
```

Remarks:

Attributes: REFID

Content: Footnote callout done in superscript

Ended by: End of line or next tag

- The reference id must refer to an ID which was established by a Footnote tag [:FN.]

Your Notes:

Front Matter :FRONTM Tag

Purpose: The Front Matter tag identifies a part of a document that contains material that serves as a guide to the document's contents and audience. Items included in the front matter are: title page, abstract, preface, table of contents, list of illustrations.

Format:

```
:FRONTM
```

Remarks:

Attributes: None

Content: No immediate text

Ended by: :BODY

- The Front Matter tag may only be placed at the beginning of a document, following the General Document tag.
- This tag forces single-column mode and causes a conditional page eject.

Front Matter :FRONTM Tag

Example:

```
:GDOC
=> :FRONTM
     :TITLEP
        :TITLE.The History of
        :TITLE.American Agriculture
        :DOCNUM.987654321
        :DATE
        :AUTHOR.John Doe
        :ADDRESS
           :ALINE.7896 Highlight Court
           :ALINE.Miami, Florida
        :EADDRESS
     :ETITLEP
     :ABSTRACT
     :PREFACE
     :TOC
     :FIGLIST
  :BODY
      . . .
```

Your Notes:

General Document :GDOC Tag

Purpose: The General Document tag identifies a general document. It is usually placed on the first input line of your document, following the Imbed control word that identifies your macro file, normally GMLLIB.SCT.

Format:

```
:GDOC [SEC='security class']
      .bottom title text
```

Remarks:

Attribute: SEC=

Content: Text for bottom title (optional)

Ended by: :EGDOC

- This tag and the end general document tag *must* be used. The rest of the document depends upon some of the procedures started by the General Document tag.
- The SEC attribute identifies the optional security classification of a document.
- To change the running title set by this tag, you should change the running title on line 2 as follows:

  ```
  .RT BOTTOM 2 /part1/part2/part3/
  ```

  Where:

  / Is used to separate the information in the three parts of the title line. This is

called a delimiter character and is determined by the first non-blank character after the line number (n).

part1 Is the portion of the title that is to be left justified.

part2 Is the portion of the title that is to be centered between the left and right margins.

part3 Is the portion of the title that is to be right justified.

Example:

```
=> :GDOC SEC='Top Secret'
      :FRONTM
         . . .
      :BODY
         . . .
      :APPENDIX
         . . .
      :BACKM
         . . .
=> :EGDOC
```

Your Notes:

Heading Zero :H0 Tag

Purpose: The Heading Zero tag prints the title within the document as well as insert a title line into the table of contents. It is used to identify "parts" of a document.

Format:

```
:H0 [ID=id] [STITLE=short title]
   [.title text]
```

Remarks:

Attributes: ID=, STITLE=

Content: Text on same line (no other tags)

Ended by: End of line (see notes)

- The heading should be entered with initial capital letters.
- If the heading is referred to by a Heading Reference tag, the ID must be defined.
- The ID attribute is used to provide a label for this part of the document. This label can be referenced from another place in the document. All IDs should have unique names.
- This tag causes a skip within the table of contents.
- This tag can be used to separate major sections in the table of contents.
- It causes a skip of five lines within the text of the output document.

Heading Zero :H0 Tag

- The STITLE attribute identifies the optional "short title." Use the short title to replace the bottom running title text.
- The "part" started by the Heading 0 is ended by another Heading 0, appendix [:APPENDIX] or back matter [:BACKM] tag.

Example:

```
:H0 ID=r1 STITLE=Dormancy.Weather and Dormancy
```

Your Notes:

Heading One :H1 Tag

Purpose: The Heading One tag prints the title within the document as well as insert a title line into the table of contents. It is used to identify "chapters" of a document.

Format:

```
:H1 [ID=id] [STITLE=short title]
   [.title text]
```

Remarks:

Attributes: ID=, STITLE=

Content: Text on line (no other tags)

Ended by: End of line (see notes)

- The heading should be entered with initial capital letters.
- If the heading is referred to by a Heading Reference tag, the ID must be defined.
- The ID attribute is used to provide a label for this part of the document. This label can be referenced from another place in the document. All IDs should have unique names.
- This tag causes a skip within the table of contents.
- This tag can be used to separate chapters in the table of contents.

Heading One :H1 Tag

- It prints the title text (underscored, boldface and capitalized) and then skips three lines within the text of the output document.
- The STITLE attribute identifies the optional "short title." Use the short title to replace the bottom running title text.
- The "chapter" started by the heading 1 is ended by another heading 1, a heading 0 [:H0], the Appendix or Back Matter tag.

Example:

```
:H1 ID=q2 STITLE=States.Environment of States
```

Your Notes:

Heading Two :H2 Tag

Purpose: The Heading Two tag identifies a second-level heading. It inserts a heading within the document and an entry into the table of contents. It is used to identify parts of a chapter of a document.

Format:

```
:H2 [ID=id][.title text]
```

Remarks:

Attributes: ID=

Content: Text on same line (no other tags)

Ended by: End of line (see notes)

- The heading should be entered with initial capital letters.
- If the heading is referred to by a Heading Reference tag, the ID must be defined.
- The ID attribute is used to provide a label for this part of the document. This label can be referenced from another place in the document. All IDs should have unique names.
- This tag can be used to separate major sections in the table of contents.
- It skips three lines, prints the second-level text in uppercase, underscored, and bold, then skips two lines before resuming text formatting. When the

Heading Two :H2 Tag

second-level heading is ready to be printed, if there are less than six lines remaining on the page, a page eject is forced.

- The second-level heading can be stopped by another heading 2 or lower number (higher priority) Heading tag, by a front matter part, or by the Body, Appendix or Back Matter tag.

Example:

```
:H2 ID=t8.Plants that Need Low Temperatures
```

Your Notes:

Heading Three :H3 Tag

Purpose: The Heading Three tag identifies a third-level heading. It inserts a heading within the document and an entry into the table of contents. It is used to identify parts of a chapter of a document.

Format:

```
:H3 [ID=id][.title text]
```

Remarks:

Attributes: ID=

Content: Text on same line (no other tags)

Ended by: End of line (see notes)

- The heading should be entered with initial capital letters.
- If the heading is referred to by a Heading Reference tag, the ID must be defined.
- The ID attribute is used to provide a label for this part of the document. This label can be referenced from another place in the document. All IDs should have unique names.
- This tag is indented two spaces as it is inserted into the table of contents.
- It skips three lines, prints the third-level text in uppercase and bold, then skips two lines before resuming text formatting. When the third-level

Heading Three :H3 Tag

heading is ready to be printed, if there are less than six lines remaining on the page, a page eject is forced.

- The third-level heading can be stopped by another Heading 3 or lower number (higher priority) Heading tag, by a front matter part, or by the Body, Appendix or Back Matter tag.

Example:

```
:H3 ID=z2.Properties of the Cadmium Layers
```

Your Notes:

Heading Four :H4 Tag

Purpose: The Heading Four tag identifies a fourth-level heading. It inserts a heading within the document as well as it inserts an entry into the table of contents. It is used to identify parts of a chapter of a document.

Format:

```
:H4 [ID=id][.title text]
```

Remarks:

Attributes: ID=

Content: Text on same line (no other tags)

Ended by: End of line (see notes)

- The heading should be entered with initial capital letters.
- If the heading is referred to by a Heading Reference tag, the ID must be defined.
- The ID attribute is used to provide a label for this part of the document. This label can be referenced from another place in the document. All IDs should have unique names.
- This tag is indented four spaces as it is inserted into the table of contents.
- It skips three lines, prints the fourth-level text only with boldface, then skips two lines before resuming text formatting. When the fourth-level

Heading Four :H4 Tag

heading is ready to be printed, if there are less than six lines remaining on the page, a page eject is forced.

- The fourth-level heading can be stopped by another Heading 4 or lower number (higher priority) Heading tag, by a front matter element, or by the Body, Appendix or Back Matter tag.

Example:

```
:H4 ID=f2.Groups of Northern Plants
```

Your Notes:

Heading Five :H5 Tag

Purpose: The Heading Five tag identifies a fifth-level heading. It only inserts a heading within the document. It is used to identify parts of a chapter of a document.

Format:

```
:H5 [ID=id][.title text]
```

Remarks:

Attributes: ID=

Content: Text on same line (no other tags)

Ended by: End of line (see notes) or another tag

- The heading should be entered with initial capital letters.
- If the heading is referred to by a Heading Reference tag, the ID must be defined.
- The ID attribute is used to provide a label for this part of the document. This label can be referenced from another place in the document. All IDs should have unique names.
- It skips one line, prints the fifth-level text in boldface, no lines are skipped before resuming text formatting. When the fifth-level heading is ready to be printed, if there are less than four lines remaining on the page, a page eject is forced.

Heading Five :H5 Tag

- The fifth-level heading can be stopped by another Heading 5 or lower number (higher priority) Heading tag, by a front matter part, or by the Body, Appendix or Back Matter tag.

Example:

```
:H5 ID=xw.Shaping and Pruning for Snow
```

Your Notes:

Heading Six :H6 Tag

Purpose: The Heading Six tag identifies a sixth-level heading. It only inserts a heading within the document. It is used to identify parts of a chapter of a document.

Format:

```
:H6 [ID=id][.title text]
```

Remarks:

Attributes: ID=

Content: Text on same line (no other tags)

Ended by: End of line (see notes)

- The heading should be entered with initial capital letters.
- If the heading is referred to by a Heading Reference tag, the ID must be defined.
- The ID attribute is used to provide a label for this part of the document. This label can be referenced from another place in the document. All IDs should have unique names.
- It skips one line, prints the sixth-level text only with underscoring, no lines are skipped before resuming text formatting. When the sixth-level heading is ready to be printed, if there are less than four lines remaining on the page, a page eject is forced.

Heading Six :H6 Tag

- The sixth-level heading can be stopped by another Heading 6 or lower number (higher priority) Heading tag, by a front matter part, or by the Body, Appendix or Back Matter tag.

Example:

```
:H6 ID=g5.Example of a Leaf
```

Your Notes:

Heading Reference :HDREF Tag

Purpose: The Heading Reference tag identifies a reference to a heading elsewhere in the document and, optionally, also mentions the page on which the reference starts.

Format:

```
:HDREF [REFID=reference-header-id]
       [PAGE=YES ]
       [     NO  ]
```

Remarks:

Attributes: REFID=, PAGE=

Content: Generated by SCRIPT/PC

Ended by: End of markup

- This tag is used in input text to refer to a heading somewhere else in the document.
- The REFID attribute identifies the mandatory reference heading-id. Its value must match the ID of the heading being referenced.
- The PAGE attribute indicates whether the heading's page number should be included in the reference.
- For all references that precede the input line that references them, twopass mode must be set as a SCRIPT/PC processing option.

Heading Reference :HDREF Tag

Example:

```
    . . .
:P.When the cold weather departs,
the trees that have survived their
dormant period, see :HDREF REFID=ca
are once again beginning to show signs
of life.
    . . .
```

Your Notes:

Highlight Phrase 1 :HP1 Tag

Purpose: This Highlight Phrase tag causes all text to be underscored until the End Highlight Phrase [:EHP1] tag is found.

Format:

```
:HP1.text
```

Remarks:

Attributes: None

Content: Text item

Ended by: :EHP1

- There are three highlighting phrase tags in the GML Starter Set.
- This tag forces underscoring.

Example:

```
This important message
:HP1.must:EHP1. be read by all
members of the club.
```

Results in:

This important message must be read by all members of the club.

Your Notes:

Highlight Phrase 2 :HP2 Tag

Purpose: This Highlight Phrase tag causes all text to be printed in bold font until the End Highlight Phrase [:EHP2] tag is found.

Format:

```
:HP2.text
```

Remarks:

Attributes: None

Content: Text item

Ended by: :EHP2

- There are three highlighting phrase tags in the GML Starter Set.
- This tag forces bold font.

Example:

```
This important message
:HP2.must:EHP2. be read by all
members of the club.
```

Results in:

This important message **must** be read by all members of the club.

Your Notes:

Highlight Phrase 3 :HP3 Tag

Purpose: This Highlight Phrase tag causes all text to be underscored as well as printed in bold font, until the End Highlight Phrase [:EHP3] tag is found.

Format:

```
:HP3.text
```

Remarks:

Attributes: None

Content: Text item

Ended by: :EHP3

- There are three highlighting phrase tags in the GML Starter Set.
- This tag forces both underscoring and bold font.

Example:

```
This important message
:HP3.must:EHP3. be read by all
members of the club.
```

Results in:

This important message **must** be read by all members of the club.

Your Notes:

Index :INDEX Tag

Purpose: Use the Index tag to specify where the index is to be inserted in your document.

Format:

```
:INDEX
```

Remarks:

Attributes: None

Content: No immediate text

Ended by: End of document

- This tag *must* be at the end of the document following the glossary (if any).
- This is an optional tag.

Example:

```
  :BODY
      . . .
      . . .
  :APPENDIX
      . . .
  :BACKM
      . . .
=>    :INDEX
:EGDOC
```

Your Notes:

Index Entry Reference :IREF Tag

Purpose: The Index Entry Reference tag creates or modifies an index entry. It can be used to provide a reference to another index.

Format:

```
:IREF [REFID=refid]  [PG={START}]
                     [    {END}]
      [SEEID=seeid]
```

Remarks:

Attributes: REFID=, PG=, SEEID=

Content: No immediate text

Ended by: End of text line

- REFID identifies the index entry term whose subject is to be used as the subject of this entry.
- PG is the same as for Index Entry Term [:I1.]
- SEEID identifies the ID of a primary index entry whose subject is a cross-reference for this subject.

Your Notes:

Index Entry Term :I1
Tag

Purpose: The Index Entry Term tag identifies a word or phrase that will be the subject of a primary index entry. This tag creates the index entry.

Format:

```
:I1 [ID=id]  [PG={START}]  [.index term]
             [    {END}]
```

Remarks:

Attributes: ID=, PG=,

Content: Line of text (no other tags)

Ended by: End of text line

- Use the ID attribute to identify a name which can be used to refer to this index entry. Each ID should be unique to each index entry with no duplicate IDs.
- The PG attribute allows a reference range to be defined via the START and END attribute values.
- The "index term" is the actual text that appears in the index.

Index Entry Term :I1 Tag

Example:

```
:P.This type of cherry tree, that more
.* **********************
:I1 ID=3C.Cherries
.* **********************
suited to pies and desserts, is known as
the Montmorency
.* **********************
:IREF REFID=3C.Montmorency
.* **********************
cherry, whose fruit is of good size and
fine flavor. However, the all time
favorite still appears to be the Bing
.* **********************
:IREF REFID=3C.Bing
.* **********************
cherry, whose sweetness is unmatched.
.* **********************
:I1 SEEID=3C.Red Fruit
.* **********************
```

The comment lines (.* *********) allow an easy method for identifying index entries in your input text. The comment lines will *not* appear in your output text or your index.

Your Notes:

List Item :LI
Tag

Purpose: The List Part tag identifies an entry in a simple, unordered, or ordered list.

Format:

```
:LI.list text
```

Remarks:

Attributes: None

Content: Implied paragraph structure

Ended by: :LI, :LP, :ESL, :EUL, or :EOL

- This tag is used to specify each entry within a simple, unordered, or ordered list.
- This tag can be used more than once within the list.
- The list item starts a new paragraph without the need for a paragraph tag. This is called an implied paragraph.
- There is no limit to the number of list entries.
- It forces a line skip before each entry unless the COMPACT attribute had been specified for the list.
- List entries for ordered lists should be entered without sequence numbers or letters. These sequence items will be inserted by SCRIPT/PC.

Example:

```
Trees:
:UL
=> :LI.Examples of Northern Trees
      :OL COMPACT
=>      :LI.Elm
=>      :LI.Maple
=>      :LI.Oak
      :EOL
=> :LI.Examples of Southern Trees
      :SL
=>      :LI.Magnolia
=>      :LI.Pecan
=>      :LI.Orange
      :ESL
:EUL
```

Results in:

```
Trees:

o   Examples of Northern Trees

    1.   Elm
    2.   Maple
    3.   Oak

o   Examples of Southern Trees

         Magnolia

         Orange

         Pecan
```

Your Notes:

List Part :LP
Tag

Purpose: The List Part tag starts the text identified by the tag under the symbol or sequence number of a list. It temporarily changes the format of the list, which is returned to normal when the next item tag is read. It normally identifies a comment or explanation which is supporting information to an item in the list (list item).

Format:

```
:LP.text
```

Remarks:

Attributes:	None
Content:	Implied paragraph structure
Ended by:	:DT, :LI, :EDL, :EOL, :ESL, or :EUL

- The list part can only occur within a list.
- This tag is optional and there may be more than one tag within a list.
- The list part starts a new paragraph without the need for a paragraph tag. This is called an implied paragraph.
- This tag divorces the paragraph text from the list entry.
- It aligns the list part with the number or bullet of the list item, or with the definition term.

Example:

```
Trees:
:UL
  :LI.Examples of Northern Trees
     :OL
       :LI.Elm
=>     :LP.The following northern trees are
       important to the furniture business.
       :LI.Maple
       :LI.Oak
     :EOL
  :LI.Examples of Southern Trees
     :OL COMPACT
       :LI.Magnolia
=>     :LP.The following trees have a
       commercial value.
       :LI.Pecan
       :LI.Orange
     :EOL
:EUL
```

Results in:

Trees:

o Examples of Northern Trees

1. Elm

The following northern trees are very important to the furniture business.

2. Maple

3. Oak

o Examples of Southern Trees

1. Magnolia
The following trees have a commercial value.
2. Orange
3. Pecan

List Part :LP
Tag

Your Notes:

Long Quotation :LQ Tag

Purpose: The Long Quotation tag identifies an excerpt, or "block quotation," which is identified by an unique format.

Format:

```
:LQ.text
```

Remarks:

Attributes: None

Content: No immediate text

Ended by: :ELQ

- The long quotation tag identifies the quote by skipping a line before and after the quote, and indenting the quote four character spaces from both the left and right margins.
- The long quotation can contain lists. However, it cannot contain figures.

Example:

```
This is an extremely volatile business.
However, as
our founder has stated many times:
=>:LQ.The rewards of the business
  makes all the risk and
  sacrifice worth it.
=>:ELQ
With this in mind, I am sure we will
have a productive year.
```

Long Quotation :LQ Tag

Results in:

```
This is an extremely volatile business.
However, as our founder has stated many
times:

    The rewards of the business makes
    all the risk and sacrifice worth
    it.

With this in mind, I am sure we will
have a productive year.
```

Your Notes:

Note :NOTE Tag

Purpose: The Note tag identifies a paragraph containing a comment or explanation that you wish to call to a reader's attention.

Format:

```
:NOTE.text
```

Remarks:

Attributes: None

Content: text

Ended by: :P, :PC, another :NOTE or any tag other than highlighting.

- The word **Note:** is placed at the beginning of your text and is highlighted in bold font.
- The note forces a line skip after it is complete.
- The note is formatted flush left with the current left margin (modified by indentation).

Example:

```
This survey shows very conflicting
results.
=>:NOTE.This survey covers only 25 years.
:P.The question of whether the earth is
heating up or cooling down has not been
determined from the test data.
```

Note :NOTE Tag

Results in:

```
This survey shows very conflicting
results.

Note: This survey covers only 25 years.

The question of whether the earth is
heating up or cooling down has not been
determined from the test data.
```

Your Notes:

Ordered List :OL Tag

Purpose: The Ordered List tag identifies a list of items, called "list items", whose order is important. The ordered list is printed with a numeric or alphabetic sequence.

Format:

```
:OL [COMPACT]
```

Remarks:

Attributes: COMPACT

Content: No immediate text

Ended by: :EOL

- The COMPACT attribute will stop line skipping between list items. It is an optional item.
- It forces a line skip before starting the list, places a numeric character in front of the list item you specified, and then indents your list item text for better visual identity.
- You may use the List Part tag in the middle of your ordered list.
- List items should not be entered with their own numbering system. The ordered list tag will number them for you.

Ordered List :OL Tag

Example:

```
  Trees:
=>:OL
    :LI.Examples of Northern Trees
=>    :OL COMPACT
        :LI.Elm
        :LI.Maple
        :LI.Oak
=>    :EOL
    :LI.Examples of Southern Trees
=>    :OL
        :LI.Magnolia
        :LI.Orange
        :LI.Pecan
=>    :EOL
=>:EOL
```

Results in:

```
Trees:

1.  Examples of Northern Trees

    a.   Elm
    b.   Maple
    c.   Oak

2.  Examples of Southern Trees

    a.   Magnolia

    b.   Orange

    c.   Pecan
```

Your Notes:

Paragraph :P Tag

Purpose: The Paragraph tag identifies the beginning of a paragraph. A paragraph is one or more sentences that relate to the same subject matter.

Format:

```
:P.text
```

Remarks:

Attributes: None

Content: Paragraph unit

Ended by: Paragraph or higher priority tag

- A line skip is forced before a new paragraph is started.
- It forces a page eject if there are less than two lines left on a page.

Example:

```
  Many more are chosen than rejected.
=>:P.While peaches are more thought
  of as belonging to Georgia. New York
  is one  of the largest commercial peach
  growers in the United States.
  Surprisingly, there are peach orchards
  to be observed beside
  Lake Ontario in Canada.
=>:P.Processing of the fruit is done
  near the orchards.
```

Paragraph :P
Tag

Results in:

Many more are chosen than rejected.

While peaches are more thought of as belonging to Georgia. New York is one of the largest commercial peach growers in the United States. Surprisingly, there are peach orchards to be observed beside Lake Ontario in Canada.

Processing of the fruit is done near the orchards.

Your Notes:

Paragraph Continuation :PC Tag

Purpose: Use the Paragraph Continuation tag to continue a paragraph that has been interrupted by an address, example, figure, list, or long quotation.

Format:

```
:PC.text
```

Remarks:

Attributes: None

Content: Paragraph unit

Ended by: Paragraph or higher priority tag

- Use this tag to continue the main paragraph after it has been interrupted.
- This tag will not cause a line skip.

Example:

```
  This is an extremely volatile business.
  However, as
  our founder has stated many times,
  :LQ.The rewards of the business make
  all the risk and
  sacrifice worth it.
  :ELQ
=>:PC.and with this in mind, I am sure
  we will have a productive year.
```

Paragraph Continuation :PC Tag

Results in:

```
This is an extremely volatile  business.
However, as our founder has  stated many
times,

    The rewards of the business make all
    the risk  and   sacrifice  worth it.

and with this in mind, I am sure we will
have a  productive year.
```

Note: The tag :LQ defines the start of a long quotation and :ELQ defines the end of the long quotation.

Your Notes:

Preface :PREFACE Tag

Purpose: The Preface tag identifies the writer's introductory remarks, such as acknowledgements, or reasons for writing the document.

Format:

```
:PREFACE
```

Remarks:

Attributes: None

Content: No immediate text

Ended by: :TOC, :FIGLIST, or :BODY

- The preface can only occur in the front matter, after the title page (and abstract, if any).
- This is an optional tag.
- A conditional page eject is set. The word **PREFACE** is printed at the top of the output page, capitalized, made bold, and underscored.

Preface :PREFACE Tag

Example:

```
:GDOC
  :FRONTM
    :TITLEP
      :TITLE.The History of
        :TITLE.American Agriculture
        :DOCNUM.987654321
        :DATE
        :AUTHOR.John Doe
        :ADDRESS
          :ALINE.7896 Highlight Court
          :ALINE.Miami, FL  34567
        :EADDRESS
    :ETITLEP
    :ABSTRACT
=>  :PREFACE
      :p.This book has been written
      to explain to my wife and children
      just what Daddy does for a living.
      I hope they think it was worth all
      the long hours, since they are my
      inspiration.
    :TOC
    :FIGLIST
       . . .
```

Your Notes:

Quote :Q
Tag

Purpose: The Quote tag identifies a phrase in which the exact words of a person or text are cited.

Format:

```
:Q.text of quotation
```

Remarks:

Attributes: None

Content: Text item

Ended by: :EQ

- The quote should only use the cited quotation.
- The quote and end quote tags causes the cited text to be bracketed by double quotes (").

Your Notes:

Simple List :SL Tag

Purpose: The Simple List tag identifies a list of items, called "list items." It differs from other lists in that the items are usually short and not emphasized. They are not printed with number systems or asterisks.

Format:

```
:SL  [COMPACT]
```

Remarks:

Attributes: COMPACT

Content: No immediate text

Ended by: :ESL

items, each identified by the List Item tag.

- The Simple List tag forces a line skip, and indents the list item to emphasize it.
- Use the COMPACT attribute to stop the line skipping between list items.
- The simple list can be interrupted by the List Part tag.

Example:

```
=> :SL
     :LI.Apples
=>     :SL COMPACT
         :LI.McIntosh
         :LI.Macoun
         :LI.Cortland
         :LI.Delicious
         :LI.Northern Spy
=>     :ESL
     :LI.Cherries
     :LI.Peaches
=> :ESL
```

Results in:

```
Apples

    McIntosh
    Macoun
    Cortland
    Delicious
    Northern Spy

Cherries

Peaches
```

Your Notes:

Table of Contents :TOC Tag

Purpose: The Table of Contents tag identifies a listing of the parts of a document and the pages on which they begin.

Format:

```
:TOC
```

Remarks:

Attributes: None

Content: Generated by SCRIPT/PC

Ended by: End of markup

- The table of contents can only occur within the front matter, after the title page (and abstract and preface, if any).
- This tag is optional.
- The :TOC tag only reserves 1 page for the table of contents. If you need more pages, you must change the setting of the &TOC symbol. For example, if you need 4 pages for your table of contents:

```
.SE &TOC = 4
:TOC
.
.
.
```

Table of Contents :TOC Tag

Example:

```
:GDOC
  :FRONTM
    :TITLEP
      :TITLE.The History of
      :TITLE.American Agriculture
      :DOCNUM.987654321
      :DATE
      :AUTHOR.John Doe
      :ADDRESS
        :ALINE.7896 Highlight Court
        :ALINE.Miami, FL  34567
      :EADDRESS
    :ETITLEP
    :ABSTRACT
    :PREFACE
=>  :TOC
    :FIGLIST
  :BODY
```

Your Notes:

Title :TITLE Tag

Purpose: The Title tag identifies the name of the document.

Format:

```
:TITLE[.text of title line]
```

Remarks:

Attributes: None

Content: Text on same line (no other tags)

Ended by: End of line

- The title is underscored for emphasis.
- It can be used as many times as it is necessary to print a title.
- The document title must be the first item on the title page, after the Title Page [:TITLEP] tag.
- Each title is left justified on the page.
- If no text follows this tag, a single line skip is generated.

Title :TITLE Tag

Example:

```
:GDOC
  :FRONTM
    :TITLEP
=>    :TITLE.The History of
=>    :TITLE.American Agriculture
      :DOCNUM.987654321
      :DATE
      :AUTHOR.John Doe
      :ADDRESS
        :ALINE.7896 Highlight Court
        :ALINE.Miami, FL  34567
      :EADDRESS
    :ETITLEP
    :ABSTRACT
    :PREFACE
    :TOC
    :FIGLIST
  :BODY
```

Your Notes:

Title Page :TITLEP Tag

Purpose: The Title Page tag identifies the part of the front matter that contains general information that must appear on the title page: title, document number, date, and author(s) and their addresses.

Format:

```
:TITLEP
```

Remarks:

Attributes: None

Content: No immediate text

Ended by: :ETITLEP

- This tag defines the start of the title page.
- It causes a conditional page eject to occur.
- It also sets up conditions to check on all other tags that appear on the title page.
- The title page can occur only as the first part of the front matter.
- The End Title Page tag forces a conditional page eject. It also causes a check to be set up so that tags that must be on the title page are flagged as errors, if you try to use them somewhere else.

Title Page :TITLEP Tag

Example:

```
:GDOC
  :FRONTM
=>  :TITLEP
      :TITLE.The History of
      :TITLE.American Agriculture
      :DOCNUM.987654321
      :DATE
      :AUTHOR.John Doe
      :ADDRESS
        :ALINE.7896 Highlight Court
        :ALINE.Miami, FL  34567
      :EADDRESS
=>  :ETITLEP
    :ABSTRACT
    :PREFACE
    :TOC
    :FIGLIST
  :BODY
```

Your Notes:

Unordered List :UL Tag

Purpose: The Unordered List tag identifies a list of items, called "list items", whose order is unimportant. The ordered list is printed with a bullet (o) or hyphen (–) to show sequences.

Format:

```
:UL [COMPACT]
```

Remarks:

Attributes: COMPACT

Content: No immediate text

Ended by: :EUL

- The COMPACT attribute will stop line skipping between list items. It is an optional item.
- It forces a line skip before starting the list, places a bullet character in front of the list item you specified, and then indents your list item text for better visual identity.
- You may use the List Part tag in the middle of your unordered list.
- List items should not be entered with their own numbering system. The unordered list tag will order them for you.

Example:

```
   Trees:
=> :UL
     :LI.Examples of Northern Trees
=>     :UL COMPACT
         :LI.Elm
         :LI.Maple
         :LI.Oak
=>     :EUL
     :LI.Examples of Southern Trees
=>     :UL
         :LI.Magnolia
         :LI.Orange
         :LI.Pecan
=>    :EUL
=> :EUL
```

Results in:

Trees:

o Examples of Northern Trees

  - Elm
  - Maple
  - Oak

o Examples of Southern Trees

  - Magnolia

  - Orange

  - Pecan

Your Notes:

More on List Tags

This section contains examples of the simple, unordered, and ordered lists using the same input text, only the tags have been changed. This should assist you in understanding how to use the various list tags.

List control tag names may start in any column and must be ended by a blank or period (.) as shown in the examples. Imbedding of lists is allowed.

In the examples that follow, the formatted output is shown first. This allows you to see how a list looks before you see the format of the file that created the list.

> **Note:** Remember that lists may be mixed and imbedded as deeply as you wish.

Simple List [:SL] and [:ESL]

A simple list creates a sequence of indented paragraphs. The following examples show a simple list without indentation first and then with indentation. Note that every list that is started must be terminated by a corresponding ending tag.

Example 1: SIMPLE list (one level) output

The following states have very cold winters:

Alaska

Maine

New Hampshire

Vermont

New York

Michigan

Minnesota

North Dakota

Montana

Idaho

Example 1: SIMPLE list (one level) input

```
:P.The following states have very
cold winters:
:SL
:LI.Alaska
:LI.Maine
:LI.New
Hampshire
:LI.Vermont
:LI.New York
:LI.Michigan
:LI.Minnesota
:LI.North Dakota
:LI.Montana
:LI.Idaho
:ESL
```

Example 2: SIMPLE list (four levels) output

The following states have very cold winters:

- Alaska
- Some New England states that are cold:
  - Maine
    - Unique among northern states:
      - Farthest North
      - Farthest East
      - Earliest official winter
    - Most snow
    - Most remote
  - New Hampshire
  - Vermont
- New York
- Michigan
- Minnesota
- North Dakota
- Montana
- Idaho

Example 2: SIMPLE list (four levels) input

```
:P.The following states have very
cold winters:
:SL
:LI.Alaska
:LI.Some New England states that are cold:
:SL
:LI.Maine
:SL
:LI.Unique among northern states:
:SL COMPACT
:LI.Farthest North
:LI.Farthest East
:LI.Earliest official winter
:ESL
:LI.Most snow
:LI.Most remote
:ESL
:LI.New
Hampshire
:LI.Vermont
:ESL
:LI.New York
:LI.Michigan
:LI.Minnesota
:LI.North Dakota
:LI.Montana
:LI.Idaho
:ESL
```

Unordered List [:UL] and [:EUL]

An unordered list creates a sequence of bulleted paragraphs. The following examples show a unordered list without indentation and then with indentation. Note that every list that is started must be terminated by a corresponding ending tag. The bullet character changes for every level.

Example 3: UNORDERED list (one level) output

The following states have very cold winters:

- Alaska
- Maine
- New Hampshire
- Vermont
- New York
- Michigan
- Minnesota
- North Dakota
- Montana
- Idaho

Example 3: UNORDERED list (one level) input

```
:P.The following states have very
cold winters:
:UL
:LI.Alaska
:LI.Maine
:LI.New
Hampshire
:LI.Vermont
:LI.New York
:LI.Michigan
:LI.Minnesota
:LI.North Dakota
:LI.Montana
:LI.Idaho
:EUL
```

Example 4: UNORDERED list (four levels) output

The following states have very cold winters:

- Alaska
- Some New England states that are cold:
  - Maine
    - — Unique among northern states:
      - Farthest North
      - Farthest East
      - Earliest official winter
    - — Most snow
    - — Most remote
  - New Hampshire
  - Vermont
- New York
- Michigan
- Minnesota
- North Dakota
- Montana
- Idaho

Example 4: UNORDERED list (four levels) input

```
:P.The following states have very
cold winters:
:UL
:LI.Alaska
:LI.Some New England states that are cold:
:UL
:LI.Maine
:UL
:LI.Unique among northern states:
:UL compact
:LI.Farthest North
:LI.Farthest East
:LI.Earliest official winter
:EUL
:LI.Most snow
:LI.Most remote
:EUL
:LI.New
Hampshire
:LI.Vermont
:EUL
:LI.New York
:LI.Michigan
:LI.Minnesota
:LI.North Dakota
:LI.Montana
:LI.Idaho
:EUL
```

Ordered List: [:OL] and [:EOL]

An ordered list creates a sequence of numbered paragraphs. The following examples show a ordered list without indentation and then with indentation. Note that every list that is started must be terminated by a corresponding ending tag.

Example 5: ORDERED list (one level) output

The following states have very cold winters:

1. Alaska
2. Maine
3. New Hampshire
4. Vermont
5. New York
6. Michigan
7. Minnesota
8. North Dakota
9. Montana
10. Idaho

Example 5: ORDERED list (one level) input

```
:P.The following states have very
cold winters:
:OL
:LI.Alaska
:LI.Maine
:LI.New
Hampshire
:LI.Vermont
:LI.New York
:LI.Michigan
:LI.Minnesota
:LI.North Dakota
:LI.Montana
:LI.Idaho
:EOL
```

Example 6: ORDERED list (four levels) output

The following states have very cold winters:

1. Alaska
2. Some New England states that are cold:
    a. Maine
        1) Unique among northern states:
            a) Farthest North
            b) Farthest East
            c) Earliest official winter
        2) Most snow
        3) Most remote
    b. New Hampshire
    c. Vermont
3. New York
4. Michigan
5. Minnesota
6. North Dakota
7. Montana
8. Idaho

Example 6: ORDERED list (four levels) input

```
:P.The following states have very
cold winters:
:OL
:LI.Alaska
:LI.Some New England states that are cold:
:OL
:LI.Maine
:OL
:LI.Unique among northern states:
:OL compact
:LI.Farthest North
:LI.Farthest East
:LI.Earliest official winter
:EOL
:LI.Most snow
:LI.Most remote
:EOL
:LI.New
Hampshire
:LI.Vermont
:EOL
:LI.New York
:LI.Michigan
:LI.Minnesota
:LI.North Dakota
:LI.Montana
:LI.Idaho
:EOL
```

Chapter 3. How to Use Symbols

Contents

Introduction to Symbols

A symbol is a name in your input file that can be replaced with something else when SCRIPT/PC starts to format your file.

An alternative to referring to control words, page numbers, strings of characters, and values that change, by their true names, is to use symbols. Symbols are nothing more than changing a temporary name to a permanent name. A symbol has a name and a value. When SCRIPT/PC finds a symbol name, it replaces that name with the symbol's current value, and after all symbol names on an input line have been replaced by their current values, SCRIPT/PC processes the input line.

There are a number of steps to using symbols in your text.

1. A symbol you want to use must first be defined, similar to someone putting a word into a dictionary.

2. Once defined, the symbol may now be written in your text. Before SCRIPT/PC will process any input line, it checks to see if there are any symbols by checking for words on the input line that begin with an ampersand (&). The ampersand (&) is used to tell SCRIPT/PC that the rest of the word is a symbol instead of text or a control word. Once recognized by SCRIPT/PC, the symbol is replaced with it's current value.

3. Once all symbols on an input line are replaced, SCRIPT/PC will then process the input line.

To get a better understanding of the process, let us look at an example of defining and using a symbol. To define a symbol you must use the Set Symbol [.SE] control word. The symbol's value can be:

1. A character string.

2. A numeric value.

3. An arithmetic expression.

4. Another symbol.

If a symbol's value contains blanks or special characters, you must enclose the whole value in single quotes.

Character Strings as Symbol Values

The following is an example of using a symbol with a character string for a value:

```
.SE long = antidisestablishmentarianism
```

I agree that the word used is very long and hard to remember to spell. However, once defined, it can be used in the following manner:

```
The longest word in the dictionary is &long..
The definition of &long is: The resistance
to separation of a church or religious body
from a country or state.
```

The results of SCRIPT/PC processing the input text is:

```
The longest word in the dictionary is
antidisestablishmentarianism. The definition
of antidisestablishmentarianism is: The
resistance to separation of a church or
religious body from a country or state.
```

As you can see, once you have defined a symbol, you can use it by inserting an ampersand in front of your symbol name. Every input line is scanned by SCRIPT/PC before processing the line to check for symbols. If any are found, they are replaced with their current symbol value before processing of the line is done. In the previous example, the whole file will look like:

```
.SE long = antidisestablishmentarianism
.
.
.
The longest word in the dictionary is
&long.. The definition of &long is:
The resistance to
separation of a church or religious body
from a country or state.
```

To use a character string that contains blanks, close the string of characters on both ends by using single quotes (').

```
.SE corp = 'International Business Machines'
```

Numeric Values as Symbol Values

- To set up an initial value:

  .SE number = 1

  .SE dime = 10

  .SE dollar = 1.00

Arithmetic Expressions as Symbol Values

Some other examples of symbol definitions are:

- To increment (add 1) a symbol:

  ```
  .SE incr = &number + 1
  ```

- To decrement (subtract 1) a symbol:

  ```
  .SE decr = &number - 1
  ```

- Make sure you use lowercase and uppercase letters as you really want them. In the following example, **test** and **TEST** are replaced with two different values. Both symbols are valid.

```
.SE test = &number + 100
.
.
.
.SE TEST = &number - 100
```

Control Words as Symbol Values

The following are examples of defining symbol values as control words:

- To cause a page eject:

```
.SE page  = '.PA 1'
```

- To cause two pages to eject:

```
.SE pages = '.PA 2'
```

- To reserve space for a figure:

```
.SE figure = '.SP 10'
```

Example:

```
.SE figure = '.SP 10'
This text is to be followed by a figure:
&figure
.CM When this page is processed, there will
.CM be 10 lines of space at this point.
The above figure can now be analyzed.
```

Results in:

```
This text is to be followed by a figure:


The above figure can now be analyzed.
```

Another Symbol as a Symbol's Value

A symbol can be used to replace another symbol's value. SCRIPT/PC now does a second replacement until a true value is established. The following are examples:

```
.SE A = 50000
.SE half   = &A
.SE profit = &half
```

If the previous set symbols are in effect:

```
This has been a successful campaign.
Our net figures are &profit for this
campaign, an increase of 35%.
I would like to congratulate all of you.
```

Results in:

```
This has been a successful campaign. Our
net figures are 50000 for this
campaign, an increase of 35%. I would
like to congratulate all of you.
```

Symbols Known to SCRIPT/PC

SCRIPT/PC has already used a number of symbols. These symbols are reserved. You may know about some of them already. Each Symbol must appear in your input text exactly as you see it, uppercase. If it is followed immediately by more text, it must be ended with a period (.).

&

This symbol by itself becomes the page number symbol. It is also used as the first character of every symbol within text that must be replaced with another value. SCRIPT/PC is always looking for an ampersand (&) in your input text. If it finds it all alone, then SCRIPT/PC inserts the page number. If other characters are included with the &, SCRIPT/PC will search for a set symbol that matches the name after the &. When it finds a matching name, the symbol is replaced by the value to the right of the = sign.

&SYSDATE

When SCRIPT/PC reads this symbol, it substitutes today's date (as known to DOS).

&SYSTIME

When SCRIPT/PC reads this symbol, it substitutes the time in hours and minutes(as known to DOS).

&SYSDAYOFM

This symbol causes substitution of the day of the month.

&SYSHOUR

When SCRIPT/PC reads this symbol, it substitutes the hour (as known to DOS).

&SYSMINUTE

When SCRIPT/PC reads this symbol, it substitutes the minutes (as known to DOS).

&SYSSECOND

When SCRIPT/PC reads this symbol, it substitutes the seconds (as known to DOS).

&SYSYEAR

When SCRIPT/PC reads this symbol, it substitutes the year (as known to DOS).

&*

This symbol is only used in tags (macros). It is replaced by all the words following the tag (macro) when the tag is read. This is how you can pass information to the tag (macro). For example, the tag for a paragraph really breaks down into:

```
.DM P
.CP 2
.SK 1
&*
.ME
```

If you look at the tag (macro) on a line by line basis, you can see how it works:

.DM p Name of tag (p for paragraph)

```
.CP 2     Page eject if less than 2 lines
.SK 1     Skip one line to start paragraph
&*        Insert text that follows the name
.ME       End this tag (macro)
```

If you now type:

```
:p.The elm tree has been destroyed by disease.
```

SCRIPT/PC will replace the &* in the tag (macro) with the words:

```
The elm tree has been destroyed by disease.
```

&*0, &*18

&*1 thru &*17

These are the symbols that are replaced by the words that are written after a tag name. For example, the first word replaces &*1 and the tenth word replaces &*10. Therefore the information (called parameters) you can pass to the tag can be identified by these 17 symbols (called token variables).

&*0 and &18 Special Definitions

- &*0 This symbol replacement will contain the count of all non-null tokens in the character string that replaces &*.

- &*18 This symbol replacement will contain all the tag text which follows the period (.).

Starter Set Symbols

The Starter Set reserves a special set of symbols for its uses. The symbols it reserves are as follows:

&rbl. Required blank

Use this symbol to force SCRIPT/PC to place a blank character in the output file. SCRIPT/PC might not place a blank character where you would want or, in some cases, would place more than one blank character in your text where you only wanted one. Use this symbol to keep text together, text that may be split apart by the formatter, if the text is at the end of a line. In the example, **$ 100.00,** you do *not* want the dollar sign ($) separated from the amount. You should type the example in the following manner:

```
$&rbl.100.00
```

which will print as **$ 100.00** every time, no matter how close to the end of the output line.

This symbol also forces SCRIPT/PC to stop placing extra spaces into your output line, as in justified text.

You may also use this symbol to create blank lines within examples (See :XMP.) or figures (See :FIG) by placing the required blank symbol on an input line by itself. Use the required blank symbol to guarantee yourself space wherever you need it.

&gml. The GML delimiter (:).

Use this symbol to replace the colon on a page where you are writing about a tag. In normal text, each colon is printed as entered. However, in SCRIPT/PC, the colon (:) is always being searched for as the start of a tag. We must, if we are writing about tags, change the colon to a symbol. If we do not, then SCRIPT/PC will not look at it as text but as a tag. For example, If you are writing

a description about a tag you developed, called **BOLD.**, then you will have to type it on your input line as:

```
&gml.BOLD
```

so that it may be printed as **:BOLD** . If you want it to be identified to SCRIPT/PC as a tag instead of text, you would type it as:

```
:BOLD
```

&amp. The ampersand (&).

This is similar to the problem you had with the colon. The ampersand (&) is always being searched for as the start of a symbol. When you really want to print an ampersand you must replace the ampersand character (&) with the ampersand symbol. Use this symbol to replace the ampersand in a symbol you only want to print, not replace:

```
&amp.HELLO
```

will be printed as **&HELLO** instead of being interpreted by SCRIPT/PC as a symbol.

&semi. The semicolon (;).

The semicolon is used by SCRIPT/PC to identify a control word separator. If you are writing about control word separators, this symbol gives you a way to type them on an input line and print them as text, not have SCRIPT/PC interpret them as separating control words. For example, to print the input line **.FO OFF;.IN 10;.SP 2** , you would type it as:

```
.SE . =  per.
   .
&per.FO OFF&semi.&per.IN 10&semi.&per.SP
```

You will also need to use this symbol if you want to use a semicolon within your own symbol values, or in the value of a tag attribute. **&semi.** will always print as a semicolon (;).

Special Characters

There are two characters reserved for special functions within SCRIPT/PC. They are the underscore and the single quote.

1. The underscore is used to force required blanks within your output file. To allow the underscore to print, you will have to change the required blank symbol (&rbl.). For example,

   ```
   .DC RB %
   ```

   The new required blank character will now be the percent sign (%).

2. The single quote has the following rules:

   a. A quote defines the start of a quoted series of characters (called a string) if the quote immediately follows a space or equal sign (=). For example,

      ```
      .SE ADDR = 'Bar  Harbor, Maine'
      ```

   b. A quote that immediately follows another character is treated as text. For example,

      ```
      .SE GOODBYE = See'ya.
      ```

      This assumes that a quoted string has not been started.

   c. A quote within a quoted string must be identified by two quotes to get the one quote into your output. For example,

```
.SE WHERE = 'See''ya at Bar Harbor, Maine'.
```

d. If an ending quote is not found on an input line that contained a starting quote, then the ending quote is assumed when the input line is completed. For example,

```
.SE GOOD = 'Vanilla ice cream;.SP 5
```

Diagnostic Mode

If you are going to modify a tag or write your own tags, you will be interested in the Diagnostic Mode control word (.ZZ). Use this control word if you want to see each control word and substitution take place. You can set it up by starting Diagnostic Mode (.ZZ ON) at the point you want to begin detailed observation. After Diagnostic Mode has been turned on, you will see the SCRIPT/PC process of decoding and formatting commands, text, tags (macros) and set symbols on the display screen. To return to normal formatting, turn Diagnostic Mode off (.ZZ OFF).

Format:

```
.ZZ  {ON}
     {OFF}
```

Where:

ON Specifies to SCRIPT/PC to turn Diagnostic Mode on.

OFF Specifies to SCRIPT/PC to turn Diagnostic Mode off.

Appendixes

Contents

Appendix A. Understanding Messages

Introduction to Messages

This section summarizes the messages that SCRIPT/PC detects and will report to you.

These messages will appear immediately upon detection and most can be stored in the error message file (d:filename.SCE), depending on the option you chose when you selected SCRIPT/PC. Some GML warning messages cannot be stored in an error message file. Also displayed will be the failing line number and filename.

The format of each message is:

Message Text

Problem: Information on why SCRIPT/PC issued the message.

Solution: Information on how you can correct the problem.

Messages are issued for two reasons, error or information only. Most of the explanations of the messages are self-explanatory; however, when a message is issued:

1. Read the message text carefully and if necessary, the explanation of the message and its solution.

2. Correct the reason for the message.

3. SCRIPT your input file again to make sure that the problem was corrected.

   Note: In some cases, correcting one problem, such as the misspelling of your macro filename, will correct a large number of messages. Remember that if you are using the IBM PC Personal Editor you *must* save your files using the NOTABS option. If you do not, page printing problems are likely to occur.

The following messages are arranged in alphabetical order.

Are you sure you want to quit (Y or N)?

Problem: You pressed the Esc (Escape) key to exit.

Solution: If you want to exit, type **Y** for YES. If you *do not* want to exit, type **N** for NO.

Arithmetic error in .SE or .RV control word.

Problem: SCRIPT/PC has detected an arithmetic error during symbol substitution.

Solution: Edit the input file that has the problem. Check for arithmetic errors, such as divide by zero. Correct the problem, save the input file and start SCRIPT/PC again.

AUTHOR macro valid only on title page. Do you want to continue (Enter Y or N)?

Problem: The tag :AUTHOR was found on a line that was not between :TITLEP (Title Page) and :ETITLEP (End Title Page).

Solution: If you answer:

Y Formatting will continue. The correction of the problem can be left for a later time.

N Formatting of your input file is ended. Exit SCRIPT/PC and edit the problem file. Make the correction, save the corrected file and start SCRIPT/PC again.

Box control word has an invalid option.

Problem: SCRIPT/PC found the input line .BX before a box was started by the box control word (.BX nn nn) or where the column starting positions (nn) are in decreasing numbers (for example, .BX 40 30 20 10) instead of increasing numbers (for example, .BX 10 20 30 40)

Solution: Exit SCRIPT/PC and edit the problem file. Check all box commands for the problems listed above. Make the correction, save the corrected file and start SCRIPT/PC again.

Box control word invalid in multicolumn mode.

Problem: SCRIPT/PC detected a box control word (.BX) while it was formatting in multicolumn (two-column) mode.

Solution: Exit SCRIPT/PC and edit the problem file. Check for the box control word (.BX) following the multicolumn mode (.MC) control word. Either delete the box control word(s) or go

back to single column mode before the box (.SC). Make the correction, save the corrected file and start SCRIPT/PC again.

Center control word text is longer than 1 line.

Problem: The line you are trying to center is too long for the current line length. SCRIPT/PC will print the problem line as-is and then continue to format your file.

Solution: Exit SCRIPT/PC and edit the problem file. Check that the length of the line(s) you are trying to center (.CE) does not exceed the line length (.LL nn). Make the correction, save the corrected file and start SCRIPT/PC again.

Choose: 1 or 2.

Problem: The character entered is not a 1 or 2.

Solution: Enter 1 for a desired 1-pass mode. Enter 2 if (for example) you have a Table-of-Contents control word or tag at the front of your document.

Choose: Number greater than starting page number.

Problem: Either the starting page number is too high or the ending page number is too low.

Solution: Check your page number range again. The starting page number *must* be lower than the ending page number, Reenter either the starting or ending page number.

Choose: Number greater than zero.

Problem: Zero (0) is not allowed as a starting page number. The starting page number must be at least 1 .

Solution: Reenter a valid page number.

Choose: S, D, or P.

Problem: The output device you selected is not S, D, or P.

Solution: Reenter either S for screen, D for disk, or P for printer.

Choose: Y or N.

Problem: You did not choose Y or N for the highlighted field. These are the only acceptable answers.

Solution: Reenter Y or N.

Column Definition not defined before Multicolumn mode selected.

Problem: The multicolumn control word (.MC) in your input file was not preceded by a column definition control word (.CD 2 nn nn).

Solution: Exit SCRIPT/PC and edit the problem file. You must establish a column definition control word (.CD 2 nn nn) *before* you issue a multicolumn control word (.MC). Make the correction, save the corrected file and start SCRIPT/PC again.

Conditional IF, AN, or OR has an invalid option.

Problem: Your input file has identified a symbol that is not defined in the conditional IF, AN, and OR control words, for example, .IF &new NE &old .IM UPDATE.SCT might contain a symbol (&new or &old) that has not been defined.

Solution: Exit SCRIPT/PC and edit the problem file. Examine your IF, AN, and OR conditional control words for a misspelled or undefined symbol (those words that start with &). Make the correction, save the corrected file and start SCRIPT/PC again.

Conditional Section control word has an invalid option.

Problem: SCRIPT/PC detected use of a invalid option (operand) in the conditional section control word (.CS n [ON | OFF][INCLUDE | IGNORE]).

Solution: Exit SCRIPT/PC and edit the problem file. Check the spelling and use of your options (operands). Make the correction, save the corrected file and start SCRIPT/PC again.

Conditional Section control word number must be 1 through 9.

Problem: SCRIPT/PC found that the numeric value of the conditional section identified in your input file did not fall between 1 and 9.

Solution: Exit SCRIPT/PC and edit the problem file. Check that all conditional section control words contain a numeric value of 1 through 9. Make the correction, save the corrected file and start SCRIPT/PC again.

Control word or tag cannot be found.

Problem: A control word or tag was found in your input file that SCRIPT/PC does not recognize. It is ignored.

Solution: Exit SCRIPT/PC and edit the input file that has the problem. Check for spelling errors (such as transposition) and either correct the spelling or remove the problem control word or tag. Save the corrected file and then start SCRIPT/PC again.

d:SCRIPT.MSG file contains error or cannot be found.

Problem: SCRIPT/PC cannot find the Error Message file or the file contains an error. This file must be on the same disk or directory that contains the program SCRIPT.EXE.

Solution: Check to ensure that the file specified is on the diskette that contains the program SCRIPT.EXE or in the current directory.

d:SCRIPT.SCR file contains error or cannot be found.

Problem: SCRIPT/PC cannot find the Display Screen file. This file must be on the same disk or directory that contains the program SCRIPT.EXE.

Solution: Check to ensure that the file specified is on the diskette that contains the program SCRIPT.EXE or in the current directory.

Define Font control word defined more than 15 fonts.

Problem: The maximum number of fonts (15) have been defined.

Solution: You cannot define any more fonts (.DF font-id definition) within this session. You have reached a maximum number of 15. Exit SCRIPT/PC and edit the input file you want to change. Remove the extra definition(s) or redefine an existing font, and then save the input file. Start SCRIPT/PC again.

Directory is full, or file specification is incorrect.

Problem: The directory cannot have another entry.

Solution: Use a new output diskette for your output.

Diskette cannot be written on.

Problem: You are not allowed to write on this diskette. It has a write protect tab or no cutout for a tab at all.

Solution: Remove this diskette from the drive and remove the write protect tab (if you wish) or replace this disk with a disk that has the tab removed. Place the diskette in the drive and try SCRIPT/PC again.

Diskette has been permanently damaged.

Problem: A permanent diskette error has occurred. SCRIPT/PC is ended. SCRIPT/PC has detected a read error on the diskette. This occurs when a bad block check or the inability to locate a sector of data on the diskette is detected.

Solution: Exit SCRIPT/PC to DOS. You must do a DISKCOPY to a new diskette from your defective one. Go to your DOS Guide to Operations manual for the correct procedure for your personal computer and the DOS version you are using. When you have completed copying your failing diskette, start SCRIPT/PC again to check for correct operation. If the error occurred on your SCRIPT/PC diskette, you may have to use your SCRIPT/PC backup diskette if the newly copied diskette cannot be made to work. If you use your backup diskette as your main diskette, be sure to make a new backup diskette.

Diskette is not ready.

Problem: This problem normally occurs when the diskette drive door is open.

Solution: Close the drive door after ensuring that the correct diskette has been placed in the drive.

Diskette space on output diskette is full.

Problem: The diskette has no more room for files.

Solution: Replace this diskette with a new diskette, close the drive door, and start SCRIPT/PC again.

DOCNUM macro valid only on title page. Do you want to continue (Enter Y or N)?

Problem: The tag :DOCNUM (Document Number) was found on a line that was not between :TITLEP (Title Page) and :ETITLEP (End Title Page).

Solution: If you answer:

Y Formatting will continue. The correction of the problem can be left for a later time.

N Formatting of your input file is ended. Exit SCRIPT/PC and edit the problem file. Make the correction, save the corrected file and start SCRIPT/PC again.

Double Width font not allowed in Multicolumn mode.

Problem: The font selected has tried to print double width characters while in multicolumn mode (two-column).

Solution: Exit SCRIPT/PC and edit the input file that has a problem.

1. Respecify the font you want to one that works in Multicolumn mode.

2. Change multicolumn mode to single-column mode (.SC) before you get to the double width font part of your file. When you have completed making the changes necessary, save the input file and start SCRIPT/PC again.

Error "xxxxxx...xxxxxxxx" occurred on line *nnn* in file xxxxx

Problem: Self-documenting.

Solution: Edit the input file with the problem. Correct the problem on the line specified. Save the input file and start SCRIPT/PC again.

Error line: xxxxxxx...xxxxxxx

Problem: Self-documenting.

Solution: Edit the input file with the problem. Correct the problem on the line specified. File the input file and start SCRIPT/PC again.

Error message file cannot be written. Error recording is turned off.

Problem: SCRIPT/PC has stopped permanently recording error messages.

Solution: Exit the SCRIPT/PC program. Insert the correct diskette that contains the error message file, or erase unused files to create enough space to write the error message file. Start SCRIPT/PC again.

File-id cannot be found on specified disk.

Esc Cancel **F10 Continue**

Problem: The filename (d:filename.ext) you entered did not match any filename on the selected input disk.

Solution:

1. If the wrong input diskette is in your drive, you may insert the correct diskette and try again by pressing F10. Processing of your input file will continue.

2. Check the drive specifier, filename and extension carefully. You may have misspelled one of them. If you have, press Esc. If you are in Interactive Mode you will be returned to the Main Menu. If you are in Batch Mode you will be returned to DOS.

3. The drive specifier may have been entered incorrectly. You must reselect your options if you had created a temporary profile, otherwise select choice 1 (Formatting a File) and type the file-id again.

Filename "xxxxxxxx.xxx" already exists. Do you want to overwrite (Y/N)?

Problem: The output filename you selected cannot be created by SCRIPT/PC because it already exists.

Solution: If you want the data within the output file to be erased and new data to be written by SCRIPT/PC, type **Y** for YES. If you want the data within the output file to be kept and you want to create a different output filename, type **N** for NO.

Filename is required.

Problem: You did not enter a filename.

Solution: Enter a valid filename.

Font specified has not been defined.

Problem: SCRIPT/PC does not recognize the font-id you have selected with the begin font control word.

Solution: Exit SCRIPT/PC and edit the input file that contains the problem. Check for a define font control word (.DF font-id definition) containing the font-id for the font you want and check the begin font control word (.BF font-id) for the correct

spelling of the font-id you have selected. Make the correction, save the input file and start SCRIPT/PC again.

Footnote space at bottom of page has been filled.

Problem: SCRIPT/PC ran out of space at the bottom of the page to put your footnotes.

Solution: Exit SCRIPT/PC and edit the problem file. Find the set symbol control word that reserves the footnote space (.SE $FN = vv) and increase the vertical line space reserved for your footnotes or cut down on the number of footnotes on the page or their length (:FN.footnote text). Make the correction, save the corrected file and start SCRIPT/PC again

Formatting is on page: nn

Problem: None - This is an information message.

Solution: None needed.

Goto control word found outside of a macro.

Problem: A Goto control word (.GO label) was found outside of a macro in a macro library or outside of a macro written in-line in your input file.

Solution: Exit SCRIPT/PC and edit the input file that contains the problem. Remove the goto control word (.GO label), save the input file and start SCRIPT/PC again.

Imbed control word nested more than 4 levels.

Problem: SCRIPT/PC detected more than 4 levels of imbedded files. SCRIPT/PC processing is ended.

Solution: Exit SCRIPT/PC. Combine your files until you have no more than 4 levels of the imbed

control word (.IM d:filename.ext). Make the correction, save the corrected file and start SCRIPT/PC again.

Invalid options for the control word .xx.

Problem: SCRIPT/PC detected a control word (represented by .xx) whose parameters (options) are not correct. SCRIPT/PC ignores the control word and continues formatting your file.

Solution: Exit SCRIPT/PC and edit the problem file. Check the failing control word for a misspelled option or you may have used an option that is not defined for this control word. Make the correction, save the corrected file and start SCRIPT/PC again.

Label pointed to by Goto control word cannot be found.

Problem: The target of a goto control word (.GO label) cannot be found.

Solution: Exit SCRIPT/PC and edit the macro definition or library that contains the problem. Correct the problem macro by:

1. Checking and correcting any label spelling errors in both the goto control word (.GO label) and the set label control word (...label) or,

2. Removing the goto control word (.GO label). Save the library file and start SCRIPT/PC again.

List structure error. Do you want to continue (Enter Y or N)?

Problem: SCRIPT/PC has determined that:

1. A List Item (:LI) has been found without a list being started or

2. An ending list tag (:ESL, :EUL, :EOL or :EDL), was found without a corresponding start list tag (:SL, :UL, :OL or :DL).

Solution: If you answer:

Y Formatting will continue. The correction of the problem can be left for a later time.

N Formatting of your input file is ended. Exit SCRIPT/PC and edit the problem file. Make the correction, save the corrected file and start SCRIPT/PC again.

Macro definition terminated by End of File.

Problem: A macro definition was not completed within the current input file.

Solution: Edit the input or library file that caused the problem. Check for a missing .ME (macro exit) control word. Correct the problem, save the input or library file, and then start SCRIPT/PC again.

Macro Exit control word found outside of a macro.

Problem: A .ME (macro exit) control word was detected without defining a macro (.DM macroname) first.

Solution: Exit SCRIPT/PC and edit the file with the problem. Either:

1. Define a macro (.DM macroname) and then define the macro steps or,

2. Remove the .ME (macro exit) control word from the file. Save the corrected file and then start SCRIPT/PC again.

Macro libraries have exceeded macro memory space.

Problem: SCRIPT/PC has run out of its allotted memory space for storing macros.

Solution: There are too many macros defined to SCRIPT/PC. You will have to remove some of them before SCRIPT/PC will process successfully. Exit SCRIPT/PC and edit your input or library file(s) to remove unwanted macros. When you have corrected the error, save the corrected file and start SCRIPT/PC again.

> **Note:** Another possible source of your problem is specifying a library more than once. A macro library (for example, GMLLIB.SCT) can be specified in the options profile or your input file (for example, .IM GMLLIB.SCT), not both. If you want to remove the macro library name from your input file, exit SCRIPT/PC, edit your input file and delete the macro library line. Save your input file, and start SCRIPT/PC again.

Macro limit on nesting has been exceeded.

Problem: This problem occurs when your macro calls another macro and it is more than the tenth time this has occurred within the same macro (10 levels of nesting are allowed).

Solution: Exit SCRIPT/PC and edit the library or input file containing the failing macro. Change your macro logic so that no more than 10 levels of macro calls are used. Save your library file and then start SCRIPT/PC again.

Macro name has been cut off at 10 characters.

Problem: A macro name was found that was longer than 10 characters. The name has been cut off after the tenth character.

Solution: Exit SCRIPT/PC and edit the file that has the problem.

1. Change the tag/control word (macro) name to a maximum of 10 characters.

2. Save the Input file.

3. Edit the library that contains the macro and change the macro name to a maximum of ten characters (.DM macroname).

4. Save the library file and then start SCRIPT/PC again.

Macro name to be erased cannot be found.

Problem: The macro you tried to delete (.DM macroname OFF) does not exist.

Solution: Exit SCRIPT/PC and edit the input file containing the problem. The problem may be:

1. Misspelling of the macro name or,

2. An extra line that should not be there.

Make your correction, save the input file and then start SCRIPT/PC again.

Macro statement tried to erase macro name.

Problem: An attempt was made to remove a macro from within another macro (.DM macroname OFF). Removal of macros can only be done on a normal input line in an input file.

Solution: Exit SCRIPT/PC and edit the library or input file containing the problem. Change the problem macro by removing the define macro control word (.DM macroname OFF). Save your library file and start SCRIPT/PC again.

Options Profile cannot be found. Using internal defaults.

Problem: The options profile you entered cannot be found. SCRIPT/PC will build a temporary options profile.

Solution:

1. Either continue with the default option or press Esc to end processing.

2. Check the filename of the profile you want. You may have misspelled it. Reenter if there has been a spelling error.

3. If you want to create a new options profile:

   a. Change the options to desired settings.

   b. Type the file-id you want on the last line of the options menu and

   c. Save the new options, use the F2 key.

Options Profile cannot be saved due to a disk error.

Problem: The profile you tried to create has not been filed due to a disk file error.

Solution: Try the operation again. If it continues to fail, the disk has a permanent error. Use your backup diskette and start SCRIPT/PC again.

Options Profile filename was not specified.

Problem: There is no profile filename specified in the options menu.

Solution:

1. Go to choice 1 on the options menu and type a valid profile filename,

2. Save the new profile and,

3. Continue with formatting or Practice Sessions.

Options Profile PROFILE.SCP cannot be found.

Problem: The default Options Profile you selected cannot be found or it contains an error.

Solution: Type **SCRIPT** and then select choice 1. When the Format a Document options menu appears, complete your choices and then continue with formatting or practice sessions.

Output filename must be specified if output is to disk.

Problem: SCRIPT/PC cannot find a file-id where it can put your output.

Solution: Either type in the output file-id you want, or type & for an output filename that will be the same as your input filename. You can add the drive specifier prior to the filename or & if the disk drive you want is not the DOS default disk.

Page format is in error.

Problem: The size of the top and bottom margins exceed the total size of the page. The command which caused the error (.TM, .BM, or .PL) is ignored.

Solution: Exit SCRIPT/PC and edit the input file that has a problem. You will probably find that there is a control word whose stated value exceeds the maximum or minimum value allowed. Check your page specifications carefully noting any that seem to be extremely large or small. If you need to, use your printer paper and a ruler to check out the specifications listed in your file. Once you have corrected the problem, save the input file and start SCRIPT/PC again.

Pass 1 formatting is on page n

Problem: None - This is an information message.

Solution: None needed.

Pass 2 formatting is on page n

Problem: None - This is an information message.

Solution: None needed.

Printer is not Ready or Online.

Problem: Your printer is not responding to SCRIPT/PC.

Solution: Make sure that the:

1. Printer is powered on.

2. Paper is loaded correctly.

3. Printer ready light is on. If the printer is on but the ready light is off, press the Online key on the printer.

   When all problems have been resolved, the printer will start printing. If there continues to be a problem, check your IBM PC Guide to Operations manual for further diagnostic action.

Printer is OFF or is out of paper.

Problem: SCRIPT/PC cannot print its output.

Solution: Check that the printer is turned on and that paper has been properly installed.

Revision Code control word cannot use * character.

Problem: The option * has been detected in the Revision Code control word. SCRIPT/PC does not support the asterisk as an option, only as a revision character.

Solution: Exit SCRIPT/PC and edit the problem file. Remove the revision code * option. Save the corrected file and start SCRIPT/PC again.

Revision Code markers are not set.

Problem: An attempt to turn on a revision code marker (.RC *n* ON) was rejected by SCRIPT/PC because the revision code was not defined (.RC *n* c).

Solution: Exit SCRIPT/PC and edit the problem file. Make sure that before you try to turn a revision code marker on (.RC *n* ON), you have defined a revision code (.RC *n* c). Make the correction, save the input file and start SCRIPT/PC again.

Revision Code markers must be turned off in reverse order.

Problem: SCRIPT/PC must have the revision markers turned off (.RC *n* OFF) in reverse sequence of how they were turned on (.RC *n* OFF). It has detected that this is not true in your file.

Solution: Exit SCRIPT/PC and edit the problem file. Check to see that the revision code markers (.RC *n* OFF) have been turned off in reverse sequence (9, 8, 7, 6, 5, 4, 3, 2, 1) in respect to how they were turned on (1, 2, 3, 4, 5, 6, 7, 8, 9). Make the correction, save the corrected file and start SCRIPT/PC again.

Right Adjust control word text is longer than 1 line.

Problem: The line in your input file is too long to be right adjusted. When right adjust is in effect, formatting is stopped.

Solution: Exit SCRIPT/PC and edit the problem file. Check that the length of the line(s) you are trying to right adjust (.RI) does not exceed the line length (.LL nn). Make the correction, save the corrected file and start SCRIPT/PC again.

Running Heading control word has an invalid option.

Problem: The Running Heading (.RH) control word has an option (operand) that SCRIPT/PC does not recognize. Legitimate operands are ON, OFF, SUS, RES, and CANCEL.

Solution: Exit SCRIPT/PC and edit the problem file. Check for a running heading (.RH) control word operand that is misspelled or undefined. Make the correction, save the corrected file and start SCRIPT/PC again.

Running Title or Split Text control word has an invalid option.

Problem: SCRIPT/PC has detected an error in defining the running title (.RT) or split text (.SX). The control word that caused the error has been ignored.

Solution: Exit SCRIPT/PC and edit the problem file. Check for spelling errors and the correct options for the running title and split text control words. Make the correction, save the corrected file and start SCRIPT/PC again.

Running Title or Split Text control word text is too long.

Problem: SCRIPT/PC has found a subfield (there are three) that is too long in the running title or split text control words. The addition of all three subfields must not be wider than the line length selected for the page. The subfield text that's too long has been cut off.

Solution: Exit SCRIPT/PC and edit the problem file. Change the running title or split text control words to delete some of the information in the subfields. Make the correction, save the corrected file and start SCRIPT/PC again.

Saving profile file ...

Problem: None - This is an information message.

Solution: None needed

SCRIPT/PC internal error *nn-nnn*.

Problem: SCRIPT/PC has detected an unrecoverable error.

Solution: Press **Esc** to return to Main Menu.

Set Symbol control word has an invalid option.

Problem: SCRIPT/PC has detected an error in the format of the set symbol control word (.SE symname = xxxxxxxx).

Solution: Exit SCRIPT/PC and edit the problem file. Check for spelling errors and a valid format. Make the correction, save the corrected file and start SCRIPT/PC again.

Set Symbol control word tried to define its own name.

Problem: SCRIPT/PC cannot resolve a symbol that sets the symbol value to the symbol name (for example, .SE push = &push).

Solution: Exit SCRIPT/PC and edit the problem file. Check the set symbol control word (.SE) for the same name on both sides of the equal (=) sign. When you have found your problem, determine a new set symbol name or value. Make the correction, save the corrected file and start SCRIPT/PC again.

Set Symbol control word tried to reset a system variable.

Problem: SCRIPT/PC found a set symbol control word (.SE symname OFF) whose name is a SCRIPT/PC supplied variable name. All SCRIPT/PC variable names cannot be removed. SCRIPT/PC continues to process your file.

Solution: Exit SCRIPT/PC and edit the problem file. Check the set symbol control words (.SE symname OFF) for a name that is a system variable (for example, SYSDATE, SYSTIME, SYSDAYOFM). Make the correction, save the corrected file and start SCRIPT/PC again.

Set Symbol control words have exceeded symbol directory space.

Problem: SCRIPT/PC allows 160 set symbols. You have exceeded this amount in your input file. SCRIPT/PC ignores this symbol and any more new ones that are detected.

Solution: Exit SCRIPT/PC and edit the problem file. Decide which set symbols you can eliminate, use .SE symname OFF. Make the correction, save the corrected file and start SCRIPT/PC again.

Subfield in a font definition is not correct.

Problem: The definition identified in the font you selected with the begin font control word does not exist.

Solution: Edit the input file. Check the define font control word for the correct spelling or correct definition. Correct the problem, save the input file and start SCRIPT/PC again.

Subroutines file SCRIPT.SBR cannot be found.

Problem: The file-id SCRIPT.SBR (subroutines) cannot be found on any disk. SCRIPT/PC is ended. This file must be on the same disk or directory that contains the program SCRIPT.EXE.

Solution: Check to ensure that the file specified is on the diskette that contains the program SCRIPT.EXE or in the current directory.disk.

Symbol and macro working space is full.

Problem: SCRIPT/PC cannot process any additional symbols or macros.

Solution: Remove unnecessary symbols or macros to create the space necessary to fully process your input file(s).

Symbol cannot be found.

Problem: SCRIPT/PC found a reference to a symbol in your input file, but could not find where the symbol is defined. SCRIPT/PC continues to format your file.

Solution: Exit SCRIPT/PC and edit the problem file. Define the symbol you want (.SE symname = value) and be sure to place it in your input file prior to its use. Make the correction, save the corrected file and start SCRIPT/PC again.

Tab Setting control word has more than 10 settings.

Problem: SCRIPT/PC detected more than 10 tab stops set with the tab setting control word (.TB *nn nn nn nn nn nn nn nn nn nn*).

Solution: Exit SCRIPT/PC and edit the problem file. Check for more than 10 entries after a tab setting control word (.TB). Make the correction, save the corrected file and start SCRIPT/PC again.

TITLE macro valid only on title page. Do you want to continue (Enter Y or N)?

Problem: The tag :TITLE was found on a line that was not between :TITLEP (Title Page) and :ETITLEP (End Title Page).

Solution: If you answer:

Y Formatting will continue. The correction of the problem can be left for a later time.

N Formatting of your input file is ended. Exit SCRIPT/PC and edit the problem file. Make the correction, save the corrected file and start SCRIPT/PC again.

Translate Character control word has an invalid option.

Problem: SCRIPT/PC has detected a translate control word error in your input file. No change is made to the input or output translate table and SCRIPT/PC continues to format your file.

Solution: Exit SCRIPT/PC and edit the problem file. Check the translate character control word (.TR) for correct format and value. Make the correction, save the corrected file and start SCRIPT/PC again.

Appendix B. Reference Material

Control Words

Alphabetical Order

This section lists the control words in alphabetical order; the BREAK column indicates whether this control word causes a change in formatting (break).

WORD	BREAK	DESCRIPTION
...	No	Set Label for Goto
.*	No	Comment Line
.AN	No	Conditional And
.AP	No	Append next input file
.BF	No	Begin new font
.BM	No	Specify bottom margin
.BR	Yes	Break in formatting
.BX	Yes	Box
.CD	No	Column definition
.CE	Yes	Center data
.CL	No	Specify column length
.CM	No	Comment
.CP	Yes	Conditional page eject
.CS	Yes	Conditional Section
.CT	No	Continue Text
.DC	No	Specify special characters
.DD	No	Name data file-id
.DF	No	Define new font
.DM	No	Define macro
.DS	Yes	Double space
.EF	Yes	End of file
.EL	No	Conditional Else
.FN	No	Footnote
.FO	Yes	Specify formatting options
.GO	No	Goto a specific label
.IF	No	Conditional If
.IL	Yes	Indent one line
.IM	No	Imbed next input file
.IN	Yes	Indent all lines
.IR	Yes	Indent from right margin
.IX	Yes	Index
.LL	Yes	Specify line length

WORD	BREAK	DESCRIPTION
.MC	Yes	Multicolumn mode
.ME	No	Macro definition end
.OF	Yes	Offset one line
.OR	No	Conditional Or
.PA	Yes	Advance to next page
.PF	No	Previous font
.PI	No	Put to index
.PL	No	Specify page length
.PM	No	Specify page margins
.PN	No	Specify page number
.PT	No	Put to table-of-contents file
.QQ	No	Quick quit
.QU	No	Quit
.RC	Yes	Revision code
.RH	Yes	Running header definition start
.RI	Yes	Right adjust lines
.RT	No	Specify running title
.RV	No	Read variable
.SC	Yes	Single column mode
.SE	No	Set symbols
.SK	Yes	Skip lines
.SP	Yes	Space lines
.SS	Yes	Single space mode
.SU	No	Substitute Symbol
.SX	Yes	Split text
.TB	No	Tab stop settings
.TC	Yes	Place table-of-contents
.TE	No	Terminal entry mode
.TH	No	Conditional Then
.TI	No	Input character translate
.TM	No	Specify top margin
.TR	No	Output character translate
.TY	No	Type on terminal

WORD	BREAK	DESCRIPTION
.UC	No	Underscore and capitalize
.UD	No	Underscore definition
.UN	Yes	Undent (indent left)
.UP	No	Upper case (capitalize)
.US	No	Underscore
.ZZ	No	Diagnostic mode

Grouped by Type

This listing lists the SCRIPT control word by the type of processing it will perform for you:

```
Break Formatting:

        .BR         Break

Character Definition:

        .DC         Define characters

Comments:

        .*
        .CM

Communicating with the User:

        .TE         Terminal enter
        .TY         Type on terminal

Conditional:

        .AN         And
        .EL         Else
        .IF         If
        .TH         Then
        .OR         Or

Data Files:

        .DD         Define data file
        .EF         End file
```

File-id Control:

.AP	Append file
.IM	Imbed file

Font Control:

.BF	Begin font
.DF	Define font
.PF	Previous font

Footnote:

.FN	Footnote

Formatting Page:

.BM	Bottom margin
.LL	Line length
.PL	Page length
.PM	Page margin
.TM	Top margin

Formatting Text:

.CE	Center
.CT	Continue text
.FO	Format
.RI	Right adjust

Highlighting:

.BX	Box
.US	Underscore
.UD	Underscore definition
.UP	Capitalize
.UC	Underscore and capitalize

Indentation:

.IL	Indent line
.IN	Indent
.IR	Indent right
.OF	Offset line
.UN	Undent

Index:

.IX	Start index
.PI	Put to index

Macro Definition and Use:

...	Set label
.DM	Define macro
.GO	Goto label
.ME	Macro end

Section Controls:

.CS	Conditional section
.RC	Revision code

Spacing and Paging:

.CP	Conditional page eject
.DS	Double space mode
.PA	Page eject
.PN	Page number
.SK	Skip
.SP	Space
.SS	Single space mode

Stop Processing:

.QQ	Quick quit
.QU	Quit

Table of Contents:

.PT	Put to table of contents
.SX	Split text
.TC	Place table of contents

Tabs:

.TB	Tab stop settings

Titles:

.RH	Running heading
.RT	Running title

Translation:

.TI	Translate input
.TR	Translate output

Variable Symbols:

.RV	Read variable
.SE	Set symbol
.SU	Substitute symbols

GML Tags

Alphabetical Order

```
:ABSTRACT   Insert ABSTRACT here
:ADDRESS    Start ADDRESS
:ALINE      Insert Address LINE here
:AUTHOR     Insert AUTHOR name(s) here

:BACKM      Start BACK Matter
:BODY       Start BODY of document

:CIT        Start citation

:DATE       Insert DATE here
:DD         Definition Description
:DL         Start Definition List
:DOCNUM     Insert DOCument NUMber here
:DT         Definition Term

:EADDRESS   End ADDRESS
:ECIT       End CITation
:EDL        End Definition List
:EFIG       End FIGure
:EFN        End FootNote
:EGDOC      End General DOCument
:EHP1       End underscore
:EHP2       End bold
:EHP3       End bold and underscore
:ELQ        End Long Quotation
:EOL        End Ordered List
:EQ         End Quotation
:ESL        End Simple List
:ETITLEP    End TITLE Page
:EUL        End Unordered List
:EXMP       End eXaMPle
```

```
:FN         FootNote
:FNREF      FootNote REFerence
:FIG        Start FIGure
:FIGCAP     FIGure CAPtion
:FIGDESC    FIGure DEScription
:FIGLIST    List of illustrations
:FIGREF     FIGure REFerence
:FRONTM     Start FRONT Matter

:GDOC       Start General DOCument

:HDREF      Heading Reference
:HP1        Start underscore
:HP2        Start bold
:HP3        Start bold and underscore
:H0         Heading 0
:H1         Heading 1
:H2         Heading 2
:H3         Heading 3
:H4         Heading 4
:H5         Heading 5
:H6         Heading 6

:INDEX      Insert INDEX here
:I1         Create Index entry
:IREF       REFer to Index entry

:LI         Line Item for list
:LP         Line Part
:LQ         Start Long Quotation

:MNOTE      Macro Note

:NOTE       Start note here

:OL         Start Ordered List

:P          Start Paragraph
:PC         Paragraph Continuation
:PREFACE    Insert PREFACE here
```

:Q	Start short Quotation
:SL	Start Simple List
:TOC	Insert Table Of Contents
:TITLEP	Start TITLE Page
:UL	Start Unordered List
:XMP	Start eXaMPle - copy text

Grouped by Function

Document Control:

:GDOC	Start Document
:EGDOC	End Document

Paragraph Control:

:P.	Start Paragraph
:PC.	Paragraph Continuation

Lists:

:LI.	Line Item for list
:LP.	Line Part for list
:OL	Start Ordered List
:EOL	End Ordered List
:UL	Start Unordered List
:EUL	End Unordered List
:SL	Start Simple List
:ESL	End Simple List
:DL	Start Definition List
:DT	Definition Term
:DD	Definition Description
:EDL	End Definition List

Figures:

```
:FIG        Start FIGure
:FIGCAP     FIGure Caption
:FIGDESC    FIGure DESCription
:FIGREF     FIGure REFerence
:FIGLIST    List of Illustrations
:EFIG       End FIGure
```

Headings:

```
:H0         Heading 0
:H1         Heading 1
:H2         Heading 2
:H3         Heading 3
:H4         Heading 4
:H5         Heading 5
:H6         Heading 6
:HDREF      HeaDing REFerence
```

Front Matter:

```
:FRONTM     Start FRONT Matter
:TITLEP     Start TITLE Page
:ETITLEP    End TITLE Page
:DOCNUM     Insert DOCument NUMber here
:DATE       Insert DATE here
:AUTHOR     Insert AUTHOR name(s) here
:ADDRESS    Start ADDRESS
:ALINE      Insert Address LINE here
:EADDRESS   End ADDRESS
:ABSTRACT   Insert ABSTRACT here
:PREFACE    Insert PREFACE here
```

Body of Document:

```
:BODY       Start BODY of Document
:APPENDIX   Start APPENDIX of document
:BACKM      Start BACK Matter
```

Highlighting:

:XMP	Start eXaMPle - copy text
:EXMP	End eXaMPle
:HP1.	Start underscore
:EHP1	End underscore
:HP2.	Start bold
:EHP2	End bold
:HP3.	Start bold and underscore
:EHP3	End bold and underscore
:LQ.	Start Long Quotation
:ELQ	End Long Quotation
:CIT	Start citation
:ECIT	End citation
:NOTE.	Start note here
:Q	Start Quote
:EQ	End Quote

Indexing:

:INDEX	Start Index
:I1	Index Entry
:IREF	Create Index Reference

Footnotes:

:FN	Footnote
:EFN	End Footnote
:FNREF	Footnote Reference

Appendix C. Differences Between SCRIPT/PC and SCRIPT/VS (DCF)

Introduction

SCRIPT/PC is a subset of the Documentation Composition Facility (DCF) Release 2, Program Number 5748-XX9 available to users of the IBM System/370. This chapter lists the differences in the characteristics of the files, marked up with SCRIPT/PC control words and tags, as they are moved from your IBM PC to the IBM System/370 host system for formatting by the DCF product SCRIPT/VS and GML. It also points out the changes necessary when you move a file from the host system to your IBM PC.

Your IBM PC Files

Files created, formatted and printed using SCRIPT/PC have some unique properties. SCRIPT/PC has been developed to take advantage of your IBM Personal Computer characteristics and to provide you with an easy way to create professional looking letters, books and other documentation using an IBM PC printer. Moving the file you have created to a System/370 host system for formatting and printing by SCRIPT/VS is an alternative way to create formatted output by taking advantage of the larger host system and the things it does best.

There's a difference in sizes and computer capabilities. Since the output devices on the larger host system do not match your printer, disks or display station, there will be changes to your setup and run procedures. You will find all the rules and messages for the host system in the DCF books available from your local IBM Branch Office. That means that when you move your file to the host system, you will be leaving all the files that supported your IBM PC input (source) files behind. This includes the menu screens, your options profile, messages and even the tag (macro) libraries. Each of those SCRIPT/PC support files was customized to take full advantage of your IBM PC and they would not be able to accomplish the same task on the larger host system. It is the purpose of this chapter to identify the differences between your input file when it is formatted on an IBM PC with SCRIPT/PC or on an IBM System/370 with SCRIPT/VS (DCF).

Your IBM System/370

When your file is sent from your IBM PC to a host system with DCF installed, it may have to have changes made to it before SCRIPT/VS would give you the same formatted output you had on the IBM PC. You are going to build a new profile of options, use a new set of messages, and possibly using a variety of printers. Since these items are well documented in the DCF literature, you will be mostly interested in how to run the file created for SCRIPT/PC on SCRIPT/VS. This chapter will make you aware of any differences.

Tags and Control Words Not Supported

Tags in the GML Starter Set

The following GML Starter Set tags are not supported by SCRIPT/PC:

Tags Not Supported by SCRIPT/PC

:I2	:IH2	:EHP0
:I3	:IH3	:PSC
:IH1	:HP0	:EPSC

If you have any questions beyond what this chapter covers, be sure to read the detailed descriptions of each GML tag in this reference book.

Control Words

The following control words are not supported by SCRIPT/PC. If you use these control words, SCRIPT/PC will flag them as errors. However, if you imbed the file **NO-OP.SCT** in the input file that contains these control words, SCRIPT/PC will ignore them. This allows you to process a file on SCRIPT/PC that was designed to run on SCRIPT/VS.

Control Words Not Supported By SCRIPT/PC

.AA	.DU	.HW	.LT	.RD	.SV
.BC	.EC	.HY	.LY	.RE	.SY
.CC	.EM	.IT	.MG	.RF	.TS
.CO	.EZ	.JU	.NL	.RN	.TU
.DH	.FL	.KP	.OC	.SA	.UW
.DI	.FS	.LB	.PP	.SL	.WF
.DL	.HM	.LI	.PS	.SO	.WZ

Differences

General Information

The following information is of a general nature.

Start Procedures

SCRIPT/PC cannot be selected from another program, however, it can be called from within a Batch File (filename.BAT).

Blank Lines in your Input File

Line 1 of your input file should contain significant data. If you leave line 1 blank, Running Headings and Page Numbers may not print correctly until the following page. All other blank lines will print as blank text lines.

Tags (Macros)

1. Definition List [:DL]

   - The attribute TERMHI='0' (print as-is) and TERMHI='2' (print bold) are the only TERMHI values supported.

2. General Document [:GDOC]

   - The header created by the SEC='line-of-text' attribute is printed "as-is". It is not capitalized or underscored.

3. Highlight Phrase 1 through Highlight Phrase 3

   - The highlighting tags cannot be used to highlight a partial word in SCRIPT/PC.

Headings

1. Headings (.H0 through .H6) are supported as control words that are macros located in a macro library called HEADDEF.SCT.

2. You may use the library "as-is" or change them, following the normal rules for writing a macro. You cannot change the headings using the Define Heading (.DH) control word in SCRIPT/PC.

3. EASYSCRIPT headings are not supported.

Index

1. The INDEX *must* be the last item in your document. No other control words or tags will be processed after the Index (.IX) control word.

2. Indexing is done on a single level. The GML tags that create an index are supported through the first level. :I2 and :I3 are not supported.

3. The sorting of the index is fixed permanently. You cannot override it.

Line Selection (.CE, .US, .UC, etc.)

The operand that identifies a specific number of lines for some control words is not supported (For example, **.CE 5**).

Look-ahead Control Words

The Keep (.KP), Widow Zone (.WZ), and Float (.FL) control words are not supported by SCRIPT/PC.

Macros

1. Macros are not transportable between the host system and the IBM PC. Macros written for DCF take advantage of the additional equipment and resources available at the System/370 host site.

2. Parameters (options or data) can be used in macro definitions. However, parameters cannot be passed to an imbedded or appended file in your input file.

Multicolumn Mode - Two Columns

1. Normal formatting occurs in this mode. However, the Balance Columns (.BC), Column Begin (.CB), and Conditional Column Begin (.CC) are not supported.

2. Column balancing occurs at the end of a page or when a return to Single-Column mode (.SC) is done.

3. The Column Definition (.CD 2 nn nn) control word must be used before you select multicolumn mode (.MC).

4. The Conditional Page Eject (.CP) control word is not supported in multicolumn (two-column) mode.

Page Formatting

1. All control words that specify page layout take effect immediately. Setting up and altering your page layout must be done with care. It should be done at the beginning of the document or just prior to a new page.

2. Page numbering is in Arabic.

3. Odd and even pages support is not available.

4. Each page eject causes the current page to be ended. However, a new page is not started until additional text or a control word that causes the page to be started is read. This is the same as the NOSTART option to the Page Eject (.PA) control word on SCRIPT/VS. NOSTART is *not* a valid option on SCRIPT/PC. Page layout changes should only be done prior to the Page Eject control word.

5. Use only the Top Margin (.TM) and Bottom Margin (.BM) control words to reserve space at the top and bottom of each page. The SCRIPT/VS heading margin and footing space control words are not supported by SCRIPT/PC.

Revision Codes

1. You must reserve space to the left of your starting column number when you are using a revision code.

2. Use the Page Margin (.PM) control word to allow enough space for a revision code to be inserted.

Tabs

1. Tabs are supported in unformatted mode (**.FO OFF**). If you use tabs while the formatter is on (**.FO ON** or **.FO LEFT**), formatting errors will occur and the tabs will not be found.

2. The only tab fill character supported is the blank character.

Control Words Detailed Differences

This section lists the differences in control words.

Append (.AP)

The only allowed option is the drive specifier, the filename and extension (**d:filename.ext**).

Box (.BX)

1. The /, NEW, and SET options are not supported.

2. If you are *not* using the IBM Graphics Printer, use the CHAR option to change the character set that is to be used to create the boxes.

Column Definition (.CD)

Only 2 column specifications may be entered.

Conditional (.AN, .TH, .EL)

1. The conditional control words: And, Then, and Else are limited to the first two characters of their functional name (**.AN, .TH and .EL**).

2. The conditional tests SYSPAGE and SYSOUT are not supported by SCRIPT/PC.

3. Unary + or - are not supported in conditional tests.

Define Character (.DC)

The only string of characters that can be defined are the CW (control word separator), PS (page number symbol), RB (required blank), and GML (GML delimiter).

Define Data File-id (.DD)

The only option allowed is the drive specifier, filename and extension (**d:filename.ext**).

Define Font (.DF)

The SCRIPT/PC fonts are only structured for the IBM PC and differ from those defined in SCRIPT/VS. Refer to the Define Font control word description for a list of all valid fonts.

Errors are not detected until the font is used by .BF.

Delay Imbed (.DI)

The delay imbed function is not supported by SCRIPT/PC.

Dictionary (.DL and .DU)

The Dictionary functions are not supported by SCRIPT/PC. You can use the IBM PC Word Proof product to do this function.

End Of File (.EF)

1. The End Of File control word can be used to leave a file at any time, unless the Define Data File-id (**.DD**) control word has been issued first. However, if this file is called again, processing will start at the top-of-file, rather than the previous stopping point.

2. For End Of File control words to step through an input file in sequential steps, the file that is to be stepped through must first be defined by a Define Data File-id control word.

Format (.FO)

The allowable options are ON, OFF, and LEFT.

GML Services (.GS)

This control word is only available for use within a GML macro and is different in syntax and use from SCRIPT/VS.

This control word is not documented for SCRIPT/PC users.

Hyphenate (.HY and .HW)

The Hyphenation functions are not supported by SCRIPT/PC.

Imbed (.IM)

The only option allowed is the drive specifier, the filename and extension (**d:filename.ext**).

Indent (.IN)

The options FOR and AFTER are not supported.

Indent Right (.IR)

The options FOR and AFTER are not supported.

Macro Exit (.ME)

The option *line* is not supported.

Page Numbering Mode (.PN)

In addition to ON and OFF, only the option *n* is supported.

Put Index (.PI)

The options ORDER and KEY are not supported.

Revision Code (.RC)

The options ON | OFF and * are not supported.

Running Title (.RT)

The option ALL is the only mode supported. You cannot override this default. ALL must not be specified with the control word. The options TOP, BOTTOM and line number are required.

Read Variable (.RV)

The equal sign is required.

Set Symbol (.SE)

1. The equal sign is required.
2. The option LIB is not supported.
3. Unresolved symbols are not flagged as errors except within a conditional control word test.
4. If a symbol cannot be resolved, it is left as-is.

Skip (.SK)

1. The options C, A, and P are not supported.

2. All Skip (.SK) control words are recognized and processed in SCRIPT/PC, even consecutive skips.

Space (.SP)

The options C, A, and P are not supported.

Split Text (.SX)

One output line is formatted. If the text is longer than one line an error message is given.

Tab Setting (.TB)

This control word defaults to the option SET. The only options allowed on the Tab Setting control word are the tab settings.

Translate Character (.TR)

Five translation pairs per control word are supported.

Translate Input (.TI)

Five translation pairs per control word are supported.

Type on Terminal (.TY)

If you end your text with an equal sign, the cursor does not do a "carriage return". It will stay on the same line that contains your message.

Underscore Definition (.UD)

This control word will be in effect when the formatter is on (**.FO ON** or **.FO LEFT**). If the formatter is off (**.FO OFF**), all blanks will be underscored.

Appendix D. Additional Help Files

The SCRIPT/PC product diskette has four additional files to assist you in the editing and proofing of input files. These files can be used if you are using one of the following IBM PC editor or proof products. The IBM PC products and the SCRIPT/PC supplied file-ids are listed below:

- IBM PC Personal Editor – SCRIPT.PRO
- IBM PC Professional Editor – SCT.PRF and PROFGML.TAG
- IBM PC Word Proof – WORDPRF.ADL

For IBM PC Personal Editor Users

The SCRIPT/PC program diskette contains the file **SCRIPT.PRO** for users of Personal Editor. You can rename the file-id for permanent use to **PE.PRO,** or leave it as-is for use when editing SCRIPT/PC input files. This profile can be started in the file currently being edited by entering MACRO SCRIPT.PRO on the command line, which sets the keys to the following:

Cursor Movement and Special Keys

Cursor up (↑), down (↓), left (←) and right (→) along with the backspace (←) keys operate in a normal manner. Other special keys are defined as follows:

```
Home - Start of line    Ctrl-Home  - Start of document
End  - End of line      Ctrl-End   - End of document
Tab  - Tab word         Shift-Tab  - Backtab word
PgUp - Up 20 lines      PgDn       - Down 20 lines
```

Function Keys

```
F1   - Help              F6   - Erase to end of line
F2   - Save file-id      F7   - Insert a line
F3   - File file-id      F8   - Delete a line
F4   - Copy a line       F9   - Split a line
F5   - Erase a line      F10  - Join two lines
```

Alternate and Function Keys

The Alternate (Alt) and specified Function (F1 thru F10) keys must be pressed at the same time:

```
Alt-F1  - Undo changes
Alt-F2  - Switch file-id
Alt-F3  - Place FILE ... NOTABS
          on command line
Alt-F4  - Drive A directory
Alt-F5  - Drive B directory
```

Control and Function Keys

The Control (Ctrl) and specified Function (F1 thru F10) keys must be pressed at the same time:

```
Ctrl-F1  - Mark a line
Ctrl-F2  - Mark a character string
Ctrl-F3  - Mark a block
Ctrl-F4  - Copy marked area
Ctrl-F5  - Move marked area
Ctrl-F6  - Fill marked area
Ctrl-F7  - Overlay marked area
Ctrl-F8  -
Ctrl-F9  - Delete marked area
Ctrl-F10 - Remove marks
```

Alternate and Other Keys for GML Tags

The following key combinations cause the listed tags to be created in your input. Additional blank lines are added where appropriate and the cursor is positioned on the normal input line:

```
Alt-0  - :H0.      Alt-U  - :UL
Alt-1  - :H1.               :LI.
Alt-2  - :H2.               :EUL
Alt-3  - :H3.      Alt-L  - :LI.
Alt-4  - :H4.      Alt-D  - :DL
Alt-5  - :H5.               :DT.
Alt-6  - :H6.               :DD.
Alt-0  - :OL                :EDL.
         :LI.      Alt-T  - :DT.
         :EOL.               :DD.
Alt-S  - :SL       Alt-F  - :FIG
         :LI.               :EFIG.
         :ESL.     Alt-C  - :FIGCAP.
Alt-X  - :XMP.
         :EXMP.

Enter (↵)  - :P.
```

For IBM PC Professional Editor Users

The SCRIPT/PC program diskette contains a profile named **SCT.PRF** to be used with Professional Editor. This profile has thirty-five data macros defined. They consist of the most commonly used GML tags and a file that contains all of the GML tags available in SCRIPT/PC. You may view them by pressing the F6 Macros function key when the Professional Editor program is loaded and the SCT.PRF profile is used.

Each element tag may be inserted into the file currently being edited by pressing the Alt and associated key. For example, to insert the :ADDRESS tag, press the Alt and A keys at the same time.

The SCRIPT/PC program diskette also contains a Professional Editor file named **PROFGML.TAG,** which includes all of the GML tags available in SCRIPT/PC. If you are creating a new document you may find it easier to start with all of the tags and then delete those you don't need. This file may be inserted into the file currently being edited when you press the Alt and 9 keys at the same time.

For IBM PC Word Proof Users

The SCRIPT/PC program diskette contains the file **WORDPRF.ADL** for users of Word Proof. All SCRIPT/PC commands, descriptive (control words) and declarative (tags), are listed in this file.

To allow Word Proof to proof all control words:

1. Select option 5 on the Word Proof Main Menu to obtain the **Profile Values** display screen.

2. Insert a period (.) in the **Control Word prefix** field.

3. Press the ↵ key to return to the Main Menu.

To allow Word Proof to proof all GML tags:

1. Select option 5 on the Word Proof Main Menu to obtain the **Profile Values** display screen.

2. Insert a colon (:) in the **Control Word prefix** field.

3. Press the ↵ key to return to the Main Menu.

Only copy the files that are necessary for the product you are using. This allows SCRIPT/PC to use your computer more efficiently.

3. Press the ↵ key to return to the Main Menu.

To allow Word Proof to proof all GML tags:

1. Select option 5 on the Word Proof Main Menu to obtain the **Profile Values** display screen.

2. Insert a colon (:) in the **Control Word prefix** field.

3. Press the ↵ key to return to the Main Menu.

Only copy the files that are necessary for the editor you are using. This allows SCRIPT/PC to use memory more efficiently.

Glossary

The glossary is a list of terms used by SCRIPT/PC. It also contains definitions of commonly used terms that relate to typographical items (those items that are the everyday language of those people who produce documents). This glossary is intended to make you familiar with the language of text processing terms and should assist you in your understanding of formatting documents. The terms in this glossary are defined as they are used in this book.

If you cannot find the term you are looking for here in the glossary, refer to the table of contents or the index for further possibilities.

ampersand - &: When an ampersand begins a string of characters, SCRIPT/PC assumes the character string is a symbol name. If the symbol name has been previously defined, the symbol name is replaced with its definition or value before being processed.

In running headings and running titles, the simple ampersand is usually the page number symbol.

The ampersand (&) is always interpreted as the page number symbol when it is found by itself.

ASCII: American National Standard Code for Information Interchange. The standard code, using a coded character set consisting of 7-bit coded characters (8-bits including parity check), used for information interchange among data processing systems, and associated equipment. The ASCII set consists of control characters and graphic characters.

attribute: A characteristic of a document other than its type or content. For example, the security level of a document or the depth of a figure.

back matter: Those sections, such as glossary and index, that are placed after the body (main sections) of a book.

base line: The horizontal line that is the reference for the bottom of characters, as in a, b, c, d, e, f, h, i, k, l, m, n, o, r, s, t, u, v, w, x, and z. Not to be confused with descenders, as in g, j, p, q, and y.

binding: The edge of the page that is to be bound, stapled, or drilled. This is normally on the left side of odd numbered pages, and on the right side of even numbered pages. SCRIPT/PC only supports page numbering on the right side of the page.

body: (1) Of a printed page, that portion between the top margin and bottom margin that contains the text. (2) Of a book, the portion of the book that is located between the front matter and back matter.

boldface: A heavy-faced type. A method of **highlighting** a word or phrase.

bottom margin: On a page, the space between the body of the page and the bottom edge of the page. The bottom margin normally contains the information defined by the running title (bottom title).

break: An interruption in the formatting of input lines; the next input line begins a new output line.

caps: Capital letters (Uppercase).

caption: Text that is next to and describes an illustration (picture or figure).

character: A symbol used with a written language. For example, a letter of the alphabet, a number (0-9), a punctuation mark, or any other symbol that has meaning to those who read it.

character set: A complete set of characters that can describe a language or a print font. For example, there is the English alphameric character set, a graphic character set, a cyrillic character set and many others.

character spacing: The space between characters in a word.

column width: The width of each text column on a page. It is controlled by the Column Width [.CL] control word. It is only used for multicolumn mode (2-column) in SCRIPT/PC.

comment: A control word line that is ignored by SCRIPT/PC. The comment control word can be either **.CM** or **.*** . The comment is useful for placing directions or editorial remarks within your input file, without processing or printing them.

comparand: One of the the items that is compared with another. The results of the comparison is either true or false. The functional unit where a comparison of comparands is done is called a comparator unit.

composition: Formatting a document or the result of formatting a document.

concatenation: The forming of an output line that contains as many complete words as the line length allows. This is done by placing the first words of an input line directly behind the last words of the previous input line. This is similar to putting individual sentences together as a paragraph.

control word: A descriptive instruction (command) within an input file that either identifies its parts or tells SCRIPT/PC how to format the document. SCRIPT/PC control words start with a period (.) followed by two characters.

control word line: An input line that contains at least one control word.

current left margin: The leftmost column that is available for text while formatting a document. Each column's left margin is defined by the Column Definition [.CD] control word (multicolumn mode). However, the current left margin can be altered by the indent, undent, indent line or offset control words.

current line: The line in the input document at which a computer program (editor or formatter) is positioned for processing.

debug: To detect, trace, and eliminate errors in SCRIPT documents or computer programs.

default value: A value that is assumed by SCRIPT/PC when you do not fill in control word item values.

document: (1) A publication, book, booklet, pamphlet, or other form of written material. (2) A machine-readable collection of lines of text or figures, usually called a source (input) document.

document conversion processor: A computer program that processes a machine-readable document that contains formatting controls. This machine-readable document is then formatted into another machine-readable document that has a different formatter language.

duplex: A mode for formatting that creates an output format that fits both sides of a sheet of paper.

EBCDIC: Extended binary-coded decimal interchange code. A coded character set.

edit: To create, review or change the contents of a file or document. For example, to insert, delete, change, rearrange, or copy lines.

editor: A computer program that allows you to edit a file.

eject: In formatting, a skip to the next page.

fill character: The character that is used to fill up a space; for example, blanks are used to fill up the space left by tabbing.

flush: Without indentation.

fold: (1) To translate the lowercase characters into uppercase characters. (2) To place the part of an input line that is too long for the current output line on the next output line.

font: An assortment of type, usually a complete character set, all of the same size and style. For example, bold, italics, pressed roman, and double width are only a few of the many fonts available in the publications world.

font set: The set of fonts to be used in formatting a source document.

footnote: A reference note, explanation, or comment placed below the column or page text, but within the body of the page.

format: (1) The shape, size, and general makeup of a printed document. (2) To prepare a document for printing in a specific manner.

formatting mode: When input lines are placed together (concatenated) and the resultant output lines are justified.

formatter: (1) That part of SCRIPT/PC that formats input lines for a specific output device. For example, the printer or disk. (2) A computer program that prepares a source (input) document to be printed.

front matter: Those sections of a book, (such as preface, abstract, table of contents, and list of illustrations), that are placed before the body (main

chapters and sections).

Generalized Markup Language (GML): A language that may be used to identify the parts of a source document without causing you to write all the SCRIPT/PC control words necessary to do the work. A shorthand method of describing desired output.

GML: See Generalized Markup Language

gutter: The space between columns when you are in multicolumn mode.

hanging indentation: The indentation of all lines of a block of text after the first line of that block of text. This is done by using the Offset or Undent control words.

head-level: The typeface and character size associated with the words that define the chapter or chapter topic. For example, the primary, secondary, or 1, 2, 3, 4, 5, and 6 level headings.

heading: Words located at the beginning of a chapter, section, topic, or top-of-page.

hexadecimal: A number system that uses the base 16. Each character position can contain the value of 0 - 15, which is written as 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F. In hexadecimal, A=10, B=11, C=12, D=13, E=14, F=15. For example, the hexadecimal number 0E = 14 (in decimal) and the hexadecimal number 1A = 26 (in decimal), each carry = +16.

indent: To place text to the right of the left margin.

indentation: The action of indenting and the condition of being indented. This is controlled by the Indent, Undent, Indent Right, Offset, and Indent Line control words.

initial caps: Capital letters that occur as the first letter of each word in a phrase. To create a phrase in initial caps is to capitalize the first letter of each word in the phrase.

initial value: A value that is assumed by SCRIPT/PC for a formatting function until you change the value with a control word. A default value is assumed **only** when the control word is found, not before.

input device: A machine that enters information into a computer system. For example, a terminal that is used to create a source document.

input line: A line, as entered into a source file, to be later processed by a formatter.

italic: A typestyle whose characters *slant upward* to the right.

justify: To insert extra blank space between words so that the left and right text margins are straight (as opposed to ragged right or ragged left). This creates a very neat page appearance as the text appears to be perfectly blocked on the left and right side.

layout: The arrangement of book matter to be printed.

leader: (1) Periods (dots) or hyphens (dashes) used to lead your eye across the paper to an important item, as in a table of contents. (2) The divider between text and footnotes on a page.

left-hand page: The page on the left when a book is opened. It is normally even page numbered.

line spacing: The space between an output base line and the next output base line (not counting ascenders or descenders).

lowercase: Small letters. For example, abcdefghijklmnopqrstuvwxyz .

machine-readable: Data in a form that a machine can read from a storage device, a data medium, or from another source.

macro: An instruction in a source language that is replaced by a series of control words, symbols, and text. It becomes a shorthand way to write a series of control words that are frequently used.

macro substitution: During formatting, the replacement of a macro by control words, symbols and text.

margin: (1) The space above, below, and on the left and right side of the body of a page. (2) The left or right limit for a column of text.

markup: The information added to a source document that enables a person or system to process the document in a specific defined manner. Markup may describe the document's characteristics, or it may specify the actual processing to be done. In SCRIPT, markup is use of GML tags, control words, and values.

offset: To indent all lines of a block of text, except for the first line.

option: Information entered when SCRIPT/PC is used to control the running of the SCRIPT/PC processor.

output device: A machine used to print, display, or store the output result of processing.

output document: A machine-readable collection of lines of text that has been formatted by the SCRIPT/PC processor. The output document can be printed or filed for future use.

output line: A line of text created by SCRIPT/PC processing your input file.

paginate: To number pages.

parameter: Any one of a set of items whose value settings define the exact desired operation of a control word.

pica: A unit of measurement approximately 1/6 inch used to measure typographical material. One pica = 12 points (12/72 inch).

pitch: A unit of measurement that says how many characters of a particular font can print in one inch of horizontal space. For example, 10-pitch means 10 characters per inch and 12-pitch means 12 characters per inch.

point: The basic unit of measurement for typographical material, 1/72 inch. For example, there are 12 points in a pica (12/72 or 1/6 inch).

proportional spacing: The spacing of characters in a printed line so that each character is given a space in proportion to its width. For example an **I** is given less space on the line than an **M** .

ragged right: The unjustified right edge of text lines.

right-hand page: The page on the right when a book is opened. It is normally odd numbered.

rule: A straight horizontal or vertical line used to separate or border the parts of a figure or box.

running heading: A heading that is repeated below the top margin area on consecutive pages in the page's body (text area).

running title: A line of data that may be repeated in the top and bottom margin area on all pages.

small caps: Capital letters that are the same size as the lowercase letters.

source document: A machine-readable collection of lines of text, figures, or tables that is the input to SCRIPT.

space: A blank area that separates words or lines.

symbol: A name in a source document that SCRIPT/PC will replace with one or more characters.

symbol substitution: The replacement, during formatting, of a symbol with a number, one or more characters, control word, or another symbol.

tab: A preset point in the output line that may be skipped to by doing a tabbing function.

tag: In GML markup, the declarative name for a type of document or part of a document that is entered in the source (input) document to identify it. For example, **:para.** could be used as a tag to identify a specific type of paragraph.

terminal: A device (machine), that usually has a keyboard and some type of display screen. It is usually capable of sending to or receiving data from a distant source or can create data on a temporary or permanent storage devices such as diskettes or fixed disk.

text line: An input line that contains only text, no control words or tags.

title: See running title.

token: One or more characters that can be treated as one item.

top margin: On a page, the space between the body or running heading and the top edge of the page.

typeface: All type of a single style. There can be several types of fonts within the same typeface.

typeset: To arrange the type on a page for printing.

unary operator: An arithmetic operator having only one term. The unary operators that can be used in absolute, relocatable, and arithmetic expressions are:positive (+) and negative (-).

underscore: To place a line under one or more characters. To *underline.*

unformatted mode: When each input line is placed in the file in the same format as it occurs. The SCRIPT/PC formatter will not change the input lines.

uppercase: Placing all type in capital letters. For example, ABCDEFGHIJKLMNOPQRSTUVWXYZ .

widow: One or more words at the end of a paragraph that are printed at the top of the next page.

word spacing: The space between words on a line, also known as wordspace.

Index

Special Characters

A

B

C

D

E

F

G

H

I

J

L

M

N

O

P

Q

R

S

T

U

W

Y

Numerals

The Personal Computer
Software Library

Reader's Comment Form

SCRIPT/PC
Book 3 Reference 1502416

Your comments assist us in improving the usefulness of our publication; they are an important part of the input used for revisions.

IBM may use and distribute any of the information you supply in any way it believes appropriate without incurring any obligation whatever. You may, of course, continue to use the information you supply.

Please do not use this form for technical questions regarding the IBM Personal Computer or programs for the IBM Personal Computer, or for requests for additional publications; this only delays the response. Instead, direct your inquiries or request to your authorized IBM Personal Computer dealer.

Comments:

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 321 BOCA RATON, FLORIDA 33432

POSTAGE WILL BE PAID BY ADDRESSEE

IBM PERSONAL COMPUTER
SALES & SERVICE
P.O. BOX 1328-C
BOCA RATON, FLORIDA 33432

Fold here

Tape Please do not staple Tape

Software included:

Diskette

IBM

System requirements:

IBM 80-column display

128KB of memory

One diskette drive

IBM Printer

International Business
Machines Corporation
P.O. Box 1328-S
Boca Raton, Florida 33432

1502416

Printed in the United States of America