CERN SGML User's Guide

Anders Berglund

CERN
CH 1211 Geneva 23
Switzerland

October 27, 1986

First Edition

CERN SGML User's Guide

Anders Berglund

CERN
CH 1211 Geneva 23
Switzerland

October 27, 1986

First Edition

1. Abstract

SGML – Standard Generalized Markup Language – ISO 8879 – is a language for representing documents in a descriptive markup form. It is intended "for publishing in its broadest definition, ranging from single medium conventional publishing to multi-media data base publishing. SGML can also be used in office document processing when the benefits of human readability and interchange with publishing systems are required."

One of the fundamental design goals of SGML has been to make the language ***independent*** of the text formatter used. One can thus expect, and is already starting to see, support for SGML from different vendors and running on a variety of computers. Some of these implementations use already existing text formatters; e.g. TeX, and SCRIPT. It is obvious that SGML can become a very useful interchange format between different systems and different installations.

It is clear that in the next few years more and more text processing is going to be performed on PCs and workstations and that mainframe (& PC) inputfile – processing – outputfile systems will be less and less used. Such systems will fade away in preference for powerful generic markup systems of a WYSIWYG type. Indeed one is starting to see systems that combine the good aspects of the What You See Is What You Get approach with the good aspects of text with markup[1]

SGML has caused a lot of interest in both the publishing industry and the computer industry. There have been created two user groups to stimulate and share experience with SGML: The SGML User's Group and EPMarkup.

It is of great interest to see that North-Holland are actively studying SGML and it may become possible for CERN authors to submit papers for publication in SGML form.

Large organizations that have adopted SGML include the Office of Official Publications of the EEC and the US Department of Defense.

[1] One such commercial system, scheduled for release first quarter 1987, was shown by Datalogics at Markup '86. It follows the ISO 8879 (SGML) standard.

2. Implementation

This document describes an implementation using SGML syntax with Waterloo SCRIPT as the text formatter. Is uses the Reference Concrete Syntax and, to a large extent, the Document Type Definition in Annex E of ISO 8879. Minor changes have been made for special element requirements, such as the AUTHLIST for the title-page.

The input for a SGML-prepared document in this implementation is a file of the following form:

```
<!DOCTYPE style>
<GDOC>
<FRONTM>
the front-matter component
</FRONTM>
<BODY>
the body component
</BODY>
<APPENDIX>
the appendix component
</APPENDIX>
<BACKM>
the back-matter component
</BACKM>
</GDOC>
```

So called ″tags″ are used to identify the elements of the document. Tags are of two forms – start-tag (e.g. <FRONTM>) and end-tag (e.g. </FRONTM>) delimiting the logical element. In many cases the end-tag is optional.

Special symbols are enterred using the ″short reference″ mapping for very common symbols and entity references for the less commonly used ones. For the entity names Annex E of ISO 8879 has been followed. Special Symbols on page 43 gives a list of these.

3. Structure of this guide

This guide is divided into several parts. The tags and attributes of each major document division – front matter, body, appendices, and back matter – are described separately. One part describes the ″basic document components″, which tags are common to all the divisions. The descriptions of the front matter, appendices and back matter refers to the ″standard″ and ″semi-standard″ layout types – Layout Types on page 49 describes the special tags for the ″non-standard″ document types (e.g. letters, memos). The tags of the ″basic document components″ are, of course, common to all styles.

In the appendices examples are shown of input files for the various document types.

Whilst strong efforts have been made to ensure that the layout styles produce documents in ″CERN taste″ it is clear that some users have very strong views on style. The style can to a certain extent be tailored by the use of the SET tag.

4. Some notes on the choice of Generic Identifiers

In the design of a set of tags and attributes the major task is the identification of the various document components for which tags should be created. Of equal importance, but much more difficult to reach agreement on, is the choice of the generic identifiers for the tags. In some cases the requirement to save key-strokes is very important, whereas in others clarity and ease of remembering the generic identifiers is more important. In this implementation the frequently used tags (mainly in the "basic document components") have short generic identifiers (e.g. P for "paragraph", whereas the less frequently used tags have longer, more explicit names (e.g. AUTHOR).

In the choice of generic identifiers for heading levels two approaches can be taken. One is to consider the heading levels to be parts, chapters, sections and so on. The other, more abstract, is to consider them to be a header of level 0, of level 1, of level 2 etc. This approach has the merit that it can be applied more logically to document types of a larger range – from large books to letters. Whilst large books are naturally divided into parts, chapters, sections ... it is much less natural to consider a letter divided into chapters and sections. Thus the generic identifiers of heading levels has been chosen to be H0, H1, ... H6 in this implementation.

5. Acknowledgements

The underlying SCRIPT macros used in this implementation are based on the "Waterloo SCRIPT GML" written by Roger Watt. They have, however, been modified to conform to local "taste" as regards layout, and addition of support for special document components. This guide is based on the Waterloo GML guide. Finally, but not least, I would like to thank Charles Goldfarb (editor of the SGML standard) for very pleasant discussions.

Contents

Figures

Tables

Part 1

Tags, Attributes, and Text

Tags are used to identify elements of a document, such as headings, paragraphs etc. There is a start tag, then the content of the element, and then the end-tag (which often can be omitted).

1. SGML Tag Syntax

A tag can appear anywhere in an input record; it does not have to start at the beginning of the record. The various ways in which a SGML tag can be specified can be represented as follows:

```
<generic_identifier>
<generic_identifier>tagtext
<generic_identifier attribute list>
<generic_identifier attribute list>tagtext
```

In this document, items enclosed in ″[″ and ″]″ characters represent optional specifications; for example,

```
<generic_identifier[ attributes]>tagtext
```

documents a tag for which attributes are optional but tagtext is required.

1.1 Generic Identifier

In its simplest form, an SGML tag consists of the SGML start tag-indicator character (<) followed by the generic identifier, followed by the tag close delimiter (>);for example,

```
<generic_identifier>
```

The generic identifier may be specified in uppercase, lowercase, or a mixture. Throughout this document, uppercase characters are used to represent things that you must specify exactly as documented, and lowercase characters are used where you have to specify something of your own choice.

1.2 End Tags

Many elements are delimited by a **start** tag and an **end** tag. The latter has the same generic identifier, but is preceeded by the end tag open delimiter ″< /″, for example,

```
</TITLEP>
```

1.3 Tag Text

Some tags either require or accept "tag text". In this case, the SGML tag is used as

```
<generic_identifier>tagtext
```

1.3.1 Heading Tagtext

Some of the elements, such as headings, require the content and start tag to be in the same record with no other tags in the content. The record end is interpreted as the end tag. These are documented as

```
<generic_identifier>Heading Content in same record
```

The first letter of each significant word in the heading data should be capitalized. The appearance of the heading in the resulting document is dependent on the layout selected.

1.3.2 Start of Text Block

With tags which are documented in the form

```
<generic_identifier>Start of text block ...
```

the tagtext represents the start of text for a block (see P, for example). It is not necessary that the content begins in the same record as the tag;

```
<generic_identifier>
Start of text block ...
```

will produce the same result.

1.3.3 Continued Text

With tags that are documented in the form

```
<generic_identifier>continued text
```

the first character of the text (if any is specified) will be placed immediately after the last character already on the output line, without any intervening blanks (see /FN, for example). Therefore, you should not enter content in the same record unless that is what you want to achieve.

1.4 Attributes

In their fullest form, some SGML tags can consist of the SGML start tag indicator, the generic identifier, an "attribute list", tag close indicator, and tagtext; for example,

```
<generic_identifier attribute list>tagtext
```

There are two forms of attributes. In the more common form, an attribute consists of an attribute name, an "=", and a value, without intervening blanks. In the less common form, an attribute name by itself is used. For example,

```
    <generic_identifier att1=value1 att2 att3=value3>tagtext
```

If an attribute's value contains anything other than letters and digits, it must be delimited by quotes or apostrophes, such as

```
    <generic_identifier attname="the value">
    <generic_identifier attname="it's value">
    <generic_identifier attname="1.5">
```

A reference summary of all tags and their attributes and values is provided in G on page 72.

2. The !DOCTYPE declaration

With the !DOCTYPE declaration the document type definition that should be used is indicated. For example

```
    <!DOCTYPE SGMLguid>
```

specifies SGMLguid tags and attributes to be used. Layout Types on page 49 gives a list of the available styles together with their intended use.

3. The GDOC and /GDOC Tags

These two tags are always required. The SEC attribute of the GDOC tag enables you to specify a ″security classification″ or to indicate that the document is a ″draft″ version. For example,

```
    <GDOC SEC="Confidential">
```

indicates that the information in the document is confidential, whereas

```
    <GDOC SEC="Draft">
```

indicates that the document is not yet in its final form.

Most of the SGML layouts utilize the value of the SEC attribute as information to be included on the title page, if a value has been specified for SEC. Some also include the SEC value in top or bottom running titles. Some layouts do not currently utilize the SEC value.

4. The INCLUDE Tag

Often you may find it useful to separate large documents into a series of smaller input files. The INCLUDE tag allows you to specify that the contents of another file are to be included during processing:

```
    <INCLUDE FILE=filename>
```

The file to be included may contain any combination of data and/or tags is appropriate.

The INCLUDE tag may also be useful if you are preparing memos (see the SGMLmemo layout), since it enables you to construct the distribution list by including one or more files containing people's names.

Part 2

The Front Matter

The front matter of a document includes the title-page components and any other preliminary information, including the tables of contents. The start and end of the front matter are identified with the FRONTM and /FRONTM tags:

```
<FRONTM>
tags and text for the front matter
</FRONTM>
```

With most of the CERN layouts, all of the pages in the front matter of a document will be numbered in lowercase roman numerals. The tags that may be used to identify the components of the front matter are described below.

5. The Title Page

The input for the title page is prepared in the following form:

```
<TITLEP>
tags for the title-page components
</TITLEP>
```

The TITLEP tag indicates the start of the information that is to appear on the title page, and the /TITLEP tag indicates the end of that information. The layout of the title-page information depends on the SGML layout that you have selected; some produce a ″full″ title page, and others produce a ″partial″ title page in which the title-page information occupies only as much of the first page as it requires. For layouts that produce a full title page, the title page will be considered as page ″i″, although a page number will not be printed on that page.

The tags that describe the components of the title page are discussed below. It is not necessary that you use all of them.

For the SGMLprep (preprint) layout two attributes should be specified; DIVISION, with value EP or TH, and NUMBER to give the reference number of the preprint:

```
<TITLEP DIVISION=TH NUMBER="4069/84">
```

5.1 Title

The title of the document is described by the TITLE tag. It is of the form

```
<TITLE>This is the Title of the Document
```

Some of the SGML layouts also cause the title to be used as a running title at the top or bottom of subsequent even- and/or odd-numbered pages. If the title is too long to be suitable for this purpose, you can provide a shortened version of that title by including the STITLE attribute:

```
<TITLE stitle="Short Title">Full Title
```

Multi-line titles can be defined by repeated use of the < TITLE > tag:

```
<TITLE>First Line of Title
<TITLE>Second Line of Title
<TITLE>Last Line of Title
```

With those layouts that use the title as a running title, only the title (or short title) from the first TITLE tag is used for the running title.

5.2 Author

The name of the author is specified as:

```
<AUTHOR>Name of Author
```

If the document has more than one author, repeat the AUTHOR tag for each author's name.

5.3 Address

If it is appropriate to place the name and address of the author's organization on the title page, then indicate the start of that information by using the ADDRESS tag, describe each line of the address information by using the ALINE tag, and indicate the end of the address information by using the /ADDRESS tag:

```
<ADDRESS>
<ALINE>Company name
<ALINE>first line of address
    .
    .
<ALINE>last line of address
</ADDRESS>
```

5.4 Author list

An alternative to AUTHOR and ADDRESS is to specify an author list, with links to the address and with (optional) notes.

```
<AUTHLIST>
<AUTHOR REFID="I1 I3">First Author's Name
<INST ID=I1>CERN, ....., Switzerland.
<INST ID=I2>Somewhere
else.
</INST>
<AUTHOR REFID=I2>Last Author's Name
<ANOTE ID=I3>Paid in part by......
</ANOTE>
</AUTHLIST>
```

The AUTHLIST tag identifies the start of the list and /AUTHLIST the end.

For each author the AUTHOR tag identifies his name and the REFID attribute is used to identify the corresponding institute and/or note.

The INST and /INST tags identify the institute and the ID attribute of the INST tag is used to give an identification to be used with the REFID attribute of the AUTHOR tag. This attribute is any ″name″ starting with a letter and 1 to 7 characters long.

The ANOTE and /ANOTE tags identify notes and the ID attribute of the ANOTE tag is used to give an identification to be used with the REFID attribute of the AUTHOR tag. This attribute is any ″name″ starting with a letter and 1 to 7 characters long.

5.5 Document Number

If the document is part of a series that has some form of publication series number, that number is specified as:

```
<DOCNUM>document number
```

5.6 Date

The DATE tag, if specified by itself, will cause today′s date to appear on the title page. If specified with text, as in

```
<DATE>date of publication
```

that text will be used as the date of publication of the document.

5.7 Copyright Line

The COPYRIGHT tag causes a copyright line to be included in the title-page information. It is of the form

```
<COPYRIGHT>[Text Line]
```

If tagtext is specified, the copyright line will consist of a copyright symbol followed by the specified tagtext. If no tagtext is specified, the information in the resulting copyright line will depend on the layout you have selected.

5.8 Copyright Notice

The COPYNOTE tag indicates the start of a ″copyright notice″ text block that is to appear after the copyright line. The copyright notice is of the form

```
<COPYNOTE>[Start of text block]
more text for the copyright notice
</COPYNOTE>
```

6. Abstract

If the document requires an Abstract, the heading and text are specified as:

```
<ABSTRACT>[Heading Text]
text of the abstract
</ABSTRACT>
```

If specified without heading text, the word "Abstract" will be used for the heading.

7. Preface

Any additional preliminary components that are required in the document must be identified as follows:

```
<PREFACE>[Heading Text]
text for this component
</PREFACE>
```

If used by itself without heading text, the PREFACE tag will use "Preface" for the heading.

8. Tables of Contents

The tables of contents are the last components of the front matter. They consist of three optional components: the table of contents, the list of figures, and the list of tables. They are specified as

```
<TOC>[Heading Text]
<FIGLIST>[Heading Text]
<TABLIST>[Heading Text]
```

The information that will appear in these components is collected automatically from the tagtext of all heading-level, figure-caption, and table-caption tags in your document. If your document is not large enough to warrant the inclusion of any or all three of these components, specify only the one(s) you want included. All three tags are ignored by the layouts in the "paper, letter, memorandum" category, since they are smaller documents that do not warrant tables of contents.

The table of contents will include the heading text of all level-zero through level-four heading tags in the document; if TOC is specified without tagtext, the heading will be "Contents". The list of figures will include the figure captions of all of the figures in the document; if FIGLIST is specified without tagtext, the heading will be "Figures". The list of tables will include the table captions of all of the tables in the document; if TABLIST is specified without tagtext, the heading will be "Tables".

Note: The tables of contents are the last components of the front matter of a document. If you are formatting your document in single-pass mode, however, the tables-of-contents entries for the body, appendix, and back matter have not yet been encountered. Therefore, the tables of contents will be printed at the end of the document, and it is your responsibility to move the resulting pages of output to their proper position in the document. If you format your document in multiple-passes mode, the tables of contents will print in their proper place, as the last components of the front matter. However, multiple-passes processing consumes additional resources.

9. Summary of Front Matter

The following table shows the different tags that can be used in the front matter. The entries are indented merely to show the hierarchical structure of the elements.

```
<FRONTM>
  <TITLEP[ ALIGN="value"]>
    <TITLE[ STITLE="text"]>Document Title

    <AUTHOR>Author's Name
      <ADDRESS>
        <ALINE>First Line of Address
          (repeat as necessary)
        <ALINE>Last Line of Address
      </ADDRESS>
                  or
    <AUTHLIST>
      <AUTHOR REFID=rrr>Name of Author
      <INST ID=id>Name and Address of Institute
    </AUTHLIST>

    <DATE>[date]
    <COPYRIGHT[ SYMBOL="symbol"]>[Heading Text]
    <COPYNOTE>[Start of text]
    </COPYNOTE>
  </TITLEP>
  <ABSTRACT[ STITLE="text"]>[Heading Text]
    (<H2> through <H6>, with basic document components)
  </ABSTRACT>
  <PREFACE[ STITLE="text"]>[Heading Text]
    (<H2> through <H6>, with basic document components)
  </PREFACE>
  <TOC>[Heading Text]
  <FIGLIST>[Heading Text]
  <TABLIST>[Heading Text]
</FRONTM>
```

Part 3

The Body

The start and end of the body of the document are identified with the BODY and /BODY tags:

```
<BODY>
tags and text for the body
</BODY>
```

10. Heading Levels -- the Divisions of the Body

The body of the document will consist of one or more level-one divisions, each with a heading. The start of each level-one division is specified as

```
<H1>Heading Text
text and "basic document components"
```

If a level-one division is large enough that it needs to be sub-divided, the start of each second-level division is specified as

```
<H2>Heading Text
text and "basic document components"
```

In the same fashion, a level-two division can be sub-divided using a H3 tag, level-three divisions by H4, level-four divisions by H5, and level-five divisions by using the H6 tag.

10.1 Layout-Dependent Considerations

The tagtext for each heading level will be composed in a style appropriate to the layout you have selected and the output device for which you are formatting the document.

Some of the SGML layouts also use the level-one and level-two headings as running titles at the tops and/or bottoms of subsequent even- and/or odd-numbered pages. If the heading text is too long to be suitable for this purpose, then you may specify the STITLE attribute, in the manner described for the TITLE tag.

The ABSTRACT and PREFACE tags are logically level-one components. Therefore, you may also use the H2, H3, ... tags to subdivide those components.

10.2 Level-Zero Components

A special tag, H0, is provided for large documents in which it is appropriate to group the level-one divisions of the body. For example, one might do this in a reference manual, thereby grouping into "parts" the level-one divisions that comprise the "chapters".

10.3 Level-One Subheadings

A H1SUB tag exists for those situations in which additional heading text is required immediately after a level-one heading. For example, the input sequence

```
<H1>Heading Text
<H1SUB>Additional Heading Text
<H1SUB>And Still More Heading Text
text and "basic document components"
```

specifies a level-one heading followed by two level-one subheadings. The tagtext for the subheadings will be composed immediately after the level-one tagtext, in the same style as the level-one heading. Level-one subheadings are not included in the table of contents.

10.4 Referring to Headings

In order to generate a reference to a heading, you must assign that heading an identifier. All of the heading-level SGML tags (H0 through H6, and all other tags equivalent to H1) provide an ID attribute that allows you define such an identifier. For example, the heading above was specified in the form

```
<H2 ID=hd2ref>Referring to Headings
```

The name of the identifier can contain from one to seven characters, and can be used in the HDREF tag to generate a reference to that particular heading. For example, the input sequence

```
This is discussed under the heading
<HDREF REFID=hd2ref>, and this
particular example is known as a
"backward reference".
```

results in: This is discussed under the heading 10.4 , and this particular example is known as a "backward reference". If tagtext is specified with the HDREF tag, it will be treated as continued text.

If the heading and the heading reference are on different pages of the composed document, then the heading number (if the style has numbered headings) followed by the words "on page" and the number of the page on which the heading appears will be generated as part of the heading reference.

Note: You will sometimes wish to generate a reference to a heading at some point in the document before that heading occurs. This is known as a forward reference ... you are referring to something that occurs later in the input sequence. In order to achieve the correct output, you must format your document in multiple-pass mode. Failure to do so will result in the text "REFID=name not defined" being printed instead of the reference.

11. Summary of Body Component

The following table shows the different tags that can be used in the body. The entries are indented merely to show the hierarchical structure of the elements.

```
<BODY>
  <H0[ STITLE="text"]>[Heading Text]
    basic document components
    <H1[ STITLE="text"]>Heading Text
      <H1SUB>Heading Text
        basic document components
        <H2[ STITLE="text"]>Heading Text
          basic document components
          (<H3> through <H6>, with basic document components)
</BODY>
```

Part 4

The Appendix Component

If the document has appendices, they normally belong after the body and before the start of the back matter (if present). The start and end of the appendix component are indicated by

```
<APPENDIX>
tags and text for each appendix
</APPENDIX>
```

The start of each appendix is specified by the H1 tag,

```
<H1>[Heading Text]
```

Part 5

The Back Matter

The back matter of a document, if present, usually consists of components such as bibliographies, glossaries, and indexes. The start and end of the back-matter components are identified with the BACKM and /BACKM tags:

```
<BACKM>
tags and text for the back matter
</BACKM>
```

The tags specific to the back matter are described below.

12. Index

SGML provides tags for generating index-item entries (discussed later). If you have used these index-item tags to create index entries for your document, then use the tag

```
<INDEX>[Heading Text]
```

to define the index heading and to cause the index entries to be composed and included in the output. The index heading is logically a level-one heading; if tagtext is not specified, the heading will be "Index".

13. The Bibliography

The heading for the bibliography is indicated with a H1 tag. The bibliography is a list of bibliographic items. Its start is indicated by the BL tag, and its end by the /BL tag. The input sequence for a bibliography will therefore be in the following form:

```
<H1>Bibliography
Text prior to the list, if desired.
<BL>
first bibliographic item
   . . .
last bibliographic item
</BL>
Text following the list, if desired.
```

13.1 The Bibliographic Items

The input sequence to define an item in the bibliography list is of the form:

```
<BIB>
Author's Name, Initials.
<CIT>Title of Cited Work</CIT>.
other information
```

The CIT and /CIT tags indicate the start and end of the title of the cited work. Most manuals of style are quite specific as to the other information that is required in a bibliographic item; if you are using a SGML layout that has been designed to adhere to the requirements of some specific manual of style, please consult that manual for the exact nature and ordering of this other information.

13.2 Referencing Bibliographic Items

At various places throughout your document, you will need to make reference to items that are in your bibliography. Most modern manuals of style suggest that you place some combination of the author's name and the date of publication in (parentheses) or [brackets] at the point of reference, since this provides both an immediate indication of the source and the least disruption in the readability of the document. Other manuals of style suggest that you use a footnote or endnote to provide that information. Still others suggest that you provide some form of numeric annotation reference number, and that the bibliographic list be numbered accordingly.

13.3 Numbered Bibliography Lists

Some of the SGML layouts will automatically number the bibliography list. To gain access to these reference numbers, you must assign a unique identifier to each of the BIB tags in your bibliography list using the ID attribute, as

```
<BL>
<BIB ID=name1>
     ...
<BIB ID=name2>
     ...
</BL>
```

The identifier name may contain no more than eight characters. If you are formating your text in single-pass mode you must also use the BIBID tag to pre-generate the reference numbers at the beginning of the document, in the same order as you have defined the items in the bibliography list:

```
<GDOC>
<BIBID REFID=name1>
<BIBID REFID=name2>
<BIBID REFID=name3>
  ...
```

At any point in your document where you must refer to an item in your bibliography list, you then use the BIBREF tag:

```
<BIBREF REFID=name>[continued text]
```

For example, the input sequence

```
As discussed in <BIBREF REFID=sgml>,
this is how you refer to a bibliographic item.
```

will produce a bibliographic reference in the style of the SGML layout you are using; in the layout being used to produce this document, the above input sequence results in: As discussed in [5], this is how you refer to a bibliographic item.

If you are using a SGML layout which numbers the items in the bibliography list and you wish to eliminate that numbering, you must use the SET tag (described in an appendix) to set the bibliographic reference style to NONE. If you are using a SGML layout that does not number the items in the bibliography list and you wish to force that numbering, you must use the SET tag to set the bibliographic reference style to NUMBER.

13.4 Labelled Bibliography Lists

A few manuals of style suggest that you devise a short "label" for each item in your bibliography list, formed by taking the first few characters of the (primary) author's last name and the year of publication, plus a letter to identify more than one work by that author in that year (if needed). To do this, you must use the SET tag (described in an appendix) to set the bibliographic reference style to LABEL, and specify the label as the ID attribute of BIB and the REFID attribute of the BIBREF tag. Do not use the BIBID tag.

13.5 Referencing a List or Range of Items

The REFID attribute can be specified in a list-notation and/or a range-notation form, as well; for example,

```
<BIBREF REFID="one two three">
<BIBREF REFID="one-two">
<BIBREF REFID="one two-three four">
```

The first is a list of three items that will be punctuated as "1,2,3". The second is a range of two items that will be punctuated as "1 – 2". The third is a list of four items that will be punctuated as "1,2 – 3,4".

14. Other Back-Matter Components

If additional back-matter components are required, they must be initiated by using the H1 tag.

15. Summary of the Back Matter

The following table shows the different tags that can be used in the back matter. The entries are indented merely to show the hierarchical structure of the elements.

```
<BACKM>
  <INDEX>[Heading Text]
  <H1>Bibliography
    Text prior to the list, if desired.
    <BL>
      <BIB>
        Author's Name, Initials.
        <CIT>Title of Cited Work</CIT>.
        other information
    </BL>
    Text following the list, if desired.
  <H1>Other Back Matter
</BACKM>
```

Part 6

Basic Document Components

There are a great many document components that may appear (almost) anywhere in the front matter, body, appendix, and back matter.

16. Paragraphs

The start of a paragraph is indicated with the P tag,

```
<P>Start of text block ...
rest of paragraph text block
```

17. Notes

A note is a block of text that is intended to qualify or clarify. The NOTE and /NOTE tags are used to create a note, in the form

```
<NOTE>Start of note block ...
more of note block ...
last of note block.
</NOTE>
```

For example, the above input sequence produces:

Note: Start of note block ... more of note block ... last of note block.

18. Footnotes

The FN and /FN tags are used to define the start and end of a block of footnote text:

```
This is the sentence containing the word
<FN>Start of text block ...
rest of footnote text block.
</FN>,
and the footnote text appears below.
```

The last word before the FN tag will have a footnote reference number appended to it, and the footnote text block will appear at the bottom of the current output page (or column, if the layout being used produces its output in multiple-columns-per-page mode). For example, the sequence above produces the following result: This is the sentence containing the word[2], and the footnote text appears below. If tagtext is specified with the /FN tag, it will be treated as continued text.

The FN tag can also take the attribute SYMBOL="string" to specify a symbol for a particular footnote. This can be useful e.g. to distinguish "translators notes" from other footnotes in a document.

18.1 Footnote Identifiers and References

The FN tag provides an ID attribute that defines a footnote identifier name:

```
<FN ID=name>Start of text block ...
```

The name can contain up to seven letters and/or digits.

The FNREF tag allows you to generate another footnote reference to a previous footnote, by specifying its identifier name as the REFID attribute:

```
<FNREF REFID=name>continued text
```

For example, the input sequence

```
One of the footnotes in this document
<FNREF REFID=foot>
was defined with ID=foot.
```

results in: One of the footnotes in this document[2] was defined with ID=foot.

19. Endnotes

A footnote is that which comes at the bottom (foot) of something. An endnote is that which comes at the end of something. Endnotes are created in much the same way as footnotes. The EN tag defines the start of the endnote text block, and the /EN tag defines its end. They are used in the following fashion:

```
Sentence containing a word
<EN>Start of endnote text block ...
rest of endnote text block.
</EN>[continued text]
for which an endnote is to be generated.
```

The above input sequence produces the following: Sentence containing a word[1] for which an endnote is to be generated.

[2] Start of text block ... rest of footnote text block. This second sentence is included to demonstrate the offset of subsequent lines of the text block.

19.1 Additional References to an Endnote

The ENREF tag allows you to make additional references to an endnote that has been specified with the ID attribute. For example, if the endnote above had been defined with

```
<EN ID=above>
```

then

```
This is an additional reference to
the previous endnote
<ENREF REFID=above>.
```

produces: This is an additional reference to the previous endnote[1].

19.2 Positioning the Endnotes

You control where you want the endnotes to be printed, by specifying the ENDNOTES tag.[3] Any already-defined endnotes will be printed at that point in the document, and endnote numbering will resume at 1 for the first endnote defined thereafter. Therefore, you can use it as many times as you wish. For example, you might define endnotes and then cause them to be printed at the end of each level-one division of the body of your document. For example, the following input sequence

```
<H2>Positioning the Endnotes
You control where you want ...
rest of text comprising the above discussion ...
results in:
<ENDNOTES >
followed by additional text.
```

results in:

[1] Start of endnote text block ... rest of endnote text block. This is additional text has been added to demonstrate a larger number of lines in the endnote text block.

followed by additional text.

20. Citations

References to other works are identified using the CIT and /CIT tags. For example,

```
This is the
<CIT>CERN SGML User's Guide</CIT>
document.
```

results in: This is the *CERN SGML User's Guide* document. If tagtext is specified with the /CIT tag, it will be treated as continued text.

[3] If you do not, the endnotes will be printed automatically by the formatter at the end of your document.

21. Quotations

21.1 Long Quotations

These are also referred to as block quotations, to distinguish them from in-line quotations. A long quotation is identified as follows:

```
text before long quotation
<LQ>[start of]
long-quotation text block
</LQ>
text after long quotation
```

The result will be displayed in a manner determined by the layout you have selected. The layout selected for this document produces:

> This is a long-quotation text block. The indentation and spacing of the text in the block are controlled by the layout you select for your document.

followed by whatever text comes after the long quote.

21.2 Short Quotations

These are also referred to as in-line quotations. The example shown in the following input sequence

```
Text before short quotation
<Q>
short-quotation text block
</Q>
text after short quotation.
```

produces the following: Text before short quotation ″short-quotation text block″ text after short quotation. If tagtext is specified with the /Q tag, it will be treated as continued text.

22. Examples

The start of a block of text representing an example is indicated with the XMP tag, and the end of the example block is indicated with the /XMP tag. For example,

```
<XMP>
This is the first input line of the
example block, and this is the
last input line of the block.
</XMP>
```

produces the following result:

```
This is the first input line of the
example block, and this is the
last input line of the block.
```

23. Highlighted Phrases

Four levels of highlighted phrases are provided by tags of the form

```
<HPn>[start of]
text to be highlighted
(n=0,1,2,3,4)
</HPn>[continued text]
```

The HP0 – /HP0 tags enclose text that will appear as normal text (in other words, the text will not be highlighted). The HP1 – /HP1, HP2 – /HP2, and HP3 – /HP3 tags result in increasing levels of highlighting.[4]

If you wish to highlight text that is within a block that is being placed in the output document exactly as entered (line by line, instead of the block mode in which normal text is produced), you can do so by specifying the highlighted-phrase tags in the form:

```
This is level <HP0>zero</HP0> highlighting.
This is level <HP1>one</HP1> highlighting.
This is level <HP2>two</HP2> highlighting.
This is level <HP3>three</HP3> highlighting.
```

For example, the above input sequence produces:

This is level zero highlighting.
This is level *one* highlighting.
This is level **two** highlighting.
This is level ***three*** highlighting.

This same in-line technique can also be used for citations and short quotations within example blocks.

24. Boxes

The BOX and /BOX tags define the start and end of a text block that is to be enclosed in a box. They are used in the manner

```
Text before box.
<BOX>
This text block is enclosed in a box.
The result has been ...
</BOX>
Text after box.
```

and result in the following: Text before box.

[4] If the document is formatted for an output device that supports multiple fonts, these levels of highlighting will be (respectively) italic, boldface, and boldface italic. If the output device does not have multiple fonts or if the fonts provided by SGML do not include those faces, alternative approaches to highlighting will result. For example, for simple terminals and line printers, underscoring is used to represent italics, overprinting is used to represent boldface, and both are used to represent boldface italics.

This text block is enclosed in a box. The result has been achieved through the use of the BOX and /BOX tags. You may include within the box any other SGML tags that do not generate headings or indicate the start or end of a major component of the document.

Text after box.

25. List Structures

25.1 Ordered Lists

In an ordered list, each item in the list is annotated in some fashion -- with arabic numerals, with letters, or with roman numerals. The OL tag identifies the start of an ordered list. The start of each item in the list is identified with the LI tag. The /OL tag identifies the end of the ordered list. For example, the sequence

```
<OL>
<LI>This is the text for ordered list item 1.
<LI>This is the text for ordered list item 2.
This second sentence demonstrates the offset of
subsequent lines of the list item.
<LI>This is the text for ordered list item 3.
</OL>
```

produces the following result:

1. This is the text for ordered list item 1.

2. This is the text for ordered list item 2. This second sentence demonstrates the offset of subsequent lines of the list item.

3. This is the text for ordered list item 3.

   Ordered lists can be nested within ordered lists, as is demonstrated by the input sequence

```
<OL>
<LI>This is ordered list item 1, level 1.
<LI>This is ordered list item 2, level 1.
<OL>
<LI>This is ordered list item 1, level 2.
</OL>
<LI>This is ordered list item 3, level 1.
</OL>
```

The maximum depth to which ordered lists can be nested is six, as is demonstrated below.

1. This is ordered list item 1, level 1.

   a. This is ordered list item 1, level 2.

i. This is ordered list item 1, level 3.

1) This is ordered list item 1, level 4.

a) This is ordered list item 1, level 5.

i) This is ordered list item 1, level 6.

25.2 Unordered Lists

Unordered lists are formed in the same way as ordered lists; the only difference is the style of annotation of the list items. The start of an unordered list is identified with the UL tag. The start of each list item is identified with the LI tag. The end of an unordered list is identified with the /UL tag. Unordered lists can be nested to a depth of 6, as is demonstrated below.

```
<UL>
<LI>This is unordered list item 1, level 1.
<UL>
<LI>This is unordered list item 1, level 2.
<UL>
<LI>This is unordered list item 1, level 3.
<UL>
<LI>This is unordered list item 1, level 4.
<UL>
<LI>This is unordered list item 1, level 5.
<UL>
<LI>This is unordered list item 1, level 6.
</UL>
</UL>
</UL>
</UL>
</UL>
</UL>
```

- This is unordered list item 1, level 1.
  - This is unordered list item 1, level 2.
    - This is unordered list item 1, level 3.
      - This is unordered list item 1, level 4.
        - This is unordered list item 1, level 5.
          - This is unordered list item 1, level 6.

25.3 Simple Lists

Simple lists are formed the same as ordered and unordered lists; the only difference is that the list items are not annotated. The start of a simple list is identified with the SL tag. The end of a simple list is identified with the /SL tag. Simple lists can be nested to a depth of 6, as is demonstrated below.

```
<SL>
<LI>This is simple list item 1, level 1.
<SL>
<LI>This is simple list item 1, level 2.
<SL>
<LI>This is simple list item 1, level 3.
<SL>
<LI>This is simple list item 1, level 4.
<SL>
<LI>This is simple list item 1, level 5.
<SL>
<LI>This is simple list item 1, level 6.
</SL>
</SL>
</SL>
</SL>
</SL>
</SL>
```

This is simple list item 1, level 1.

This is simple list item 1, level 2.

This is simple list item 1, level 3.

This is simple list item 1, level 4.

This is simple list item 1, level 5.

This is simple list item 1, level 6.

25.4 List Items with Headings

As an alternative to LI, the LIHD tag can also be used to identify items in an ordered, unordered, or simple list. For example,

```
<LIHD>List Item Heading
```

The LIHD tag functions in the same fashion as LI with the exception that it treats the tagtext as a heading. The tagtext should therefore not end with any form of punctuation. The text on the lines following the LIHD tag will become the body of the list item. The following demonstrates the effect of using LIHD in the six levels of ordered lists.

```
<OL>
<LIHD>List Item Heading at Level 1
This is the text following the List Item Heading,
and is formatted as the body of the list item.
<OL>
```

```
<LIHD>List Item Heading at Level 2
This is the text following the List Item Heading,
and is formatted as the body of the list item.
<OL>
<LIHD>List Item Heading at Level 3
This is the text following the List Item Heading,
and is formatted as the body of the list item.
<OL>
<LIHD>List Item Heading at Level 4
This is the text following the List Item Heading,
and is formatted as the body of the list item.
<OL>
<LIHD>List Item Heading at Level 5
This is the text following the List Item Heading,
and is formatted as the body of the list item.
<OL>
<LIHD>List Item Heading at Level 6
This is the text following the List Item Heading,
and is formatted as the body of the list item.
</OL>
</OL>
</OL>
</OL>
</OL>
</OL>
```

1. *LIST ITEM HEADING AT LEVEL 1*

   This is the text following the List Item Heading, and is formatted as the body of the list item.

   a. *List Item Heading at Level 2*

      This is the text following the List Item Heading, and is formatted as the body of the list item.

      i. *LIST ITEM HEADING AT LEVEL 3*: This is the text following the List Item Heading, and is formatted as the body of the list item.

         1). *List Item Heading at Level 4*: This is the text following the List Item Heading, and is formatted as the body of the list item.

            a) *List Item Heading at Level 5*: This is the text following the List Item Heading, and is formatted as the body of the list item.

               i) *List Item Heading at Level 6*: This is the text following the List Item Heading, and is formatted as the body of the list item.

Note that you can mix LI and LIHD tags in the same list, as has been done in the unordered list below:

```
<UL>
<LI>This is a list item
and looks just like the list items in the previous examples.
<LIHD>List Item with Heading
This is the text following the List Item Heading,
and is formatted as the body of the list item.
<LI>And this is another list item without a heading.
</UL>
```

- This is a list item and looks just like the list items in the previous examples.
- *LIST ITEM WITH HEADING*

  This is the text following the List Item Heading, and is formatted as the body of the list item.
- And this is another list item without a heading.

25.5 Referring to List Items

An identifier can be assigned to a list item on the LI or LIHD tags, and then a reference can be generated later to a list item:

```
<OL>
<LI ID=one>List item one.
<LIHD ID=two>List item two.
</OL
These are references to list items
<LIREF REFID=one>
and <LIREF REFID=two>
from the above list.
```

produces:

1. List item one.
2. *LIST ITEM TWO.*

These are references to list items "1." and "2." from the above list.[5]

25.6 Definition Lists

The start of a definition list is identified with the DL tag. The end of a definition list is identified with the /DL tag. The manner of identifying list items for definition lists is different than for ordered, unordered, or simple lists. Two SGML tags must be used. They are of the form

```
<DT>term
<DD>text of definition list item
```

The DT tag identifies the definition term, and the DD tag identifies the definition description.

[5] Because of the nature of the annotation styles for items in simple and unordered lists, there seems to be little point in referencing their list items; it works, but seems pointless.

You can have multiple DT tags per DD tag, and multiple DD tags per DT tag. Definition lists can be nested to a depth of 6, as is demonstrated below.

```
<DL>
<DT>term
<DD>This is definition list item 1, level 1.
<DL>
<DT>term one
<DT>term two
<DD>This is definition list item 1, level 2.
It has multiple terms for one definition.
<DL>
<DT>term
<DD>This is definition list item 1, level 3.
<DD>It has multiple definitions for one term.
<DL>
<DT>term
<DD>This is definition list item 1, level 4.
<DL>
<DT>term
<DD>This is definition list item 1, level 5.
<DL>
<DT>term
<DD>This is definition list item 1, level 6.
</DL>
</DL>
</DL>
</DL>
</DL>
</DL>
```

term This is definition list item 1, level 1.

> **term one**
> **term two** This is definition list item 1, level 2. It has multiple terms for one definition.
>
> > **term** This is definition list item 1, level 3.
> >
> > It has multiple definitions for one term.
> >
> > > **term** This is definition list item 1, level 4.
> > >
> > > > **term** This is definition list item 1, level 5.
> > > >
> > > > > **term** This is definition list item 1, level 6.

An attribute TSIZE=n, where n is a number (default 10) can be specified to increase or decrease the space reserved for the term.

```
<DL TSIZE=15>
<DT>term
<DD>This is definition list item 1, level 1.
</DL>
```

long term This is definition list item 1, level 1.

25.7 Definition Lists with Term and Description Headings

The DTHD and DDHD tags provide a means for creating definition terms and descriptions as headings. For example, the input sequence

```
<DL>
<DTHD>heading
<DDHD>description heading
<DT>term
<DD>text of definition list item
</DL>
```

produces

heading *description heading*

term text of definition list item

25.8 Glossary Lists

Glossary lists are defined in a manner similar to definition lists. They are useful when the term is a phrase instead of a single word. The start of a glossary list is identified with the GL tag. The end of a glossary list is identified with the /GL tag. Each item in the list is defined in the fashion:

```
<GT>word or phrase
<GD>text of glossary list item
```

Six levels of glossary lists are provided. A glossary list appears as follows on output.

```
<GL>
<GT>This is a glossary term
<GD>This is glossary description 1.
This second sentence demonstrates the offset of subsequent lines of
a glossary description.
<GT>This is another glossary term
<GD>This is glossary description 2.
This second sentence demonstrates the offset of subsequent lines of
a glossary description.
</GL>
```

This is a glossary term: This is glossary description 1. This second sentence demonstrates the offset of subsequent lines of a glossary description.

This is another glossary term: This is glossary description 2. This second sentence demonstrates the offset of subsequent lines of a glossary description.

25.9 Nesting Various Types of Lists

The various types of lists can be intermixed, as shown below.

```
<OL>
<LI>This is ordered list item 1, level 1.
This sentence serves only to pad out the block
of text to demonstrate offsets.
<LI>This is ordered list item 2, level 1.
<DL>
<DT>first
<DD>This is definition item 1, level 2.
This sentence pads the block to demonstrate offsets.
<DT>second
<DD>This is definition item 2, level 2.
<UL>
<LI>This is unordered list item 1, level 3.
<GL>
<GT>glossary term 1
<GD>level 4
</GL>
<LI>This is unordered list item 2, level 3.
<SL>
<LI>This is simple list item 1, level 4.
</SL>
</UL>
<DT>third
<DD>This is definition item 3, level 2.
This sentence serves only to pad out the block of text to
illustrate offsets.
</DL>
<LI>This is ordered list item 3, level 1.
</OL>
We are now back to normal text outside the list structure.
```

1. This is ordered list item 1, level 1. This sentence serves only to pad out the block of text to demonstrate offsets.

2. This is ordered list item 2, level 1.

   first This is definition item 1, level 2. This sentence pads the block to demonstrate offsets.

   second This is definition item 2, level 2.

   - This is unordered list item 1, level 3.

     glossary term 1: level 4

   - This is unordered list item 2, level 3.

     This is simple list item 1, level 4.

   third This is definition item 3, level 2. This sentence serves only to pad out the block of text to illustrate offsets.

3. This is ordered list item 3, level 1.

We are now back to normal text outside the list structure.

25.10 Interrupting the Items in a List -- the List Paragraph

The LP tag indicates the start of a list paragraph which interrupts the items in a list. The text of the list paragraph will be positioned so that it appears at the same level of indentation as the annotation symbols for the list. For example, the input sequence

```
<OL>
<LI>This is the first list item.
<LP>This is the first line of the list paragraph.
<LI>This is the second list item.
</OL>
```

produces the following result:

1. This is the first list item.

This is the first line of the list paragraph.

2. This is the second list item.

The LP tag can be used anywhere in any type of list structure.

26. Tables and Figures

Tables and figures may be defined anywhere in your document. If there is room in the current output column, the table or figure will appear at exactly the position in which it was defined. Otherwise, it will be promoted to the top of the next output column.

26.1 Figures

The SGML tags to define a figure are used as follows:

```
Last text line of paragraph in which the
first reference to the figure is made.
<FIG>
Text for body of figure.
Each input line will appear as entered.
<FIGCAP>This is the Figure Caption
<FIGDESC>And this is the figure description,
which is optional text providing additional
descriptive information about the figure.
</FIGDESC>
</FIG>
```

The above input sequence produces the following ... Last text line of paragraph in which the first reference to the figure is made.

```
Text for body of figure.
Each input line will appear as entered.
```

Figure 1: This is the Figure Caption. And this is the figure description, which is optional text providing additional descriptive information about the figure.

The definition of the figure is then followed by an input line defining the start of the next paragraph, or a heading.

26.2 Tables

The SGML tags to define a table are used as follows:

```
    Last text line of paragraph in which the
    first reference to the table is made.
    <TAB>
    <TABCAP>This is the Table Caption
    <TABDESC>And this is the table description,
    which is optional text providing additional
    descriptive information about the table.
    </TABDESC>
    Text for body of table.
    Each input line will appear as entered.
    </TAB>
```

This is text that comes before the table, just to show how formatting is currently being handled.

Table 1: This is the Table Caption

And this is the table description, which is optional text providing additional descriptive information about the table. Note that the table description, if present, MUST be terminated with the /TABDESC tag!

```
This is the body of the table.
Each input line will appear as entered.
```

This is the first line of the next paragraph, to show that we are back to the normal formatting environment.

26.3 Defining the Body of a Table or Figure

The body of a table or figure may consist of textual information, or blank lines (if the body is to be pasted in after the document has been printed), or both.

26.3.1 Single cell material

A common class of tables are so called "single cell" tables. These have single line entries arranged in columns. The SINCELL tag identifies the start of the single cell material and the /SINCELL tag the end. Each row is identified by the ROW tag and ended by an /ROW. Each column is identified by a T tag. The alignment of any column can be specified by the ALIGN=value attribute on each T tag. The values permitted are: LEFT, RIGHT, CENTRE, and CHAR, for left alignment, right alignment, centering, and alignment on a character. If ALIGN=CHAR is specified a CHAR="character" attribute specifying the character to align on can also be given – default is ".". The ALIGN and CHAR attributes can also be specified on the ROW tag and are then applied to all columns in the row. By default the alignment of a column in a row is the same as that of the same column in the previous row. For the first row the default is to left align. For example the sequence:

```
<TAB ID=XSECT>
<TABCAP>Differential cross-section
<SINCELL>
<ROW ALIGN=CENTRE><T>t    <T> range <T> dσ/dt <T> error </ROW>
<ROW ALIGN=CHAR  ><T>0.05<T>  0.10 <T>  124.0 <T>  15.0 </ROW>
<ROW>             <T>0.15<T>  0.10 <T>  100.0 <T>  10.0 </ROW>
<ROW>             <T>0.25<T>  0.10 <T>   70.0 <T>   6.0 </ROW>
</SINCELL>
</TAB>
```

produces

Table 2: Differential cross-section

t	range	dσ/dt	error
0.05	0.10	124.0	15.0
0.15	0.10	100.0	10.0
0.25	0.10	70.0	6.0

Errors: except for errors that are easy to correct, for example attempt to align on a non-existing character, you may encounter a message indicating that the entries in the table are too wide to fit in on the page. The size of the characters can, on some printers, be reduced and you can try <TAB ID=name FONT=FOOT> to reduce the size to that of footnotes.

26.3.2 Pasted-In Material

Both the FIG and TAB tags have a DEPTH attribute. This allows you to specify the number of blank lines that are to be generated as the body of the table or figure. For example, specifying

```
<FIG DEPTH=36>
```

will cause 36 blank lines to be generated as the body of this particular figure.

26.3.3 Textual Material

Lines of text that occur after the FIG tag (for figures) and the TABCAP and /TABDESC tags (for tables), will be output in a one-line-in/one-line-out fashion.

26.3.4 Textual and Pasted-In Material

You can specify the DEPTH attribute and also provide textual information. For a figure, the blank lines will be generated before the textual information. For a table, the blank lines will be generated after the textual information.

26.3.5 Graphics Inserted in the Figure

Several types of computer generated graphics can be included in a figure and printed together with the text on certain output devices (e.g. the APA6670 and IBM 3812 printers). Information on graphics sources and the method to convert them into a form that can be used with SGML is given in Graphic Sources on page 45.

To specify the inclusion of a graph the PICTURE tag is used. With the attribute NAME the name of the file containing the graphic information is specified. An optional attribute HMOVE with value CENTRE (default), LEFT or RIGHT is used to specify how the graphic should be positioned, for example:

```
<FIG>
<PICTURE NAME=GRAPHICS>
<FIGCAP>This is the Figure Caption
<FIGDESC>And this is the figure description,
which is optional text providing additional
descriptive information about the figure.
</FIGDESC>
</FIG>
```

26.4 Referencing Tables and Figures

The tags and text that define a table or figure should be positioned at the end of the paragraph that first references the table or figure. In some cases, you may wish to refer to that table or figure by its number, before or after the tags and text that define it. The FIGREF and TABREF tags are provided for this purpose.[6] If tagtext is specified with these tags, it will be treated as continued text.

For example, assume that you must now make your first reference to a particular table, before you have defined the table. At the point where you wish to reference it, use the TABREF tag to assign it an identifier:

[6] For forward references when the document is formatted in single-pass mode, these tags pre-generate the figure or table number.

```
As is illustrated in
<TABREF REFID=example>,
the rest of the sentence ...
```

Identifiers can be a maximum of 7 characters long. The name that you chose for a table or figure identifier must be unique (you cannot have two figures with the same identifier, but you can have a figure and a table with the same identifier). The above input sequence produces the following result: As is illustrated in table 2, the rest of the sentence ...

Then, at the end of the paragraph in which that sentence appears, use the ID attribute of the TAB tag:

```
<TAB ID=example>
<TABCAP>The Table Caption
body of table
</TAB>
```

The table that the above defines appears below.

Table 2: The Table Caption

```
body of table
```

26.5 Forward Out-of-Sequence References

The first use of a particular table or figure identifier assigns the identifier a value that is one greater than the previous table or figure number. Thus, tables and figures will each be numbered in the order in which their identifiers occur in the input. Two special tags, TABID and FIGID, allow you to pre-generate these values, at the beginning of your document, so that the identifiers will already have values when they are encountered as attributes of the FIGREF, TABREF, FIG, or TAB tags. For example,

```
<TABID REFID=dog>
<TABID REFID=cat>
```

will assign successive values to the identifiers. Thereafter, you can make out-of-sequence references to these two tables.

Once you have associated an identifier with a table, you can use the TABREF tag to make other references to it, anywhere thereafter in your document.

Note: This is only needed when you format your document in single-pass mode.

26.6 Referencing a List or Range of Figures or Tables

The REFID attribute can be specified in a list-notation and/or a range-notation form, as well; for example,

```
<FIGREF REFID="one two three">
<FIGREF REFID="one-two">
<FIGREF REFID="one two-three four">
```

The first is a list of three items that will be punctuated as ″1, 2, and 3″. The second is a range of two items that will be punctuated as ″1 – 2″. The third is a list of four items that will be punctuated as ″1, 2 – 3, and 4″.

27. Process-Specific Code

It is sometimes desirable to include in a document′s input file information whose appearance in the resulting output is dependent on the output device for which the document is being processed.

```
<PSC PROC="device1 device2 ... deviceN">
process-specific code block
</PSC>
```

For example, a block to be processed only if the output device is a line printer, an IBM APA6670, or a Xerox 2700 would appear as:

```
<PSC PROC="PRINT APA6670 X2700">
contents of block
</PSC>
```

If the output device for which the document is being processed is not one of those specified in the PROC= list (or if no PROC= list is specified), the contents of the block will not be processed.

28. Tags for Constructing Indexes

Index terms are created throughout the document by using the I1, I2, I3, and IREF tags, and the resulting index is printed in the back matter by using the INDEX tag.

The IX attribute of these tags enables you to create up to nine indexes for a document. When any of the tags are used without the IX attribute, index number 1 is assumed.

The I1, I2, and I3 tags can also take a combination of PG, SEE, ID, REFID, and SEEID attributes. These are described in more detail below.

28.1 Index Terms and Page References

An index consists of some number of index terms, each of which has a corresponding list of one or more page references to the locations in the document where that information is discussed. In the simplest case, a page reference will be the number of the page that was being produced at the time the index term was added to the index.

Three levels of index terms can be created. When an index is printed, the index terms appear in alphabetical order, by level. For example,

```
level-one term ... list of page references
   level-two term ... list of page references
      level-three term ... list of page references
next level-one term ... list of page references
```

28.2 The Index-Term Tags

The index term specified with the I1 tag is added to the index as a level-one index term, and the current page number in the document is added to its list of page references. When an index term is specified with the I2 tag, that term is added to the index as a level-two term under the most-recently-specified level-one index term. Similarly, the I3 tag adds its index term as a level-three term under the most recent level-one and level-two index terms.

28.3 An Example

The following example demonstrates the use of the I1, I2, and I3 tags to add several index terms to the index for a document about house pets. Each line shown as "(text on page ..)" represents the number of the page that was being produced at the time the tag below it was encountered.

```
(text on page 10)
<I1>cats
(text on page 34)
<I2>proper care
(text on page 42)
<I3>life span
```

If these were the only index terms created in the document, the resulting index, when printed, would appear as follows:

```
cats ... 10
   proper care ... 34
      life span ... 42
```

28.4 Index-Term Identifiers

The ID attribute enables you to assign an index-identifier name to a particular index term. This makes it considerably easier to add lower-level index terms without having to re-specify the higher-level tag again. For example, we can extend the previous example as follows:

```
(text on page 57)
<I3 ID=groom>grooming
```

The ID attribute assigns an identifier to the resulting index term. If used with I2 or I3, this identifier also refers to this index term's higher-level terms. The use of ID above defines "groom" as an identifier that can thereafter be used as a REFID or SEEID attribute to refer to the three-level index term "cats, proper care, grooming". If used as the REFID attribute of a subsequent I2 or I3 tags, the index term of that tag will be added under the higher-level index term(s) named by identifier "groom", instead of under the most-recently-specified term.

28.5 Special Page References

The PG and SEE attributes apply to the manner in which information is added to the list of page references for an index term. If MAJOR is specified for the PG attribute, the current page number is added at the front of the list of page references. With the SEE attribute, the value specified is added to the list of page references as a "see item", instead of the current page number.

```
(text on page 82)
<I3 PG=MAJOR>grooming
<I3 SEE=diet>grooming
```

The SEEID attribute enables you to specify an identifier whose index term(s) will be generated as the "see" reference, instead of specifying the "see" reference as the text of the SEE attribute.

28.6 Repeating an Index Term

The IREF tag specifies that another occurrence of an index term that already exists is to be added to the index. It is used in the form

```
<IREF REFID=identifier>
```

The current page number will be added to the list of page references for the index term(s) associated with the specified identifier name. For example,

```
(text on page 94)
<IREF REFID=groom>
```

would cause the number 94 to be added to the list of page references for "cats, proper care, grooming" of the identifier "groom".

28.7 The Index-Header Tags

The IH1, IH2, and IH3 tags function in a manner identical to the I1, I2, and I3 tags with the exception that no page reference is added to the index term that they add to the index.

28.8 Printing the Resulting Index in the Back Matter

The INDEX tag is used in the back matter to cause the printing of the index terms and page references. For example, all of the examples presented above, plus the INDEX tag, would appear in the following sequence:

```
(text on page 10)
<I1>cats
(text on page 34)
```

```
<I2>proper care
(text on page 42)
<I3>life span
(text on page 57)
<I3 ID=groom>grooming
(text on page 64)
<I3>diet
(text on page 82)
<I3 PG=MAJOR>grooming
<I3 SEE=diet>grooming
(text on page 94)
<IREF REFID=groom>
  ... rest of body ...
</BODY
<BACKM
<INDEX>[Heading Text]
```

If no tagtext is specified, the heading will be "Index". The index terms and page references will appear below the heading, as:

```
cats ... 10
  proper care ... 34
    diet ... 64
    grooming ... 82, 57, see "diet", 94
    life span ... 42
```

28.9 Multiple Indexes

Each of the tags described above accepts an IX attribute, whose value must be an integer between 1 and 9. This allows you to specify index items for up to 9 separate indexes. If the IX attribute is not specified, the tag applies to index number 1. For example, in a document that requires one index for authors and another index for subjects, the index tags might be used as shown below:

```
<I1 IX=1>Ima N. Author
<I1 IX=2>index term for subject index
     ...
<BACKM>
<INDEX IX=1>Index of Authors
<INDEX IX=2>Subject Index
```

Part 7

Special Symbols

29. Greek characters and Common mathematical symbols

For ease of typing greek characters and the most common mathematical symbols "short references" are used. The choice has been made to be a @ followed by one character. For example @p will result in a π. The characters available are listed in the table below:

a α	b β	g γ	d δ	e ε	z ζ	h η	q θ	i ι	k κ
l λ	m μ	n ν	x ξ	o ο	p π	r ρ	s σ	t τ	u υ
f ϕ	c χ	y ψ	w ω	G Γ	D Δ	Q Θ	L Λ	X Ξ	P Π
S Σ	U Υ	F Φ	Y Ψ	W Ω	I ∫	A →	M ±	R √	N ∇
B ∞	* ×	: ·	| ≠	≡	E ∂	T †	O ∮	↑	@ ↓
[←	{ ⊂	} ⊃	¬ ∩	∧ ∪	K ℓ	j ∝	Z ∓	V ∧	v ∨
< ≤	> ≥	, ⟨	. ⟩	? ∥	/ ⊥	_ $\bar{\mathrm{p}}$	+ $^{+}$	– $^{-}$	0 ^{0}
1 ^{1}	2 ^{2}	3 ^{3}	4 ^{4}	5 ^{5}	6 ^{6}	7 ^{7}	8 ^{8}	9 ^{9}	= $_{0}$
! $_{1}$	" $_{2}$	# $_{3}$	$ $_{4}$	% $_{5}$	& $_{6}$	′ $_{7}$	($_{8}$	) $_{9}$	

30. Other symbols

For the less commonly used symbols, and for completeness also the greek alphabet, the entities in Annex E of ISO 8879 have been implemented. Entity references is a name delimited by a & and a ; The ; can be omitted if the entity reference is followed by a space. For example &alpha;&beta would produce αβ when printed. A list of the available symbols is given below:

alpha α	beta β	gamma γ	Gamma Γ
delta δ	Delta Δ	epsi ε	zeta ζ
eta η	theta θ	Theta Θ	iota ι
kappa κ	lambda λ	Lambda Λ	mu μ
nu ν	xi ξ	Xi Ξ	pi π
Pi Π	rho ρ	sigma σ	Sigma Σ
tau τ	upsi υ	Upsi Υ	phi ϕ
Phi Φ	chi χ	psi ψ	Psi Ψ
omega ω	Omega Ω	sup0 ^{0}	sup1 ^{1}
sup2 ^{2}	sup3 ^{3}	sup4 ^{4}	sup5 ^{5}
sup6 ^{6}	sup7 ^{7}	sup8 ^{8}	sup9 ^{9}
sub0 $_{0}$	sub1 $_{1}$	sub2 $_{2}$	sub3 $_{3}$
sub4 $_{4}$	sub5 $_{5}$	sub6 $_{6}$	sub7 $_{7}$
sub8 $_{8}$	sub9 $_{9}$		

int ∫	oint ∮	rar →	lar ←
uar ↑	dar ↓	plusmn ±	mnplus ∓
sub ⊂	sup ⊃	cap ∩	cup ∪
radic √	nabla ∇	infin ∞	times ×
dot ·	ne ≠	equiv ≡	part ∂
dagger †	ell ℓ	prop ∝	and ∧
or ∨	le ≤	ge ≥	dbra ⟨
dket ⟩	par ‖	perp ⊥	supplus ⁺
supmn ⁻	suplpar ⁽	suprpar ⁾	subplus ₊
submn ₋	sublpar ₍	subrpar ₎	pbar p̄
rArr ⇒	lArr ⇐	uArr ⇑	dArr ⇓
hArr ⇔	harr ↔	ap ≈	sime ≃
ape ≅	sim ∼	wedgeq ≙	deg °
oplus ⊕	ominus ⊖	otimes ⊗	odot ⊙
osol ⊘	phmmat ℳ	lagran ℒ	hamilt ℋ
bernou ℬ	order ℴ	real ℜ	image ℑ
inodot ı	jnodot ȷ	aleph ℵ	lambar ƛ
planck ℏ	nubar ν̄	lbar ƚ	forall ∀
exists ∃	tprime ‴	utrif ▲	permil ‰
vepsi ∈	vphi φ	vtheta ϑ	vsigma ς
Lt ≪	Gt ≫	compfn °	squ □
divide ÷	subE ⊆	supE ⊇	isin ∈
notin ∉	obar ‾	oarrow →	ohat ˆ
otilda ˜	odarrow ↔		

Part 8

Graphic Sources

31. GKS Metafile

Not implemented yet – awaiting some work from the graphics section since November '85.

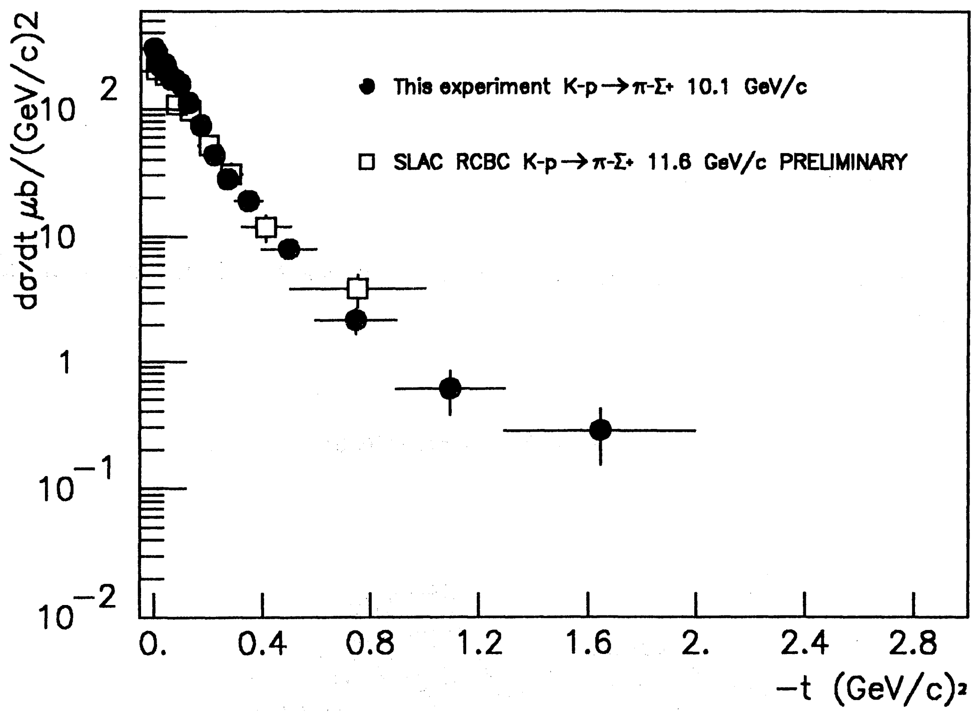


Figure 2: Sample graph from Hplot

32. Apollo bit-maps

The creation of the images on the Apollo and the method of transfering the resulting graphics file to VM/CMS is described in a separate guide [6].

The Apollo file, once transfered to VM/CMS, is then converted into the appropriate format for inserting in a document by using the APOLA67 command. This command has the format:

```
APOLA67 fn [ ( OUTMODE fm scale ]
  where
     fn is the file-name of the APOLLO file
     OUTMODE fm can be added to specify a file-mode for the result
     scale can take the value 1X or 2X (default) to indicate the scale
```

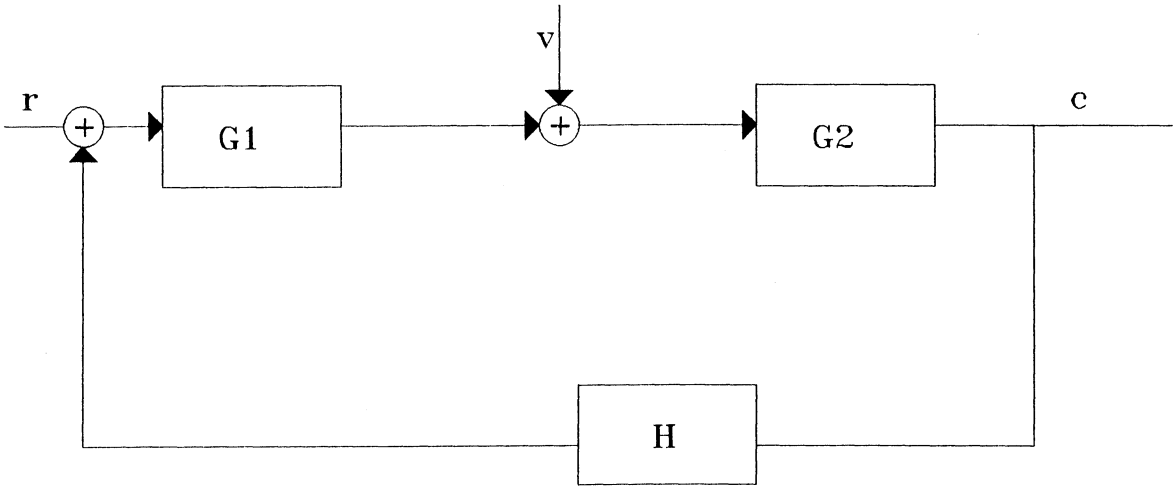


Figure 3: Sample graph from Apollo

Each pixel in the bit-map from the Apollo is printed either as a 2 by 2 "dot" or as one dot depending on the scale selected.

33. MacPaint Drawings

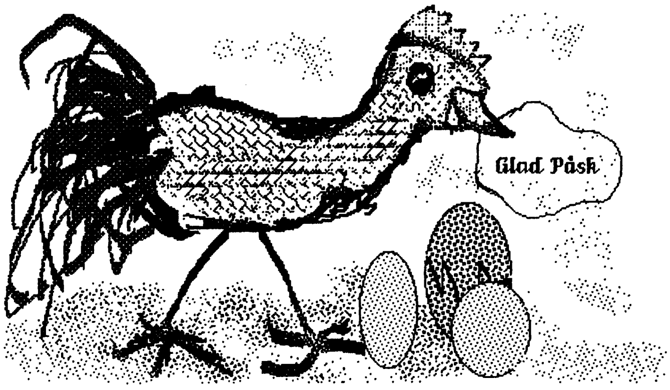


Figure 4: Sample graph from MacPaint

The MacPaint file is first transferred to the VM/CMS system using KERMIT to transfer a binary file. The file-name is recommended to be 7 characters long and the file-type has to be MACPAINT. The use of MacPaint and the transfer is described in more detail in a separate guide [7].

For each file the MACPA67 utility is invoked to convert the MacPaint file into the appropriate format. This command has the format:

```
MACPA67 fn [( OUTMODE fm scale ]
  where
     fn is the file-name of the MACPAINT file
     OUTMODE fm can be added to specify a file-mode for the result
     scale can take the value 2X or 3X (default) to indicate the scale
```

Each pixel in the bit-map from MacPaint is printed either as a 3 by 3 "dot" or as a 2 by 2 "dot" depending on the scale selected. At 3X magnification some stair-case effects are visible, whereas at 2X magnification they are negligible.

34. Scanned Images

A scanner capable to digitize both line drawings and half-tone pictures (photographs) is on order, but has not yet arrived. This chapter will be updated when the scanner has arrived.

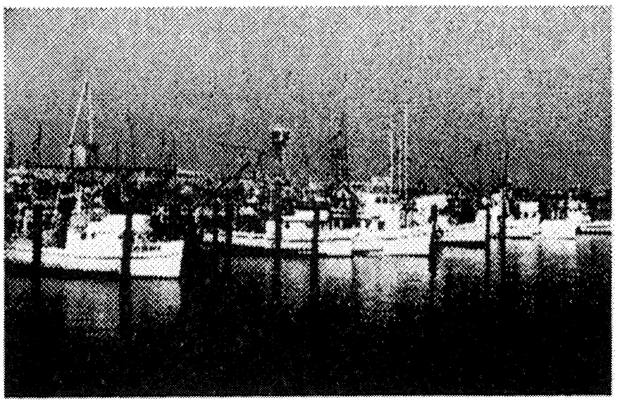


Figure 5: Sample scanned image

Part 9

Layout Types

The ″standard″ types are standard merely in the sense that they all have the same tags and you can thus format the same input file using different styles without any changes to the input file.

″Semi-Standard″ types have the same tags as the standard types, but some special, required, attributes on one or more tags.

″Non-Standard″ types have some special tags, normally in the front-matter. Such types include the letter type.

35. ″Standard″ Types

SGMLman intended for ″manuals″. Level zero headings are called ″Parts″ and level one headings are called ″Chapters″.

SGMLguid intended for ″guides″.

36. ″Semi-Standard″ Types

SGMLwrup intended for ″writeups″, all tags as in the ″standard″ layouts, but does not allow an ″author list″.

SGMLprep intended for ″preprints″, in EP or TH division styles. It has the same tags as the ″standard″ layouts, but two additional attributes on the <TITLEP> tag.

37. ″Non-Standard″ Types

SGMLclet a layout style for CERN official letters. This is only supported for the APA6670 printer, since it requires graphic capabilities for overlaying the letterhead.

SGMLfoil a layout style for making transparencies.

SGMLmemo

a layout style for memos.

SGMLmins a layout style for minutes of meetings.

37.1 SGMLclet Tags

```
<GDOC[ SEC="classification"]>
  <TO>
    Name of Recipient (or Company Name)
    First Line of Recipient's Address
    (etc)
    Last Line of Recipient's Address
  <PHONE>telephone number of sender
  <REF YOUR="reference" OUR="reference">
  <DATE>[date]
  <ATTN>Attention Name
  <SUBJECT>Heading Text
  <OPEN>Opening Salutation
    basic document components
  <CLOSE[ DEPTH=n]>Closing salutation
    Author's Name
    Author's title and/or
    Author's address, etc
  </CLOSE>[ABC/xyz]
  <DISTRIB>
    <DIST>label
      names, one per line
  </DISTRIB>
</GDOC>
```

37.2 SGMLmemo Tags

```
<GDOC[ SEC="classification"]>
  <DATE>[date]
  <DISTRIB[ COLUMNS=n]>
    <DIST>[label]
      list of names, one per line
  </DISTRIB>
  <SUBJECT[ MODE=BLOCK]>[Heading Text]
    (additional subject text, if BLOCK mode)
  </SUBJECT>
    basic document components
  <CLOSE
    name of author
    etc
  </CLOSE>[abc/xyz]
</GDOC>
```

37.3 SGMLmins Tags

```
<GDOC[ SEC="classification"]>
  <TITLE>Name of Corporation
  (repeat as necessary)
  <TITLE>Name of Group holding meeting
  <DATE[ TYPE=AGENDA]>[date of meeting]
  <DISTRIB[ COLUMNS=n]>
    <DIST>[label]
```

```
      names, one per line
  </DISTRIB>
  <BODY>
    <OL
      <LIH>Heading Text (discussion item)
        Details of discussion, etc.
    </OL>
    The meeting adjourned at (time).
  <CLOSE
    <TYPIST>ABC/xyz
    <RECORDER>Name of Secretary
      Secretary's Title
      (etc)
  </CLOSE>
</GDOC>
```

Part 10

Language support

There is support for generating the ″fixed″ text – like ″page″, ″chapter″ – in the following languages: French, German, Swedish, and Dutch. The language is selected by the use of the < SET > tag before the < GDOC > tag. For example:

```
<SET ITEM=LANG VALUE=FRENCH>
```

For typing the national use characters the entities in Annex E of ISO 8879 have been implemented. Entity references is a name delimited by a & and a ; The ; can be omitted if the entity reference is followed by a space. For example &eacute;l&egrave;ve and hall&aring are both valid uses of entity references and result in élève and hallå

A list of the available symbols is given below:

aacute á
Aacute Á
agrave à
Agrave À
acirc â
Acirc Â
auml ä
Auml Ä
atilde ã
Atilde Ã
aring å
Aring Å
ae æ
AE Æ
ccedil ç
Ccedil Ç
eacute é
Eacute É
egrave è
Egrave È

ecirc ê
Ecirc Ê
euml ë
Euml Ë
iacute í
Iacute Í
igrave ì
Igrave Ì
icirc î
Icirc Î
iuml ï
Iuml Ï
ntilde ñ
Ntilde Ñ
oacute ó
Oacute Ó
ograve ò
Ograve Ò
ocirc ô
Ocirc Ô

ouml ö
Ouml Ö
otilde õ
Otilde Õ
oe œ
OE Œ
oslash ø
Oslash Ø
ssharp ß
uacute ú
Uacute Ú
ugrave ù
Ugrave Ù
ucirc û
Ucirc Û
uuml ü
Uuml Ü
yuml ÿ
Yuml Ÿ

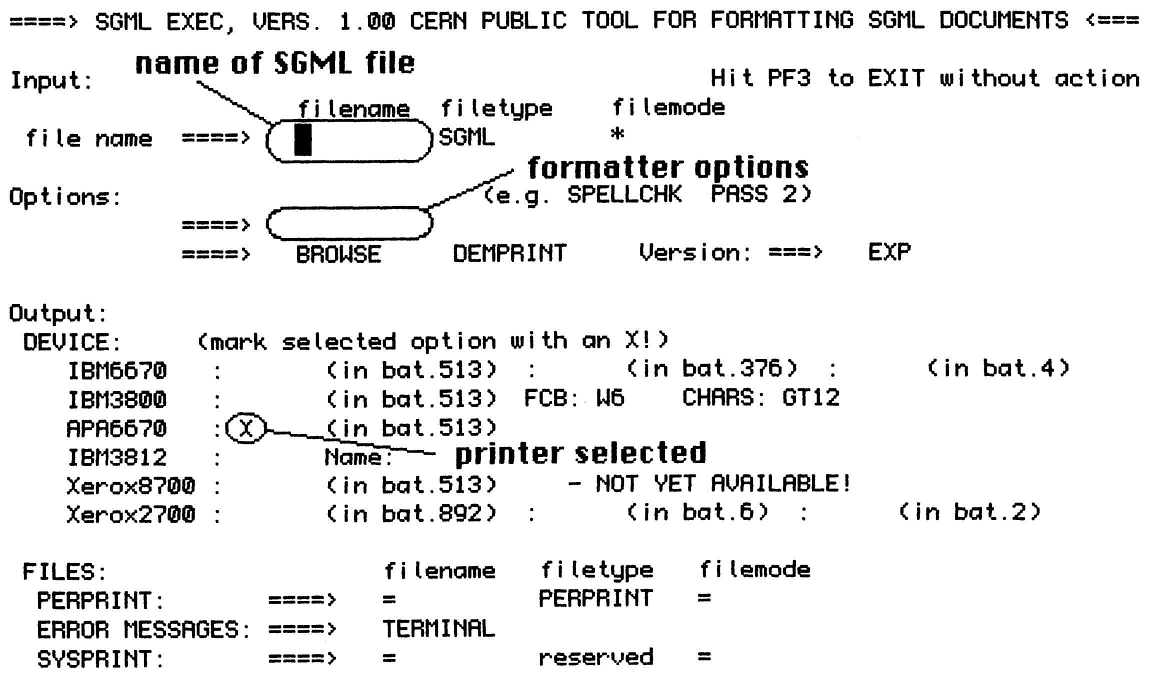
Part 11

Usage in VM/CMS

The input file is prepared using XEDIT. The file-type is required to be SGML. For example:

```
XEDIT myfile SGML
```

A menu driven exec has been proviced to make the formatting as easy as possible. This is invoked by typing the SGML command. This will display the following menu:



```
====> SGML EXEC, VERS. 1.00 CERN PUBLIC TOOL FOR FORMATTING SGML DOCUMENTS <===

Input:                                                 Hit PF3 to EXIT without action
                     filename   filetype    filemode
 file name   ====> (█         )SGML         *

Options:                               (e.g. SPELLCHK  PASS 2)
             ====> (          )
             ====>   BROWSE     DEMPRINT     Version: ===>    EXP

Output:
 DEVICE:      (mark selected option with an X!)
    IBM6670   :          (in bat.513)  :        (in bat.376)  :        (in bat.4)
    IBM3800   :          (in bat.513)  FCB: W6     CHARS: GT12
    APA6670   :(X)       (in bat.513)
    IBM3812   :          Name:
    Xerox8700 :          (in bat.513)     - NOT YET AVAILABLE!
    Xerox2700 :          (in bat.892)  :       (in bat.6)  :       (in bat.2)

 FILES:                     filename   filetype    filemode
  PERPRINT:        ====>    =          PERPRINT    =
  ERROR MESSAGES:  ====>    TERMINAL
  SYSPRINT:        ====>    =          reserved    =
```

name of SGML file

formatter options

printer selected

Figure 6: SGML panel in VM/CMS

where you should enter the name of your input SGML file. Optionally parameters for SCRIPT can be added – e.g. PASS 2 FROM 10 TO 15 (to use 2 pass mode and only print pages 10 to 15) – and a different output device from the default selected.

One should note that the defaults can be changed permanently by using the DEFAULTS SET SGML command.

Part 12

Usage in WYLBUR/MVS

To run a job that is to be printed on a 6670 laser printer you use the following JCL:

```
// JOB ,TIME=(0,15),MSGCLASS=Z
// EXEC SGML,
// SPRM='MEMO,I6670,PERICOM'
//S.SYSPRINT DD SYSOUT=I,DEST=<destination>
//S.SYSIN DD *
Text of your document (+ SGML tags)
```

Valid destinations are: DEST = I6670 (local, building 513), DEST = SB6670 (building 4) and DEST = IR (building 376).

For output on the 3800 laserprinter, the SPRM parameter of the SGML procedure should be omitted and the SYSOUT class changed to W:

```
// JOB ,TIME=(0,15),MSGCLASS=Z
// EXEC SGML
//S.SYSPRINT DD SYSOUT=W
//S.SYSIN DD *
Text of your document (+ SGML tags)
```

For output on the APA6670 laserprinter use the following JCL:

```
// JOB ,TIME=(0,15),MSGCLASS=Z
// EXEC SGMLA67
//S.SYSIN DD *
Text of your document (+ SGML tags)
```

For output on the 2700 laserprinter use the following JCL:

```
// JOB ,TIME=(0,15),MSGCLASS=Z
// EXEC SGMLX27,DEST=<destination>
//S.SYSIN DD *
Text of your document (+ SGML tags)
```

Valid destinations are: DEST = NA2700 (building 892), DEST = PS (building 6), and DEST = OC2700 (building 2).

Appendix A

Sample Standard SGML Document

The following input shows all the tags that can be used for the "standard" layouts together with the resulting formatted document for some of the layout styles available.

```
<!DOCTYPE SGMLxxxx>
<GDOC SEC='SEC value, if any'>
<FRONTM>
<I1 id=front>front material
<TITLEP>
<I2 refid=front>title page
<TITLE stitle='SGML Sample'>The CERN SGML
<TITLE>Sample Document
<DOCNUM>DOCNUM text, if any
<AUTHLIST>
<AUTHOR REFID="I1 I3">First Author's Name
<INST ID=I1>CERN, ....., Switzerland.
</INST>
<INST ID=I2>Somewhere
else.
</INST>
<AUTHOR REFID=I2>Last Author's Name
<ANOTE ID=I3>Paid in part by......
</ANOTE>
</AUTHLIST>
<DATE>date
<COPYRIGHT>
</TITLEP>
<ABSTRACT>
<I2 refid=front>abstract
This is the body of the abstract.  The abstract will begin on the same
page as the "title page" material with some layouts, and at the top of
a new page with others.
</ABSTRACT>
<PREFACE>First Preface Heading
<I2 refid=front>preface
This is the body of this preface.  The first preface will begin on the
same page as the "abstract" material with some layouts, and at the top
of a new page with others.
<FN>The layouts in the "thesis" and "manual" categories cause the
abstract, preface, and all other level-one headings
to begin at the top of a new page.</FN>
<H2>Level-Two Heading in the Front Material
The following is a reference to an "endnote".
<EN>Endnotes are very much like footnotes, except that YOU control
their placement.</EN>
<PREFACE>Last Preface Heading
<I2 refid=front>preface
Note that you can title a "preface" whatever you want, and that if you
don't supply heading text, it will be titled "Preface".
<P>
```

```
<FIGREF refid=frontm>
below demonstrates that a figure can occur in the front material.
<FIG id=frontm>
body of figure
<FIGCAP>This is the Figure Caption
<FIGDESC>And this is the "figure description".
The "figure description" is additional text amplifying the
"figure caption", which should be as terse as you can make it.
</FIG>
</PREFACE>
<TOC>
<I2 refid=front>tables of contents
<FIGLIST>
<TABLIST>
</FRONTM>
<BODY>
<I1 id=body>body
<H0>First Level-Zero Heading
<I2 refid=body>heading levels 0 to 6
This is the first sentence after a heading.
This is the last sentence before the start of a new paragraph.
<P>This is the first sentence of a new paragraph.
This is the last sentence before a heading.
<H1 ID=fh1>First Level-One Heading
This is the first sentence after a heading.
This is the last sentence before a heading.
<H2>First Level-Two Heading
This is the first sentence after a heading.
This is the last sentence before a heading.
<H3>First Level-Three Heading
This is the first sentence after a heading.
This is the last sentence before a heading.
<H4>First Level-Four Heading
This is the first sentence after a heading.
This is the last sentence before a heading.
<H5>First Level-Five Heading
This is the first sentence after a heading.
This is the last sentence before a heading.
<H6>First Level-Six Heading
This is the first sentence after a heading.
This is the last sentence before a heading.
<P>
This forces text to the next page after a previous level-one heading.
 ...
This forces text to the next page after a previous level-one heading.
<H1>Another Level-One Heading
<H2>Long Quotations
<I2 refid=body>long quotation
This is the first sentence after a heading.
This is the last sentence before a long quotation.
<LQ>This is the first sentence of a long quotation.
This is the last sentence of a long quotation.</LQ>
This is the first sentence after a long quotation.
<H2>Footnotes and Endnotes
<I2 refid=body>footnotes
```

```
<I2 refid=body>endnotes
This is the first sentence after a heading.
This is the text immediately before the footnote,
<FN>This is the first sentence of a footnote.
This is the last sentence of a footnote.</FN>
and this is the text immediately after the footnote.
This is the text immediately before the endnote,
<EN>This is the first sentence of a endnote.
This is the last sentence of a endnote.</EN>
and this is the text immediately after the endnote.
This is the last sentence before a heading.
<H0>Second Level-Zero Heading
<H1>Figures and Tables
<I2 refid=body>figures
<FIGREF refid=body>
below shows the layout of a figure in the main-body material.
<FIG id=body>
body of figure
<FIGCAP>Figure Caption
<FIGDESC>The "figure description" is additional text
amplifying the "figure caption".
</FIG>
</BODY>
<APPENDIX>
<I1>appendix
<H1>First Appendix, Showing Figure and Table Examples
<FIGREF refid=example>
below shows the layout of a figure.
<FIG id=example>
body of figure
<FIGCAP>Figure Caption
<FIGDESC>The "figure description" is additional text
amplifying the "figure caption".
</FIG>
<P>An example of the layout of a table
is shown in <TABREF refid=example>.
<TAB id=example>
<I2 refid=body>tables
<TABCAP>Table Caption
<TABDESC>The "table description" is additional text
amplifying the "table caption".
</TABDESC>
body of table
</TAB>
<H1>Second Appendix Heading
This is the first sentence after a heading.
This is the last sentence before a heading.
<H2>Level-Two Heading in an Appendix
This is the first sentence after a heading.
This is the last sentence before a heading.
<P>A sample reference <HDREF REFID=fh1> to a header.
</APPENDIX>
<BACKM>
<I1 id=backm>back material
<H1>Level-One Heading in Back Material
```

```
This is the first sentence after a heading.
This is the last sentence before a heading.
<FN>With most layouts, the back material will start at the top of a new
page, even if the level-one headings thereafter do not.</FN>
Endnotes can also occur in the back material.
<EN>This is the text of an endnote defined in the back material.</EN>
<P>
<FIGREF refid=backm>
below shows the layout of a figure in the back material.
<FIG id=backm>
body of figure
<FIGCAP>Figure Caption
<FIGDESC>The "figure description" is additional text
amplifying the "figure caption".
</FIG>
<H1>Positioning the Endnotes
<I2 refid=body see='back material'>endnotes
<I2 refid=backm>endnotes
The endnotes defined earlier in this document are printed below.
<ENDNOTES>
<INDEX>Sample Index
<H1>Bibliography
<BL>
<BIB>
<CIT>Waterloo SCRIPT GML User's Guide</CIT>,
Department of Computing Services, University of Waterloo.
Waterloo, 1986.
</BL>
</BACKM>
</GDOC>
```

Appendix B

Sample CERN offical letter

```
<!DOCTYPE SGMLclet>
<GDOC>
<TO>
The Addressee
Street
City
Country
<PHONE>83 33 63
<REF YOUR=abc OUR=xyz>
<DATE>
<SUBJECT>This is the subject
<OPEN>Dear Sir,
<P>
Many thanks for your letter of xxxx xxnd 1985.
This is a line of test for the first paragraph of this letter.
<P>A second paragraph for this test letter.
And a few more words to fill up this second paragraph.
<CLOSE>Yours sincerely,
Name of Sender
</CLOSE>
</GDOC>
```

Appendix C

Sample memo

```
<!DOCTYPE SGMLmemo>
<GDOC SEC="Sec value if any">
<DATE>
<DISTRIB>
<DIST>To
Name of Recipient 1
Name of Recipient 2
</DISTRIB>
<SUBJECT>Engraving of Terminal Keys for VM/CMS
</SUBJECT>
<P>
With the new "cheap" pool terminal we have been offered to have made
glue-on labels printed with any text we like. These labels
apparently stick on very nicely and also leave the original engraving
of the keys visible.
<CLOSE>
A.Berglund
</CLOSE>
</GDOC>
```

Appendix D

Sample foil

```
<!DOCTYPE SGMLFOIL>
<GDOC>
<TITLEP>
<TITLE>Languages for Describing Mathematics
<TITLE>- an End-User's Point of View
<AUTHOR>Anders Berglund
<ALINE>CERN, Geneva, Switzerland
</TITLEP>
<H0>CERN European Laboratory for Particle Physics
<UL COMPACT=1>
<LI>The Convention establishing <HP2>CERN</HP2>
- (Conseil Européen pour la Recherche
Nucléaire) - entered into force in 1954.
</UL>
<H1>Text Processing in the Computer Centre
<HP1>The primary rôle</HP1> of the Computer Centre is
<HP1>data processing of experimental data,</HP1> but nonetheless...
<UL>
<LI><HP1>Processing represents a major application,</HP1>
(10% of number of MVS jobs, more for VM/CMS -
many CERN staff are their own secretary).
<LI><HP2>Text Processing</HP2>
is based on
<HP2>Waterloo SCRIPT</HP2>
<LI>meaning
<HP2>MARKED-UP TEXT</HP2>
as opposed to
<HP2>WYSIWYG,</HP2>
<LI>and on <HP2>GENERIC</HP2> rather than <HP2>SPECIFIC</HP2>
markup.
<LI><HP2>Output Devices</HP2>
for text processing include:
<UL COMPACT=2>
<LI>"research" (APA6670) laser printer (1)
<LI>IBM 3812 printers (currently 2, 3 coming)
<LI>Xerox 2700 laser printers (3)
<LI>Xerox 8700 laser printer (1)
<LI>Compugraphic MCS8400 Photo-typesetter (1)
<LI>IBM 6670s (2)
<LI>IBM 3800 model 1 (1)
</UL>
<LI>We are in the process of a conversion from "typewriter" fonts
to typographic fonts, and from plain text only to full exploitation
of
<HP2>all-points-addressability.</HP2>
<LI>mostly non-graphic terminals in use (almost 2000),
some Apollo workstations, PCs, McIntoshes
</UL>
</GDOC>
```

Appendix E

Sample minutes and agendas

```
<!DOCTYPE sgmlmins>
<GDOC SEC="SEC value, if any">
<TITLEP>
<TITLE>Minutes of the XXmeeting
<DATE>date when it was
<DISTRIB>
<DIST>Present
S.One
S.B.Else
A.Third
B.Fourth
A.Participant
</DISTRIB>
</TITLEP>
<BODY>
<OL>
<LI>The bla bla was discussed.
<LI>Some other topic.
</OL>
<CLOSE>
<TYPIST>AB/xyz
<RECORDER>AB
</CLOSE>
</GDOC>
```

Appendix F

The <SET> Tag

The <SET> tag sets a global parameter for the document. You can specify as many <SET> tags as you wish, *before* the <GDOC> tag. Any <SET> that follows the <GDOC> will be ignored.

In the following descriptions of the <SET> tag, things in uppercase letters must be specified exactly as shown; things in lowercase must be replaced by a value of your chosing.

F.1 Number of Heading Levels

Some publishing bodies demand that different heading styles be used for the same heading levels, depending on the *number of levels* of headings contained in the document;

```
    <SET ITEM=HEADLEVEL VALUE=number>
```

allows you to specify this number.

F.2 Page Settings

The following settings alter the size of the output paper for which a layout was designed, or alter the positioning of the output on that paper. The tag is specified as

```
    <SET ITEM=item VALUE=number>
    or
    <SET ITEM=item VALUE="cm">
```

″cm″ is a measurement such as ″16cm″ (sixteen centimetres), or ″79″ (79 character units) and the items are those below.

PAGE the number of lines that can fit on the page (number)

HOR the horizontal width of the paper (cm)

PAGE and HOR must only be used if you are producing your output on paper that is different in size than the paper for which the layout was designed; the items below alter the placement of the output on the paper.

LEFT the width of the left margin (cm)

LINE the width of the output line (cm)

COLS the number of columns of text per page (number)

F.3 Line-Spacing Modes

Some of the layouts produce single-spaced output and others produce double-spaced output. You can force a particular spacing by specifying:

```
<SET ITEM=LS VALUE=number>
<SET ITEM=SS VALUE=number>
```

The LS value defines the number of blank lines that will be produced between output lines in the normal line-spacing mode, and the SS value defines the number of blank lines that will be produced between output lines in single-spacing mode. A value of 0 implies single-spacing mode (no extra blank lines between output lines). For example, if you were using a layout that double-spaced text and single-spaced components such as <XMP> blocks, specifying

```
<SET ITEM=LS VALUE=2>
<SET ITEM=SS VALUE=1>
```

would result in triple-spacing for text that would otherwise have been double-spaced, and double-spacing for text that would otherwise have been single-spaced.

F.4 Font Selections for Special Output Devices

If the document source file is processed with any of the APA6670, XEROX 2700, or IBM 6670 options of the formatter, default font selections will be made for you. Currently only the APA6670 and IBM 3812 printers offer the full flexibility in proportionally spaced characters of different sizes.

SGML automatically defines five font-identifier names and their associated type styles and point sizes.

TITLE the font identifier for the family of proportionally-spaced fonts that will be used for all TITLE tagtext

HEAD the font identifier for the family of proportionally-spaced fonts that will be used for all heading text (H0-H6, ABSTRACT, and PREFACE tagtext)

MONO the font identifier for the family of mono-width fonts that will be used for the text in the body of figures, tables, and examples

FOOT the font identifier for the family of proportionally-spaced fonts that will be used for footnotes

TEXT the font identifier for the family of proportionally-spaced fonts that will be used for all other text

The default assignment of the MONO font for the example, figure, and/or table blocks can be changed to use the TEXT font -- include the process-specific-code block shown below, with those <SET> tags for which you want the TEXT font used instead of the MONO font:

```
<PSC PROC="APA6670">
<SET TAG=FIG ITEM=FONT VALUE=TEXT>
<SET TAG=TAB ITEM=FONT VALUE=TEXT>
<SET TAG=XMP ITEM=FONT VALUE=TEXT>
</PSC>
<GDOC>
     ...
```

For the IBM 6670 printer one font from a set of four can be selected by

```
<SET ITEM=FONT VALUE=font_number>
```

where font_number can be

301 a proportionally spaced font with normal and bold letters, but without accented characters

302 same as 301, but with accented characters, but without the bold ones

351 a mono spaced font with normal and bold letters, but without accented characters

352 same as 351, but with accented characters, but without the bold ones

For the XEROX 2700 printer only one font is available. It corresponds to font 351 plus 352 of the IBM 6670 printer.

F.5 Relative Page Numbering

Relative numbering of pages with respect to level-one headings in the body and appendix material can be requested by specifying

```
<SET ITEM=NUMB VALUE=RELATIVE>
```

With layouts for which this is not possible, an error message will result. With layouts for which relative page numbering is the default, absolute page numbering can be achieved by

```
<SET ITEM=NUMB VALUE=ABSOLUTE>
```

F.6 Bibliography-List Styles

The annotation style for items in the bibliography list and for references to them can be set by specifying

```
<SET TAG=BIB ITEM=STYLE VALUE="style">
```

The possible values are:

NUMBER The items in the bibliography list will be numbered. This is the default for all of the general-purpose SGML layouts, and for a few others.

LABEL Some manuals of style suggest labelling the items instead of numbering them, with a label of the form CCCCCyya, where CCCCC is the first five characters of the author's last name, "yy" is the last two digits of the year of publication of the work (optional, used if the bibliography cites more than one work by the same author), and "a" is an alphabetized sequence (optional, used if the bibliography cites more than one work by the same author in the same year). If you wish to use this style, then you must also use the CCCCCyya label as the value of the ID attribute of each < BIB > tag in the bibliography list.

NONE It is possible to have the items in the bibliography listed without a number or label. If you do this, you do not need to specify an ID attribute with the < BIB > tag, and it is impossible to make reference to items in the bibliography list using the < BIBREF > tag.

F.7 Figure and Table Names

For example, if you want to have "Example"s instead of "Table"s and "Illustration"s instead of "Figure"s, then in your SCRIPT input file, before the < GDOC > tag, specify

```
<SET TAG=FIG ITEM=NAME VALUE="Illustration">
<SET TAG=TAB ITEM=NAME VALUE="Example">
```

Note that you must still use the figure and table tags to define these "example"s and "illustration"s.

F.8 Figure and Table Numbering

With layouts that number the headings produced by < H1 > in the < BODY > component of a document, you can set the manner in which figures and tables are numbered within those level-one headings.

```
<SET TAG=FIG ITEM=NUMB VALUE="style">
<SET TAG=TAB ITEM=NUMB VALUE="style">
```

ABSOLUTE
The default for all layouts.

RELATIVE
The figures or tables will be numbered relative to the level-one heading within which they occur (for example, 1.1, 1.2, etc within the first level-one heading in the < BODY > material, and A.1, A.2, etc within the first level-one heading within the < APPENDIX > material).

F.9 Figure and Table Printing

Each of the layouts controls the level to which headings cause any generated-but-not-yet-printed figures and tables to be printed. With some layouts, only < H0 > causes this, so that a figure defined within one < H1 > might not print until after the start of the next < H1 > component. With other layouts, all heading levels cause previously-generated figures and tables to print. You can set the level to which headings will cause figures and tables to print. For example, to set this for < H0 >, < H1 >, and < H2 > only,

```
<SET ITEM=FTLEV VALUE=2>
```

F.10 Footnote Numbering

```
<SET TAG=FN ITEM=NUMB VALUE="style">
```

The ″style″ values are:

ABSOLUTE

the default for all layouts; footnote numbering will be sequential throughout the document

RELATIVE

footnote numbering will start at 1 within each level-one heading.

F.11 Heading-Level Names

With several of the layouts, each level-zero heading is referred to as a ″Part″ and each level-one heading is referred to as a ″Chapter″ in the output document. If you are using one of the layouts that does this, and the names ″Part″ and ″Chapter″ are not appropriate, you can specify names of your choosing for them. For example, you can cause each level-one heading to be referred to as a ″Section″ by specifying

```
<SET TAG=H1 ITEM=NAME VALUE="Section">
```

F.12 Heading Levels in the Table of Contents

For those layouts that produce tables of contents, the heading text of heading levels zero through four will be included in the table of contents. You cannot extend this to heading levels five and six, but you can confine it to less than four. For example, if you wanted to exclude heading levels three and four from the table of contents, then specify

```
<SET TAG=TOC ITEM=LEVEL VALUE=2>
```

Any attempt to set a VALUE less than 1 will be treated as though you had specified ″1″; as a result, heading levels two and lower will be omitted from the table of contents.

F.13 Notes

You can specify the word to be generated instead of ″Note″, by

```
<SET TAG=NOTE ITEM=NAME VALUE="name for notes">
```

F.14 Ordered-List Annotation Styles

The items in an ordered list can be annotated using arabic numerals, uppercase or lowercase alphabetics, or uppercase or lowercase roman numerals. If the default annotation styles and delimiter characters are not appropriate for your document, you can establish your own by specifying (for example)

```
<SET TAG=OL ITEM=STYLE VALUE="n. a. i. n) a) i)">
```

The definition of VALUE above shows the default annotation style and delimiter characters for the six levels of ordered lists. The ″style″ character can be any of ″n″ (for ″arabic numerals″), ″a″ or ″A″ (for lowercase or uppercase ″alphabetics″), and ″i″ or ″I″ (for lowercase or uppercase ″roman numerals″). The ″delimiter″ can be any single character after the ″style″ character, or you can specify both a left and a right delimiter character on either side of the ″style″ character, such as ″(a)″.

F.15 Unordered-List Annotation Styles

The items in an unordered list can be annotated using different symbols. The default symbols can be changed by specifying (for example)

```
<SET TAG=UL ITEM=STYLE VALUE="* - -- + ++ -">
```

The definition of VALUE above sets the symbols for six levels of ordered lists to be * for the first level, – for the second level and so on.

F.16 Indentation for Basic Document Components

The number of character spaces that certain basic document components will be indented throughout the document can be SET by

```
<SET TAG=tagname ITEM=IN VALUE=number>
```

The indentation can be set for any tag, including <P>, <LQ>, <XMP>, and all of the <xL> ″start-of-list″ tags. For example, the indentation for definition-list terms can be set by

```
<SET TAG=DL ITEM=IN VALUE=number>
```

If the tagname you specify does not have such a setting, an error message will be produced.

F.17 Vertical Spacing for Document Components

The number of blank lines before (and in appropriate cases, after) a document component can be SET by

```
<SET TAG=tagname ITEM=SK VALUE=number>
```

If the tagname you specify does not have such a setting, an error message will be produced.

Appendix G

Summary of Tags, Attributes, and Values for the Standard Tagset

The following is a reference summary of the tags supported by the "standard tag set" layouts, including their attributes and values. The information is organized in alphabetical order by tagname. Tags with corresponding end-of tags are shown as <TAGNAME> – </TAGNAME>, and unless otherwise specified, the attributes pertain to the start-of tag. If shown as <TAGNAME> –[</TAGNAME>], the end-of tag is optional. Values that are described as "default" are the default values for the standard-tagset layouts that are of a general-purpose nature. Layouts designed for particular publishing bodies will have defaults peculiar to those bodies' style requirements.

<ABSTRACT> [Heading Text] – [</ABSTRACT>]

STITLE = "Short Title"

<ADDRESS> – </ADDRESS>

<ALINE> one line of author address

<ANOTE> –[</ANOTE>]

ID = idname

<APPENDIX> –[</APPENDIX>]

<AUTHLIST> –[</AUTHLIST>]

<AUTHOR> Author's Name

BYLINE = "byline string"

<AUTHOR> Author's Name inside <AUTHLIST>

REFID = "idname(s)" – – required

<BACKM> –[</BACKM>]

<BIB>

ID = idname

<BIBID>

REFID = idname – – required

<BIBREF>

REFID = "idname(s)" – – required
PP = "page reference" – – do not use if REFID specifies a list or range

<BL> – </BL> see OL

<BODY> –[</BODY>]

<BOX> – </BOX>

<CIT>[text] – </CIT>[continued text]

<COPYRIGHT>[text]

SYMBOL="string" – – characters to be used as the copyright symbol

<COPYNOTE>[text] – </COPYNOTE>

<DATE>[date substitute]

<DL> – </DL>

BREAK – – forces the definition description to start on the output line below the definition term (see also TSIZE below)
TERMHI=n – – (default 2) the highlighted-phrase level for the definition term
TSIZE=n – – (default 10) the number of character spaces to be reserved for definition-term text; if a term is longer, the description will start on the next output line
see also OL

<DOCNUM> document number

<DT> definition term

<DD> definition description

<DTHD> definition term heading

<DDHD> definition description heading

<EN>[text] – </EN>[continued text]

ID=idname – – if not specified, this endnote cannot be referenced by the ENREF tag

<ENDNOTES>

<ENREF>

REFID=idname – – required

<FIG> – </FIG>

ID=idname – – if not specified, the table or figure number will be generated without your having any way in which to reference it
DEPTH=n – – the number of blank lines to be left for the body of the table or figure; useful when the body is to be pasted in
FONT=fontid – – MONO (default) or TEXT; the font in which the body of the table or figure is to be produced; ignored for devices that do not support fonts
PLACE=
FLOAT – – (default) if the table or figure will not fit in the current output column, it will appear at the top of the next column and the remainder of the current column will be filled with the text that follows the /FIG or /TAB tag

ANY -- no attempt will be made to ensure that the table or figure will fit in the current output column; if it does not, it will be split across a column or page boundary

INLINE -- if the table or figure will not fit in the current output column, the rest of the column will be left blank and the table or figure will be positioned at the top of the next column

PAGE -- enough blank lines will be added to the bottom of the table or figure (after the frame, if there is one) to ensure that it completely fills the output column

TOP -- the remainder of the current column will be left blank and the table or figure will be positioned at the top of the next column

The following is not yet implemented. You can specify it, but you will get the result of FLOAT instead.

BOTTOM -- the table or figure will be floated to the bottom of the current output column if there is room for it, or to the top of the next output column if not; the remainder of the current output column will be filled with the text that follows the /FIG or /TAB tag

FRAME=

BOX -- (default) figure or table will be enclosed in a box

NONE -- no frame will be used

RULE -- a horizontal rule will be used both before and after

'string' -- the specified characters will be repeated across the top and bottom

CAP=

LONG -- (default for figures) the figure caption will be block-indented to the right of the figure number; if specified for a table, the table number and table caption will be centered on separate lines

SHORT -- (default for tables) the table number and caption will be centered if they both will fit on the same output line, otherwise they will be handled in the block-indented manner described above for figures; if specified for figures, the figure number and caption will be centered on the same line

<FIGCAP> [Heading Text]

STITLE="Heading Text" -- if specified, the STITLE heading text will be placed into the FIGLIST instead of the FIGCAP heading text

<FIGDESC> [text] - [</FIGDESC>]

<FIGID>

REFID=idname -- required

<FIGLIST> [Heading Text] see TOC

<FIGREF>

REFID=idname -- required

<FN> [text] - </FN> [continued text]

ID=idname

SYMBOL=

NONE -- this footnote will not have an annotation symbol

'string' -- the specified character string will be used as the annotation symbol

if not specified, the next number in the footnote sequence will be used as the annotation symbol

<FNREF>

REFID = idname – – required

< FRONTM > –[< /FRONTM >]

< GDOC > – < /GDOC > required

SEC = "text" – – the document's security classification
HY = – – the text hyphenation mode to be used
ON – – an attempt will be made to hyphenate any word that is too long to fit in the space remaining on an output line
OFF – – no hyphenation will be performed
USER – – only words that contain hyphens will be candidates for hyphenation
JU = – – the text justification mode to be used
ON – – extra white space will be added between words so that all output lines will be completely filled (ignored for the last output line in a paragraph)
HALF – – only half the effect of ON; produces semi-ragged line endings
OFF – – no extra white space will be added; produces extremely ragged line endings

< GL > – < /GL > see OL

< GT > glossary term

< GD > glossary description

< H0 > ,1,2,3,4,5,6

ID = idname – – heading-identifier name, for use by HDREF
N = "symbol" – – for use with H1 only, and ignored if the layout does not number them. If the value is numeric, it will be used to number this level-one heading and subsequent level-one headings will continue in sequence (unless specified with another N =). If the value is not numeric, it will be used to number this level-one heading, but subsequent level-one headings will continue from the numbering sequence of the previous level-one heading.
STITLE = "Short Title" – – (H0,1,2 only)

< H1SUB > may only be used immediately after a H1 (or equivalent) or another H1SUB tag

< HDREF >

REFID = idname – – required

< HP0 > ,1,2,3 > [text] – < /HP0 > ,1,2,3 > [continued text]

< INCLUDE >

FILE = filename – – required

< I1 > ,2,3 > text, < IH1 > ,2,3 > text Any combination of ID, REFID, and IX can be specified together. No more than one of PG, SEE, or SEEID may be specified together. Several additional constraints are defined below.

PG = – – if not specified, the current page number will be used as the page – reference text
MAJOR – – the current page number will be placed at the beginning of the list of page references, to indicate that this is the major reference for this index term

START – – specifies that the current page number is the start of a range of pages within which this index term occurs

END – – specifies that the current page number is the end of a range of pages within which this index term occurs

The intended function of START and END is to generate a page-range reference of the form "n – m" where n is the start of the range and m is the end. As currently implemented, they have no effect other than to add the current page number to the list of page references.

'text' – – can be anything other than MAJOR, START, and END. The text between the apostrophes will be used as the page reference, instead of the current page number. If the text is a single blank, the index term will be added to the index without a page reference.

SEE = "text" – – the text to be used as a "see item" in the reference list, instead of the current page number

ID = idname – – index-identifier name; may contain up to seven letters and/or digits, and cannot be specified as the ID attribute of any other index – related tag

REFID = idname – – must have already been defined as the ID value of a previous I1,2,3 or IH1,2,3 tag. The index term will be added under the higher-level term(s) of the named index-id, instead of under the most recent higher-level index term(s). REFID is invalid with the I1 tag.

SEEID = idname – – must have already been defined as the ID value of a previous I1,2,3 or IH1,2,3 tag. The index term(s) of the specified index-identifier name will be used as the text of the "see" reference.

IX = n – – an index number, from 1 to 9; if omitted, the index term will be added to index number 1

< INDEX > [Heading Text]

IX = n – – specifies which index is to be printed (a numeric value between 1 and 9); if omitted, index number 1 is printed

COLUMNS = n – – the number of columns in which the index is to be printed (a numeric value between 1 and 9). With layouts that produce the document in multiple-columns mode, this attribute is ignored. With layouts that produce the document in single-column mode, a value of 2 will be used if the attribute is omitted.

< INST > – [**< /INST >**]

ID = idname

< IREF >

REFID = idname – – required

< LI > [text] for ordered, unordered, and simple lists

< LIHD > Heading Text for ordered, unordered, and simple lists

< LP > [text]

< LQ > [text] – **< /LQ >**

< NOTE > [text] – [**< /NOTE >**]

NAME = "text" – – will be used instead of the word **Note**

< OL > – **< /OL >**

COMPACT = n

- **0** – – no compaction; there will be a blank line before the first list item, between each list item, and after the last list item
- **1** – – with a layout that produces its output in single-spaced mode, will eliminate the blank line between items within the list; with a layout that produces its output in double-spaced mode, causes the text of the list items to be produced in single-spaced mode, but with a blank line between each list item
- **2** – – a single-spaced list with no blank lines before, between, or after list items

< P > [text]

< PC > [text]

< PICTURE >

NAME = name of picture file
HMOVE =
- **CENTRE** – – (default) picture will be centered
- **LEFT** – – picture will be left adjusted
- **RIGHT** – – picture will be right adjusted

< PREFACE > [Heading Text] – [< /PREFACE >]

STITLE = "Short Title"

< PSC > – < /PSC >

PROC = "list of output devices" – – if the output device to which the document is being produced is not one of those specified in the list, or if no list is specified, the contents of the block will be ignored

< Q > [text] – < /Q > [continued text]

< ROW > – < /ROW >

ALIGN = – – specifies the alignment for all columns in a row
- **LEFT** – – left alignment of the columns
- **RIGHT** – – right alignment of the columns
- **CENTRE** – – centering of the columns
- **CHAR** – – alignment on a character

CHAR = "character" – – character to align on (if ALIGN = CHAR is specified). Default "."

< SINCELL > – < /SINCELL > start and end of single cell material (in a table)

< SL > – < /SL > see OL

< T >

ALIGN = – – specifies the alignment for the column
- **LEFT** – – left alignment of the column
- **RIGHT** – – right alignment of the column
- **CENTRE** – – centering of the column
- **CHAR** – – alignment on a character

CHAR = "character" – – character to align on (if ALIGN = CHAR is specified). Default "."

<TAB> – </TAB> for these and TABCAP, TABDESC–/TABDESC, TABID, and TABREF, see the corresponding tags for figures

<TABLIST> [Heading Text] see TOC

<TITLE> Document Title

STITLE = "Short Title"

<TITLEP> – </TITLEP>

ALIGN =
CENTER – – (default for most layouts) the title-page material will be centered
LEFT – – the title-page material will be left-aligned
RIGHT – – the title-page material will be right-aligned (default for GMLss)
DIVISION = – – only for SGMLprep style
EP – – EP division titlepage
TH – – TH division titlepage
NUMBER = "string" – – reference number for the perprint in the SGMLprep style

<TOC> [Heading Text]

PLACE =
TOP – – the resulting output will begin at the top of a page (default for TOC)
INLINE – – the resulting output will start a few lines below the previous output (default for FIGLIST and TABLIST); if PLACE = INLINE is specified with TOC and the document is SCRIPTed in single-pass mode, PLACE = TOP will be assumed

<UL> – </UL> see OL

<XMP> – </XMP>

ALIGN =
LEFT – – (default) each input line will be left-aligned on output; any line that is too long will be output as multiple left-aligned lines
BLOCK – – the formatting mode for the document will also be used for the lines of the example; the result will be block of text
CENTER – – each input line will be centered on output, and any line that is too long will be output as multiple centered lines
LINE – – each input line will be output exactly as it appeared in the input
RIGHT – – each input line will be right-aligned on output, and any line that is too long will be output as multiple right-aligned lines
DEPTH = n – – the number of blank lines to be left for the body of the example; useful when the body is to be pasted in
FONT = fontid – – MONO (default) or TEXT; the font in which the body of the example is to be produced; ignored for devices that do not support fonts
IN = n – – indentation value for the example block
PLACE =
INLINE – – (default) the example block will be kept together; if it cannot fit in the current column, a new column will be started and the example block will appear at the top of this new column
ANY – – the example block may be split across columns or pages

1. Index

2. Bibliography

1. *Document Composition Facility Generalized Mark-up Language: Starter Set Reference*, SH20-9187-0. IBM Corporation, 1980.

2. *Waterloo SCRIPT Reference Manual* for Waterloo SCRIPT – Version 85.1 (85SEP20). Waterloo: Department of Computing Services, University of Waterloo.

3. *Waterloo SCRIPT User's Guide* for Waterloo SCRIPT – Version 85.1 (85SEP20). Waterloo: Department of Computing Services, University of Waterloo.

4. *GML Reference Summary and SCRIPT Reference Summary* for Waterloo SCRIPT – Version 85.1 (85SEP20). Waterloo: Department of Computing Services, University of Waterloo.

5. ISO 8879. *Information processing – Text and Office systems – Standard Generalized Markup Language (SGML)*, Geneva: International Organization for Standardization, 1985.

6. F.Hedman. *A Collection of Programs to Make Pictures and Slides on the Apollo*, CERN, Geneva, 1986.

7. F.Hedman. *Kermit, Macintosh and VM/CMS*, CERN, Geneva, 1986.