

Con-nect

Con-form User's Guide

Version 3.4.2

November 2016

This document applies to Con-nect Version 3.4.2.

Specifications contained herein are subject to change and these changes will be reported in subsequent release notes or new editions.

Copyright © 1985-2016 Software AG, Darmstadt, Germany and/or Software AG USA, Inc., Reston, VA, USA, and/or its subsidiaries and/or its affiliates and/or their licensors.

The name Software AG and all Software AG product names are either trademarks or registered trademarks of Software AG and/or Software AG USA, Inc. and/or its subsidiaries and/or its affiliates and/or their licensors. Other company and product names mentioned herein may be trademarks of their respective owners.

Detailed information on trademarks and patents owned by Software AG and/or its subsidiaries is located at <http://softwareag.com/licenses>.

Use of this software is subject to adherence to Software AG's licensing conditions and terms. These terms are part of the product documentation, located at <http://softwareag.com/licenses/> and/or in the root installation directory of the licensed product(s).

This software may include portions of third-party products. For third-party copyright notices, license terms, additional rights or restrictions, please refer to "License Texts, Copyright Notices and Disclaimers of Third-Party Products". For certain specific third-party license restrictions, please refer to section E of the Legal Notices available under "License Terms and Conditions for Use of Software AG Products / Copyright and Trademark Notices of Software AG Products". These documents are part of the product documentation, located at <http://softwareag.com/licenses> and/or in the root installation directory of the licensed product(s).

Use, reproduction, transfer, publication or disclosure is prohibited except as specifically provided for in your License Agreement with Software AG.

Document ID: CMF-USERGUIDE-342-20161125

Table of Contents

Con-form User's Guide	vii
1 Con-form Instructions	1
Instruction Syntax	2
Option Syntax	5
Suppressing the Recognition of Instructions	6
2 Writing A Simple Document	7
General Information	8
Jagged Edge	9
Left and Right Justification	10
Starting Text on a New Line	12
Creating Blank Lines	16
Including Comments in the Source Text	19
3 Document Layout	23
Parts of a Formatted Page	25
Instructions for Defining the Document Layout	27
Left and Right Margins	29
Gutter	33
Blank Lines at the Top of the Page	38
Blank Lines at the Bottom of the Page	39
Top Titles and Bottom Titles	40
Multi-Line Top and Bottom Titles	54
Line Length for Top and Bottom Titles	57
Page Numbering	60
Page Headers	66
Footnotes	71
Number of Lines on a Page	78
Line Spacing and Character Spacing	79
4 Positioning Text	85
Text Alignment	87
Indenting Text	93
Page Breaks	102
Availability of Remaining Lines on a Page	105
5 Emphasizing Text	117
Boldface	118
Underlining and Underscoring	122
Italic Text	127
Superscript Text	128
Subscript Text	129
Backspacing	130
Escape Sequences	131
6 Hyphenation	135
Hyphenation in the Formatted Output	137
Hyphenated Word in the Source Text	143

7 Tabs and Boxes	145
Tabs	146
Boxes	153
8 Formatting Large Documents	167
Embedding Documents	169
Saving and Restoring the Settings	173
Chapter Titles and the Table of Contents	176
Index Entries and the Index Summary	194
9 Variables	197
Defining Your Own Variables	199
Arithmetic Calculations	204
Text Functions	212
Decimal Numbers	215
The Variable Character	219
System Variables	221
10 Conditional and Repetitive Processing	225
Conditional Processing	227
Repetitive Processing	232
Defining the Relation	236
Defining the Consequence	239
Logical Operators	240
11 Macros	243
General Information	244
Defining a Macro	245
Calling a Macro	248
Local Macro Variables	248
12 Controlling the Output	253
Defining the Pages to be Output	254
Defining the Paper Tray	256
Blank Spaces	258
Translating Characters	261
Altering the Text Orientation	268
Problem Handling	270
13 Formatting a Document	271
The FORMAT Command	272
Printer Profiles	272
Formatting Profiles	273
14 Special Characters	275
Ampersand (&)	276
Apostrophe (')	276
Circumflex (^)	276
Comma (,)	277
Commercial-at (@)	277
Dollar (\$)	277
Hash (#)	279

Period (.)	279
Semicolon (;)	279
Slash (/)	280
Space	281
15 Form Letters	283
Using the COMPOSE Statement	284
Testing Your Form Letter	286
16 Deprecated Instructions	289
.>> - Label	291
.GO - Go to the Specified Label	291
.LO - Lower-Case	292
.OP LOC - Shiftlock Character	292
.OP SHI - Shift Character	294
.PN - Page Number	294
.TY - Type Information	294
.UP - Upper-Case	295

Con-form User's Guide

Con-form is a text formatter which is automatically installed with Con-nect.

To write your text, you can use the Con-nect editor (see the *Con-nect User's Guide*, section *Text Processing*).

This documentation provides a step-by-step introduction to Con-form. It starts with simple text formatting instructions and gradually introduces more complex features.

Many Con-form instructions are illustrated by examples. The examples show both how you specify an instruction in your source text and how the formatted output looks. As a rule, the term "output", as used in the examples, refers to the printed version of the formatted document.

This documentation is subdivided into the following sections:

Con-form Instructions	Explains how to specify Con-form instructions in your source text. It also explains how you can suppress the recognition of Con-form instructions.
Writing a Simple Document	Introduces the most important instructions that you need to produce a simple document.
Document Layout	Explains how to define the document layout.
Positioning Text	Explains features such as centering or right-adjusting text, indenting paragraphs and defining page breaks.
Emphasizing Text	Explains features such as boldface printing and underlining. Furthermore, it explains how to use escape sequences so that you can define, for example, italic text.
Hyphenation	Explains how to activate hyphenation for the English, French or German language.
Tabs and Boxes	Explains how to set tab stops and create boxes.
Formatting Large Documents	Explains how to merge several documents into a single output document ("embedding"). It also explains how to create a table of contents and an index.
Variables	Explains how to work with variables.
Conditional and Repetitive Processing	Explains how to define consequences that are to be processed when the result of a relation is true.
Macros	Explains how to work with macros.
Controlling the Output	Explains how to define the pages to be output (printed) and the paper tray to be used. It also explains how to replace multiple blanks with a single blank and how to translate characters.
Formatting a Document	Explains how to format a document, use a printer profile and define a formatting profile.
Special Characters	Explains the characters that have a special meaning in Con-form.
Form Letters	Explains how to create a form letter which contains variables that are to be substituted with information from an Adabas file. Furthermore, it explains how to test whether your form letter produces the desired results before you have the variables replaced with data from the database.
Deprecated Instructions	Lists all Con-form instructions which are still retained for compatibility with older Con-form versions. However, it is recommended that you do not use these instructions.

The following topics are covered below:

- **List of all Instructions**
- **List of all Options**

List of all Instructions

The following is an alphabetical list of all Con-form instructions.

- **.** (comment)**
- **.> (label)**
- **.AN (logical operator)**
- **.BF (boldface)**
- **.BM (bottom margin)**
- **.BP (bold print)**
- **.BR (line break)**
- **.BT (bottom title for all pages)**
- **.BX (box)**
- **.CB (compress blanks)**
- **.CE (center text)**
- **.CH (chapter)**
- **.CO (comment)**
- **.CS (character spacing)**
- **.CV (compute variable)**
- **.DU (dump workspace)**
- **.EB (bottom title for even pages)**
- **.EC (end-of-line character)**
- **.EF (end of file)**
- **.EI (end If-condition)**
- **.EL (else)**
- **.EM (embed)**
- **.EP (end of processing)**
- **.ET (top title for even pages)**
- **.EW (end While-loop)**
- **.FI (filling)**

- **.FM (footer margin)**
- **.FN (footnote)**
- **.FS (footer space)**
- **.GO (go to specified label)**
- **.HC (header clear)**
- **.HL (header lines)**
- **.HM (header margin)**
- **.HP (hyphen connection)**
- **.HS (header space)**
- **.IC (ignore instructions)**
- **.IF (start If-condition)**
- **.IL (insert lines)**
- **.IP (insert pages)**
- **.IX (index)**
- **.JU (justification)**
- **.LL (line length)**
- **.LM (left margin)**
- **.LO (lower-case)**
- **.LS (line spacing)**
- **.MA (start macro)**
- **.ME (end macro)**
- **.MX (exit from macro)**
- **.NL (need lines)**
- **.NP (new page)**
- **.OB (bottom title for odd pages)**
- **.OF (offsetting)**
- **.OP (options)**
- **.OR (logical operator)**
- **.OT (top title for odd pages)**
- **.PH (print header)**
- **.PL (page length)**
- **.PM (page-numbering mode)**
- **.PN (page number)**

- **.PS (paragraph start)**
- **.PT (put to table of contents)**
- **.RA (right-adjustment)**
- **.RM (right margin)**
- **.RS (restore context)**
- **.SA (save context)**
- **.SB (skip blanks)**
- **.SC (set chapter number)**
- **.SL (skip lines)**
- **.SU (substitution)**
- **.SV (set variable)**
- **.TB (tab stops)**
- **.TH (then)**
- **.TI (temporary indentation)**
- **.TM (top margin)**
- **.TO (text orientation)**
- **.TR (translate character to character)**
- **.TS (translate character to string)**
- **.TT (top title for all pages)**
- **.TY (type information)**
- **.UL (underline)**
- **.UP (upper-case)**
- **.US (underscore)**
- **.WH (start While-loop)**
- **.WX (exit from While-loop)**

List of all Options

The following is an alphabetical list of all Con-form options.

- **BIN (paper bin)**
- **BRN (suppress line breaks)**
- **BXH (horizontal box line)**
- **BXV (vertical box line)**
- **CHA (chapter numbering)**

- **CHI (chapter indentation)**
- **CHL (chapter levels)**
- **CSE (instruction separator character)**
- **DAS (characters after decimal)**
- **DEC (decimal character)**
- **ECH (echo instructions)**
- **EMN (embed nobreak)**
- **END (end-of-line character)**
- **ESC (escape character)**
- **GEV (gutter for even pages)**
- **GOD (gutter for odd pages)**
- **GUT (gutter for all pages)**
- **HYA (characters after the hyphen)**
- **HYB (characters before the hyphen)**
- **HYP (hyphenation)**
- **LOC (shiflock character)**
- **NPG (number of pages to output)**
- **PAG (page number)**
- **PGF (page formatting)**
- **PNS (page-number character)**
- **PTC (switch .PT on/off)**
- **REM (remainder of division)**
- **RND (round result of arithmetic calculation)**
- **ROM (Roman page-numbering)**
- **SHI (shift character)**
- **SSF (single sheet feeder)**
- **STA (start output at specific page)**
- **STO (stop output at specific page)**
- **TRI (thousands separator character)**
- **ULB (underscore blanks)**
- **VSG (variable character)**

1 Con-form Instructions

- Instruction Syntax 2
- Option Syntax 5
- Suppressing the Recognition of Instructions 6

This chapter explains how to specify Con-form instructions in your source text. It also explains how to suppress the recognition of Con-form instructions.


The following topics are covered below:

Instruction Syntax

Each description of a Con-form instruction is preceded by a diagram which shows all the possibilities for specifying the instruction in question. For example:

```
.RA number  
.RA ON  
.RA OFF
```

Each instruction is indicated in upper-case letters and must be specified as shown in the above diagram. Each parameter for which you must substitute a number, character or text is indicated in lower-case italics.

 **Note:** All instructions are indicated in upper-case letters. However, you can specify instructions and parameters in any combination of upper- and lower-case letters.

The following symbols are used within the syntax descriptions:

[]	Elements contained within square brackets are optional.
{ }	Elements contained within braces indicate that one (that is, one only) of the elements must be specified.
...	A term preceding an ellipsis may be repeated.

The Period

You must begin each Con-form instruction with a period (.). For example:

```
.BR
```

As a rule, you begin an instruction in the very first column of a line. However, it is possible to mix text and instruction in a line. See the description of the instruction separator character for further information.

When you specify more than one period, the subsequent input is not interpreted as a Con-form instruction.

The Parameters

With certain instructions, you must specify one or more parameters.

When an instruction requires a parameter, you must insert a space character between the instruction and the parameter. For example:

```
.FI ON
```

When an instruction requires more than one parameter, you must insert a comma (,) between the parameters. For example:

```
.BX 10,55
```

The Instruction Separator Character


When you define several instructions in the same line, you must insert the instruction separator character between the instructions (initially, the instruction separator character is the semicolon). For example:

```
.FI ON;.JU ON
```

You can also mix text and instructions in a line. The instruction separator character followed by a period (;.) always indicates that the subsequent input is an instruction (you must *not* include a blank between the instruction separator character and the period). For example:

```
This is text;.IL;This is more text;.IL;And this is also text
```

```
.CE ON;Centered Heading;.CE OFF;.IL;Text below the heading.
```

 **Caution:** The instruction separator character is only recognized, when it is followed by a period. When you specify a blank between the instruction separator character and the period, the subsequent input is not interpreted as an instruction.

.OP CSE - A Different Instruction Separator Character

```
.OP CSE=character
```

Initially, the instruction separator character is the semicolon. However, you can define a different character. For example, to define the colon (:) as the instruction separator character, you must specify:

```
.OP CSE=:
```

Option Syntax

.OP - Option

```
.OP keyword=value[,keyword=value]...
```

Options are special instructions that determine several operating characteristics of Con-form. When necessary, you can redefine the initial value of an option.

You must insert a space character between .OP and the keyword. You must insert an equal sign between the keyword and the value. You must not insert blanks before and after the equal sign.

The following example shows how to specify an option. In this case, CSE is the keyword and the colon (:) is the value.

```
.OP CSE=:
```

You can combine several options in the same line. To do so, you must insert a comma between the options. The following example shows how to define three different options in the same line:

```
.OP CSE=: ,END=% ,ESC=/'
```

When a character is already used for a different purpose (for example, the comma is used to separate the options of the .OP instruction), it is important that you enclose the value in apostrophes. For example:

```
.OP DEC=' , '
```

The options are described in the different sections of this documentation which deal with the topic in question.

Suppressing the Recognition of Instructions

.IC - Ignore Instructions

```
.IC ON  
.IC OFF
```

Since Con-form interprets a period in the first column of a line as the instruction identifier, you must not use this character in the first column of your text. However, you can suppress the recognition of instructions. This enables you to use the period in the first column and to output Con-form instructions in the formatted text.

To suppress the recognition of instructions, you must specify the following instruction:

```
.IC ON
```

All information after this instruction will also appear in the formatted version. To switch this feature off, so that Con-form instructions are executed again, you must specify the following instruction beginning in the *first* column of a line (this corresponds to the initial setting):

```
.IC OFF
```

2 Writing A Simple Document

▪ General Information	8
▪ Jagged Edge	9
▪ Left and Right Justification	10
▪ Starting Text on a New Line	12
▪ Creating Blank Lines	16
▪ Including Comments in the Source Text	19

This chapter introduces you to the most important instructions that you need to produce a simple document. Thus, this chapter only considers the "body" of the document, which is the pure text.

The following topics are covered below:

Text outside the body of the document comprises the page titles at the top and bottom of a page, page headings, footnotes, index entries and entries in the table of contents. These are explained later in this documentation.

General Information

When you write text and do not use Con-form instructions within your text, the formatted version of your text will look exactly as the unformatted version. This is because filling is initially switched off (see *.FI - Filling*).

The only difference you will notice are three blank lines at the top of each page. This is due to the initial values *.HS 1* (see *.HS - Header Space*) and *.HM 2* (see *.HM - Header Margin*). Furthermore, when you print the formatted document, you will notice a left margin consisting of 10 blank spaces before every printed line. This is due to the initial value *.OP GUT=10* (see *Gutter*). When you do not specify otherwise, the initial Con-form values apply to the layout of your document.

Jagged Edge

.FI - Filling

```
.FI ON
.FI OFF
```

Filling is the process by which Con-form fills up each line between the defined margins with the maximum number of words.

The advantages of filling are:

- you can edit your text without regard to line length - thus you can insert additional words in the middle of a paragraph as a separate line of text;
- you can specify different values for the left and right margins - your text is automatically arranged between the new margins.

Initially, filling is switched off (.FI OFF). In this case, the defined value for the right margin is not considered and the formatted text looks exactly as the unformatted source text.

If you want to arrange your text between the defined left and right margins so that the right margin is printed with a jagged edge, you must switch filling on. However, since filling interacts with justification (which is initially switched on), you must take care to switch justification off:

```
.FI ON;.JU OFF
```

Even for writing a simple document, it is recommended that you specify left and right margins (see [Left and Right Margins](#)) and switch filling on. For example:

```
.LM 0;.RM 60
.FI ON;.JU OFF
```



Note: When you switch filling on, but do not specify the left and right margin, the initial values .LM 0 and .RM 72 apply.

Left and Right Justification

.JU - Justification

```
.JU ON  
.JU OFF
```

When filling *and* justification are switched on, spaces are added between words so that the text is justified between the left and right margins. In this case, the right margin of the formatted version is no longer printed with a jagged edge.

Lines which are immediately followed by a break (for example, the last line of a paragraph) are not justified.

Initially, justification is switched on (.JU ON). This means, when you specify .FI ON at the beginning of your document, justification is automatically activated. However, if you want to switch justification on, it is recommended that you always specify both instructions in order to avoid confusion, namely:

```
.FI ON; .JU ON
```

When you have specified .JU for the body of the text, this is not automatically applied to the footnotes (see [.FN - Footnote](#)).

Example

This example illustrates the instructions `.FI` and `.JU` which are used for filling and justification. It also contains the instructions `.LM` and `.RM` which are used to define the left and right margins (see [Left and Right Margins](#)). Furthermore, it introduces the `.SL` instruction which is used to create blank lines (see [.SL - Skip Lines](#)).

Source Text

```
.LM 0;.RM 60
By default, filling is switched off (.FI OFF) and justification is
switched on (.JU ON). When filling is switched off, words are
not borrowed or carried over from one line to another.
The left and right margins which have been specified are not considered
and the formatted text looks exactly like the unformatted text.
.SL 1
.FI ON
Now filling has been switched on (.FI ON). Since justification is
switched on by default (.JU ON), spaces are added between words to cause
an even right-hand margin, i.e. the text is justified between both margins.
Justification can only take place when filling has been switched on.
.SL 1
.JU OFF
Now justification has been switched off (.JU OFF). This means that
the right-hand margin remains ragged.
However, since filling is still switched on (.FI ON), short lines are
filled up with words from other lines.
The left and right margins are considered.
```

Formatted Output

By default, filling is switched off (`.FI OFF`) and justification is switched on (`.JU ON`). When filling is switched off, words are not borrowed or carried over from one line to another. The left and right margins which have been specified are not considered and the formatted text looks exactly like the unformatted text.

Now filling has been switched on (`.FI ON`). Since justification is switched on by default (`.JU ON`), spaces are added between words to cause an even right-hand margin, i.e. the text is justified between both margins. Justification can only take place when filling has been switched on.

Now justification has been switched off (`.JU OFF`). This means that the right-hand margin remains ragged. However, since filling is still switched on (`.FI ON`), short lines are filled up with words from other lines. The left and right margins are considered.

Starting Text on a New Line

When filling is switched on, all text is arranged so that a line is filled with the maximum amount of words. To make sure that your formatted text starts on a new line, you can specify line breaks. There are several different ways to specify line breaks.

.BR - Break

```
.BR
```

You can use the .BR instruction to start text on a new line.

.EC - A Different End-of-Line Character

```
.EC character
```

Initially, the end-of-line character is the dollar sign (\$). The text after the end-of-line character is placed on the next line.

The end-of-line character must be entered at the end of a line in your source text. When it is not entered at the end of the line, it is interpreted as normal text.

For example, to define the percent sign (%) as the new end-of-line character, you must specify:

```
.EC %
```

.OP END - A Different End-of-Line Character

```
.OP END=character
```

Instead of the .EC instruction, you can specify .OP END. Thus, you can also specify the following to define the percent sign (%) as the new end-of-line character:

```
.OP END=%
```


Blank Spaces

One or more blank spaces at the beginning of a line in your source text cause a break in filling. However, the formatted text also has the blank spaces in the beginning of the line.

A line break does not occur when the following instruction has been specified:

```
.SB ON
```

See *.CB - Compress Blanks* for further information.

Blank Lines

You can insert blank lines in your source text. In the formatted text, the text after the blank lines starts at the beginning of a new line.

Instructions Which Cause a Break in Filling

When you use one of the instructions listed below, the text after the instruction also starts on a new line.

```
.BF  
.BX  
.CE  
.CH  
.EF  
.EM  
.FI  
.IL  
.IP  
.LM  
.NL  
.NP  
.OF  
.PH  
.PS  
.RA  
.SL  
.TI  
.UL
```

Example

This example illustrates the `.BR` instruction and the end-of-line character which is used to start a new line in the formatted document. Initially, the end-of-line character is the dollar sign (`$`). This example also shows how to define another end-of-line character using the `.EC` instruction. Furthermore, it shows how to start a new line either by using space characters at the beginning of a line, or by simply leaving blank lines in the source text.

Source Text

```
.LM 0;.RM 65
.FI ON
To make sure that text starts in a new line, you can specify
an end-of-line-character in your text. By default, this is
the $ character; all text after the $ is placed in the next line. The
$ must be entered at the end of a line.$
If the $ is not issued at the end of the line, it is interpreted
as normal text.
.SL 1
You can also specify a different end-of-line character using one of the
following instructions: .OP END=char or .EC char.
.SL 1
.EC &
The ampersand has been defined as the new end-of-line character.&
Thus, you can now output the $ when it is used at the end of a line:$
.BR
However, you can also use the .BR instruction to start text in a
new line.

Breaks are also caused by the insertion of blank lines in the text.
A space character at the beginning of a line also causes a break in
filling. However, the formatted text also has the space character at the
beginning of the line.
```

Formatted Output

```
To make sure that text starts in a new line, you can specify an
end-of-line-character in your text. By default, this is the $
character; all text after the $ is placed in the next line. The $
must be entered at the end of a line.
If the $ is not issued at the end of the line, it is interpreted
as normal text.

You can also specify a different end-of-line character using one
of the following instructions: .OP END=char or .EC char.

The ampersand has been defined as the new end-of-line character.
Thus, you can now output the $ when it is used at the end of a
```

line:\$

However, you can also use the .BR instruction to start text in a new line.

Breaks are also caused by the insertion of blank lines in the text.

A space character at the beginning of a line also causes a break in filling. However, the formatted text also has the space character at the beginning of the line.

Creating Blank Lines

You can either leave blank lines in your source text or specify them explicitly using one of the following instructions. For information on blank lines which have been defined for the top and bottom of each page, see [Blank Lines at the Top of the Page](#) and [Blank Lines at the Bottom of the Page](#).

Basically, the .SL and .IL instructions are similar. However, it is recommended that you use the .IL instruction, when you want to insert blank lines at the top of a page.



Note: You can also use the .PS instruction which automatically inserts one blank line in the formatted output (see [.PS - Paragraph Start](#)).

.SL - Skip Lines

```
.SL number  
.SL
```

If you want to separate two paragraphs, it is recommended that you use the .SL instruction. The separation of the two paragraphs is achieved either by the blank lines or by a page break. The .SL instruction does not cause blank lines at the top of a page.

For example, to specify 3 blank lines, you must use the following instruction:

```
.SL 3
```

If you do not specify a parameter, one line is skipped.

If the parameter specified is greater than the number of lines remaining on the page, the excess lines are ignored. For example, if you specify .SL 10 and only 2 more blank lines can be added to the page, the remaining 8 lines are ignored.

.IL - Insert Lines

```
.IL number  
.IL
```

If you want to insert blank lines at the top of a page (for example, to reserve blank space for a diagram that will be added later to the printed page) it is recommended that you use the `.IL` instruction (and not the `.SL` instruction).

In addition, you must also specify either the instruction `.NL` or `.NP`. For example, to specify 10 blank lines at the top of a page, you can use one of the following instruction combinations:

```
.NL 10;.IL 10
```

```
.NP;.IL 10
```

If the parameter specified with the `.IL` instruction is greater than the number of lines remaining on the page, the excess lines are inserted at the top of the following page.

If you do not specify a parameter with the `.IL` instruction, one blank line is inserted.

You can also enter `.IL 0` to cause a break in filling. No blank lines are produced in this case. The instruction `.IL 0` is therefore equivalent to the `.BR` instruction.

Example

This example illustrates the different ways of defining blank lines.

Source Text

```
.LM 0;.RM 60  
.FI ON;.JU ON
```

You can either leave blank lines in your source text, or specify a Con-form instruction.

The instructions `.SL` and `.IL` are similar. However, when you want to insert blank lines at the top of a page, it is recommended that you use the `.IL` instruction.

```
.SL 1
```

When you want to separate two paragraphs, it is recommended that you use the `.SL` instruction. The two paragraphs are separated by either blank lines or a page break. In contrast to the `.IL` instruction, the `.SL` instruction does not produce blank lines at the top of a page.

Formatted Output

You can either leave blank lines in your source text, or specify a Con-form instruction.

The instructions `.SL` and `.IL` are similar. However, when you want to insert blank lines at the top of a page, it is recommended that you use the `.IL` instruction.

When you want to separate two paragraphs, it is recommended that you use the `.SL` instruction. The two paragraphs are separated by either blank lines or a page break. In contrast to the `.IL` instruction, the `.SL` instruction does not produce blank lines at the top of a page.

Including Comments in the Source Text

A comment is additional information in your source text which is not meant to appear in the formatted document. It is indicated with either the `**` or the `.CO` instruction.

A comment can contain any characters and it must not exceed one line in length. For example, the end-of-line character (initially, this is the dollar sign) is ignored when it is used in a comment.

A comment does not cause a break in filling.

The instructions `**` and `.CO` work slightly different. If the complete line of text is to be ignored, it is recommended that you use the `**` instruction.

`` - Ignore All Instructions After the Comment**

```
** text
```

In contrast to the `.CO` instruction, the remainder of the line in which `**` has been entered is ignored, i.e. an instruction after the `**` instruction is not executed.

The instruction separator character (initially, this is the semicolon) is ignored. Thus, it can be used within the comment. For example:

```
** This is a comment; it will not appear in the formatted output
```

`.CO` - Execute All Instructions After the Comment

```
.CO text
```

The `.CO` instruction can be used within a line containing several instructions, i.e. other instructions can stand before and after the `.CO` instruction, separated from it by the instruction separator character. Initially, the instruction separator character is the semicolon.

If you want to use the instruction separator character within a comment, you must repeat it. Otherwise, the text after the instruction separator character appears in the formatted version. For example:

```
.FI ON;.CO This is a comment;; the next instruction will be executed;.JU ON
```


Example

This example illustrates the two different instructions that can be used to enter comments in the source text.

Source Text

```
.LM 0;.RM 60
.FI ON
You can include comments in your source text which will not occur
in the formatted version.
.** This is a comment; all instructions in this line are ignored;.LM 10
.** If you want to enter several comment lines, each comment line
.** must begin with either the .** or .CO instruction.
A comment does not cause a break in filling.
.SL 1
.CO This is another comment;;the following instruction is considered;.LM 10
Now the new left margin is in effect which was been specified after the
comment which began with .CO.
```

Formatted Output

You can include comments in your source text which will not occur in the formatted version. A comment does not cause a break in filling.

Now the new left margin is in effect which was specified after the comment which began with .CO.

3 Document Layout

- Parts of a Formatted Page 25
- Instructions for Defining the Document Layout 27
- Left and Right Margins 29
- Gutter 33
- Blank Lines at the Top of the Page 38
- Blank Lines at the Bottom of the Page 39
- Top Titles and Bottom Titles 40
- Multi-Line Top and Bottom Titles 54
- Line Length for Top and Bottom Titles 57
- Page Numbering 60
- Page Headers 66
- Footnotes 71
- Number of Lines on a Page 78
- Line Spacing and Character Spacing 79

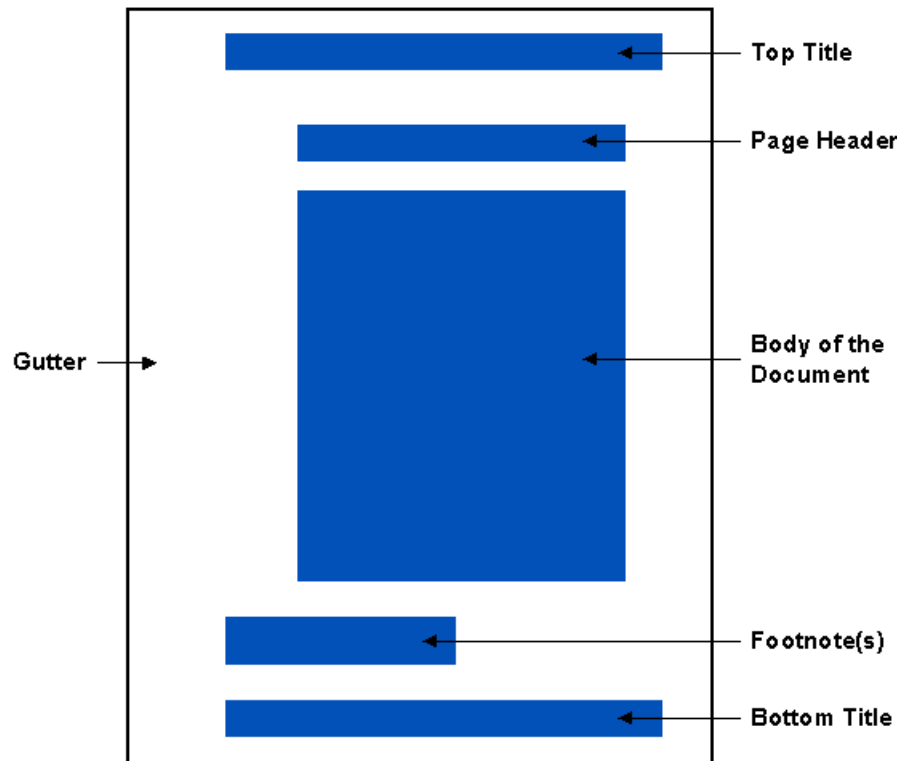
This chapter explains how you can define your individual document layout. When you do not specify otherwise, the initial Con-form values apply to your document layout.

The following topics are covered below:

You can create a formatting profile that is to be processed *before* each of your source documents (see [Formatting Profiles](#)). The formatting profile can contain, for example, your default settings for the page layout.

Parts of a Formatted Page

The following diagram gives an overview of the different parts of a formatted page.



Gutter

The gutter is the white space on the left of each page. You can define different gutter values for even- and odd-numbered pages.

Top Title

The top title appears at the head of each page. For example, when you are writing a book, you can print the book title and chapter name at the top of each page. You can also print the number of the current page at the top of each page. You can define different top titles for even- and odd-numbered pages.

Page Header

The page header appears at the head of each page, between the top title and the body of the document. You can also print the header within the body of the document.

Body of the Document

The body of the document contains the pure text.

Footnote(s)

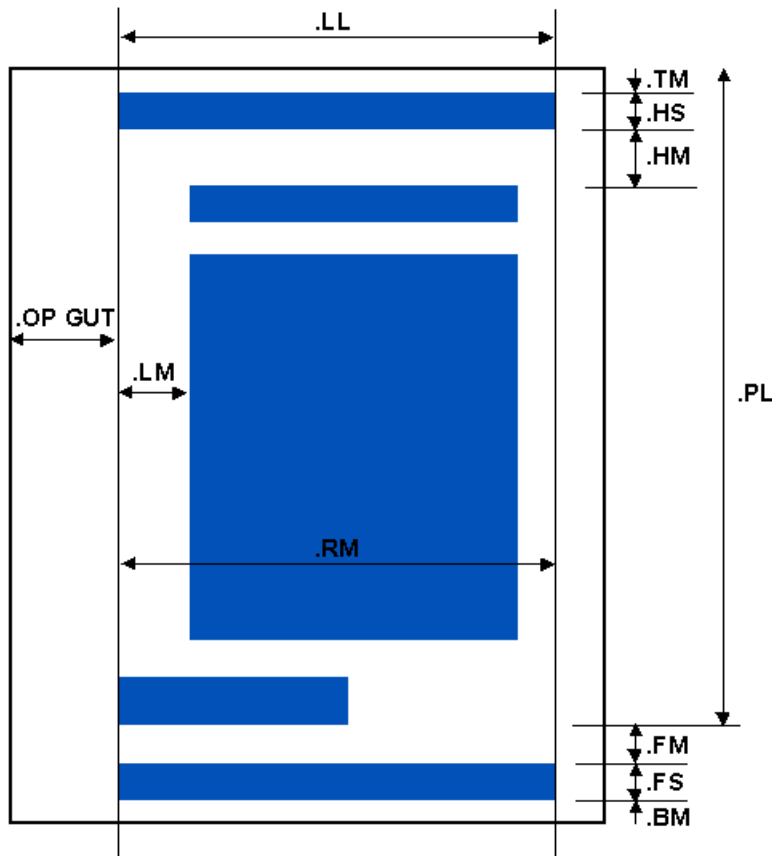
A footnote is additional information below the text. It refers to a text passage on the currently printed page.

Bottom Title

The bottom title appears at the foot of each page. For example, you can print the number of the current page at the bottom of each page. In contrast to the top title, you can also define that the number of the next page is to be printed in the bottom title. You can define different bottom titles for even- and odd-numbered pages.

Instructions for Defining the Document Layout

The following diagram shows the instructions that are used to define the document layout.



.LM and .RM

These instructions define the columns of the left and right margins.

.OP GUT

This instruction defines the number of characters for the gutter on all pages. To define the gutter for odd- and even-numbered pages, you must use the instructions `.OP GOD` and `.OP GEV`.

.TM, .HM, .BM and .FM

These instructions define the number of blank lines between the different parts of the page.

.HS and .FS

These instructions define the number of lines reserved for the top and bottom titles.

.LL

This instruction defines the line length (i.e. the number of characters) for the top and bottom titles.

.PL

This instruction defines the page length, i.e. the number of lines on the page including top margin, top title, page header, text and footnotes.

Left and Right Margins

The left and right margins define the line length for the page header and the body of the document. They do not affect the top and bottom titles (see [.LL - Line Length](#)).

.LM - Left Margin

```
.LM number  
.LM +number  
.LM -number  
.LM
```

The initial setting of the left margin is in column 0. It is not possible to define a negative column for the left margin, i.e. you must not specify a column smaller than 0.

The text always starts after the column which has been defined for the left margin. For example, if column 10 has been defined for the left margin, the text starts in column 11.

The current value is stored in the variable \$LN (see [Modifiable System Variables](#)).

There are several ways to define another position for the left margin with the .LM instruction. The following examples illustrate how to use this instruction.

You define the number of a column. If column 10 is to be the position of the left margin, you must specify:

```
.LM 10
```

If the current position of the left margin is in column 10 and you want to move it 5 columns to the right so that it starts in column 15, you must specify:

```
.LM +5
```

If the current position of the left margin is in column 15 and you want to move it 10 columns to the left so that it starts in column 5, you must specify:

.LM -10

When you do not specify a parameter, this instruction is equivalent to .LM 0 which is the initial setting of this instruction.

.RM - Right Margin

```
.RM number  
.RM +number  
.RM -number  
.RM
```

The initial setting of the right margin is in column 72. The maximum value is 190.

The right margin is placed behind the text. For example, if column 60 has been defined for the right margin, the last character of the text line is printed in column 59.

The current value is stored in the variable \$RM (see [Modifiable System Variables](#)).

There are several ways to define another position for the right margin with the .RM instruction. The following examples illustrate how to use this instruction.

You define the number of a column. If column 60 is to be the position of the right margin, you must specify:

```
.RM 60
```

If the current position of the right margin is in column 60 and you want to move it 5 columns to the right so that it starts in column 65, you must specify:

```
.RM +5
```

If the current position of the right margin is in column 65 and you want to move it 10 columns to the left so that it starts in column 55, you must specify:

```
.RM -10
```

When you do not specify a parameter, this instruction is equivalent to .RM 72 which is the initial setting of this instruction.

Example

This example illustrates the different ways to enter the .LM and .RM instructions which are used to set the left and right margins.

Source Text

```
.FI ON;.JU ON
No left and right margins have been defined for this part
of the text. Therefore, the defaults apply, i.e. the initial
values for the margins are .LM 0 and .RM 72.
.SL 1
.LM 10;.RM 60
Now column 10 has been defined for the left margin and column
60 has been defined for the right margin.
.SL 1
.LM +5;.RM -5
Now the left margin has been moved 5 columns to the right
so that it starts in column 15, and the right margin has been
moved 5 columns to the left so that it starts in column 55.
.SL 1
.LM -5;.RM +5
Now the left and right margins have been moved back to their
previous positions.
```

Formatted Output

No left and right margins have been defined for this part of the text. Therefore, the defaults apply, i.e. the initial values for the margins are .LM 0 and .RM 72.

Now column 10 has been defined for the left margin and column 60 has been defined for the right margin.

Now the left margin has been moved 5 columns to the right so that it starts in column 15, and the right margin has been moved 5 columns to the left so that it starts in column 55.

Now the left and right margins have been moved back to their previous positions.

Gutter

The white space formed by the inside margins of two facing pages (as of a book) is referred to as "gutter".

In Con-form you define the gutter by specifying the number of blank spaces before the *left* margin. For example, when you define a larger gutter value, the whole text is shifted to the right.

The initial setting for the gutter is 10 spaces before the left margin; this value applies to all pages, no matter whether they are even- or odd-numbered pages.



Note: When your environment allows you to modify the character spacing (see [.CS - Character Spacing](#)), Con-form interprets the number of blank spaces in tenths of inches. In this case, it is possible that the character spacing used in the gutter differs from the character spacing used in the text.

In addition to the gutter value which applies to all pages, you can also define different gutter values for even- and odd-numbered pages. This is useful if you are producing a document which is to be bound in book format. In this case, it is recommended that you use a smaller value for the even-numbered pages to shift the text to the left, and a larger value for the odd-numbered pages to shift the text to the right.

The gutter determines the position of the left margin on the printed page. However, the gutter does *not* affect the column number of the left margin (defined with .LM). For example, when the left margin starts in column 10 and you define a larger value for the gutter, the left margin still starts in column 10, although the gutter space has been increased.

The gutter is defined with the .OP instruction, as described on the following pages. When you format the document, you can only see the gutter in the printed version (the gutter is not shown when you display the formatted version on the screen).

.OP GUT - Gutter Value for All Pages

```
.OP GUT=number
```

This instruction affects both even- and odd-numbered pages.

If you want to define a different value for all even- and odd-numbered pages, you must define the number of spaces that are to be left blank before the left margin.

For example, if 15 spaces are to be left blank before the left margin, you must specify:

```
.OP GUT=15
```

Since this instruction is not processed in the middle of a printed page, it becomes effective only at the beginning of the next page.

.OP GEV - Gutter Value for Even Pages

```
.OP GEV=number
```

This instruction affects the even-numbered pages which are usually the left-hand pages in a book.

If you want to define a different value for the even-numbered pages, you must define the number of spaces that are to be left blank before the left margin.

For example, if 9 spaces are to be left blank before the left margin of each even-numbered page, you must specify:

```
.OP GEV=9
```

Since this instruction is not processed in the middle of a printed page, it becomes effective only at the beginning of the next even-numbered page.

.OP GOD - Gutter Value for Odd Pages

```
.OP GOD=number
```

This instruction affects the odd-numbered pages which are usually the right-hand pages in a book.

If you want to define a different value for the odd-numbered pages, you must define the number of spaces that are to be left blank before the left margin.

For example, if 21 spaces are to be left blank before the left margin of each odd-numbered page, you must specify:

```
.OP GOD=21
```

Since this instruction is not processed in the middle of a printed page, it becomes effective only at the beginning of the next odd-numbered page.

Example

This example illustrates .OP instructions which are necessary to define the gutter for even- and odd-numbered pages. Note that the position of the left margin on the printed page is determined by the values which have been defined for the gutter. Furthermore, the .NP instruction which is used to print text on the next page is introduced (see [.NP - New Page](#)).

Source Text

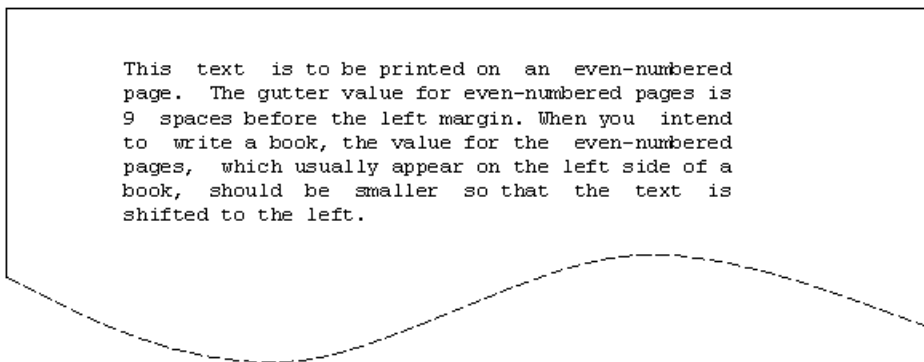
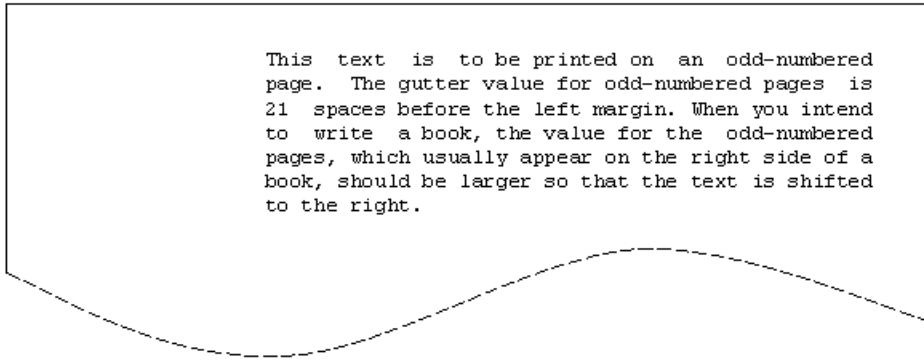
```
.** The gutter value for odd pages is 21:  
.OP GOD=21  
.** The gutter value for even pages is 9:  
.OP GEV=9  
.LM 0;.RM 50  
.FI ON;.JU ON
```

This text is to be printed on an odd-numbered page. The gutter value for odd-numbered pages is 21 spaces before the left margin.
When you intend to write a book, the value for the odd-numbered pages, which usually appear on the right side of a book, should be larger so that the text is shifted to the right.

```
.NP
```

This text is to be printed on an even-numbered page. The gutter value for even-numbered pages is 9 spaces before the left margin.
When you intend to write a book, the value for the even-numbered pages, which usually appear on the left side of a book, should be smaller so that the text is shifted to the left.

Formatted Output



Blank Lines at the Top of the Page

.TM - Top Margin

```
.TM number
```

This instruction defines the number of blank lines above the top title of every page. The initial setting is .TM 0.

For example, if 2 blank lines are to appear above the top title, you must specify:

```
.TM 2
```

The current value is stored in the variable \$TM (see [Modifiable System Variables](#)).

.HM - Header Margin

```
.HM number
```

This instruction defines the number of blank lines between the top title and the header. If a header has not been specified, it is the number of blank lines between the top title and the text. The initial setting is .HM 2.

For example, if 3 blank lines are to appear between the top title and the header, you must specify:

```
.HM 3
```



Note: Initially, one line is reserved for the top title (.HS 1; see [.HS - Header Space](#)), and there are two blank lines between the top title and the header (.HM 2). This accounts for the 3 blank lines which always appear at the beginning of a formatted text when you specify neither a top title nor a header.

The current value is stored in the variable \$HM (see [Modifiable System Variables](#)).

Blank Lines at the Bottom of the Page

.BM - Bottom Margin

```
.BM number
```

This instruction defines the number of blank lines below the bottom title of every page.

This instruction is redundant in any case, since the bottom title is always followed by a form feed. However, if you want to define, for example, 4 blank lines at the bottom of a page, you must specify:

```
.BM 4
```

The current value is stored in the variable \$BM (see [Modifiable System Variables](#)).

.FM - Footer Margin

```
.FM number
```

This instruction defines the number of blank lines between the text (or the last footnote if there are any) and the bottom title of every page. The initial setting is .FM 2.

For example, if 3 blank lines are to appear between the footnotes and the bottom title, you must specify:

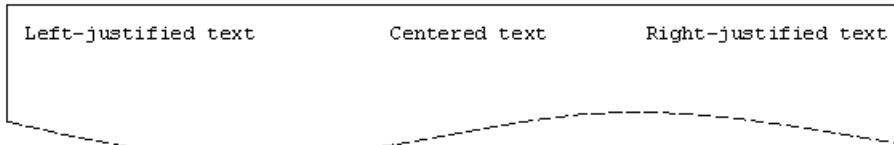
```
.FM 3
```

The current value is stored in the variable \$FM (see [Modifiable System Variables](#)).

Top Titles and Bottom Titles

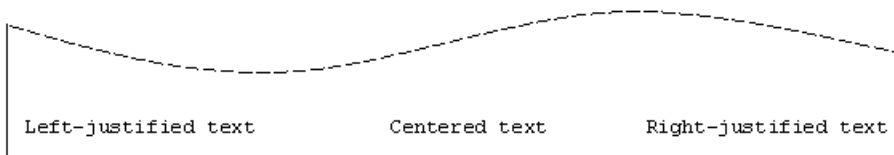
The *top title* is printed at the head of each page and can consist of several lines. Initially, one line is reserved for the top title (.HS 1; see [.HS - Header Space](#)).

The information in the top title can be printed in three different text areas, as illustrated in the following diagram:



The *bottom title* is printed at the foot of each page and can consist of several lines. Initially, one line is reserved for the bottom title (.FS 1; see [.FS - Footer Space](#)).

In analogy to the top title, the information in the bottom title can also be printed in three different text areas, as illustrated in the following diagram:



Both the top and bottom title are *not* affected by the left and right margins which have been defined with .LM and .RM. Instead, they are controlled by the .LL instruction (see [.LL - Line Length](#)).

You must take care not to specify a longer text than can fit in the top or bottom title; otherwise the information in the three different areas is automatically truncated in order to avoid overlapping.

There are several restrictions for the top and bottom title:

- The text must not include the slash (/), since the slash is used as the separator character between the parameters for the three different text areas.
- The parameter for the first text area must not start with a number, since numbers are used for multi-line top and bottom titles.
- Initially, the semicolon is used to separate several instructions in the same line. Therefore, you must repeat the instruction separator character (";;") if it is to appear in the top or bottom title.
- A parameter must not exceed 79 characters. This also applies when the parameter includes variables. After the substitution of the variables, the parameter must not exceed 79 characters.

If the page number is to be printed in the top or bottom title, you must include a hash (#) in the parameter for the relevant text area (see [Page Numbering](#)).

See also: [Line Spacing and Character Spacing at the Top and Bottom of a Page](#).

You can either define a top and bottom title which applies to all pages, or you can define different top and bottom titles for even- and odd-numbered pages. The latter is useful, if you are producing a document which is to be bound in book format.

All instructions for the top and bottom titles which are described on the following pages use the same parameter format; therefore, the parameter format has only been explained with the .TT (top title for all pages) and .BT (bottom title for all pages) instructions.

Top Title

If the following instructions are specified *before* any text is entered, the top title appears on the first page of the printout. If they are specified *after* text is entered, the top title does *not* appear on the first page but on all following pages. This feature does not apply to bottom titles.

.TT - Top Title for All Pages

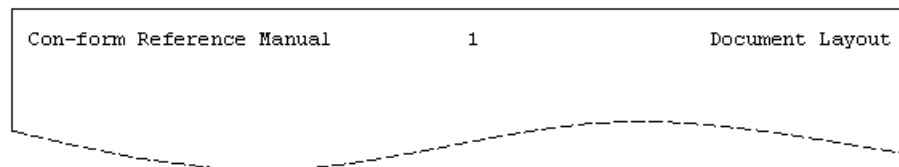
```
.TT text/text/text  
.TT  
.TT number text/text/text  
.TT number
```

This instruction defines the text strings which are to be printed in the top title of all even- and odd-numbered pages.

To specify your actual top title, you must replace the parameters of the .TT instruction with the actual top title. For example, to show the manual title on the left, the page number in the center and the chapter name on the right, you specify the following instruction:

```
.TT Con-form Reference Manual/#/Document Layout
```

The above instruction causes the following formatted output:

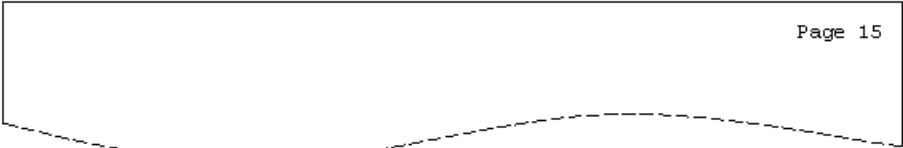


```
Con-form Reference Manual      1      Document Layout
```

The three text areas must be separated by slashes (see the above instruction). You must not omit one of the slashes; otherwise the information is not printed as intended. This is particularly important when you specify only one text area, as in the following example:

```
.TT //Page #
```

If the currently printed page has page number 15, the above instruction causes the following formatted output:



Multi-Line Top Title

If the top title is to consist of more than one line, you must first reserve the required number of lines (see *.HS - Header Space*).

To define a multi-line top title, each line of text for the top title must be preceded by a line number between 1 and 9. You must not leave a blank between the line number and the text. For example:

```
.TT 1//This is the first line of the top title  
.TT 2//This is the second line of the top title
```

The lines of text can be defined in any order. You must not define more than 9 lines for each top title.

Clearing the Top Title

To clear the top title so that it is no longer printed on the following pages, you must use the instruction without a parameter:

```
.TT
```

To clear a specific line of the top title, you must use the instruction with the number of that line. For example, to clear the second line of the top title, you must specify:

```
.TT 2
```


.ET - Top Title for Even Pages

```
.ET text/text/text  
.ET  
.ET number text/text/text  
.ET number
```

This instruction defines the text strings which are to be printed in the top title of all even-numbered pages.

See the description of the [.TT instruction](#) for further information.

Clearing the Top Title for Even Pages

To clear the top title so that it is no longer printed on the following even-numbered pages, you must use the instruction without a parameter:

```
.ET
```

To clear a specific line of the top title, you must use the instruction with the number of that line. For example, to clear the second line of the top title, you must specify:

```
.ET 2
```

.OT - Top Title for Odd Pages

```
.OT text/text/text  
.OT  
.OT number text/text/text  
.OT number
```

This instruction defines the text strings which are to be printed in the top title of all odd-numbered pages.

See the description of the [.TT instruction](#) for further information.

Clearing the Top Title for Odd Pages

To clear the top title so that it is no longer printed on the following odd-numbered pages, you must use the instruction without a parameter:

.OT

To clear a specific line of the top title, you must use the instruction with the number of that line. For example, to clear the second line of the top title, you must specify:

.OT 2

Bottom Title

.BT - Bottom Title for All Pages

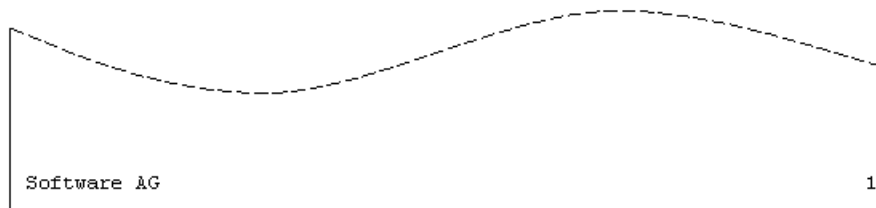
```
.BT text/text/text  
.BT  
.BT number text/text/text  
.BT number
```

This instruction defines the text strings which are to be printed in the bottom title of all even- and odd-numbered pages.

To specify your actual bottom title, you must replace the parameters of the .BT instruction with the actual bottom title. For example, to show the company name on the left and the page number on the right, you specify the following instruction:

```
.BT Software AG//#
```

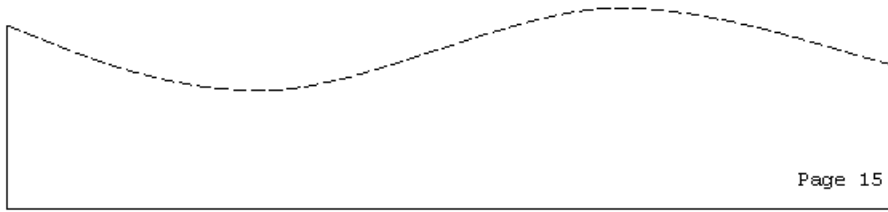
The above instruction causes the following formatted output:



In analogy to the top title, the three text areas of the bottom title must also be separated by slashes (see the above instruction). You must not omit one of the slashes; otherwise the information is not printed as intended. This is particularly important when you specify only one text area, as in the following example:

```
.BT //Page #
```

If the currently printed page has page number 15, the above instruction causes the following formatted output:



Multi-Line Bottom Title

If the bottom title is to consist of more than one line, you must first reserve the required number of lines (see *.FS - Footer Space*).

To define a multi-line bottom title, each line of text for the bottom title must be preceded by a line number between 1 and 9. You must not leave a blank between the line number and the text. For example:

```
.BT 1//This is the first line of the bottom title  
.BT 2//This is the second line of the bottom title
```

The lines of text can be defined in any order. You must not define more than 9 lines for each bottom title.

Clearing the Bottom Title

To clear the bottom title so that it is no longer printed on the following pages, you must use the instruction without a parameter:

```
.BT
```

To clear a specific line of the bottom title, you must use the instruction with the number of that line. For example, to clear the second line of the bottom title, you must specify:

```
.BT 2
```

.EB - Bottom Title for Even Pages

```
.EB text/text/text  
.EB  
.EB number text/text/text  
.EB number
```

This instruction defines the text strings which are to be printed in the bottom title of all even-numbered pages.

See the description of the [.BT instruction](#) for further information.

Clearing the Bottom Title for Even Pages

To clear the bottom title so that it is no longer printed on the following even-numbered pages, you must use the instruction without a parameter:

```
.EB
```

To clear a specific line of the bottom title, you must use the instruction with the number of that line. For example, to clear the second line of the bottom title, you must specify:

```
.EB 2
```

.OB - Bottom Title for Odd Pages

```
.OB text/text/text  
.OB  
.OB number text/text/text  
.OB number
```

This instruction defines the text strings which are to be printed in the bottom title of all odd-numbered pages.

See the description of the [.BT instruction](#) for further information.

Clearing the Bottom Title for Odd Pages

To clear the bottom title so that it is no longer printed on the following odd-numbered pages, you must use the instruction without a parameter:

.OB

To clear a specific line of the bottom title, you must use the instruction with the number of that line. For example, to clear the second line of the bottom title, you must specify:

.OB 2

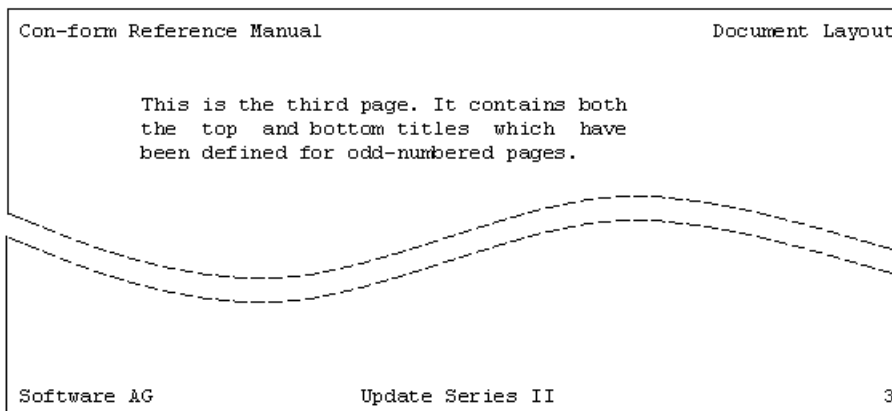
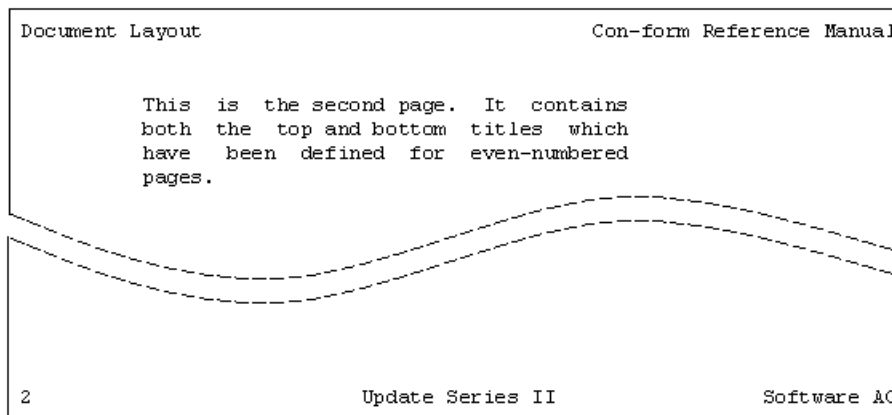
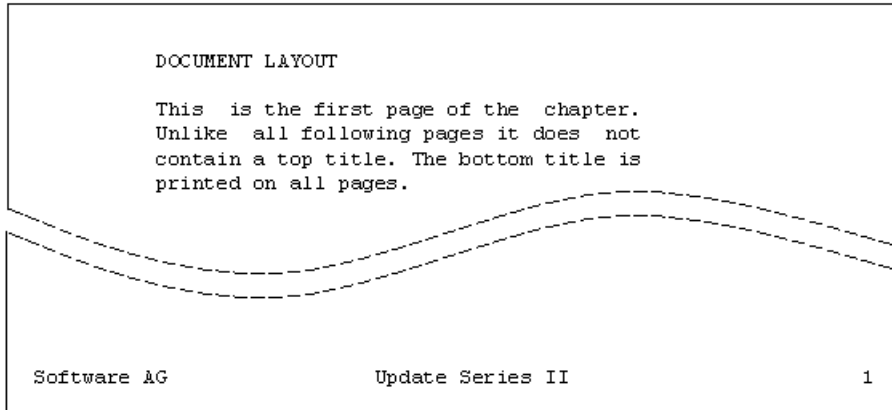
Example

This example illustrates the different instructions that are used to create top and bottom titles on odd- and even-numbered pages. Note that the first page does not contain a top title.

Source Text

```
.LM 10;.RM 50
DOCUMENT LAYOUT
.** The top title is defined AFTER the chapter name has been specified.
.** Thus, the first page of the chapter has a different appearance.
.** The top title will be printed on all following pages.
.** This description does NOT apply to the bottom title; it will be
.** printed on all pages.
.OT Con-form Reference Manual//Document Layout
.ET Document Layout//Con-form Reference Manual
.OB Software AG/Update Series II/#
.EB #/Update Series II/Software AG
.SL 1
.FI ON;.JU ON
This is the first page of the chapter. Unlike all following pages
it does not contain a top title. The bottom title is printed on
all pages.
.NP
This is the second page. It contains both the top and bottom titles
which have been defined for even-numbered pages.
.NP
This is the third page. It contains both the top and bottom titles
which have been defined for odd-numbered pages.
```


Formatted Output



Multi-Line Top and Bottom Titles

If the top or bottom title is to consist of more than one line, you must first reserve the required number of lines using the instructions `.HS` or `.FS`.

To define a multi-line top and bottom title, you use the instructions as explained in section [Top Titles and Bottom Titles](#). Each line of text for the title must be preceded by a line number between 1 and 9.

.HS - Header Space

```
.HS number
```

This instruction defines the number of lines that is to be reserved for the top title. The initial setting is `.HS 1`.

The current value is stored in the variable `$HS` (see [Modifiable System Variables](#)).

Since one line is initially reserved, you need not use this instruction when the top title consists of only one line.

For example, if you want to reserve 3 lines for the top title, you must specify:

```
.HS 3
```

If the parameter you specify is greater than the number of lines in the top title, the excess will appear as blank lines between the top title and the body of the document.

The length of the header space influences the length of the body of the document: the longer the header space, the shorter is the body of the document.

.FS - Footer Space

```
.FS number
```

This instruction defines the number of lines that is to be reserved for the bottom title. The initial setting is `.FS 1`.

The current value is stored in the variable `$FS` (see [Modifiable System Variables](#)).

Since one line is initially reserved, you need not use this instruction when the bottom title consists of only one line.

For example, if you want to reserve 2 lines for the bottom title, you must specify:

```
.FS 2
```

If the parameter you specify is greater than the number of lines in the bottom title, the excess will appear as blank lines between the body of the document and the bottom title.

The length of the footer space does not influence the length of the body of the document.

Example

This example illustrates how to create multi-line top and bottom titles on odd- and even-numbered pages. This example introduces the **.LL instruction** which is used to define the line length for top and bottom titles and the **.NP instruction** which causes a form feed.

Source Text

```

.** 2 lines will be reserved for header and footer space
.HS 2;.FS 2
.** The line length for top and bottom titles:
.LL 60
.OT 1//This is a top title for odd pages
.OT 2//This line appears only on odd pages
.ET This is a top title for even pages//
.OB 1Software AG//Update Series II
.OB 2//Page #
.EB 1Update Series II//Software AG
.EB 2Page #//
.SL 1
.LM 10;.RM 50
.FI ON;.JU ON
.** Assign page number 3 to this page
.NP 3
This is an odd-numbered page. It has a right-justified top title
which consists of 2 lines. Therefore, 2 lines of header space have
been reserved with the .HS instruction.
.SL 1
The bottom title consists of 2 lines and shows the current page
number on the right side.
.NP
This is an even-numbered page. It has a left-justified top title.
Unlike the top title for odd-numbered pages, the top title on this
page consists of a single line. Since 2 lines of header space have
been reserved, an additional blank line will appear below the top
title.
.SL 1
The bottom title on this page also consists of 2 lines and shows
the current page number on the left side.
```

Formatted Output

This is a top title for odd pages
This line appears only on odd pages

This is an odd-numbered page. It has a right-justified top title which consists of 2 lines. Therefore, 2 lines of header space have been reserved with the .HS instruction.

The bottom title consists of 2 lines and shows the current page number on the right side.

Software AG Update Series II
Page 3

This is a top title for even pages

This is an even-numbered page. It has a left-justified top title. Unlike the top title for odd-numbered pages, the top title on this page consists of a single line. Since 2 lines of header space have been reserved, an additional blank line will appear below the top title.

The bottom title on this page also consists of 2 lines and shows the current page number on the left side.

Update Series II Software AG
Page 4

Line Length for Top and Bottom Titles

The top and bottom titles start immediately after the gutter.

Both the top and bottom title are not affected by the left and right margins which have been defined with .LM and .RM. Instead, they are controlled by the .LL instruction.

.LL - Line Length

```
.LL number
```

This instruction defines the maximum number of characters to be printed in the top and bottom titles.

The initial setting is .LL 72. The maximum value is 190.

The current value is stored in the variable \$LL (see [Modifiable System Variables](#)).

To define a different line length, you must specify the number of characters in the line. For example, to specify a line length of 60 characters, you must specify:

```
.LL 60
```

You will receive an error message if the output text line is longer than the limit specified with this instruction.

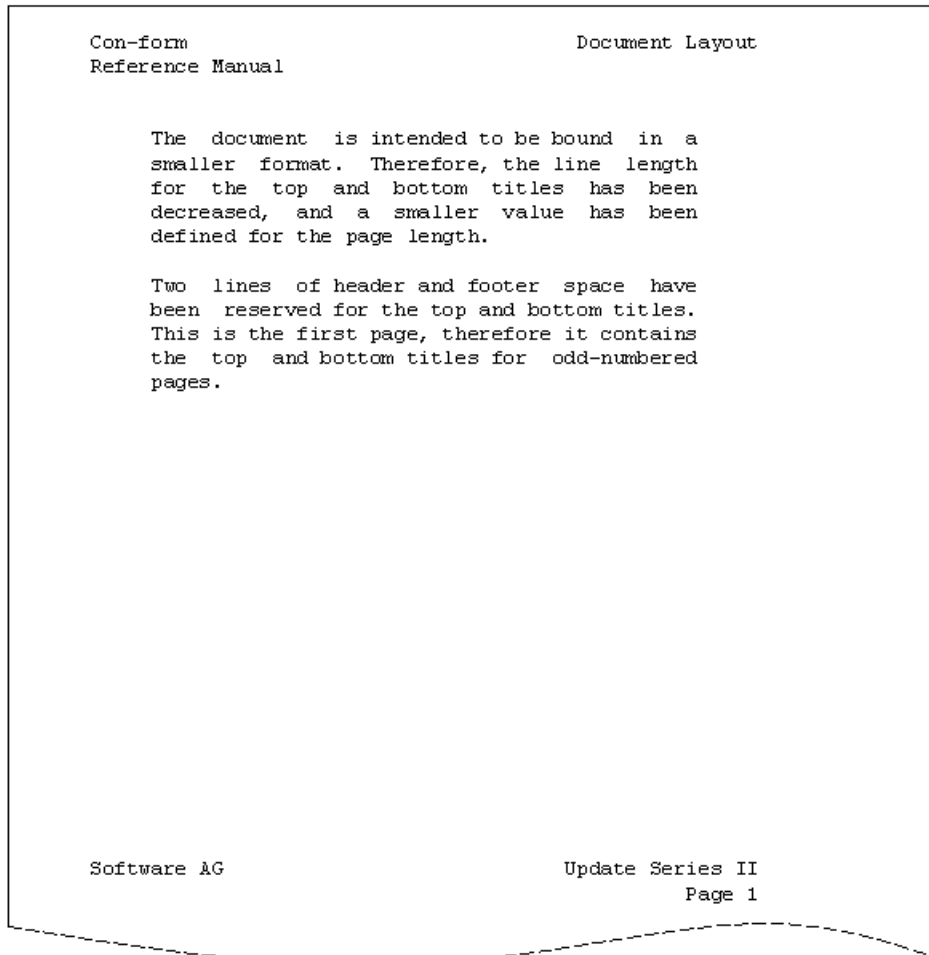
Example

This example illustrates how to define the line length for the top and bottom titles. Furthermore, this example introduces the **.PL instruction** which is used to define the page length. These instructions are of importance when you want to produce a book in a smaller format. The second page is automatically printed on a new sheet of paper. When binding the document in book format, you just have to cut off the excess paper to obtain the required page format.

Source Text

```
.LL 55;.PL 30
.HS 2;.FS 2
.OT 1Con-form//Document Layout
.OT 2Reference Manual//
.ET 1Document Layout//Con-form
.ET 2//Reference Manual
.OB 1Software AG//Update Series II
.OB 2//Page #
.EB 1Update Series II//Software AG
.EB 2Page #//
.SL 1
.LM 5;.RM 50
.FI ON;.JU ON
The document is intended to be bound in a smaller format.
Therefore, the line length for the top and bottom titles has been
decreased, and a smaller value has been defined for the page length.
.SL 1
Two lines of header and footer space have been reserved for the
top and bottom titles.
This is the first page, therefore it contains the top and bottom
titles for odd-numbered pages.
.NP
.** All text for the following pages would be entered after this
.** comment.
```

Formatted Output



Page Numbering

Initially, the page-number character is the hash sign (#). You can use it in the top and bottom titles, and with the **.SV instruction**. When used in the bottom title, the page-number character represents either the number of the current page or the number of the next page (see **.OP PAG - Number of the Current or Next Page**). Otherwise, it always represents the number of the current page.

The current page number is stored in the variable \$PN (see **Modifiable System Variables**).

.OP PNS - A Different Page-Number Character

```
.OP PNS=character
```

You can define a different page-number character, if you want to use a sign other than the hash. For example, if you want to define the exclamation mark (!) as the new page-number character, you must specify:

```
.OP PNS=!
```

Now you must use the exclamation mark instead of the hash when you want to print the page number in the top or bottom title.

When you use the page-number character within the body of the document, it is interpreted as normal text (i.e. the page-number character is printed in the formatted version, not the page number).

.OP PAG - Number of the Current or Next Page

```
.OP PAG=EQU  
.OP PAG=DIF
```

This instruction only affects the page number in the bottom title. The initial setting of this option is EQU.

The following instruction defines that the number of the *current* page is to be printed in the bottom title:


```
.OP PAG=EQU
```

The following instruction defines that the number of the *next* page is to be printed in the bottom title - the last page does not contain a bottom title:

```
.OP PAG=DIF
```

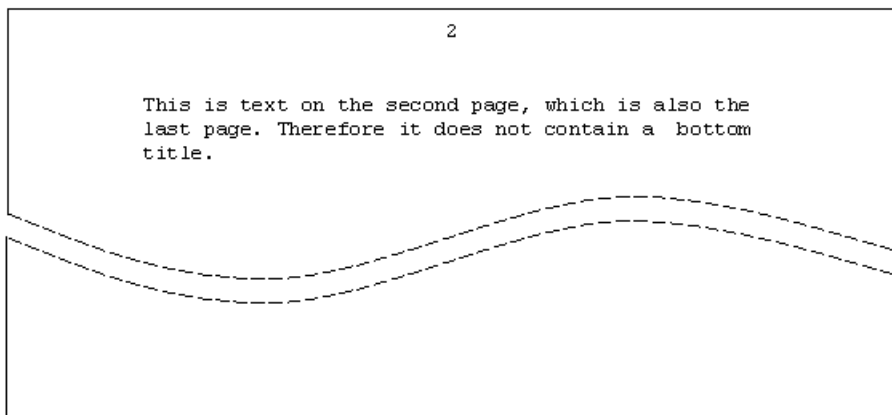
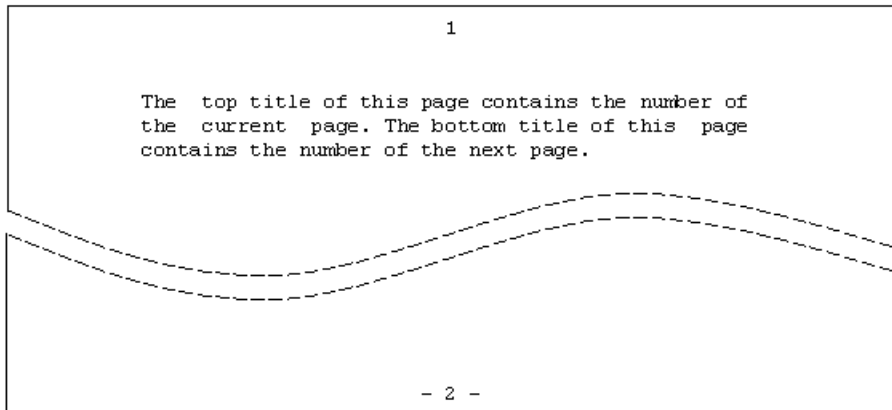
Example

This example illustrates how you can use the .OP PAG instruction to print the number of the next page in the bottom title. This is useful as an indicator that there are still pages which follow.

Source Text

```
.OP PAG=DIF
.TT /#/
.BT /- # -/
.LM 10;.RM 60
.FI ON;.JU ON
The top title of this page contains the number of the current page.
The bottom title of this page contains the number of the next page.
.NP
This is text on the second page, which is also the last page.
Therefore it does not contain a bottom title.
```

Formatted Output



.PM - Page-Numbering Mode

```
.PM A  
.PM R  
.PM r
```

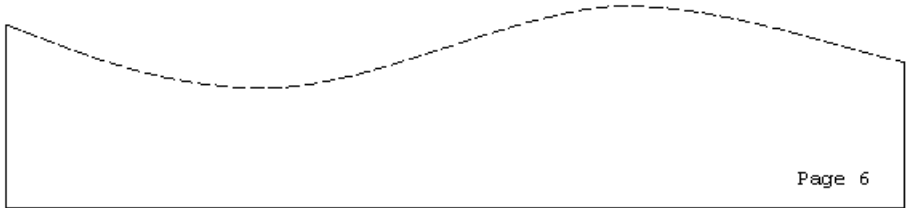
The page numbers can be printed either in Arabic or Roman numerals. The initial setting is Arabic.

Arabic Page Numbers

To define Arabic page numbers, you must specify:

```
.PM A
```

The following is an example of the resulting output:



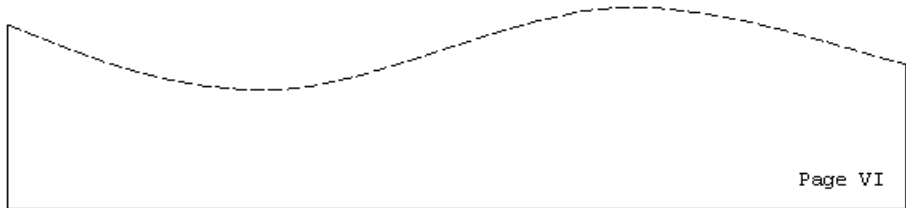
Upper-case Roman Page Numbers

To define upper-case Roman page numbers, you can specify:

```
.PM R
```

It is important that you specify the parameter in upper-case.

The following is an example of the resulting output:



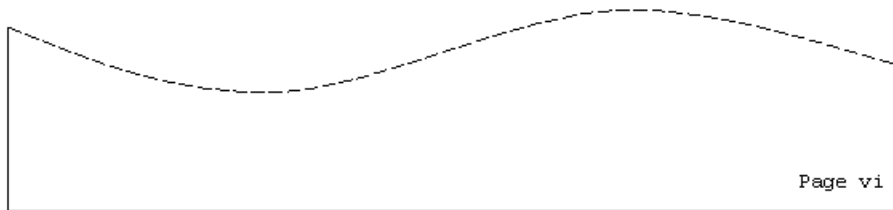
Lower-case Roman Page Numbers

To define lower-case Roman page numbers, you can specify:

```
.PM r
```

It is important that you specify the parameter in lower-case.

The following is an example of the resulting output:



.OP ROM - Roman Page Numbering

```
.OP ROM=UPPER  
.OP ROM=LOWER
```

The initial setting of this option is UPPER (i.e. Roman page numbers are printed in upper-case).

Instead of the .PM R instruction, you can also specify the following instruction to define upper-case Roman page numbers:

```
.OP ROM=UPPER
```

Instead of the .PM r instruction, you can also specify the following instruction to define lower-case Roman page numbers:

```
.OP ROM=LOWER
```

Page Headers

The page header appears at the head of each page, between the top title and the body of the document. You can also specify that the page header is printed within the body of the document.

The number of blank lines between the top title and the page header is determined by the header margin (initially .HM 2; see [.HM - Header Margin](#)).

Unlike the top and bottom titles, the page header is affected by the values which have been defined for the body of the document (such as left and right margins). The body of the document, however, is not affected by the values which have been defined for the page header.

See also: [Line Spacing and Character Spacing in Header Lines](#).

.HL - Header Lines

```
.HL instruction  
.HL text
```

There are two different uses of the .HL instruction. You can either define the text which appears in the page header or an instruction which applies to the page header. However, you must not mix text and instructions in one .HL instruction.

For example, to define the header text, you specify the following:

```
.HL This is the header line
```

To specify an instruction which applies to the page header, for example, to center the page header, you specify the following:

```
.HL .CE ON
```

If you want to define a new page header, you must first clear the current page header using the .HC instruction, and then define the new page header using the .HL instruction once more.

.HC - Header Clear

```
.HC
```

This instruction clears the page header which has been defined with the .HL instruction. To clear the current page header, you must specify the following:

```
.HC
```

The .HC instruction does not have a parameter.

.PH - Print Header

```
.PH
```

This instruction prints the page header which has been defined with the .HL instruction at the current position of the running text. The instructions which have been defined for the page header are also considered. For example, if the header at the top of the page has been centered with the .HL .CE ON instruction, it is also centered within the body of the document.

To print the page header at the current text position, you must specify the following:

```
.PH
```

The .PH instruction does not have a parameter.

You can also define a page header at some point within your text, and then issue the .PH instruction so that the header is output immediately. The header will then appear automatically at the top of all following pages (unless it is cleared with the .HC instruction).

Since the page header is automatically printed at the head of each page, the .PH instruction is only required, when you want to print the page header within the body of the document.

Example

This example illustrates how to define a page header and how to print it within the body of the document. Furthermore, this example introduces the **.CE instruction** which is used to center text.

Source Text

```
.LL 60
.TT Con-form Reference Manual//Page #
.FI ON;.JU ON
.LM 0;.RM 60
.HL .CE ON
.HL This is the page header.
.HL It is printed at the top of each page.
.HL This page header has been centered.
.HL .SL 1
.** You should always define a blank line with the .HL instruction.
.** so that the page header and the body of the document are always
.** separated.
These lines represent the body of the document.
Note that 2 blank lines have automatically been included between the
top title and the page header.
This is due to the initial value for the header margin (.HM 2).
.SL 1
```


You can print the page header within the body of the document. The instruction which has been defined for the page header will be considered.

```
.SL 1
```

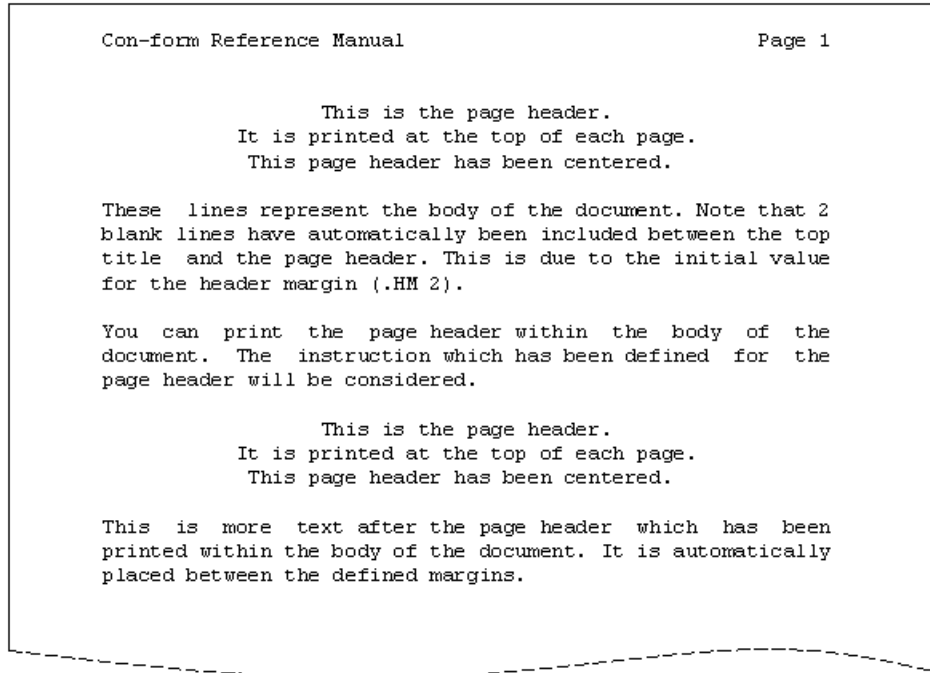
```
.PH
```

```
.** You need not define a blank line here, since one blank line
```

```
.** has been defined for the header with the .HL instruction.
```

This is more text after the page header which has been printed within the body of the document. It is automatically placed between the defined margins.

Formatted Output



Footnotes

.FN - Footnote

```
.FN text  
.FN ON  
.FN OFF
```

You can define text which is to be printed as a footnote at the bottom of the current page - between the body of the document and the bottom title. Con-form automatically prints a short horizontal line before the first footnote on a page.

The number of blank lines between the footnotes and the bottom title is determined by the footer margin (initially .FM 2; see [.FM - Footer Margin](#)).

See also: [Line Spacing and Character Spacing in Footnotes](#).

There are two different ways to use the .FN instruction.

One Line of Footnote Text

To define a single line of footnote text, you specify the .FN instruction followed by the line of text which is to be printed as the footnote. For example:

```
.FN This is the footnote.
```

If several of these instructions are issued, each instruction contains one line of text.

Unlike the page header, the footnotes are not affected by the values which have been defined for the body of the document (such as left and right margins). The footnote always starts in column 0 and you must take care, that you do not define a longer footnote text than can fit on the page. However, you can also output a footnote as a block of text as described below. In this case, you can also define the layout for a footnote.

Several Lines of Footnote Text

You can output the footnote as a block of text. To do so, you must enclose the footnote text and all instructions which apply to the footnote between the following two instructions:

```
.FN ON
```

```
.FN OFF
```

When you use these instructions, you can define left and right margins for a specific footnote as well as filling and justification.

Initially, filling is switched on for the footnote. If you do not define the left and right margins, the initial values apply (.LM 0 and .RM 72).

When you do not define justification for the footnote, its use is dependent on the settings for the body of the document. If justification is used in the body of the document, it is also used for the footnote.

The layout you have defined for a footnote always applies to this specific footnote. If you want the next footnote to appear in the same way, you must define the layout once more.

It is not possible to conditionally suppress the processing of the .FN OFF instruction by using it in an **.IF construction**.

Numbering Footnotes

Footnotes are not automatically numbered by Con-form. If you want a number to appear before a footnote, you must explicitly define it.

Manually

When you specify each footnote note number by yourself, this has the disadvantage that you have to renumber all footnotes, if you insert additional footnotes at the beginning of the document.

In the example below, "/1/" has been included in the document text, so that it is clear to which passage in the text the footnote refers.

```
This belongs to the body of the document /1/
.FN /1/ This is the first footnote.
and this is the body of the document continued.
```

The document text is formatted as follows (within the defined margins):

```
This belongs to the body of the document /1/ and this is the body
of the document continued.
```

The footnote is formatted as follows:

```
/1/ This is the first footnote.
```

Automatically

You can leave the responsibility for numbering the footnote to Con-form. The number of the current footnote is stored in the variable \$FN (see *Modifiable System Variables*). Each time, the variable \$FN is processed in the text, its current value is increased by one.

To number the footnotes consecutively, you must assign the value of \$FN to another variable using the *.SV instruction*. You must repeat the *.SV* instruction for each footnote that you want to define. For example:

```
.SV number=&$FN
This belongs to the body of the document /&number/
.FN /&number/ This is the first footnote.
and this is the body of the document continued.
.SL 1
.SV number=&$FN
You must not forget to update the "number" variable
.FN /&number/ This is the second footnote.
/&number/, otherwise the footnote number is not increased.
```

The document text is formatted as follows (within the defined margins):

```
This belongs to the body of the document /1/ and this is the body
of the document continued.
```

```
You must not forget to update the "number" variable /2/, otherwise
the footnote number is not increased.
```

The footnote is formatted as follows:

```
/1/ This is the first footnote.  
/2/ This is the second footnote.
```

Example

This example illustrates the several different ways in which footnotes can be defined. It shows how to number footnotes automatically, and how to define the layout for a footnote which has been defined as a block of text.

Source Text

```
.LM 5;.RM 60  
.FI ON;.JU ON  
The following text contains footnotes. The footnote is always  
printed on the page which also contains the reference to  
the footnote.  
.** &$FN is a special variable which is incremented each time it is  
.** used. You use it for consecutive numbering of footnotes within a  
.** document.  
To include a reference for the footnote within the running text,  
you must also define a variable in which the current footnote value is  
stored. Thus, you can automatically place the correct footnote value  
in the section of the page which points to the footnote below.  
.SV number=&$FN  
.** You must repeat the above instruction for each footnote which you  
.** include in your text.  
The first footnote applies to this sentence  
.FN /&number/ This is the first footnote.  
.** The following line contains the remainder of the sentence, namely  
.** the footnote indicator and the full stop.  
/&number/.  
.IL 1  
.** The following paragraph will contain another footnote, therefore  
.** the "number" variable must be updated:  
.SV number=&$FN  
If you want to create additional footnotes, you must not forget to update  
the "number" variable /&number/.  
.FN /&number/ This is the second footnote.  
It is also possible to update the "number" variable in the middle  
.SV number=&$FN  
.FN /&number/ This is the third footnote.  
of the sentence that you are currently writing /&number/.  
.NP  
.SV number=&$FN  
This is an example of a block of text which has been defined  
as a footnote /&number/.  
.FN ON  
.** You can define the left and right margins for the footnote.  
.** You can also specify filling and justification. This settings
```

```
.** apply only to this one footnote.  
.LM 10;.RM 50  
.FI ON;.JU ON  
.OF +4  
/&number/ This is a longer footnote. Specific left and right margins  
have been defined for the footnote. Otherwise, the defaults .LM 0  
and .RM 72 would have been used.  
Furthermore, the lines after the first line have been offset  
4 columns to the right.  
To stop offsetting, the .OF instruction must be issued  
once more, this time without a parameter.  
.OF  
.FN OFF  
The filling and justification instructions, which have been issued  
for the footnote do not apply to the body of the document.
```

Formatted Output - First Page

The following text contains footnotes. The footnote is always printed on the page which also contains the reference to the footnote. To include a reference for the footnote within the running text, you must also define a variable in which the current footnote value is stored. Thus, you can automatically place the correct footnote value in the section of the page which points to the footnote below. The first footnote applies to this sentence /1/.

If you want to create additional footnotes, you must not forget to update the "number" variable /2/. It is also possible to update the "number" variable in the middle of the sentence that you are currently writing /3/.

/1/ This is the first footnote.
/2/ This is the second footnote.
/3/ This is the third footnote.

Formatted Output - Second Page

This is an example of a block of text which has been defined as a footnote /4/. The filling and justification instructions, which have been issued for the footnote do not apply to the body of the document.

/4/ This is a longer footnote. Specific left and right margins have been defined for the footnote. Otherwise, the defaults .LM 0 and .RM 72 would have been used. Furthermore, the lines after the first line have been offset 4 columns to the right. To stop offsetting, the .OF instruction must be issued once more, this time without a parameter.

Number of Lines on a Page

.PL - Page Length

```
.PL number
```

This instruction defines the number of lines to be printed on a page. The initial setting is .PL 57.

The page length includes the top margin (.TM), top title (i.e. the number of lines which have been reserved with .HS), the header margin (.HM), the text and the footnotes.

The footer margin (.FM), bottom margin (.BM), and the number of lines which have been reserved for the bottom title (.FS) are *not* included in this number.

The current value for the page length is stored in the variable \$PL (see [Modifiable System Variables](#)).

To define a different page length, you must specify the number of lines on a page. For example, if you print on shorter paper and want to specify a page length of 40 lines, you must specify:

```
.PL 40
```

Definitions for the page length are only considered at the top of a page. Thus, a modified page length does not become active at the position where the .PL instruction occurs in the source text. The new page length is only activated after the actual page break has occurred. If you want to define the page length for the whole document, you must define it *before* the document text.

Line Spacing and Character Spacing

.LS - Line Spacing

```
.LS S  
.LS D  
.LS number
```

You can define the amount of space between the text lines in the formatted output.

Single Line Spacing

Initially, single line spacing is active. This corresponds to the following instruction:

```
.LS S
```

Double Line Spacing

To activate double line spacing, you must specify:

```
.LS D
```

Number of Lines per Inch

This feature must be used in conjunction with a Con-nect printer profile (see the *Con-nect User's Guide*, section *Printer Profiles*). You must include your line spacing definitions in the printer profile, so that the .LS instruction results in the desired output. For example, for the instruction .LS 060 you must specify the following symbol in your printer profile:

```
LI060
```

You can define the number of lines per inch that are to be printed. The initial setting for the .LS *number* instruction is 6 lines per inch which corresponds to the following:

```
.LS 060
```

The parameter must be exactly 3 digits long. Leading zeros as in `.LS 060` must always be specified.

The following example illustrates how to define the number of lines per inch:

```
.LS 030
Here, the line spacing is 3 lines per inch.
This has been achieved with the .LS 030 instruction.
.SL 1
.LS 040
Here, the line spacing is 4 lines per inch.
This has been achieved with the .LS 040 instruction.
.SL 1
.LS 060
Here, the line spacing is 6 lines per inch.
This has been achieved with the .LS 060 instruction.
.SL 1
.LS 080
Here, the line spacing is 8 lines per inch.
This has been achieved with the .LS 080 instruction.
.SL 1
.LS 120
Here, the line spacing is 12 lines per inch.
This has been achieved with the .LS 120 instruction.
```

Example

This example illustrates the instructions for single and double line spacing. It also shows how these instructions interact with the [.SL instruction](#).

Source Text

```
.LM 5;.RM 55
.FI ON;.JU ON
By default, single line space is active (.LS S). You can also specify
double line spacing (.LS D).
.SL 1
.LS D
For this paragraph double line spacing has been specified. With double
line spacing, one blank line is inserted between the text lines.
.SL 1
When you use the .SL instruction with double line spacing,
the space between the two paragraphs is increased, i.e. 3 blank lines
are inserted.
.LS S
.SL 1
```

Now single line spacing has been defined BEFORE the .SL instruction. Therefore, a single blank line is inserted before this paragraph.

Formatted Output

By default, single line space is active (.LS S). You can also specify double line spacing (.LS D).

For this paragraph double line spacing has been specified. With double line spacing, one blank line is inserted between the text lines.

When you use the .SL instruction with double line spacing, the space between the two paragraphs is increased, i.e. 3 blank lines are inserted.

Now single line spacing has been defined BEFORE the .SL instruction. Therefore, a single blank line is inserted before this paragraph.

.CS - Character Spacing

```
.CS number
```

This feature must be used in conjunction with a Con-nect printer profile (see the *Con-nect User's Guide*, section *Printer Profiles*). You must include your pitch definitions in your printer profile, so that the .CS instruction results in the desired output. For example, for the instruction .CS 100 you must specify the following symbol in your printer profile:

```
CI100
```

You can define the number of characters per inch (referred to as "pitch") that are to be printed in a line. The initial setting for the .CS instruction is 10 characters per inch which corresponds to the following:

```
.CS 100
```

The parameter must be exactly 3 digits long. Leading zeros as in .CS 060 must always be specified.

The following example illustrates how to define the number of characters per inch:

```
.CS 060
Here, the character spacing is 6 characters per inch.
This has been achieved with the .CS 060 instruction.
.SL 1
.CS 100
Here, the character spacing is 10 characters per inch.
This has been achieved with the .CS 100 instruction.
.SL 1
.CS 120
Here, the character spacing is 12 characters per inch.
This has been achieved with the .CS 120 instruction.
.SL 1
.CS 150
Here, the character spacing is 15 characters per inch.
This has been achieved with the .CS 150 instruction.
```

Line Spacing and Character Spacing at the Top and Bottom of a Page

If many .LS and .CS instructions are used in your text, it is recommended that you define separate .LS and .CS values for the top and bottom titles. Otherwise, it may happen that on the different pages the top and bottom titles have different line and character spacing.

Individual values for line spacing and character spacing can be defined for the following:

- top title (.TT, .ET and .OT) including blank lines (.TM und .HM)
- bottom title (.BT, .EB and .OB) including blank lines (.FM)

To do so, you specify the instructions .LS and .CS directly before the instructions that affect the top and bottom titles. For example:

```
.LS 030;.CS 060
.TT The top titles are printed with .LS 030 and .CS 060
.LS 040;.CS 100
.BT The bottom titles are printed with .LS 040 and .CS 100
The body of the document is also printed with .LS 040 and .CS 100.
.LS 060;.CS 120
Now the body of the document is printed with .LS 060 and .CS 120.
```

In the above example, different values are defined for the top title, the bottom title and the body of the document. The following example uses the same value for the top and bottom titles:

```
.LS 030;.CS 060
.TT The top titles are printed with .LS 030 and .CS 060
.BT The bottom titles are also printed with .LS 030 and .CS 060
.LS 060;.CS 120
The body of the document is printed with .LS 060 and .CS 120.
If you omit the instructions .LS 060 and .CS 120, the same
values are used in the top and bottom titles and in the
body of the document (.LS 030 and .CS 060).
```

Since the .LS instruction can cause the bottom lines to be positioned differently on different pages, an internal calculation is done, and if required, one blank line is printed. For this blank line the most fitting line spacing definition from the Con-nect printer profile is used. It is therefore recommended that you define as many line spacing definitions as possible in your printer profile (for example, LI060) so that the bottom lines can evenly be aligned on all pages.

See also: [Top Titles and Bottom Titles](#), [Blank Lines at the Top of the Page](#), and [Blank Lines at the Bottom of the Page](#).

Line Spacing and Character Spacing in Header Lines

For header lines, separate values for line spacing and character spacing can be defined in the `.HL` instruction. For example:

```
.HL .LS 060  
.HL .CS 120  
.HL This is the page header.
```

If you do not define line and character spacing with the `.HL` instruction, the values currently defined for the body of the document are used. It is recommended that you define the line spacing with the `.HL` instruction. Otherwise, the layout of the headers may differ on different pages.

For detailed information on the `.HL` instruction, see [.HL - Header Lines](#).

Line Spacing and Character Spacing in Footnotes

For footnotes, separate values for line and character spacing can be defined. To do so, you must place the `.LS` and `.CS` instructions in a separate footnote block as shown in the example below:

```
.FN ON  
.LS 060  
.CS 120  
.FN OFF
```

The values defined in the above footnote block apply to *all* footnotes that follow in your source text. Therefore, this block must occur before the first footnote that is to be output.

If you want to use other values for the next footnote, you have to place another footnote block before this footnote. The values defined in this footnote block are then valid for all following footnotes.



Note: The concept of a separate footnote block for line and character spacing is an exception. To define a different layout for the footnotes (for example, left and right margins), you must still define it separately for each footnote.

For detailed information on footnotes, see [.FN - Footnote](#).

4 Positioning Text

- Text Alignment 87
- Indenting Text 93
- Page Breaks 102
- Availability of Remaining Lines on a Page 105

This chapter covers the following topics:

Text Alignment

In addition to a jagged edge (see [.FI - Filling](#)) or a left and right justification (see [.JU - Justification](#)) you can also center or right-adjust your text.

.CE - Centered Text

```
.CE number  
.CE ON  
.CE OFF
```

This instruction is used to center a line of text between the left and right margins. As long as centering is in effect, the instructions for filling (.FI) and justification (.JU) are not carried out.

Centering can be combined with, for example, underlining (see [.UL - Underline](#)) or boldface printing (see [.BF - Boldface](#)).

There are two different ways to use the .CE instruction.

Defining the Number of Lines to be Centered

You can specify the number of lines to be centered. For example, to center the next two lines, you specify:

```
.CE 2
```

Centering a Block of Text

You can also center a block of text. To do so, you must enclose the text to be centered between the following two instructions:

```
.CE ON
```

```
.CE OFF
```

Example

This example illustrates how to center text. Furthermore, it introduces the instructions **.BP** and **.UL**.

Source Text

```
.LM 0;.RM 60
.FI ON;.JU ON
.CE 3
This is centered text. Since filling
and justification do not apply, each line of the
source text is output in a separate line.
.SL 1
In the above example, three lines of the source text have been
centered. All following text is adjusted between the left and
right margins as specified with the .LM and .RM instructions.
.SL 1
.CE ON
.** This is an example for using bold print with centered text.
.BP
You can also center a block of text.
To do so, the text must be enclosed between
the instructions .CE ON and .CE OFF.
.SL 1
You must always take care
that each line to be centered
fits between the specified left and right margins.
.** This is an example for underlining centered text. The
.** following line is to be underlined. Note that the .UL
.** instruction must be entered AFTER the text to be underlined.
Otherwise the line cannot be centered correctly.
.UL *
.CE OFF
.SL 1
This is text which follows the centered text block. It is
adjusted between the left and right margins as specified with
the .LM and .RM instructions.
```

Formatted Output

This is centered text. Since filling
and justification do not apply, each line of the
source text is output in a separate line.

In the above example, three lines of the source text have
been centered. All following text is adjusted between the
left and right margins as specified with the .LM and .RM
instructions.

You can also center a block of text.

To do so, the text must be enclosed between
the instructions .CE ON and .CE OFF.

You must always take care
that each line to be centered
fits between the specified left and right margins.
Otherwise the line cannot be centered correctly.

This is text which follows the centered text block. It is
adjusted between the left and right margins as specified
with the .LM and .RM instructions.

.RA - Right-Adjusted Text

```
.RA number  
.RA ON  
.RA OFF
```

This instruction is used to align text lines with a straight right margin; the left margin will appear ragged. As long as right-adjustment is in effect, the instructions for filling (.FI) and justification (.JU) are not carried out.

Right-adjustment can be combined with, for example, underlining (see [.UL - Underline](#)) or boldface printing (see [.BF - Boldface](#)).

There are two different ways to use the .RA instruction.

Moving Text to the Defined Right Margin

You can right-adjust your text so that it is moved to the right margin which has been defined with the .RM instruction. To do so, you must enclose the text to be right-adjusted between the following two instructions:

```
.RA ON
```

```
.RA OFF
```

Defining a Different Right Margin

You can also specify a different right margin for the right-adjusted text. For example, if the right margin is to fall in column 50, you must enclose the text to be right-adjusted between the following two instructions:

```
.RA 50
```

```
.RA OFF
```

Example

This example illustrates how to right-adjust text. Furthermore, it introduces the instructions **.BF** and **.US**. When you compare this example with the previous [example for centering text](#), you will see that different instructions have been used there (**.BP** and **.UL**).

Source Text

```
.LM 0;.RM 60
.FI ON;.JU ON
.RA ON
.** This is an example for using bold print with right-adjusted text.
.BF 1
This is right-adjusted text.
.BF
Since filling and justification do not apply,
each line of the source text is output
in a separate line.
.RA OFF
.SL 1
All text following the above text is adjusted between the left and
right margins as specified with the .LM and .RM instructions.
.SL 1
.RA 40
The right margin for this part
of the right-adjusted text
has been defined in
.** This is an example for underscoring right-adjusted text.
.** Unlike, the .UL instruction, the .US instruction which has been
.** used in this example, must be entered BEFORE the text to be
.** right-adjusted.
.US
column 40.
.RA OFF
.SL 1
All text following the above instruction will again be adjusted
between the defined left and right margins.
```

Formatted Output

```

                This is right-adjusted text.
            Since filling and justification do not apply,
                each line of the source text is output
                    in a separate line.

All text following the above text is adjusted between the
left and right margins as specified with the .LM and .RM
instructions.
```

The right margin for this part
of the right-adjusted text
has been defined in
column 40.

All text following the above instruction will again be
adjusted between the defined left and right margins.

Indenting Text

You can indent the first line of a paragraph so that the left margin of the first line is at a different position than the left margin of the remainder of the paragraph.

You must not use both the `.TI` instruction and the [.OF instruction](#) in the same paragraph.

.TI - Temporary Indentation

```
.TI number  
.TI +number  
.TI -number
```

The `.TI` instruction sets the left margin for the first line. All following lines use the left margin which has been defined with the `.LM` instruction. The `.TI` instruction causes a break in filling.

The current indentation is stored in the variable `$IN` (see [Modifiable System Variables](#)).

There are three different ways to use the `.TI` instruction.

Defining a Specific Column

You can define the column in which the first line is to start. For example, if the first line is to start in column 5 (and all other lines start in column 0), you must specify:

```
.TI 5
```

Moving to the Right

For example, the current left margin is in column 10 - if you want to move the first line 3 columns to the right so that it starts in column 13, you must specify:

```
.TI +3
```

Moving to the Left

For example, the current left margin is in column 10 - if you want to move the first line 4 columns to the left so that it starts in column 6, you must specify:

```
.TI -4
```

You cannot define a negative left margin. Therefore, you must take care not to specify a column smaller than 0.

Example

This example illustrates the `.TI` instruction which is used to indent the first line of a paragraph.

Source Text

```
.LM 5;.RM 60
.FI ON;.JU ON
.TI 20
The first line of this paragraph starts in column 20. All following lines
start at the left margin. Since the left margin starts in column 5,
the first line is indented 15 characters to the right so that it starts
in column 20.
.IL 1
.TI +15
This first line in this example starts in the same column as the first
line in the previous example.
.IL 1
.TI -3
When the left margin is greater than 0, you can also indent the first line
to the left. However, the value must not be greater than the left margin
value, since you cannot indent text further left than the left margin.
```

Formatted Output

The first line of this paragraph starts in column 20. All following lines start at the left margin. Since the left margin starts in column 5, the first line is indented 15 characters to the right so that it starts in column 20.

This first line in this example starts in the same column as the first line in the previous example.

When the left margin is greater than 0, you can also indent the first line to the left. However, the value must not be greater than the left margin value, since you cannot indent text further left than the left margin.

.OF - Offsetting

```
.OF number  
.OF +number  
.OF -number  
.OF
```

The first line of the paragraph uses the left margin which has been defined with the .LM instruction. The .OF instruction sets the left margin for all following lines; this instruction causes a break in filling.

The current indentation is stored in the variable \$IN (see [Modifiable System Variables](#)).

There are four different ways to use the .OF instruction.

Defining a Specific Column

You can define the column in which all lines after the first line are to start. For example, if the first line starts in column 0 and all following lines are to start in column 5, you must specify:

```
.OF 5
```

Moving to the Right

For example, the first line starts in column 10 - if you want to move all following lines 6 columns to the right so that they start in column 16, you must specify:

```
.OF +6
```

Moving to the Left

For example, the first line starts in column 10 - if you want to move all following lines 4 columns to the left so that they start in column 6, you must specify:

```
.OF -4
```

You cannot define a negative left margin. Therefore, you must take care not to specify a column smaller than 0.

Canceling the .OF instruction

Each .OF instruction (as described above) defines a new setting. Thus, the previous .OF instruction is automatically canceled.

If you want to cancel the effects of the .OF instruction so that the left margin which has been defined with the .LM instruction will be in effect again, you must specify .OF without a parameter:

```
.OF
```

You can also cancel the effects of the .OF instruction by defining a new left margin using the .LM instruction.

Example

This example illustrates the `.OF` instruction which is used to indent all lines of a paragraph, except for the first line.

Source Text

```
.LM 5;.RM 60
.FI ON;.JU ON
.OF 9
1. You can use the .OF instruction, for example, to create a
numbered list.
.SL 1;.OF 9
2. In contrast to the .TI instruction, the .OF instruction applies to all
lines after the first line.
.SL 1;.OF 9
3. The first line is not affected by the .OF instruction.
It starts at the left margin.
.SL 1;.OF +4
4. The lines in this paragraph (except for the first line) start in the
same column as the lines in the previous example.
.SL 1;.OF -5
5. You can also specify negative offset values. In this example, the
first line starts at the left margin (which is in column 5), and the
remaining lines start in column 0.
.SL 1;.OF
Issuing .OF without a parameter switches offsetting off and all lines
begin at the left margin which has been defined with the .LM instruction.
```

Formatted Output

```
1. You can use the .OF instruction, for example, to
create a numbered list.

2. In contrast to the .TI instruction, the .OF
instruction applies to all lines after the first
line.

3. The first line is not affected by the .OF
instruction. It starts at the left margin.

4. The lines in this paragraph (except for the first
line) start in the same column as the lines in the
previous example.

5. You can also specify negative offset values. In
this example, the first line starts at the left margin
(which is in column 5), and the remaining lines start in
column 0.
```

Issuing `.OF` without a parameter switches offsetting off and all lines begin at the left margin which has been defined with the `.LM` instruction.

.PS - Paragraph Start

.PS

The .PS instruction can be used for creating paragraphs. The first line of each paragraph is automatically indented three columns to the right.

The .PS instruction automatically executes the following four instructions:

.BR

Causes a break in filling.

.NL 4

If less than 4 lines are available on the current page, the new paragraph starts on a new page. See [.NL - Need Lines](#) for further information.

.SL

The .SL instruction interacts with line spacing.

When you use single line spacing, one blank line is inserted between the previous and the next paragraphs (unless a page break occurs).

When you use double line spacing, three blank lines are inserted (unless a page break occurs).

.TI +3

The first line of the new paragraph is indented three columns to the right.

Example

This example illustrates the .PS instruction and shows how it interacts with double line spacing.

Source Text

```
.LM 5;.RM 55  
.FI ON;.JU ON  
.LS D;.PS  
For this paragraph double line spacing has been specified. With double  
line spacing, one blank line is inserted between the text lines.  
.PS  
When you use the .PS instruction with double line spacing,  
three blank lines (instead of one blank line as with single line  
spacing) will be inserted between the paragraphs.
```

Formatted Output

For this paragraph double line spacing has been
specified. With double line spacing, one blank
line is inserted between the text lines.

When you use the .PS instruction with double
line spacing, three blank lines (instead of one
blank line as with single line spacing) will be
inserted between the paragraphs.

Page Breaks

You can use the following instructions to control the page breaks. If you do not specify page breaks, the initial values for the document layout apply and the form feed occurs after the bottom title. Or, if no bottom title has been defined, the form feed occurs after the footer space (i.e. the blank line which has initially been reserved for the bottom title).

.NP - New Page

```
.NP number  
.NP
```

The following instruction causes a form feed.

```
.NP
```

Before the form feed occurs, Con-form prints the defined bottom title and footnotes. All text after the .NP instruction starts on the next page.

A form feed does not occur, when the text to be printed has already been positioned at the top of a page. For example, when you specify .NP at the very beginning of your text, a form feed does not occur.

You can also specify a new page number for the next page. For example, if you want to assign the number 6 to the next page, you must specify:

```
.NP 6
```

You cannot produce more than one form feed using this instruction. If you want to insert several blank pages in your text, you must use the .IP instruction.

The current page number is stored in the variable \$PN (see [Modifiable System Variables](#)).

.IP - Insert Pages

```
.IP number  
.IP
```

This instruction inserts the specified number of blank pages in the output text. If top and bottom titles and a page header have been defined, this information is also printed on the "blank" pages.

For example, if you want to insert 3 blank pages, you must specify:

```
.IP 3
```

If the above instruction is processed in the middle of a page, the remainder of the page is left blank, and the specified number of blank pages is inserted. The text which has been defined after the .IP instruction is printed at the top of the page which is output after the defined number of blank pages.

When you use this instruction without a parameter, one blank page is inserted.

.OP PGF - Page Formatting

```
.OP PGF=ON  
.OP PGF=OFF
```

You can switch page formatting off, so that page breaks are not considered. In this case, only the body of the document is printed. Instead of starting text on a new page, it is started in a new line. This can be useful, for example, when you only want to print the pure text for proofreading.

When page formatting is switched off, top and bottom titles, page headers and footnotes are not printed. Furthermore, entries in the table of contents, index entries and text blocks which have been defined with .NL KEEP and .NL FLOAT are not considered.

However, when page formatting is switched off, the document layout is still influenced by the instructions which have been defined, for example, for the left and right margins, filling and justification, centered text and blank lines.

To switch page formatting off for the whole document, you must specify the following instruction at the beginning of the document:

```
.OP PGF=OFF
```

Initially, page formatting is switched on. This corresponds to the following:

```
.OP PGF=ON
```

Availability of Remaining Lines on a Page

.NL - Need Lines

```
.NL number  
.NL  
.NL KEEP  
.NL FLOAT  
.NL OFF
```

The `.NL` instruction is useful if you want to ensure that a paragraph does not start close to the bottom of a page. It is also useful if you want to leave space for a figure to be inserted, without running the risk that the space is split between two successive pages.

This instruction finds out whether the required number of lines is available on the current output page. If less than the required number of lines is available, the text is output on a new page. If the required number of lines is available, this instruction merely causes a break in filling, i.e. the text starts on a new line.

The number of the remaining lines on the page is stored in the variable `$LC` (see [Modifiable System Variables](#)).

The different forms of the `.NL` instruction are described below.

.NL number

For example, if you want to make sure that there are still 8 lines left on the current page, so that your information will fit as a whole on the page, you must specify:

```
.NL 8
```

If 8 lines are available, the text starts on a new line of the current page. If 8 lines are not available, the text is output on the next page.

If you want to reserve blank lines for a diagram that will be added later to the printed page, you must also specify the `.IL` instruction as in the following example:

```
.NL 10;.IL 10
```

Using the .NL instruction without a parameter is equivalent to .NL 7.

.NL OFF

This instruction is only used with the instructions .NL KEEP and .NL FLOAT which are explained on the following pages. It defines the end of the text block.

It is not possible to conditionally suppress the processing of the .NL OFF instruction by using it in an **.IF construction**.

.NL KEEP

You can enclose a text block between the following two instructions:

```
.NL KEEP
```

```
.NL OFF
```

Con-form then checks automatically whether the required number of lines for this text block is available on the current page. If the required number of lines is available, the text block is printed on the current page. If not, it is printed on the next page.

Example

This example illustrates how to reserve blank lines for a diagram, and it shows the results of the .NL KEEP instruction.

Source Text

```
.TT Con-form Reference Manual//The KEEP Buffer
.BT //Page #
.LM 10;.RM 60
.FI ON;.JU ON
You can reserve blank lines for a diagram that will later be added
to the printed page. On this page 30 blank lines have been
reserved for the diagram.
.IL 1
In the case that less than 30 lines are available on this page,
the blank lines will automatically be inserted at the top of the
next page.
.NL 30;.IL 30
All blank lines above this line have been reserved for a diagram.
If the text in the following KEEP buffer is longer than can fit on
this page, it is automatically printed on the next page. Otherwise,
it is printed on the current page.
.IL 2
.NL KEEP
The .NL KEEP Instruction
.SL 1
You can use the .NL KEEP instruction to store a section of text in a
buffer. If the number of lines required for the text in the buffer is
available on the current page, the text is printed on that page.
.SL 1
If the number of lines required for the text in the buffer is not
available on the current page, the text is automatically printed
on the next page.
.NL OFF
```

Formatted Output - First Page

Con-form Reference Manual

The KEEP Buffer

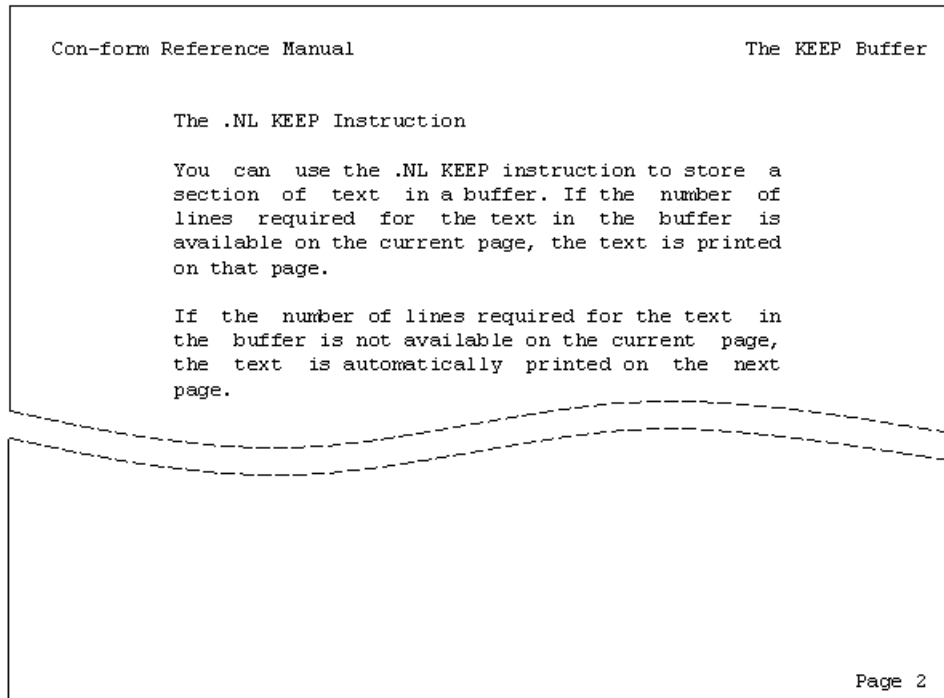
You can reserve blank lines for a diagram that will later be added to the printed page. On this page 30 blank lines have been reserved for the diagram.

In the case that less than 30 lines are available on this page, the blank lines will automatically be inserted at the top of the next page.

All blank lines above this line have been reserved for a diagram. If the text in the following KEEP buffer is longer than can fit on this page, it is automatically printed on the next page. Otherwise, it is printed on the current page.

Page 1

Formatted Output - Second Page



.NL FLOAT

You can enclose a text block between the following two instructions:

```
.NL FLOAT
```

```
.NL OFF
```

Con-form then checks automatically whether the required number of lines for this text block is available on the current page.

When the lines are available, the text block is printed immediately.

When the required number of lines is not available on the current page, `.NL FLOAT`, unlike the `.NL KEEP` instruction, processes text which follows the `.NL OFF` instruction until the current page is full. The defined text block is then printed on the next page.

The `.NL FLOAT` instruction is appropriate, when it does not matter whether the text block which has been enclosed between the above instructions is placed on the current or on the next page.

Example

This example illustrates how to reserve blank lines for a diagram, and it shows the results of the `.NL FLOAT` instruction.

Source Text

```
.TT Con-form Reference Manual//The FLOAT Buffer
.BT //Page #
.LM 10;.RM 60
.FI ON;.JU ON
You can reserve blank lines for a diagram that will later be added
to the printed page. On this page, 30 blank lines have been reserved.
.NL 30;.IL 30
All blank lines above this line have been reserved for a diagram.
.IL 2
.NL FLOAT
The .NL FLOAT Instruction
.SL 1
You can use the .NL FLOAT instruction to store a section of text in a
buffer. If the number of lines required for the text in the buffer is
available on the current page, the text is printed on that page.
.SL 1
If the number of lines required for the text in the buffer is not
available on the current page, the current page will be filled with
the text which follows the .NL OFF instruction. The text in the
```

buffer will then be printed at the top of the next page.

```
.SL 1
```

The `.NL FLOAT` instruction is appropriate, for example, when the position of the text in the buffer does not matter.

```
.SL 1
```

```
.NL OFF
```

This is text which has been defined after the `.NL FLOAT` instruction in the source text. It will be printed before the text in the `FLOAT` buffer if the current page has not enough lines for the text in the buffer.

Formatted Output - First Page

Con-form Reference Manual	The FLOAT Buffer
<p>You can reserve blank lines for a diagram that will later be added to the printed page. On this page, 30 blank lines have been reserved.</p>	
<p>All blank lines above this line have been reserved for a diagram.</p>	
<p>This is text which has been defined after the .NL OFF instruction in the source text. It will be printed before the text in the FLOAT buffer if the current page has not enough lines for the text in the buffer.</p>	
<p>Page 1</p>	

Formatted Output - Second Page

Con-form Reference Manual	The FLOAT Buffer
<p>The .NL FLOAT Instruction</p>	
<p>You can use the .NL FLOAT instruction to store a section of text in a buffer. If the number of lines required for the text in the buffer is available on the current page, the text is printed on that page.</p>	
<p>If the number of lines required for the text in the buffer is not available on the current page, the current page will be filled with the text which follows the .NL OFF instruction. The text in the buffer will then be printed at the top of the next page.</p>	
<p>The .NL FLOAT instruction is appropriate, for example, when the position of the text in the buffer does not matter.</p>	
<p>Page 2</p>	

.OP BRN - Suppress Line Breaks Caused by .NL

```
.OP BRN=ON  
.OP BRN=OFF
```

As a rule, the .NL instruction causes a break in line filling when the required number of lines is available on the current page. However, you can switch this feature off, so that the filling process is not interrupted.

To suppress the breaks in line filling, you must specify:

```
.OP BRN=OFF
```

The initial value of the option is:

```
.OP BRN=ON
```

Example

This example illustrates the results of the .OP BRN=OFF instruction.

Source Text

```
.LM 0;.RM 60  
.FI ON;.JU ON  
.OP BRN=OFF  
As a rule, the .NL instruction causes a break in line filling.  
However, you can switch this feature off.  
.SL 1  
In this example, the BRN option has been switched off. You can  
check whether the required number of lines is available on the  
current page.  
.NL 8  
If the required number of lines is available, the filling process  
is NOT interrupted.
```

Formatted Output

As a rule, the .NL instruction causes a break in line filling. However, you can switch this feature off.

In this example, the BRN option has been switched off. You can check whether the required number of lines is available on the current page. If the required number of lines is available, the filling process is NOT interrupted.

5 Emphasizing Text

- Boldface 118
- Underlining and Underscoring 122
- Italic Text 127
- Superscript Text 128
- Subscript Text 129
- Backspacing 130
- Escape Sequences 131

This chapter covers the following topics:

The symbols which are used in escape sequences represent printer instructions which have previously been defined in conjunction with the appropriate printer manual. The following features can only be defined using escape sequences: italic, superscript, subscript and backspacing.

Boldface

There are two instructions that can be used for boldface printing: `.BF` and `.BP`. In addition to these instructions, you can also use escape sequences to invoke boldface printing.

Both instructions `.BF` and `.BP` can be combined with centering (see [.CE - Centered Text](#)), right-adjustment (see [.RA - Right-Adjusted Text](#)) and with the [.UL instruction](#). Boldface printing can only be combined with underscoring when you use escape sequences.

.BF - Boldface

```
.BF number  
.BF
```

The `.BF` instruction prints all text which follows the instruction in boldface until `.BF` is issued once more.

As a rule, the `.BF` instruction should be used when you want to print large sections of text in boldface. The `.BF` instruction causes a break in filling (i.e. the subsequent text starts in a new line). Thus, it cannot be used to print single words in a line in boldface.

To switch boldface printing on, you must specify:

```
.BF 1
```



Note: The value after the `.BF` instruction must be any number greater than zero.

All text that follows the `.BF` instruction is printed in boldface.

To switch boldface printing off, you must issue the `.BF` instruction once more, this time without a parameter:

```
.BF
```

.BP - Bold Print

```
.BP
```

The .BP instruction prints only the text which follows in the next line of the source text in boldface.

As a rule, the .BP instruction should be used when you want to print single words within a line in boldface. The .BP instruction does not cause a break in filling. However, you cannot emphasize single letters within a word (this can only be achieved using escape sequences).

To print the text line after the instruction in boldface, you must specify:

```
.BP
```

In contrast to the .BF instruction, bold printing need not be switched off.

Using Escape Sequences for Boldface Printing

When you use escape sequences, you can also print a part of a word in boldface and combine boldface printing with underscoring (see [Using Escape Sequences for Underscoring Text](#)). Escape sequences do not cause a break in line filling.

Before you can issue an escape sequence, you must define the escape character (see [.OP ESC - Escape Character](#)).

To print part of a word in boldface, the following symbols must be included in the source text:

M1	Switches boldface printing on.
M0	Switches boldface printing off.

The following example shows how to print part of a word in boldface:

```
.OP ESC=/
This should be printed in /M1bold/M0face.
```

The above instructions cause the following formatted output:

This should be printed in **boldface**.

Example

This example illustrates the instructions `.BF` and `.BP`. It also shows how to use escape sequences.

Source Text

```
.LM 0;.RM 60
.FI ON;.JU ON
.BF 1
All text in this paragraph is printed in boldface. To switch boldface
printing off, the .BF instruction must be issued once more, this time
without a parameter.
.BF
.SL 1
In this sentence, a
.BP
single
word is printed in boldface. The word to be printed in boldface must
immediately follow the .BP instruction and it must be entered in a
.BP
separate line.
.SL 1
If you want to use escape sequences, you must first define the escape
character.
.OP ESC=/
In this example, the escape character is the slash.
You can now print /M1several words/M0 or a part of a word in
/M1bold/M0face.
.SL 1
.** The following instruction cancels the escape character.
.OP ESC=NULL
If you want to include the slash (/) in the formatted output, you must
cancel the escape character or define another escape character.
```

Formatted Output

All text in this paragraph is printed in boldface. To switch boldface printing off, the `.BF` instruction must be issued once more, this time without a parameter.

In this sentence, a **single** word is printed in boldface. The word to be printed in boldface must immediately follow the `.BP` instruction and it must be entered in a **separate line**.

If you want to use escape sequences, you must first define the escape character. In this example, the escape character

is the slash. You can now print **several words** or a part of a word in **boldface**.

If you want to include the slash (/) in the formatted output, you must cancel the escape character or define another escape character.

Underlining and Underscoring

There are two instructions that can be used to underline or underscore text: `.UL` and `.US`. In addition to these instructions, you can also use escape sequences to invoke underscoring.

`.UL` must be entered *after* the text to be underlined, whereas `.US` must be entered *before* the text to be underscored.

.UL - Underline

```
.UL character  
.UL
```

The `.UL` instruction always underlines the previous line of the source text. If you do not specify an underline character, the hyphen (-) is used. The underline character occupies an extra output line.

The `.UL` instruction causes a break in filling (i.e. the subsequent text starts in a new line). Thus, it cannot be used to underline single words within a line.

You must specify the `.UL` instruction *after* the text to be underlined, for example:

```
This text is to be underlined with the hyphen character.  
.UL
```

You can define another underline character. For example, to define the asterisk (*) as the underline character, you specify:

```
.UL *
```

Underlining can be combined with centering (see [.CE - Centered Text](#)), right-adjustment (see [.RA - Right-Adjusted Text](#)) and boldface printing (see [.BF - Boldface](#)).

.US - Underscore

```
.US
```

The .US instruction underscores only the text which follows in the next line of the source text. In contrast to the .UL instruction, you cannot define another underline character. Your text is always underscored with a solid line. Underscoring does not produce an extra output line.

The .US instruction does not cause a break in filling. Thus, you can underscore single words in a sentence. However, you cannot underscore single letters within a word (this can be achieved using escape sequences).

You must specify the .US instruction *before* the text to be underscored, for example:

```
.US
This line is to be underscored.
```

Underscoring can be combined with centering (see [.CE - Centered Text](#)) and right-adjustment (see [.RA - Right-Adjusted Text](#)). Underscoring can only be combined with boldface printing when you use escape sequences.

.OP ULB - Underscore Blanks

```
.OP ULB=ON
.OP ULB=OFF
```

When you use the .US instruction, all blanks in the formatted output are underscored. This corresponds to the initial setting:

```
.OP ULB=ON
```

To switch this option off so that blanks are not underscored, you must specify:

```
.OP ULB=OFF
```



Caution: The instruction .OP ULB=OFF is only recognized when filling is switched on (.FI ON).

Using Escape Sequences for Underscoring Text

When you use escape sequences, you can also underscore a part of a word and combine underscoring with boldface printing. Escape sequences do not cause a break in line filling.

Before you can issue an escape sequence, you must define the escape character (see [.OP ESC - Escape Character](#)).

To underscore part of a word, the following symbols must be included in the source text:

U1	Switches underscoring on.
U0	Switches underscoring off.

The following example shows how to underscore part of a word:

```
.OP ESC=/  
You can /U1under/U0score a part of a word.
```

The above instructions cause the following formatted output:

```
You can underscore a part of a word.
```

When you use escape sequences, you can combine underscoring with boldface printing. For example:

```
.OP ESC=/  
You can /M1/U1combine/U0/M0 underscoring with boldface printing.
```

The above instructions cause the following formatted output:

```
You can combine underscoring with boldface printing.
```

Example

This example illustrates the instructions `.UL` and `.US`. It also shows how to use escape sequences.

Source Text

```
.LM 0;.RM 60
.FI ON;.JU ON
This line is underlined with an equal sign (=).
.UL =
The .UL instruction has been entered after the line to be underlined.
The .UL instruction causes a break in filling.
.SL 1
In this sentence, a
.US
single
word is underscored. The word(s) to be underscored must immediately
follow the .US instruction and must be entered in a
.US
separate line.
.SL 1
If you want to use escape sequences, you must first define the escape
character.
.OP ESC=/
In this example, the escape character is the slash.
You can now underscore /Ulseveral words/U0 or p/Ular/U0t of a word.
.SL 1
You can also /M1/Ulcombine/U0/M0 underscoring with boldface printing.
.** The following instruction cancels the escape character.
.OP ESC=NULL
.SL 1
If you want to include the slash (/) in the formatted output, you must
cancel the escape character or define another escape character.
```

Formatted Output

```
This line is underlined with an equal sign (=).
=====
The .UL instruction has been entered after the line to be
underlined. The .UL instruction causes a break in filling.

In this sentence, a single word is underscored. The word(s)
to be underscored must immediately follow the .US
instruction and must be entered in a separate line.

If you want to use escape sequences, you must first define
the escape character. In this example, the escape character
is the slash. You can now underscore several words or part
of a word.

You can also combine underscoring with boldface printing.

If you want to include the slash (/) in the formatted
```

Emphasizing Text

output, you must cancel the escape character or define another escape character.

Italic Text

Italic text can only be defined using escape sequences.

Before you can issue an escape sequence, you must define the escape character (see [.OP ESC - Escape Character](#)).

To print italic text, the following symbols must be included in the source text:

I1	Switches italic printing on.
I0	Switches italic printing off.

The following example shows how to print a word in italics:

```
.OP ESC=/
This should be printed in /I1italics/I0.
```

The above instructions cause the following formatted output:

```
This should be printed in italics.
```

You can also print part of a word or several words in italics.

You can also combine italic text with underscoring. For example:

```
.OP ESC=/
This should be printed in /I1/U1underscored italics/U0/I0.
```

The above instructions cause the following formatted output:

```
This should be printed in underscored italics.
```

Superscript Text

Superscript text can only be defined using escape sequences.

Before you can issue an escape sequence, you must define the escape character (see [.OP ESC - Escape Character](#)).

To print superscript text, the following symbols must be included in the source text:

E1	Switches superscript printing on.
E0	Switches superscript printing off.

The following example shows how to print part of a word in superscript:

```
.OP ESC=/  
This is printed in /E1super/E0script.
```

The above instructions cause the following formatted output:

```
This is printed in superscript.
```

You can also combine superscript text with bold text, italics and underscoring. For example:

```
.OP ESC=/  
This is printed in bold /M1/E1super/E0/M0script.
```

The above instructions cause the following formatted output:

```
This is printed in bold superscript.
```

Subscript Text

Subscript text can only be defined using escape sequences.

Before you can issue an escape sequence, you must define the escape character (see [.OP ESC - Escape Character](#)).

To print subscript text, the following symbols must be included in the source text:

S1	Switches subscript printing on.
S0	Switches subscript printing off.

The following example shows how to print part of a word in subscript:

```
.OP ESC=/  
This is printed in /S1sub/S0script.
```

The above instructions cause the following formatted output:

```
This is printed in subscript.
```

You can also combine subscript text with bold text and italics. For example:

```
.OP ESC=/  
This is printed in italic /I1/S1sub/S0/I0script.
```

The above instructions cause the following formatted output:

```
This is printed in italic subscript.
```

Backspacing

Backspacing can only be defined using escape sequences.

Backspacing is helpful, when you want to print two characters at the same position (for example, the Danish character ø).

Before you can issue an escape sequence, you must define the escape character (see [.OP ESC - Escape Character](#)).

To use backspacing, the following symbol must be included in the source text:

B	Goes back one character.
---	--------------------------

The following example shows how to use backspacing to create the Danish character ø (since you use the slash (/) within your source text, you must not define it as the escape character):

```
.OP ESC=#  
The price is 10 Danish o#B/re.
```

The above instructions cause the following formatted printout:

```
The price is 10 Danish øre.
```

Escape Sequences

Escape sequences are helpful when you want to define italic text, subscript text, superscript text and backspacing, or if you want to combine, for example, boldface printing with underscoring, since this cannot be achieved using the Con-form instructions.

You can also define hyphenation points using escape sequences (see [Defining Hyphenation Points](#)).

The symbols which are used in the escape sequences have already been described in the previous sections. This section provides general information on how to use escape sequences. You can define the following symbols in your source text:

B	Goes back one character.
E1	Switches superscript printing on.
E0	Switches superscript printing off.
I1	Switches italic printing on.
I0	Switches italic printing off.
M1	Switches boldface printing on.
M0	Switches boldface printing off.
U1	Switches underscoring on.
U0	Switches underscoring off.
S1	Switches subscript printing on.
S0	Switches subscript printing off.

For each symbol, you must define the appropriate printer command sequence (which is documented in your printer manual) in your printer profile. See the *Con-nect User's Guide*, section *Printer Profiles*.

You can also define the following symbols in your source text (see [Using Escape Sequences to Alter the Text Orientation](#)):

A1	Switches left-to-right orientation on.
A0	Switches left-to-right orientation off.
Z1	Switches right-to-left orientation on.
Z0	Switches right-to-left orientation off.

.OP ESC - Escape Character

```
.OP ESC=character  
.OP ESC=NULL
```

Before you can issue an escape sequence, you must define the escape character. For example, to specify the slash (/) as the escape character, you must specify:

```
.OP ESC=/  

```

When you define the slash as the escape character you cannot use it as part of the normal text, since Con-form always expects an escape sequence with it.

Canceling the Escape Character

If you want to include the slash (which has been defined as the escape character with the above instruction) in your formatted output, you must either define another escape character or cancel the effects of the escape character using the following instruction:

```
.OP ESC=NULL
```

Defining Your Own Symbols

You can define your own symbols. You must also define them in your printer profile. See the *Connect User's Guide*, section *Printer Profiles*.

Your symbols must be enclosed either in parentheses or in apostrophes.

For example, you have defined a symbol called "Courier" which invokes this font. When you want to use parentheses, you include the following instructions in your source text:

```
.OP ESC=/  
/(COURIER)
```

When you want to use apostrophes, you include the following instructions in your source text:

```
.OP ESC=/  
/"COURIER"
```


Example

This example illustrates how to use escape sequences.

Source Text

```
.LM 0;.RM 60
.FI ON;.JU ON
.OP ESC=/
You can use escape sequences to emphasize a part of a word or a
longer passage of text, for example, to define
/M1bold/M0, /U1underscored/U0,
/I1italic/I0, /S1sub/S0script and /E1super/E0script text.
You can combine bold text with /M1/U1underscoring/U0/M0,
or italic text with /I1/U1underscoring/U0/I0.
.SL 1
/I1You can output longer passages of text in italics and
combine it with /S1sub/S0script or /E1super/E0script text./I0
.SL 1
/M1You can output longer passages of text in boldface
and combine it with /S1sub/S0script or /E1super/E0script text./M0
.SL 1
.OP ESC=#
You can use backspacing, for example, to output the Danish word
"o#B/re" in your printed document. When you want to output the
slash (/), you must make sure that it is not defined as the current
escape character.
.SL 1
.** The escape character is still the hash (#)
.CE ON
You can output #U1centered#U0 text.
.CE OFF;.SL 1
.RA ON
You can output #M1right#M0-adjusted text.
.RA OFF
```

Formatted Output

You can use escape sequences to emphasize a part of a word or a longer passage of text, for example, to define **bold**, underscored, *italic*, _{sub}script and ^{super}script text. You can combine bold text with **underscoring**, or italic text with *underscoring*.

You can output longer passages of text in italics and combine it with _{sub}script or ^{super}script text.

You can output longer passages of text in boldface and combine it with **_{sub}script** or **^{super}script** text.

You can use backspacing, for example, to output the Danish word "øre" in your printed document. When you want to output the slash (/), you must make sure that it is not defined as the current escape character.

You can output centered text.

You can output **right**-adjusted text.

6 Hyphenation

- Hyphenation in the Formatted Output 137
- Hyphenated Word in the Source Text 143

This chapter covers the following topics:

In justified text (.FI ON;JU ON), hyphenation reduces the number of spaces which have to be inserted in the formatted output.

When you arrange your text with a jagged edge (.FI ON;JU OFF), hyphenation reduces the raggedness of the right margin.

The hyphenation option is available for the following languages: English, French and German.

Hyphenation in the Formatted Output

The following options only affect words at the end of an output line. Hyphenation works only when filling has been switched on (.FI ON).

.OP HYP - Hyphenation

```
.OP HYP=language-code  
.OP HYP=OFF
```

Initially, hyphenation is switched off.

To activate hyphenation, you must define the language code for the desired language. The following language codes are available:

E	English
F	French
G	German (old orthography rules)
D	German (new orthography rules)

For example, to define hyphenation for the English language, you specify:

```
.OP HYP=E
```

When hyphenation is switched on, long words at the end of an output line are hyphenated according to the grammatical rules for the chosen language.

To switch hyphenation off, you specify:

```
.OP HYP=OFF
```

.OP HYB - Characters Before the Hyphen

```
.OP HYB=number
```

ou can define the minimum number of characters that must remain before the hyphen (i.e. at the end of the line). For example, to define a minimum of 2 characters, you specify:

```
.OP HYB=2
```

When you define a minimum of 2 characters before the hyphen, all hyphenation points in the following word are considered:

```
in-har-moni-ous
```

When you define a minimum of 3 characters before the hyphen, the word can only be hyphenated as follows:

```
inhar-moni-ous
```

.OP HYA - Characters After the Hyphen

```
.OP HYA=number
```

You can define the minimum number of characters that must remain after the hyphen (i.e. in the beginning of the new line). For example, to define a minimum of 3 characters, you specify:

```
.OP HYA=3
```

When you define a minimum of 3 characters after the hyphen, all hyphenation points in the following word are considered:

```
in-har-moni-ous
```

When you define a minimum of 4 characters after the hyphen, the word can only be hyphenated as follows:

in-har-monious

When you specify the following combination, the word can only be hyphenated in the middle, namely "inhar-monious":

.OP HYA=4, HYB=3

Defining Hyphenation Points

You can define hyphenation points to ensure that a word is not split at an inappropriate point. For example, Con-form automatically hyphenates the word "record" as follows:

```
re-cord
```

This is correct when the word is used as a verb. However, when the word is used as a noun, it must be hyphenated in a different position:

```
rec-ord
```

To define hyphenation points, you must first define the escape character (see [.OP ESC - Escape Character](#)). You can then enter the escape sequence, i.e. the escape character followed by an hyphen. For example:

```
.OP ESC=/  
The Guinness Book of Rec/-ords records many rec/-ords.
```

As a result, the word is hyphenated only at the points you specify, and no longer at the predefined points.

Disallowing the Hyphenation of a Word

When you want to ensure that a word is not hyphenated at all, you can put the escape sequence (i.e. the escape character followed by an hyphen) before the word in question. For example:

```
.OP ESC=/  
/-present
```

Instead, you can also put the escape sequence after the word. For example:

```
.OP ESC=/  
present/-
```


Example

This example illustrates how to switch on hyphenation for the English language, how to define the minimum number of characters before and after the hyphen, and how to define the hyphenation points.

Source Text

```
.LM 0;.RM 50
.FI ON;.JU ON
.OP HYP=E;.** Hyphenation for the English language
.OP HYB=2;.** The minimum number of characters before the hyphen is 2.
.OP HYA=3;.** the minimum number of characters after the hyphen is 3.
You can specify hyphenation for the following languages: English, French and
German. You can also specify the minimum number of characters which
must remain before and/or after the hyphen.
.IL 1
You must use escape sequences to define the hyphenation points.
Thus, you can make sure that a word is split at the position you specified
and not at an inappropriate point.
To do so, you must first define an escape character.
.OP ESC=/
In this example, the slash has been defined as the escape character.
You can now enter the slash followed by an hyphen to define the
hyphenation point.
.IL 1
For example, you can make sure that the noun "rec/-ord" is split correctly.
The hyphen must be in a different position for the verb "re/-cord".
.OP HYP=OFF;.IL 1
Initially, hyphenation is switched off.
```

Formatted Output

You can specify hyphenation for the following languages: English, French and German. You can also specify the minimum number of characters which must remain before and/or after the hyphen.

You must use escape sequences to define the hyphenation points. Thus, you can make sure that a word is split at the position you specified and not at an inappropriate point. To do so, you must first define an escape character. In this example, the slash has been defined as the escape character. You can now enter the slash followed by an hyphen to define the hyphenation point.

For example, you can make sure that the noun "rec-ord" is split correctly. The hyphen must be in a

Hyphenation

different position for the verb "record".

Initially, hyphenation is switched off.

Hyphenated Word in the Source Text

The following instruction only affects words which have been hyphenated at the end of a source text line. It works only when filling has been switched on (.FI ON).

.HP - Hyphen Connection

```
.HP ON
.HP OFF
```

You can hyphenate a word at the end of a source text line and specify that the hyphen will not appear in the formatted output.

To specify that hyphenated words which occur at the end of a source text line are to be reunited in the formatted version, you specify the following:

```
.HP ON
```

Initially, this feature is switched off and words which have been hyphenated at the end of a source text line are also hyphenated in the formatted output, even if the word then occurs in the middle of a line. This corresponds to the following instruction:

```
.HP OFF
```

Example

This example illustrates how hyphenated words at the end of a source text line can be reunited in the formatted output.

Source Text

```
.LM 0;.RM 60
.FI ON;.JU ON
.HP ON
```

You can specify that long words that you have hyphenated at the end of a line in your source text are to be reunited in the formatted version of the document. However, this works only when filling has been switched on. This does not affect words which have been hyphenated within a source text line.

```
.HP OFF;.IL 1
```

By default, this feature is switched off.

This means that the words which you have hyphenated at the end of a source text line will also be hyphenated in the formatted version. Furthermore, a space will be included after the hyphen (when justification has been switched on, several spaces may be included).

This does not affect words which have been hyphenated within a source text line.

Formatted Output

You can specify that long words that you have hyphenated at the end of a line in your source text are to be reunited in the formatted version of the document. However, this works only when filling has been switched on. This does not affect words which have been hyphenated within a source text line.

By default, this feature is switched off. This means that the words which you have hyphenated at the end of a source text line will also be hyphenated in the formatted version. Furthermore, a space will be included after the hyphen (when justification has been switched on, several spaces may be included). This does not affect words which have been hyphenated within a source text line.

7 Tabs and Boxes

▪ Tabs	146
▪ Boxes	153

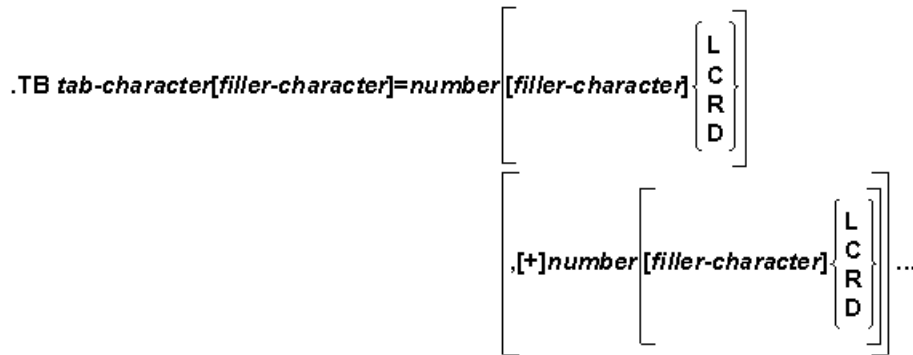
This chapter covers the following topics:

Tabs

When you use tabs to create tables, it is recommended that you switch filling off (`.FI OFF`), so that the columns in your table are vertically straight.

When filling is switched on, Con-form first tabulates your input and then executes the `.FI ON` instruction. This may lead to undesired results.

.TB - Set Tab Stops




.TB OFF

This instruction is used to define the position for each tab stop. In its basic form, you have to define a tab character and the tab position.


For example, to define the asterisk (*) as the tab character and the tab position as 10 columns to the right of the current left margin, you must specify:

```
.TB *=10
```

The tab character is the character you enter in your source text in order to move the text which is preceded by it to the defined tab position. You can define any character except a blank space as the tab character. If you want to define the instruction separator character (initially, this is the semicolon) as the tab character, you must repeat it (i.e. you must specify ";;").

 **Note:** Each tab stop occupies at least one space.

The tab position defines the column where the tab stop is to be set. The position of a tab stop is influenced by the current left margin which has been defined with the `.LM` instruction. For example, if you defined the left margin as `.LM 10`, your text starts in column 11. Thus, if you define a tab stop as 10 columns to the right of the current left margin, the tabulated text starts in column 21.

 **Note:** If the left margin is shifted using the `.TI` or `.OF` instruction, all tab positions are shifted accordingly.

If you want to define several tab stops, you must separate the column numbers by commas; you must not include spaces. For example:

```
.TB *=10,20,30,40
```

Each column number except the first can be preceded by a plus (+) sign. This denotes that the tab stop is to be set n columns to the right of the previous tab stop. The following two examples are therefore equivalent:

```
.TB *=5,10,18,23,30,40,50,65
```

```
.TB *=5,+5,+8,+5,+7,+10,+10,+15
```

You can define a maximum of 20 tab stops.

Each time you define new tab stops using the .TB instruction, any tab stops which have been defined previously are deleted. The TAB key on your keyboard is not used by Con-form.

Tab Stop Alignment

To align a tab stop, you specify one of the following letters after the column number:

L	Left-justified alignment
C	Centered alignment
R	Right-justified alignment
D	Decimal alignment

Left-justified Alignment

To create a left-justified tab stop, you can specify the letter L after the column number. For example:

```
.TB *=10L
```

The text is aligned so that it starts in the specified column.



Note: If you do not specify a letter, all tab stops are left-justified.

Centered Alignment

To create a centered tab stop, you must specify the letter C after the column number. For example:

```
.TB *=30C
```

The text is aligned so that its center coincides with the specified column.

Right-justified Alignment

To create a right-justified tab stop, you must specify the letter R after the column number. For example:

```
.TB *=50R
```

The text is aligned so that it ends in the specified column.

Decimal Alignment

To create a decimal tab stop, you must specify the letter D after the column number. For example:

```
.TB *=25D
```

The text is aligned so that the decimal character coincides with the specified column. If your text does not contain a decimal character, one is assumed directly to the right of your text.

Initially, the decimal character is the period (.). You can define a different decimal character. For example, to define the comma as the new decimal character, you specify:

```
.OP DEC=','
```

Since the comma is used by certain Con-form instructions to separate parameters, it is important that you enclose it in apostrophes as shown above. If you want to define, for example, the slash (/), you need not use the apostrophes.

Filler Character

Filler characters appear whenever a tab occurs. You can define any character except the equal sign (=) as the filler character. If you want to define the instruction separator character (initially, this is the semicolon) as the filler character, you must repeat it (i.e. you must specify ";;"). When you do not specify a filler character, spaces are used.

If you want to define the period (.) as the filler character, you specify it before the equal sign:

```
.TB *. =10
```

For example, you can enter the following:

```
.FI OFF
.TB *. =15
Athens*07:48
Frankfurt*09:05
London*10:34
```


The above source text causes the following formatted output:

```
Athens.....07:48
Frankfurt.....09:05
London.....10:34
```

When you use one of the letters that is used to define the type of tab stop alignment (L, C, R or D), you can also define the filler character after the equal sign. In this case, you must specify the filler character between the column number and the letter. You can then define different filler characters for each tab stop. For example:

```
.TB *=10-L,30.C,50_R
```

The filler characters always appear *before* the defined tab stop.

 **Note:** When you have defined filler characters before and after the equal sign, the character before the equal sign is only considered if a column number (after the equal sign) is not followed by a filler character.

Canceling the Tab Character

If you want to print the character which has been defined as the tab character in your formatted output, you can cancel the tab character as follows:

```
.TB OFF
```

You can also define a different tab character. For example, if the current tab character is the asterisk and you want to use the hash (#), you specify:

```
.TB #=10
```

Example

This example is divided into three parts. The first part illustrates how to create tables using tabs and how to align them decimally. The second part shows that the tab character is not ignored by the **.IC instruction**. The third part shows how to use tabs with the **.OF instruction** which sets the left margin for all lines except the first line.

Source Text

```
.LM 0;.RM 60
.FI OFF
.SL 1
.** The following example shows how to create a simple table.
.** The last two columns are aligned decimally
.TB *=10,30D,40D
2 doz*Eggs*2.39*4.78
3 pints*Milk*.99*2.97
1 piece*Butter*2.29*2.29
2 lbs*Meat*15.99*31.98
.SL 1
Total***42.02
.SL 2
.** The following example shows that the tab character is not ignored
.** by the .IC instruction.
Breaks are caused by the following instructions:
.TB #=10,20,30,40
.IC ON
.BF#.EF#.IP#.OF#.SL
.BX#.EM#.LM#.PH#.TI
.CE#.FI#.NL#.PS#.UL
.CH#.IL#.NP#.RA
.IC OFF
.SL 2
.** The following example shows how to use tabs with .OF
```

```
.FI ON;.JU ON  
.TB *=15  
.OF 15
```

Offsetting:*The tab stop has been set so that the remaining text in the first line begins in the same column as the text that has been offset.

Formatted Output

```
2 doz    Eggs          2.39    4.78  
3 pints  Milk           .99     2.97  
1 piece  Butter         2.29    2.29  
2 lbs    Meat          15.99   31.98  
  
Total                                42.02
```

Breaks are caused by the following instructions:

```
.BF      .EF      .IP      .OF      .SL  
.BX      .EM      .LM      .PH      .TI  
.CE      .FI      .NL      .PS      .UL  
.CH      .IL      .NP      .RA
```

Offsetting: The tab stop has been set so that the remaining text in the first line begins in the same column as the text that has been offset.

Boxes

You can create either empty boxes or boxes that are filled with text. You can also create tables using boxes.

.BX - Draw a Box

```
.BX [-]number [, [+]number]... , [±]number
.BX
.BX OFF
.BX CANCEL
```

To draw a box, you must proceed as follows:

1. You must first define the positions of the vertical lines. For example, if you want to draw a box starting in column 10 and ending in column 55, you specify:

```
.BX 10,55
```

This instruction draws the first horizontal line of the box. The numbers you entered determine the columns in which the horizontal line is to start and end, and the positions of the vertical box lines.



Note: When the first parameter is 0, the left vertical box line is not drawn.

You must specify at least two parameters with the .BX instruction. You can specify a maximum of 20 parameters.

Each column number except the first can be preceded by a plus sign (+). This denotes that the vertical box line is to be set *n* columns to the right of the previous line. The following two examples are therefore equivalent:

```
.BX 10,55
```

```
.BX 10,+45
```

The column positions specified with the .BX instruction (unlike the .TB instruction) are absolute positions. They are not influenced by the current position of the left margin.

2. Next, you must specify the height of the box. The number of lines within the box is determined by the text you enter now and/or the number of blank lines you specify. For example, if you want to create an empty box, you specify the number of blank lines:

```
.IL 5
```

This instruction causes the vertical box lines to be drawn at the specified positions. There is no text between both vertical lines.

3. Finally, to draw the final horizontal box line and to return to normal text processing, you specify:

```
.BX OFF
```

The instructions in steps 1 to 3 cause the following formatted output:

```
+-----+
|       |
|       |
|       |
|       |
|       |
|       |
+-----+
```



Note: Some devices use the exclamation mark (!) instead of the vertical bar. The exclamation mark has also been used in all these examples.

Horizontal Box Lines

To draw several horizontal lines within the box, you must use the .BX instruction without parameters as illustrated below:

```
.BX 10,55
.IL 1
.BX
.IL 1
.BX
.IL 1
.BX OFF
```

The above instructions cause the following formatted output:

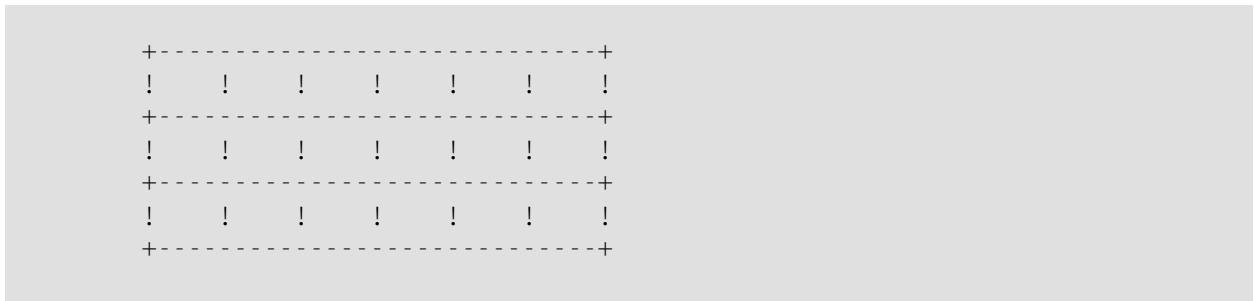
```
+-----+
!                                     !
+-----+
!                                     !
+-----+
!                                     !
+-----+
```

Grid

To draw a grid with more than two vertical lines, you must specify more than two parameters with the first .BX instruction. The remaining .BX instructions (which have no parameters) draw lines at the specified positions. For example:

```
.BX 10,15,20,25,30,35,40
.IL 1
.BX
.IL 1
.BX
.IL 1
.BX OFF
```

The above instructions cause the following formatted output:



Superimposed Boxes

To draw a series of superimposed boxes with vertical lines at different positions, you must repeat the `.BX` instruction with the new parameters. If the minus sign is used with a parameter, the vertical line is not printed at that column (this is explained in detail later in this chapter). For example:

```
.BX -10,25,35,-50
.IL 1
.BX 10,20,30,40,50
.IL 3
.BX 10,50
.IL 1
.BX OFF
```

The above instructions cause the following formatted output:

```
-----
                !      !
-----
!          !          !          !          !
!          !          !          !          !
!          !          !          !          !
+-----+
!                                     !
+-----+
```

Boxes Containing Text

When you want your text contained in a box, you must make sure that the parameter for the right margin (defined with `.RM`) is lower than the parameter for the right vertical line. The number of lines within the box is determined by the text you enter after the `.BX` instruction. For example:

```
.FI ON;.JU ON
.BX 10,55
.LM 10;.RM 54
The parameter for the right margin must be lower than the parameter for
the right vertical line. Otherwise, the right border of the box does not
appear correctly.
.BX OFF
```

The above instructions cause the following formatted output:

```
+-----+  
!The parameter for the right margin must be!  
!lower than the parameter for the right!  
!vertical line. Otherwise, the right border!  
!of the box does not appear correctly.      !  
+-----+
```

If you prefer the text with a jagged edge, you must make sure that justification has been switched off. However, you must not switch filling off (.FI ON;JU OFF).

```
+-----+
!When filling is switched off, the text appears in the same way as you
!entered it, i.e. the specifications for the right margin are
!not considered.                                     !
+-----+
```

It is your responsibility to ensure that your text fills the box in the required manner. You must always specify the appropriate left and right margins for the text that is to appear within a box.

```
+-----+
This!example shows what happens, if you
do not ensure that the correct margins
are specified.                                     !
+-----+
```

You can define the left and right margins so that blank spaces appear before and after a text line, and you can insert blank lines before and after the text. For example:

```
.FI ON;.JU ON
.BX 10,55
.LM 12;.RM 52
.IL 1
The left and right margins of the text are specified so that 2 blank spaces
appear before and after each text line in the box.
Furthermore, one blank line appears before and after the text.
.IL 1
.BX OFF
```

The above instructions cause the following formatted output:

```
+-----+
!                                     !
! The left and right margins of the text !
! are specified so that 2 blank spaces !
! appear before and after each text line !
! in the box. Furthermore, one blank line !
! appears before and after the text.     !
!                                     !
+-----+
```

Boxes without a Left Side

To create a box without a left side, you must insert a minus sign (-) before the parameter for the first column. For example:

```
.BX -10,50
```

The following examples illustrate how this feature can be used:

```
.FI ON;.JU ON
.LM 10;.RM 50
.BX -10,51
The values for the left vertical line and the left margin are the same.
.BX OFF
.IL 1
.BX -11,51
The value for the left vertical line has been increased by one.
Now the box starts in the same column as the text.
.BX OFF
.IL 1
.BX -15,51
The value for the left vertical line is more than the value for
the left margin.
.BX OFF
```

The above instructions cause the following formatted output:

```
-----
The values for the left vertical line!
and the left margin are the same.    !
-----

-----
The value for the left vertical line has!
been increased by one. Now the box!
starts in the same column as the text. !
-----

-----
The value for the left vertical line is!
more than the value for the left margin.!
-----
```

Boxes without a Right Side

To create a box without a right side, you must insert a minus sign (-) before the parameter for the last column. For example:

```
.BX 10,-54
```

The following examples illustrate how this feature can be used:

```
.FI ON;.JU ON
.LM 10;.RM 54
.BX 10,-54
The values for the right vertical line and the right margin
are the same.
.BX OFF
.IL 1
.BX 10,-60
The value for the right vertical line is more than the value for
the right margin.
.BX OFF
.IL 1
.BX 10,-45
The value for the right vertical line is less than the value for
the right margin.
.BX OFF
```

The above instructions cause the following formatted output:

```
+-----+
!The values for the right vertical line and
!the right margin are the same.
+-----+

+-----+
!The value for the right vertical line is
!more than the value for the right margin.
+-----+

+-----+
!The value for the right vertical line is
!less than the value for the right margin.
+-----+
```

Boxes without a Bottom

To create a box without a bottom, you must use the `.BX CANCEL` instruction instead of `.BX OFF`. In this case, the final horizontal box line is not drawn. For example:

```
.BX 10,55
.IL 2
.BX CANCEL
```

The above instructions cause the following formatted output:


```
+-----+
!
!
!
```

You can also draw boxes without a bottom and without a left or right side. For example:

```
.FI ON;.JU ON
.LM 12;.RM 30
.BX 10,-33
.IL 1
This box has no right side and no bottom.
.IL 1
.BX CANCEL
```

The above instructions cause the following formatted output:

```
+-----+
!
! This box has no
! right side and no
! bottom.
!
```

 **Note:** It is not possible to create a box without a top.

Tables

To create a table, you define the tab stops using the `.TB` instruction and the box lines using the `.BX` instruction. For example:

```
.LM 0;.RM 65
.TB #=10,22
.BX 0,9,21,-67
Keyword#Value#Description
.BX
BIN#number#Select a paper tray
BRN#ON/OFF#Suppress breaks in line filling
BXH#character#Define character for horizontal box lines
BXV#character#Define character for vertical box lines
CHA#NUM/UNN#Chapter numbering
.BX CANCEL
```

The above instructions cause the following formatted output:

```
-----
Keyword ! Value      ! Description
-----
BIN      ! number    ! Select a paper tray
BRN      ! ON/OFF    ! Suppress breaks in line filling
BXH      ! character ! Define character for horizontal box lines
BXV      ! character ! Define character for vertical box lines
CHA      ! NUM/UNN  ! Chapter numbering
```

In the following example, the **.OF instruction** has been used to indent the text of the third column, i.e. it sets the left margin for all text following the first line.

```
.LM 2;.RM 58
.FI ON;.JU OFF
.BX 1,13,23,60
.TB #=12,22
.OF +22
Chapter 6#.OP HYP#Define language code
.BX;.OF +22
#.OP HYB#Number of characters before the hyphen
.BX;.OF +22
#.OP HYA#Number of characters after the hyphen
.BX;.OF +22
#.HP#Hyphenated words at the end of a source text line are not
hyphenated in the formatted output.
.BX;.OF +22
Chapter 7#.TB#Define positions of tab stops
.BX;.OF +22
#.BX#Draw a box
```

```
.BX;.0F +22
#.0P BXH#Define character for the horizontal box line
.BX;.0F +22
#.0P BXV#Define character for the vertical box line
.BX OFF
```


The above instructions cause the following formatted output:

```

+-----+
! Chapter 6 ! .OP HYP ! Define language code      !
+-----+
!           ! .OP HYB ! Number of characters before the !
!           !           ! hyphen                          !
+-----+
!           ! .OP HYA ! Number of characters after the  !
!           !           ! hyphen                          !
+-----+
!           ! .HP      ! Hyphenated words at the end of a      !
!           !           ! source text line are not            !
!           !           ! hyphenated in the formatted        !
!           !           ! output.                            !
+-----+
! Chapter 7 ! .TB      ! Define positions of tab stops      !
+-----+
!           ! .BX      ! Draw a box                          !
+-----+
!           ! .OP BXH ! Define character for the          !
!           !           ! horizontal box line                !
+-----+
!           ! .OP BXV ! Define character for the vertical !
!           !           ! box line                           !
+-----+

```

.OP BXH - Horizontal Box Line

```
.OP BXH=character
```

The initial character for the horizontal box line is the hyphen (-). However, you can specify any character you want. For example, to specify the asterisk (*) as the new character, you specify:

```
.OP BXH=*
```


When you specify a different character for the horizontal box line, the character for the box corners (+) is no longer in effect.

.OP BXV - Vertical Box Line

```
.OP BXV=character
```

The initial character for the vertical box line is the vertical bar (|). However, you can specify any character you want. For example, to specify the asterisk (*) as the new character, you specify:

```
.OP BXV=*
```

 **Note:** Some devices use the exclamation mark (!) instead of the vertical bar.

The following example illustrates how to create a box where the horizontal and vertical lines are made up of asterisks:

```
.OP BXH=*
.OP BXV=*
.BX 10,55
.IL 2
.BX OFF
```

The above instructions cause the following formatted output:

```
*****
*                                     *
*                                     *
*****
```

8 Formatting Large Documents

- Embedding Documents 169
- Saving and Restoring the Settings 173
- Chapter Titles and the Table of Contents 176
- Index Entries and the Index Summary 194

This chapter covers the following topics:

When you define chapter titles, a table of contents is automatically created. When you define index entries, an index is automatically created.

The different parts of a formatted document are output in the following sequence:

1. document text (including all embedded documents at the specified positions)
2. index
3. table of contents

Embedding Documents

You can merge several source documents into a single output document (i.e. the formatted version of all source documents). To do so, you use the `.EM` instruction.

The documents to be embedded may either contain short texts (e.g. standard phrases which you include in a letter) or long texts (e.g. a chapter of a book).

When you write a book, you can create a document which contains all `.EM` instructions and which thus controls the output of all chapters of the book.

.EM - Embed

```
.EM document-name
```

To embed a document in the current document (i.e. the document containing the `.EM` instruction), you must specify the `.EM` instruction with the name of the document that you want to embed. For example, to embed the document named "preface", you must specify:

```
.EM preface
```

If the document name consists of several words which are separated by blanks, the document name must be specified in single quotation marks. For example:

```
.EM 'new strategy'
```

No special instruction is needed at the end of an embedded document. Con-form automatically returns to the document which contained the `.EM` instruction and continues processing the information following the `.EM` instruction. However, if required, you can also use the `.EF` instruction (see below).

The document specified by the `.EM` instruction may also contain `.EM` instructions. This process may be continued up to a maximum nesting depth of 5.

.EF - End of File

```
.EF
```

When you want to embed only the beginning of a document, you can include the .EF instruction in the document that you want to embed.

Upon encountering this instruction, Con-form returns control to the document which contains the .EM instruction and continues processing the information following the .EM instruction.

All information in the embedded document which follows the .EF instruction is not considered.

.EP - End of Processing

```
.EP
```

You can include the .EP instruction in any document (it need not necessarily be an embedded document) to stop formatting.

In the case of an embedded document, Con-form does not return control to the document which contains the .EM instruction.

It is not possible to conditionally suppress the processing of the .EP instruction by using it in an **.IF construction**. Formatting is always stopped.

.OP EMN - Embed Nobreak

```
.OP EMN=ON  
.OP EMN=OFF
```

You can only use the instruction .OP EMN=ON when filling is switched on (.FI ON).

When you embed a document in another document, the text *after* the embedded document always starts in a new line - even if filling is switched on.

To disallow the line break, you must specify the following:

```
.OP EMN=ON
```

As a result, the text of the current document continues in the same line as the text of the embedded document.

Initially, this option is switched off. This corresponds to the following:

```
.OP EMN=OFF
```

Example

This example illustrates how to embed a document. When the embedded document contains no specific instructions, it is formatted according to the instructions given in the document which contains the .EM instruction. However, instructions in the embedded document (e.g. values for margins, filling and justification) also apply to the document which contains the .EM instruction. Therefore, you must take care to reset these values at the end of the embedded document.

Source Text - Document to be Embedded

```
.LM 10;.RM 60
.FI ON;.JU OFF
This is the document to be embedded. When it is embedded in another
document, it is formatted according to the instructions defined
here.
.LM 0;.RM 70
.FI ON;.JU ON
```

Source Text - Document Containing the .EM Instruction

```
.LM 0;.RM 70
.FI ON;.JU ON
To include the text from another document into the current document,
you must specify the .EM instruction with the name of the document to
be embedded.
.IL 1
.EM doc2
.IL 1
When you do not reset the values in the embedded document, they are also
applied to the main document.
```

Formatted Output

To include the text from another document into the current document, you must specify the .EM instruction with the name of the document to be embedded.

This is the document to be embedded. When it is embedded in another document, it is formatted according to the instructions defined here.

When you do not reset the values in the embedded document, they are also applied to the main document.

Saving and Restoring the Settings

If a part of your document is to be formatted differently than the main part of it, you can save the current settings of the document with the `.SA` instruction. For the next section of text, you can then modify the settings as required. Later you specify the `.RS` instruction to restore the original settings which have been saved with the last `.SA` instruction and continue processing.



Note: This feature is helpful when you want to embed a document.

.SA - Save Context

```
.SA
```

To save the current settings of the document, you must specify the `.SA` instruction.

You can then modify the settings as required. For example, you can define different left and right margins for the text after the `.SA` instruction.

The following actions automatically issue the `.SA` instruction:

- `.NL KEEP`
- `.NL FLOAT`
- The processing of footnotes (`.FN`)
- The processing of header lines (`.HL`)

To restore the original settings which have been saved with the last `.SA` instruction, you must use the `.RS` instruction (see below).

.RS - Restore Context

```
.RS
```

To restore the settings which have been saved with the last `.SA` instruction, you must specify the `.RS` instruction.

As a result, formatting continues with the settings which were in effect before the last `.SA` instruction was executed.

The following actions automatically issue the `.RS` instruction:

- `.NL OFF`
- The completion of the processing of footnotes (`.FN`)
- The completion of the processing of header lines (`.HL`)

Example

This example illustrates the instructions `.SA` and `.RS`.

Source Text

```
.LM 0;.RM 70
.FI ON;.JU ON
You can use the .SA instruction to save the current settings (for
example, the settings for the margins, filling and justification).
.SA
.** Define different settings for the following lines.
.LM 10;.RM 50
.FI ON;.JU OFF
.IL 1
Different margins have been defined for this part of the document and
justification has been switched off.
.IL 1
.** Restore the previous settings.
.RS
When you use the .RS instruction, the previously stored settings are
restored and formatting continues with the settings which were in effect
before the .SA instruction was executed.
```

Formatted Output

You can use the `.SA` instruction to save the current settings (for example, the settings for the margins, filling and justification).

Different margins have been defined for
this part of the document and
justification has been switched off.

When you use the `.RS` instruction, the previously stored settings are restored and formatting continues with the settings which were in effect before the `.SA` instruction was executed.

Chapter Titles and the Table of Contents

You define chapter titles with the `.CH` instruction. The titles are output in the current position of the text as well as in the table of contents. The table of contents is automatically output at the end of the document, starting on a new page. When you also specify index entries, the table of contents is output after the index.

You can use the `.PT` instruction to control the layout of the table of contents.

.CH - Chapter

```
.CH number
.CH +number
.CH -number
.CH
```

To define chapter titles and make entries in the table of contents, you must use the `.CH` instruction. The text which follows the `.CH` instruction - before the next line break - is the chapter title. For example:

```
.CH 1
Chapter One
```

The title is output at the current position in the text and in the table of contents. The parameter after the `.CH` instruction defines the level (see [Defining Different Levels for Titles](#)).

Con-form automatically checks whether at least 7 free lines are available on the current page (i.e. Con-form implicitly performs the instruction `.NL 7`). If seven lines are not available, Con-form starts a new page and outputs the title on the new page. Thus, a chapter or section will not start near the bottom of a page.

The Importance of Line Breaks after the Title

When filling is switched on (.FI ON), a line break must occur after the title. Otherwise, all text between the .CH instruction and the next line break is interpreted as the title and included in the table of contents. For example:

```
.FI ON;.JU ON
.CH 1
This is the chapter title.
When filling has been switched on, you must ensure that a line break
occurs after the chapter title.
.IL 1
All text between the .CH instruction and the next line break is included
in the table of contents.
```

The above instructions cause the following formatted output:

```
1. This is the chapter title. When filling has been switched on, you
   must ensure that a line break occurs after the chapter title.

All text between the .CH instruction and the next line break is included
in the table of contents.
```

The table of contents is automatically output at the end of the formatted document:

```
1. This is the chapter title. When filling has been switched on, you
   must ensure that a line break occurs after the chapter title..... 1
```

To define a line break, you can either leave blank lines in the source text or specify the end-of-line character which is usually the dollar sign (\$). See [Starting Text on a New Line](#) for further information concerning line breaks.

Defining Different Levels for Titles

To define different levels, you must specify the level as a parameter with the `.CH` instruction. For example:

Instruction	Level	Numbering Style
<code>.CH 1</code>	Chapter Title	1
<code>.CH 2</code>	Section Title	1.1
<code>.CH 3</code>	Subsection Title	1.1.1
<code>.CH 4</code>	Subsubsection Title	1.1.1.1



Note: The parameter after the `.CH` instruction is not the number of the chapter, but the level of the title. The chapter number is automatically increased by one when the next `.CH 1` instruction (i.e. a title on the first level) is encountered.

You can define up to ten different levels for titles and up to 99 entries for each level. Each time a `.CH` instruction is processed, a chapter number is automatically built. The chapter number is stored in the variable `$CH` (see *Modifiable System Variables*).

When you specify the level as a parameter with the `.CH` instruction, the titles are initially numbered. However, you can switch chapter numbering off (see the instruction `.OP CHA`).

Initially, the entries in the table of contents are indented. However, you can switch indentation off (see the instruction `.OP CHI`).



Note: When you do not want to use different levels for your titles, you need not specify a number with the `.CH` instruction. In this case, the titles are neither numbered nor indented in the table of contents.

The following example shows how to define titles on different levels:

```
.CH 1
Chapter Title
This is the text below the chapter title.
.IL 1
.CH 2
Section Title
This is text below the second level section title. In the table of
contents, all titles below level 1 are automatically indented.
.IL 1
.CH 3
Subsection Title
This is text below the third level subsection title.
.IL 1
.CH 1
```

Chapter Title
 This is chapter two.

The above instructions cause the following formatted output:

```

1. Chapter Title
   This is the text below the chapter title.

1.1 Section Title
   This is text below the second level section title. In the table of
   contents, all titles below level 1 are automatically indented.

1.1.1 Subsection Title
   This is text below the third level subsection title. The table
   of contents is output at the end of the formatted document.

2. Chapter Title
   This is chapter two.
```

The table of contents is output as follows:

```

1. Chapter Title..... 1
   1.1 Section Title..... 1
     1.1.1 Subsection Title..... 1
2. Chapter Title..... 1
```

Defining the Number of Levels Higher or Lower than the Current Level

You can also use a plus or minus sign to define the number of levels higher or lower than the current level. When you specify the instruction `.CH +0`, the next title is output at the same level as the current title.

The following example shows how to define titles on different levels:

```
.CH 1
Chapter One
To define the next chapter title at the same level, you specify .CH +0.
.IL 1
.CH +0
Chapter Two
To define a section title within the second chapter, you specify the number
of levels higher or lower.
.IL 1
.CH +1
This is the section title
.CH +1
A subsection
.CH -1
Another section title
.CH +1
First subsection
.CH +0
Second subsection
.CH +0
Third subsection
.IL 1
.CH 1
Chapter Three
```

The above instructions cause the following formatted output:

```
1. Chapter One
To define the next chapter title at the same level, you specify .CH +0.

2. Chapter Two
To define a section title within the second chapter, you specify the number
of levels higher or lower.

2.1 This is the section title
2.1.1 A subsection
2.2 Another section title
2.2.1 First subsection
2.2.2 Second subsection
2.2.3 Third subsection
```


3. Chapter Three

The table of contents is output as follows:

1. Chapter One.....	1
2. Chapter Two.....	1
2.1 This is the section title.....	1
2.1.1 A subsection.....	1
2.2 Another section title.....	1
2.2.1 First subsection.....	1
2.2.2 Second subsection.....	1
2.2.3 Third subsection.....	1
3. Chapter Three.....	1

Emphasizing a Title

At the current position in the text, the title is not automatically formatted in a specific way. For example, to emphasize the title or print blank lines before and after it, you must include the appropriate instructions in the source text:

```
.CH 1;.US:.* Underscore the next line
Chapter One
.II 1
.CH 1;.BP:.* Print the next line in boldface
Chapter Two
.* Print the next chapter title in italics.
.* The escape sequence /I1 was entered in a separate line.
.* When the escape sequence is entered directly in front of the words
.* "Chapter Three", the number is not printed in italics.
.OP ESC=/
/I1
.CH 1
Chapter Three/I0
```

The above instructions cause the following formatted output:

1. Chapter One
2. Chapter Two
3. *Chapter Three*

The table of contents is not influenced by the above instructions:

1.	Chapter One.....	1
2.	Chapter Two.....	1
3.	Chapter Three.....	1

To control the layout of the table of contents, you must use the **.PT instruction**.

.OP CHA - Chapter Numbering

```
.OP CHA=NUM
.OP CHA=UNN
```

When you specify the level as a parameter with the `.CH` instruction, the titles are initially numbered. This corresponds to the following instruction:

```
.OP CHA=NUM
```

To switch chapter numbering off, you must specify:

```
.OP CHA=UNN
```

For example:

```
.OP CHA=UNN
.CH 1
Chapter 1
This is the first chapter.
.CH 2;.IL 1
Section 1.1
This is the first section.
.CH 2;.IL 1
Section 1.2
This is the second section.
.CH 1;.IL 1
Chapter 2
This is the second chapter.
```

The above instructions cause the following formatted output:

```
Chapter 1
This is the first chapter.

Section 1.1
This is the first section.

Section 1.2
This is the second section.

Chapter 2
This is the second chapter.
```

The table of contents is output as follows:

Chapter 1.....	1
Section 1.1.....	1
Section 1.2.....	1
Chapter 2.....	1

Initially, the entries in the table of contents are indented. However, you can switch indentation off (see the instruction **.OP CHI**).

.OP CHI - Chapter Indentation in the Table of Contents

```
.OP CHI=ON
.OP CHI=OFF
```

When you specify the level as a parameter with the `.CH` instruction, the titles in the table of contents are initially indented - according to the level you specified. This corresponds to the following instruction:

```
.OP CHI=ON
```

To switch indentation off, you must specify:

```
.OP CHI=OFF
```

For example:

```
.OP CHI=OFF
.CH 1
Chapter One
This is the first chapter.
.CH 2;.IL 1
First Section
This is the first section.
.CH 2;.IL 1
Second Section
This is the second section.
.CH 1;.IL 1
Chapter Two
This is the second chapter.
```

The above instructions cause the following formatted output:

```
1. Chapter One
This is the first chapter.

1.1 First Section
This is the first section.

1.2 Second Section
This is the second section.

2. Chapter Two
This is the second chapter.
```

The table of contents is output as follows:

1. Chapter One.....	1
1.1 First Section.....	1
1.2 Second Section.....	1
2. Chapter Two.....	1

.OP CHL - Chapter Levels in the Table of Contents

```
.OP CHL=number
```

When you specify the level as a parameter with the .CH instruction, you can define a maximum of 10 different levels. This corresponds to the initial setting:

```
.OP CHL=10
```

You can specify the maximum level of titles that is included in the table of contents. All titles with levels greater than the number specified with the .OP CHL instruction are not included in the table of contents. For example:

```
.OP CHL=2
.CH 1
Chapter Title
The chapter title at level 1 is included in the table of contents.
.CH 2;.IL 1
Section Title
The section title at level 2 is included in the table of contents.
.CH 3;.IL 1
Subsection Title
The subsection title at level 3 is not included in the table of contents.
```

The above instructions cause the following formatted output:

```
1. Chapter Title
The chapter title at level 1 is included in the table of contents.

1.1 Section Title
The section title at level 2 is included in the table of contents.

1.1.1 Subsection Title
The subsection title at level 3 is not included in the table of contents.
```

The table of contents is output as follows:

```
1. Chapter Title..... 1
  1.1 Section Title..... 1
```

When you do not want to include any titles in the table of contents, you must specify the following:

```
.OP CHI=0
```

When you specify the above instruction at the beginning of a document, a table of contents is not created. When you specify the above instruction after you have already included several titles in the table of contents, the titles after this instruction are not included in the table of contents.

.SC - Set Chapter Number

```
.SC chapter-number
```

This instruction is only valid when .OP CHA=NUM is in effect, i.e. when chapter numbering is switched on.

You can define another chapter number for the *current* chapter or section. The next .CH *n* instruction then determines the chapter or section level. For example:

```
.CH 1
Chapter One
.SC 4;.** Set the current chapter number to 4.
.CH 1;.** Start a new chapter. The chapter number is increased by 1.
Chapter Five
```

The above instructions cause the following formatted output:

- 1. Chapter One
- 5. Chapter Five

The table of contents is output as follows:

1. Chapter One.....	1
5. Chapter Five.....	1

As long as you do not specify a new level 1 chapter title (using the instruction .CH 1), all sections and subsections receive the chapter number which has been defined for the current chapter. When the instruction .CH 1 occurs after the .SC instruction, the chapter number is increased by one. For example:

```
.CH 1
Chapter Title
.SC 4;.** Set the current chapter number to 4.
.CH 2;.** Output a section title at level 2.
Section Title
.SC 7;.** Set the current chapter number to 7.
.CH 2;.** Output another section title at level 2.
Section Title
.CH 3;.** Output a subsection title at level 3.
First Subsection Title
.SC 7.5;.** Set the current section number to 7.5
.CH 3;.** Output another subsection title at level 3.
Second Subsection Title
.SC 15;.** Set the current chapter number to 15.
```

```
.CH 1;** Start a new chapter with number 16.  
Chapter Title
```

The above instructions cause the following formatted output:

```
1. Chapter Title  
4.1 Section Title  
7.1 Section Title  
7.1.1 First Subsection Title  
7.5.1 Second Subsection Title  
16. Chapter Title
```

In the table of contents, the titles are indented according to their levels:

```
1. Chapter Title..... 1  
  4.1 Section Title..... 1  
    7.1 Section Title..... 1  
      7.1.1 First Subsection Title..... 1  
      7.5.1 Second Subsection Title..... 1  
16. Chapter Title..... 1
```

.PT - Put to Table of Contents

```
.PT instruction  
.PT text
```

You use the .PT instruction to control the layout of the table of contents. You can define text as well as instructions (or macros with parameters as necessary) with the .PT instruction. However, you must not mix text and instruction within a single .PT instruction.

For example, to print a centered and underscored title above the table of contents, you specify the following instructions:

```
.PT .CE 1;.US  
.PT Table of Contents
```

You can use the **.HL** instruction (to define a header which appears at the top of each page) as a parameter of the .PT instruction. Like the .PT instruction, the .HL instruction is also specified with either an instruction or text. For example:

```
.PT .HL .CE 1  
.PT .HL Table of Contents
```

The .PT instruction must be entered at the beginning of the document. When you enter it, for example, at the end of the document, it may happen that the formatted output does not appear as desired.

The table of contents is always output at the end of the document. However, when you have printed the document, you can re-arrange the printed pages and move the table of contents to the beginning of the document. In this case, it is useful when a different page number style (for example, Roman page numbers) has been defined for the table of contents.

When you specify a top and/or bottom title for the document and do not define different top or bottom titles for the table of contents, the table of contents also contains the top and bottom titles of the main document. When page numbering has been defined, consecutive numbering continues from the last page of the actual document.



Tip: When you work in Con-nect, you can file a formatted version of the document and then modify the formatted version of the table of contents in the Con-nect editor.

The following example illustrates how to output the page number in the bottom title of the table of contents and how to define upper-case Roman page numbers with number 1 as the first page.

```
.FI ON;.JU ON
.PT .NP 1
.PT .PM R
.PT .BT //ToC #
.PT .US;.** Underscore the following line.
.PT Table of Contents
.PT .IL 1
.CH 1;.** Chapter numbering is switched on by default.
The Table of Contents
.IL 1
.CH 2
The .PT Instruction
.IL 1
You use the .PT instruction to influence the layout of the table of
contents. You can specify text or instructions with the .PT
instruction. However, you must not mix text and instructions within a
single .PT instruction.
.IL 2
.CH 2
The .OP PTC Instruction
.IL 1
By default, the .PT instruction is switched on. You can switch it
off so that the text or instruction after a .PT instruction is not
applied to the table of contents.
```

The above instructions cause the following formatted output:

1. The Table of Contents

1.1 The .PT Instruction

You use the .PT instruction to influence the layout of the table of contents. You can specify text or instructions with the .PT instruction. However, you must not mix text and instructions within a single .PT instruction.

1.2 The .OP PTC Instruction

By default, the .PT instruction is switched on. You can switch it off so that the text or instruction after a .PT instruction is not applied to the table of contents.

The table of contents is output as follows:

Table of Contents

1. The Table of Contents.....	1
1.1 The .PT Instruction.....	1
1.2 The .OP PTC Instruction.....	1

The page number is printed in the bottom title of the table of contents:

ToC I

.OP PTC - Switch .PT Instruction On/Off

```
.OP PTC=ON
.OP PTC=OFF
```

Initially, the .PT instruction is in effect, i.e. the text or instruction after the .PT instruction is applied to the table of contents. This corresponds to the following instruction:

```
.OP PTC=ON
```

To switch this feature off, so that the text or instruction after the .PT instruction is not applied to the table of contents, you specify:

```
.OP PTC=OFF
```

Index Entries and the Index Summary

.IX - Index

```
.IX text
```

Index entries are made using the .IX instruction. For example:

```
.IX Index entries;; how to specify
```

The text to be included in the index can be up to 60 characters long and may contain any printable characters. If you want to include the instruction separator character (initially, this is the semicolon) in the index, you must repeat it (i.e. you must specify ";;").

The text defined with the last .IX instruction is stored in the variable \$IX (see [Modifiable System Variables](#)).

Unlike the .CH instruction, the .IX instruction does not output the index entry at the current position in the text. It stores the index entry together with the number of the page on which the entry has been defined.

All index entries are automatically sorted alphabetically and output at the end of the document, starting on a new page. When your document also contains a table of contents, the index summary is output between the document text and the table of contents.

Lower-case letters precede the corresponding upper-case letters in the index summary. The sorting sequence is "aAbBcC ... zZ".



Note: A specific sorting sequence can be defined while installing Con-form.

When you used the .CH instruction in your source text, the index summary automatically receives the heading "Index". In addition, when chapter numbering was active (.OP CHA=NUM), the heading "Index" is preceded by the next chapter number. When the .CH instruction was not used in your source text, the heading "Index" does not appear.

"Index *n*" (where *n* is the page number) is automatically output at the bottom of all odd-numbered pages. Therefore, you should not use the .BT instruction to define page numbers in the bottom title. Consecutive numbering continues from the last page of the actual document - even if you did not define page numbers for your document. A page number is not output on an even-numbered page.

Example

This example illustrates how to define index entries. Filling has been switched on. Therefore, the index entries do not cause line breaks in the formatted document. The .CH instruction has been used in the source text. Therefore, the index receives a chapter number and a table of contents is created. In this example, the index is output on an even-numbered page. Therefore, the page does not contain a page number.



Note: When you work in Con-nect and file a formatted version of this example document, the document format is Cnf (instead of Txt). This is because the index contains an entry starting with a period (.IX).

Source Text

```
.FI ON;.JU ON
.CH 1
Defining Index Entries
.IX 1
Index entries are made using the .IX instruction.
.IX .IX instruction
The text to be included in the index can be up to 60 characters long
.IX Restrictions for index entries
and it may contain any printable characters.
.IX Characters, allowed in an index entry
All index entries are sorted alphabetically and output at the end
of the document.
.IX Sorting sequence
```

Formatted Output - Document Text

1. Defining Index Entries

Index entries are made using the .IX instruction. The text to be included in the index can be up to 60 characters long and it may contain any printable characters. All index entries are sorted alphabetically and output at the end of the document.

Formatted Output - Index

2. Index

```
Characters, allowed in an index entry ... 1
Restrictions for index entries ... 1
```

Sorting sequence ... 1

.IX instruction ... 1

Formatted Output - Table of Contents

1. Defining Index Entries.....	1
2. Index.....	2

9 Variables

▪ Defining Your Own Variables	199
▪ Arithmetic Calculations	204
▪ Text Functions	212
▪ Decimal Numbers	215
▪ The Variable Character	219
▪ System Variables	221

This chapter covers the following topics:

Defining Your Own Variables

If a long word or phrase occurs frequently in the source text, you can assign it to a variable. Subsequently it is only necessary to specify the variable.

.SV - Set Variable

```
.SV variable-name=value
```

You can define a variable and then assign a value to it. For example:

```
.SV name=Smith
```

Each variable must be identified by a unique name. The variable name (the parameter before the equal sign) can only contain letters or digits. It must contain at least one character and can be up to 100 characters long. The variable name must not start with the variable character (see below) and must not contain blanks. No distinction is made between upper-case and lower-case.

The parameter after the equal sign can either be text (can consist of several words and may contain blanks) or an arithmetic calculation (see [Arithmetic Calculations with .SV](#)).

	Maximum value for text	Maximum value for arithmetic calculation
Parameter after equal sign	249 characters	100 characters
Intermediate result	CSIZE minus 10 (in KB)	31 digits
Final result	CSIZE minus 10 (in KB)	29 digits



Notes:

1. Starting with Con-form version 3.4.1, the maximum value for text may exceed 100 characters. In previous versions, this was restricted to 100 characters.
2. CSIZE is the size of Con-nect buffer area. The administrator defines it in the Natural parameter module NATPARM. The maximum CSIZE value can be 512 KB. Thus, the maximum that can be used for an intermediate and final text result is 502 KB.

No distinction is made between variables which hold text and variables which hold numeric values. A variable can even hold text at one instant and a numeric value at a later instant. The type of value which a variable currently holds is determined by the most recent .SV instruction which assigned a value to that variable.

Using the Variable in the Document Text

After you have defined a variable with the `.SV` instruction, you can include it in the document text. It must be preceded by the variable character. Initially, the variable character is the ampersand (`&`). For example:

```
&variable
```

When you format the document, the variable is replaced with the defined value.

Substitution (see the [.SU instruction](#)) is automatically switched on by the `.SV` instruction. This means that the text that is preceded by the variable character is interpreted as a variable and replaced accordingly.

Associating Text Strings and Variables

You can associate any number of text strings and variables.

To output a text string or another variable *after* the variable (without a blank in between), you must specify a period (`.`) directly after the first variable. The period is not required after a text string which is followed by a variable. For example:

```
.SV var1=butter  
.SV var2=fly  
.SV dollars=200  
&var1.&var2  
&var1.milk  
dragon&var2  
$&dollars
```

The above instructions cause the following formatted output:

```
butterfly  
buttermilk  
dragonfly  
$200
```

Using Compound Names

A variable can be formed by resolving a compound name.

A compound name contains two or more simple variables and is resolved by substituting the value of each variable, working from right to left. For example:

```
.SV V1wakeup=Good Morning
.SV V2night=wakeup
.SV V3=night
&V1&V2&V3
```

The above instructions cause the following formatted output:

```
Good Morning
```

The variable in the above example is resolved from right to left in successive steps:

```
&V1&V2&V3
&V1&V2night
V1wakeup
Good Morning
```

Punctuation Mark After a Variable

To output a punctuation mark after the variable, you must enter a period (.) followed by the required punctuation mark. For example, to end a sentence with a period, you must repeat the period:

```
.SV text=In this case, you must repeat the period
&text..
.SV more=This is followed by a semicolon
&more.;
```

The above instructions cause the following formatted output:

```
In this case, you must repeat the period.
This is followed by a semicolon;
```

Arithmetic Calculations with .SV

The parameter of the .SV instruction can also be a numeric value. You can specify an integer, a fractional number with a decimal sign, or a number with a thousands separator character (see the [.OP TRI instruction](#)).

You can use the following arithmetic operators:

+	Addition
-	Subtraction
*	Multiplication
/	Division

In the .SV instruction, a calculation is evaluated by applying the operators from left to right. All operators have equal priority. Parentheses are not evaluated.



Tip: It is recommended that you use the .CV instruction for arithmetic operations since it evaluates parentheses. See [.CV - Compute Variable](#).

You can specify fractional numbers in the .SV instruction. However, if you do not specify the number of characters which are to be output after the decimal character (see the [.OP DAS instruction](#)), the result is rounded to the nearest integer. For example:

```
.SV number1=2.5*3+2
The result of number1 is &number1
.OP DAS=2
.SV number2=2.5*3+2
The result of number2 is &number2
```

The above instructions cause the following formatted output:

```
The result of number1 is 9
The result of number2 is 9.50
```



Caution: You must specify the .OP DAS instruction before the .SV instruction. Otherwise, the result is not rounded as desired.

You can modify the value of a parameter. For example:

```
.SV counter=11  
.SV counter=&counter+1  
&counter
```

The above instructions cause the following formatted output:

```
12
```

Initially, the result of a division is rounded to the nearest integer. However, when you specify the number of characters which are to be output after the decimal character (see the [.OP DAS instruction](#)), the result is rounded accordingly. For example:

```
.OP DAS=2  
.SV number=2/3  
&number
```

The above instructions cause the following formatted output:

```
0.67
```

Arithmetic Calculations

It is recommended that you use the `.CV` instruction for arithmetic operations since it evaluates parentheses (in contrast to the `.SV` instruction).

`.CV` - Compute Variable

```
.CV variable-name=arithmetic-expression
```

You can define a variable and then compute its value using an arithmetic expression. An arithmetic expression consists of one or more constants, variables or system variables.

A constant is an integer, a fractional number with a decimal sign, or a number with a thousands separator character (see the [.OP TRI instruction](#)). Arithmetic operators (see below) must not be used as thousands separator characters. The number of digits in the constant must not exceed 29. The number of digits after the decimal sign must not exceed 7. For example:

```
.OP TRI=' , '
.CV const1=15
.CV const2=0.123
.CV const3=1234567890123456789012.1234567
.CV const4=1,999
```

A variable used in an arithmetic expression must have a numeric value. This variable must have been defined in a previous `.SV` or `.CV` instruction.

A system variable can be used in an arithmetic expression if it has a numeric value.

You can use the following arithmetic operators:

()	Parentheses
*	Multiplication
/	Division
+	Addition
-	Subtraction

Parentheses are evaluated. The arithmetic operation is processed in the following order:

1. Parentheses
2. Multiplication and division (left to right)
3. Addition and subtraction (left to right)

When the divisor is 0, the result of a division operation is 0.

When the result of a .CV calculation leads to an overflow, the variable is set to 5 asterisks (*****). For example:

```
.CV var1=-25000
&var1
.CV var2=15-5+4/2-10
&var2
.CV var3=(1+2) * 3/ (1+8)
&var3
.CV var4=&var1+100000
&var4
.CV var5= 10000000000000000 * 10000000000000000
.IF &var5 = *****
.TH
Your result is too big
.EL
&var5
.EI
```

The above instructions cause the following formatted output:

```
-25000
2
1
75000
Your result is too big
```

Precision of Results for Arithmetic Operations

The following table gives an overview of the precision rules that are used in an arithmetic calculation with the .CV instruction.

Operation	Digits before decimal character	Digits after decimal character (intermediate result)	Digits after decimal character (final result)
Addition/Subtraction	$F_i + 1$ or $S_i + 1$ (whichever is greater)	F_d or S_d (whichever is greater, maximum 7)	DAS and RND are applied to the last intermediate result.
Multiplication	$F_i + S_i + 2$	$F_d + S_d$ (maximum 7)	
Division	$F_i + S_d$	If .OP REM=OFF: F_d or value of DAS option (whichever is greater, maximum 7). In addition, if .OP RND=ON, this number of digits is internally increased (+1). If .OP REM=ON: value of DAS option. This value is also used for remainder.	

The following abbreviations are used in the above table:

F	First operand.
S	Second operand.
i	Digits before decimal character.
d	Digits after decimal character.

The number of digits after the decimal character is handled separately for intermediate and final result. Intermediate results are arithmetic operations *before* rounding and applying of the option DAS. The final result is last intermediate result *after* rounding and applying of the option DAS.

In the final result (formatted output), the digits after the decimal character are defined by the option DAS.

When the option RND is switched ON, rounding is performed. When the option RND is switched OFF, the last digits are truncated.

When the option REM is switched ON, the result of a division operation is not rounded (regardless of the setting defined for .OP RND).

Example:

```
.OP DAS=0  
.OP RND=ON  
.CV VAR=(3+5.25) / 1.5 - 4
```

The final result of the above operation is 2, as indicated in the following table:

	Operation	Result
Intermediate operation 1	$3+5.25$	8.25
Intermediate operation 2	$8.25/1.5$	5.500 (since .OP RND=ON, the number of digits after decimal character is internally increased (+1)).
Intermediate operation 3	$5.500-4$	1.500
Final operation	Apply DAS and RND	2

.OP RND - Round Result of .CV

```
.OP RND=ON  
.OP RND=OFF
```

This instruction can only be used with the .CV instruction.

Initially, the result of the .CV instruction is rounded to the nearest integer. This corresponds to the following instruction:

```
.OP RND=ON
```

For example:

```
.OP DAS=3  
.OP RND=ON  
.CV var1=0 + 0.2555  
&var1  
.OP RND=OFF  
.CV var2=0 + 0.2555  
&var2
```

The above instructions cause the following formatted output:

```
0.256  
0.255
```

.OP REM - Remainder of Division with .CV

```
.OP REM=ON  
.OP REM=OFF
```

This instruction can only be used with the `.CV` instruction.

Initially, this option is switched off. This corresponds to the following instruction:

```
.OP REM=OFF
```

If the remainder of a division operation is to be moved to the modifiable system variable `$RR`, you must specify the following instruction:

```
.OP REM=ON
```

The initial value of `$RR` is 0 (zero).

The result of a division operation is not rounded (regardless of the setting defined for `.OP RND`). The format of the remainder depends on the options defined with the instructions `.OP DAS` and `.OP TRI`.

For example:

```
.OP DAS=2  
.OP REM=ON  
&$RR  
.CV number=2/3  
&number  
&$RR
```

The above instructions cause the following formatted output:

```
0  
0.66  
0.02
```

Text Functions

You can use text functions in conjunction with the `.SV` or `.CV` instruction. You must always specify an apostrophe (') after the function code.

Text functions can also be used with the instructions `.IF` and `.WH`.

The following function codes are available:

Code	Explanation	Example
E	Check whether the specified variable exists. The parameter is the name of a variable. If the variable exists, the value 1 is returned. If the variable does not exist, the value 0 is returned.	<pre>.SV var=Smith .SV test=var If variable "var" exists, type 1, else type 0: &E'&test</pre>
H	Convert text string to hexadecimal value. The parameter can be an arbitrary text string. Each character in the parameter string is converted to the corresponding EBCDIC hexadecimal value.	<pre>.SV var=H'123 The hexadecimal value of 123 is &var. .SV name=Smith The hexadecimal value of Smith is &H'&name.</pre>
I	Invert rendition of text string. The parameter is an arbitrary text string. The direction in which the contents of the variable is interpreted is altered.	<pre>.SV abc=xyz .SV def=I'&abc The inverted value is &def (zyx).</pre>
L	Convert text string to lower-case. The parameter can be an arbitrary text string. Each upper-case letter in the parameter string is replaced by the lower-case equivalent. Other characters are left unchanged.	<pre>.SV var=CATS Don't forget to feed your &L'&var tonight.</pre>
N	Determine length of text string. The parameter can be an arbitrary text string. The number of characters in the parameter string is returned.	<pre>.SV date=3.11.00 The length of this string is &N'&date</pre>
R	Convert parameter to a Roman number. The parameter is an unsigned decimal number. The number is returned as the corresponding Roman number in upper-case format.	<pre>.SV year=R'2000 &year will be a wonderful year. .SV vol=8 Give me volume &R'&vol of the encyclopaedia.</pre>

Code	Explanation	Example
U	Convert text string to upper-case. The parameter is an arbitrary text string. Each lower-case letter in the parameter string is replaced by the upper-case equivalent. Other characters are left unchanged.	<pre>.SV var=Smith Please call Mr. &U'&var before you go home.</pre>
X	Convert hexadecimal value to a character. The parameter must be a hexadecimal value. The hexadecimal value is converted to its one-character long equivalent. You can specify only one hexadecimal value as the parameter.	<pre>.SV hex=C5 The hexadecimal value C5 denotes &X'&hex .SV hexa=X'C5 The hexadecimal value C5 denotes &hexa</pre>

Specifying the Text Function as a Parameter

You can specify a text function as a parameter of the `.SV` or `.CV` instruction. To do so, you must specify the required function code followed by the parameter. You can then include the variable in the running text; it must be preceded by the variable character. Initially, the variable character is the ampersand (&). For example:

```
.SV var=H'123
The hexadecimal value of 123 is &var
```

The above instructions cause the following formatted output:

```
The hexadecimal value of 123 is F1F2F3
```

Specifying the Text Function in the Running Text

When you use the text function in the running text, the function code must be preceded by the variable character. Furthermore, you must specify a defined variable after the text function. The variable must also be preceded by the variable character. Initially, the variable character is the ampersand (&). For example:

```
.SV var=123
The hexadecimal value of 123 is &H'&var
```

The above instructions cause the following formatted output:

The hexadecimal value of 123 is F1F2F3

Decimal Numbers

You can use the following options to specify how decimal numbers are to be output when arithmetic calculations are included in variables.

.OP DAS - Number of Characters After the Decimal Character

```
.OP DAS=number
```

Initially, the result of an arithmetic calculation is rounded to the nearest integer. To avoid this, you must specify the number of characters which are to be output after the decimal character. The decimal character is defined using the [.OP DEC instruction](#).

For example, if 2 characters are to be output after the decimal character, you must specify:

```
.OP DAS=2
```

You can specify a maximum of 7 characters after the decimal character.

The decimal character is *not* inserted automatically when specifying the `.SV` or `.CV` instruction as in the following example:

```
.OP DAS=2  
.SV number=1  
&number
```

The above instructions cause the following formatted output:

```
1
```

To output the above variable with a decimal character, you should perform the following additional calculation step:

```
.OP DAS=2  
.SV number=1+0  
&number
```

The above instructions cause the following formatted output:

1.00

.OP DEC - A Different Decimal Character

```
.OP DEC=character
```

Initially, the Natural decimal character is used. By default, this is the period (.). You can define a different decimal character. For example, to define the comma as the new decimal character, you must specify:

```
.OP DEC=','
```

Since the comma is used by certain Con-form instructions to separate parameters, it is important that you enclose it in apostrophes as shown above. If you want to define, for example, the slash (/), you need not use the apostrophes.

.OP TRI - Thousands Separator Character

```
.OP TRI=character  
.OP TRI=ON  
.OP TRI=OFF
```

Initially, a thousands separator character is not defined.

You can specify the character that is to be used as the thousands separator character. For example, to define the comma as the thousands separator character, you must specify:

```
.OP TRI=','
```

Since the comma is used by certain Con-form instructions to separate parameters, it is important that you enclose it in apostrophes as shown above. If you want to define, for example, the slash (/), you need not use the apostrophes.

The thousands separator character is inserted in the formatted output as the result of an arithmetic calculation. It is *not* inserted when specifying the .SV or .CV instruction as in the following example:

```
.OP TRI=','  
.SV number=2000  
&number
```

The above instructions cause the following formatted output:

```
2000
```

To output the above variable with a thousands separator character, you should perform the following additional calculation step:

```
.OP TRI=', '  
.SV number=2000+0  
&number
```

The above instructions cause the following formatted output:

```
2,000
```

If you no longer want to use the thousands separator character, you can deactivate it using the following instruction:

```
.OP TRI=OFF
```

If you want to reuse the previously defined, deactivated thousands separator character, you can activate it using the following instruction:

```
.OP TRI=ON
```

If you specify `.OP TRI=ON` and a thousands separator character has not yet been defined, the comma is used by default.

The Variable Character

All variables in the document text (i.e. the variables you defined using the `.SV` or `.CV` instruction as well as the system variables) must be preceded by the variable character. If required, you can define another variable character or switch the recognition of the variable character off.

.OP VSG - A Different Variable Character

```
.OP VSG=character
```

This instruction defines the character that is used to distinguish text and variables.

Initially, the variable character is the ampersand (&). However, you can define a different character. For example, to define the paragraph sign (§) as the variable character, you must specify:

```
.OP VSG=§
```

It is not possible to use the Dollar sign (\$) as the variable character for system variables. Thus, it is not possible to specify, for example, `$$PL`. However, it is possible to define the following:

```
.SV PL=&§PL  
.OP VSG=§  
§PL
```

.SU - Substitution

```
.SU ON  
.SU OFF
```

When substitution is switched on, each string in the source text which is preceded by the variable character is interpreted as a variable and replaced accordingly.

Initially, substitution is switched off. However, it is automatically switched on by the `.SV` or `.CV` instruction. This corresponds to the following:

```
.SU ON
```

Substitution is also switched on by macro calls *with* parameters. It is not switched on by macro calls without parameters.

You can switch the recognition of the variable character off. This is necessary if the source text contains strings that include the variable character and you do not want them to be interpreted as variables.

To switch substitution off and thus cancel the effect of the variable character, you must specify:

```
.SU OFF
```


System Variables

In addition to the variables you define yourself (using the `.SV` or `.CV` instruction), you can also use system variables in your source text. System variables can contain values such as the current date and time, or the current text margins.

When you specify a system variable, it must be preceded by the variable character. Initially, the variable character is the ampersand (&). For example, to use the system variable `$DT`, you must specify it as follows:

```
&$DT
```

In addition to inserting a variable in the running text, you can also use it with the instructions `.IF` and `.WH`.

Fixed System Variables

Fixed system variables cannot be modified. Con-form automatically replaces the variables below with their actual values when the document is formatted.

Variable	Explanation	Value
<code>\$CN</code>	Century (the first two digits of the year).	19 or 20
<code>\$DA</code>	Day.	1...31
<code>\$DD</code>	Day name in Danish.	Mandag...Søndag
<code>\$DF</code>	Day name in French.	Lundi...Dimanche
<code>\$DG</code>	Day name in German.	Montag...Sonntag
<code>\$DN</code>	Day name in English.	Monday...Sunday
<code>\$DT</code>	Date. For the months 1 through 9 a blank is inserted before the number of the month (for example, 9. 1.93).	dd.mm.yy
<code>\$DY</code>	Julian date.	1...366
<code>\$HO</code>	Hour.	00...23
<code>\$MD</code>	Month name in Danish.	Januar...December
<code>\$MF</code>	Month name in French.	Janvier...Décembre
<code>\$MG</code>	Month name in German.	Januar...Dezember
<code>\$MI</code>	Minute.	00...59
<code>\$MN</code>	Month name in English.	January...December
<code>\$MO</code>	Month number.	1...12
<code>\$SE</code>	Second.	00...59
<code>\$YE</code>	Year.	00...99

The following example illustrates how to use fixed system variables:

```
The current date is: &$DN., &$MN &$DA., &$CN.&$YE..
```

The above instructions cause the following formatted output:

```
The current date is: Wednesday, October 25, 2000.
```

Modifiable System Variables

The following variables can be used in complex formatting situations. Initially, they are set to default values. However, when one of the instructions shown in the right column is issued, the value of the variable is modified.

Variable	Explanation	Modified by
\$BM	Bottom margin.	.BM
\$CH	Chapter number.	.CH, .SC
\$FM	Footer margin.	.FM
\$FN	Footnote counter. The variable \$FN is incremented every time it is referenced. It is used for consecutive numbering of footnotes.	
\$FS	Footer space.	.FS
\$HM	Header margin.	.HM
\$HS	Header space.	.HS
\$IN	Indentation.	.LM, .OF, .TI, .CS
\$IX	Last index entry.	.IX
\$LC	Remaining lines on page.	.SV, .LS
\$LL	Line length.	.LL, .CS
\$PL	Page length.	.PL, .LS
\$PN	Current page number. When the variable \$PN is replaced in your text, Arabic page numbering is used. With the .SV or .CV instruction and in the top and bottom titles, the page-number character can be used instead of \$PN. In this case, the page number is always output according to your specifications (either Arabic or Roman numbers). Initially, the page-number character is the hash (#).	.NP, .PN
\$RM	Right margin.	.RM, .CS
\$RR	Remainder of division when .OP REM=ON.	.CV
\$TM	Top margin.	.TM

The above modifiable system variables return values which have been internally saved by Con-form. However, it is not guaranteed that the system variables always produce the same results in different environments and with different versions of Con-form.

The following example illustrates how to use modifiable system variables. It also introduces the **.IF instruction**.

When less than 5 lines are available on the current page, the box is output on the next page. The current page number is output in the box.

```
.SL 1
.IF &$LC < 5;.NP
.BX 10,50
.LM 10;.RM 49
.IL 1;.CE 1
This box is printed on Page &$PN..
.IL 1;.BX OFF
```

The above instructions cause the following formatted output:

When less than 5 lines are available on the current page, the box is output on the next page. The current page number is output in the box.

```
+-----+
!           !
!   This box is printed on Page 1.   !
!           !
+-----+
```


10 Conditional and Repetitive Processing

▪ Conditional Processing	227
▪ Repetitive Processing	232
▪ Defining the Relation	236
▪ Defining the Consequence	239
▪ Logical Operators	240

This chapter covers the following topics:

When you use the **.EP instruction** in an .IF construction, formatting is always stopped.

Conditional Processing

.IF - Start Conditional Processing

```
.IF relation
```

Conditional processing is started using the .IF instruction. The .IF instruction defines a relation (see [Defining the Relation](#)).

When the result of the relation is true, the defined consequence is processed (see [Defining the Consequence](#)). When the result of the relation is false, the defined consequence is skipped.

When you use the .IF instruction, you can either define a simple consequence (see below) or a compound consequence (see the [.TH instruction](#)).

The Simple Consequence

A simple consequence consists of either a single line of text or an instruction after the .IF instruction. In the following example, the output "Congratulations" is the simple consequence:

```
.IF &age = 50  
Congratulations
```

The simple consequence terminates the conditional statement.



Note: You should not use the .EI instruction (End-If) after a simple consequence.

You can also use the logical operators .AN or .OR (see [Logical Operators](#)). In the following example, the output "Dear Mr." is the simple consequence:

```
.IF &age = 50;.AN &sex = M  
Dear Mr.
```

If the consequence is an instruction, you can define it in the same line as the .IF instruction. In the following example, the instruction .NP is the simple consequence:

```
.IF &$LC < 5;.NP
```


.TH - Define a Compound Consequence (Then)

```
.TH
```

A compound consequence consists of several text lines and/or instructions after the .IF instruction. It must be introduced with the .TH instruction and terminated with the .EI instruction. For example:

```
.IF &$DN = Monday
.TH
.NP
&$MN &$DA., &$YE
.IL 1
This is the start of a new week.
.EI
```

In the above example, the instructions and text between .TH and .EI form the compound consequence (start a new page, print the current date, insert one blank line and output the specified text).

The instructions .TH and .EI form a pair, i.e. when you use the .TH instruction, it must be followed by a matching .EI instruction.



Note: It is possible to nest relations. The maximum nesting depth is 10.

.EL - Define an Alternative (Else)

```
.EL
```

When you use the .TH instruction, you can also define an alternative. If the result of the defined .IF relation is false, the alternative is processed (instead of the consequence after the .TH instruction).

An alternative can consist of several text lines and/or instructions after the .EL instruction. For example:

```
.IF &sex = M  
.TH  
Mr  
.EL  
Ms  
.EI
```

If the variable &sex in the above example does not have the value M, the alternative between the instructions .EL and .EI (include "Ms" in the formatted text) is processed.

The rules which apply to the consequence (see [Defining the Consequence](#)), also apply to the alternative.



Note: When you use the .EL instruction, you must also use the instructions .TH and .EI.

.EI - End Conditional Processing (End-If)

```
.EI
```

The .EI instruction is used to indicate the end of a compound consequence (including the alternative) which has been introduced using the .TH instruction.



Note: You must use the .EI instruction when you have previously used the .TH instruction.

The whole .IF construction (i.e. all specifications between .IF and .EI) must be specified within the same document.

Repetitive Processing

.WH - Start While-Loop

```
.WH relation
```

Repetitive processing is started using the .WH instruction. The .WH instruction defines a relation (see [Defining the Relation](#)).

The relation is followed by a consequence which can consist of several text lines and/or instructions that are to be repeated (see [Defining the Consequence](#)).

The end of the loop must be indicated by the .EW instruction.

As long as the result of the relation is true, the defined consequence is processed. If the result of the relation is false, the While-loop is terminated and processing continues after the .EW instruction.

The following is an example for a simple While-loop:

```
.SV counter=0
.WH &counter <= 3;.** Loop 3 times.
.SV counter=&counter+1;.** Increment the value of the variable &counter.
Number &counter
.EW
&counter was the last number.
```

The above instructions cause the following formatted output:

```
Number 1
Number 2
Number 3
3 was the last number.
```

The result of the relation must change from true to false. If the result is always true, Con-form loops endlessly. If the result of the relation is false the first time the .WH instruction occurs, it is skipped without being processed at all.



Note: It is not possible to nest While-loops.

Processing a List of Variables with Similar Names

When you use the .WH instruction, you can process a list of variables which have similar names that end with a number.

To do so, you must define an additional numeric variable and increment its value within the loop. Then you must combine the beginning of the name variable with the number variable as shown in the example below:

```
.SV name1=James
.SV name2=Lars
.SV name3=Jason
.SV name4=Kirk
.SV number=1;.** Define the additional numeric variable.
.WH &number <= 4
.** Combine the beginning of the variable &name with the variable &number:
&name&number
.SV number=&number+1;.** Increment the value of the variable &number.
.EW
```

The above instructions cause the following formatted output:

```
James
Lars
Jason
Kirk
```

.WX - Exit from While-Loop

```
.WX
```

The `.WX` instruction can be used to exit from a While-loop before its end is reached and to continue processing after the `.EW` instruction. In the following example, the occurrence of the name "Lars" terminates the execution of the While-loop:

```
.SV name1=James
.SV name2=Lars
.SV name3=Jason
.SV name4=Kirk
.SV number=1
.WH &number <= 4
&name&number
.IF &name&number = Lars
.WX
.SV number=&number+1;
.EW
.SL 1
&name&number was the last name in the loop.
```

The above instructions cause the following formatted output:

```
James
Lars

Lars was the last name in the loop.
```

.EW - End While-Loop

`.EW`

The `.EW` instruction must be used in conjunction with the `.WH` instruction. It indicates the end of the While-loop.

It is not possible to conditionally suppress the processing of the `.EW` instruction by using it in an **.IF construction**.

Defining the Relation

Both the .IF and .WH instructions must be followed by a relation on the same line. A relation consists of two values separated by a relational operator.

If you want to define more than one relation with the .IF or .WH instruction, you must separate the relations by the logical operators .AN or .OR (see [Logical Operators](#)).

Relational Operators

You must specify one of the following relational operators between the two values of a .IF or .WH instruction:

Operator		Description
=	EQ	Equal to
<>	NE	Not equal to
<	LT	Less than
<=	LE	Less than or equal to
>	GT	Greater than
>=	GE	Greater than or equal to
?		Substring
*		Subset

Most relational operators have alternative forms as shown in the table above. For example, the relational operator "Equal to" can be defined by using either the equal sign (=) or the abbreviation EQ. Thus, the following two instructions are identical:

```
.IF &age = 50  
.IF &age EQ 50
```

You must include at least one blank space before and after the relational operator.

The first value (to the left of the relational operator) must be a variable (see [Variables](#)).

The second value (to the right of the relational operator) can either be a variable or a constant (for example, a number or name).

In some cases, it is not necessary to specify the second value. For example, if you want to process a consequence only when the value of a variable is not blank, you can specify:


```
.IF &street NE;.TH;.BR
&street
.EI
```

Enclosing a Constant Within Apostrophes

If a constant includes blank spaces or commas, the constant must be enclosed within apostrophes. For example:

```
.IF &company <> 'Software AG'
```

It is possible to include an apostrophe in a constant which is enclosed within apostrophes by repeating the apostrophe. For example:

```
.IF &company <> 'Software AG''s'
```

It is possible to include apostrophes within a parameter which is not enclosed within apostrophes. For example:

```
.IF &company <> AG's
```

Any number of apostrophes may appear within the constant as long as the first character is not an apostrophe. For example:

```
.IF &company <> Software'AG's
```

Substring (?)

The substring operator ? is used to determine whether the left string is a substring of the right string.

In the following example, the left string is a subset of the right string. Therefore, the relation is considered as true.

```
.IF rat ? scratch
```

In the following example, the left string is *not* a subset of the right string. Therefore, the relation is considered as false.

```
.IF cat ? scratch
```



Note: When you use the substring operator, the first value need not necessarily be a variable. It can also be a constant.

Subset (*)

The subset operator `*` is used to determine whether each character which occurs in the left string also occurs in the right string.

In the following examples, all characters of the left string also occur in the right string. Therefore, all relations are considered as true.

```
.IF a * scratch
.IF chat * scratch
.IF ttt * scratch
```

In the following example, *not* all characters of the left string occur in the right string. Therefore, the relation is considered as false.

```
.IF mat * scratch
```



Note: When you use the subset operator, the first value need not necessarily be a variable. It can also be a constant.

Defining the Consequence

A consequence can be an instruction and/or text. It is processed when the result of a relation is true.

If the consequence is an instruction, you can specify it in the same line as the `.IF` or `.WH` instruction, or in the same line as the last `.AN` or `.OR` operator - separated by the instruction separator character (initially, this is the semicolon). For example:

```
.IF &age > 60;.TH
```

If the consequence is text, it must start in a separate line. For example:

```
.IF &age = 50
Congratulations
```



Note: The above rules also apply for an alternative that is defined using the `.EL` instruction.

Logical Operators

The logical operators `.AN` and `.OR` can be used with the `.IF` and `.WH` instructions.

These operators are treated as separate instructions. Therefore, they must be separated from the preceding relation by the instruction separator character (initially, this is the semicolon). For example:

```
.IF &age = 50;.AN &sex = M
```

When you specify several logical operators in the same line, they are processed from left to right.

However, you can also specify the `.AN` or `.OR` instruction in a separate line. For example:

```
.IF &age = 50  
.AN &sex = M
```

.AN - And

```
.AN relation
```

When you use the logical operator `.AN`, *all* relations must be true. Otherwise the defined consequence is not processed. For example:

```
.SV age=65  
.SV sex=M  
.IF &age GE 65;.AN &sex = F  
Dear Grandma
```

Since only the first relation is true in the above example, the words "Dear Grandma" are not output.

.OR - Or

```
.OR relation
```

When you use the logical operator `.OR`, at least one of the relations must be true. Otherwise the defined consequence is not processed. For example:

```
.SV age=65  
.SV sex=M  
.IF &age GE 65;.OR &sex = F  
Dear Grandma
```

In the above example, only the first relation is true. However, the words "Dear Grandma" are output, since it is only required that one of the two relations is true.

11 Macros

▪ General Information	244
▪ Defining a Macro	245
▪ Calling a Macro	248
▪ Local Macro Variables	248

This chapter covers the following topics:

General Information

A macro is a collection of Con-form instructions and/or text which is useful when the same sequence of instructions or the same passage of text is required repeatedly within one or several source documents.

A macro can contain any Con-form instructions, with the exception of a second .MA instruction (i.e. macros cannot be nested). A macro can contain further macro calls.

You must not define recursive macros (i.e. a macro must not call itself, neither directly nor through other macros).

As a rule, all macros you define should be contained in a separate document, the so-called "formatting profile" which is always processed before your source document (see [Formatting Profiles](#)).

When you do not define a macro in the formatting profile, you can only use it in the document in which it has been defined. In this case, the macro call is only recognized as a valid instruction when it occurs after the macro definition.

Another possibility to make a macro generally available in your cabinet is to create a separate document for each macro definition. You must then specify the following instructions in your source text to call the macro:

```
.EM documentname  
.macroname
```

The *documentname* in the above example is the name of the document that contains the macro definition (see the [.EM instruction](#)). The *macroname* is the name of the macro that is to be called.

Defining a Macro

.MA - Start Macro

```
.MA macro-name
```

The .MA instruction is used to indicate the beginning of the macro definition.

The .MA instruction must be followed by the name of the macro in the same line. The name can be up to 100 characters long. Upper-case and lower-case letters in the macro name are not distinguished.

For example, to define a macro called "defaults", you must specify:

```
.MA defaults
```



Tip: It is recommended that a macro name has at least three characters, so that it cannot be confused with a Con-form instruction.

You can now specify the macro definition (i.e. all required instructions and/or text lines) below the .MA instruction. The following is an example of a macro which contains your default settings:

```
.MA defaults  
.LM 10;.RM 60  
.FI ON;.JU ON  
.OP ESC=/  
.OP HYP=E  
.IF &$LC < 5:.NP  
.ME
```

You must specify the .ME instruction after the macro definition to define the end of the macro.

Disabling a Standard Con-form Instruction

When you use the name of a standard Con-form instruction as the name of a macro, the standard instruction is no longer processed. Instead of the standard instruction, the macro with the same name is processed every time it is specified in the source document.

This feature can be used to disable a standard Con-form instruction. For example, the macro below has no effect, since no instruction or text has been defined for it. However, it disables the Con-form instruction .NP which normally causes a form feed. This is useful when you want to print a document for proofreading where it is more important to save paper than to maintain the final page layout.

.MA NP
.ME

.MX - Exit from Macro

```
.MX
```

You can insert the .MX instruction at any point within the macro definition to exit the macro before its end is reached and to continue processing after the .ME instruction. The macro definitions after the .MX instruction are not processed.

When you use the .MX instruction with the **.IF instruction**, it is recommended that you only use it with a simple consequence that does not require the .EI instruction.



Note: When you use the .MX instruction with a compound consequence (.TH) or alternative (.EL) and you exit from the macro before the .EI instruction occurs, Con-form does not recognize that you have specified the .EI instruction and outputs an error message.

.ME - End Macro

```
.ME
```

The .ME instruction is used to indicate the end of the macro definition.

You can optionally specify the name of the macro after the .ME instruction. For example:

```
.ME defaults
```

It is not possible to conditionally suppress the processing of the .ME instruction by using it in an **.IF construction**.

Calling a Macro

To call a macro, you must specify its name in the source text, preceded by a period. For example, to call the macro with the name "defaults", you must specify:

```
.defaults
```

As a result, the instructions and/or text lines which have been defined for the macro are processed. Processing begins immediately following the `.MA` instruction and continues until either an `.MX` instruction or the `.ME` instruction which indicates the end of the macro is encountered.

Local Macro Variables

When you define local macro variables, you can pass, for example, a name and address as parameters to a macro which produces a standard letter.

In addition to the local macro variables, you can also use variables which have been defined with the `.SV` instruction or system variables in the macro definition or as parameters in the macro call. However, you should be careful when you design macros which alter the values of these variables; confusion can easily result if several macros modify the same variable.

Substitution (see the [.SU instruction](#)) is automatically switched on by a macro call *with* parameters. Substitution is not switched on by a macro call without parameters.

Defining a Macro that Requires Parameters

You can include up to 99 local macro variables in the macro definition. The local macro variables are named \$1, \$2, \$3, ... \$99. Each local macro variable in the macro definition must be preceded by the variable character. Initially, the variable character is the ampersand (&).

Later, when you call the macro, you must specify the parameters (i.e. the appropriate values for the variables) after the macro name (see [Calling a Macro that Requires Parameters](#)).

In the following macro definition, the variable \$1 determines the number of blank lines to be inserted in the document text, and variable \$2 determines the text to be output below the blank lines:

```
.MA figure
.IL &$1
.CE 1
&$2
.SL 2
.ME
```

The variable \$1 in the macro definition must be the first parameter in the macro call. The variable \$2 in the macro definition must be the second parameter in the macro call. Thus, to call the above macro, you must specify the macro call with two parameters:

```
.figure 5 Macros
```

As a result, 5 blank lines (variable \$1) are inserted in the document text and the centered text string "Macros" (variable \$2) is output below the blank lines.

Determining the Number of Parameters in the Macro Call

The local macro variable \$0 contains the number of parameters which have been specified after the macro name. For example:

```
.figure 5 Macros
Number of parameters in the current macro call: &$0
```

In the above example, two parameters have been defined after the macro name. As a result, the value 2 is assigned to the local macro variable \$0. Thus, above example causes the following formatted output:

```
Number of parameters in the current macro call: 2
```

When you call a macro without a parameter, the value 0 (zero) is assigned to the local macro variable \$0.

Ensuring that the Macro is Called with the Correct Number of Parameters

You can use the local macro variable \$0 to ensure that a macro is processed with the correct number of parameters. For example:

```
.MA figure
.IF &$0 NE 2
.TH
*****
This macro requires two parameters.
Parameter 1: Number of lines to be inserted for the figure.
Parameter 2: The caption for the figure.
*****
.EL
.IL &$1
.CE 1
&$2
.EI
.ME
```

In the above example, the .IF instruction is used to ensure that exactly two parameters are specified with the macro call. When more or fewer than 2 parameters are defined, a message is output.

Calling a Macro that Requires Parameters

When a macro requires parameters, you must supply the appropriate values after the macro name - separated from it by a single space. For example:

```
.chap1 Introduction
```

Enclosing a Parameter Within Apostrophes

If a parameter contains spaces or commas, you must enclose the parameter within apostrophes. For example:

```
.chap1 'How to Use this Manual'
```

When you omit the apostrophes in the above example, Con-form tries to process five parameters, each word being a separate parameter.



Note: Macro parameters that do *not* include spaces or commas may be enclosed within apostrophes if desired, but this is not necessary.

You can include an apostrophe in a macro parameter which is enclosed within apostrophes by repeating the apostrophe. For example:

```
.chap1 'All in a Day''s Work'
```

You can include apostrophes within a parameter which is not enclosed within apostrophes. For example:

```
.chap1 Day's
```

Any number of apostrophes may appear within the parameter as long as the first character is not an apostrophe. For example:

```
.chap1 All'in'a'Day's'Work
```

Defining More Than One Parameter

If the macro has more than one parameter, you must insert a comma and/or space between two parameters. The example below has two parameters, separated by a comma (it is also possible to separate the parameters by a space):

```
.example 15, 'Defining a Macro'
```

Each parameter you specify in the macro call is assigned to a local macro variable. In the above example, the value 15 is assigned to the variable \$1 and the text string "Defining a Macro" is assigned to the variable \$2.

The local macro variables are only available within the called macro. As soon as the execution of the macro is completed, the local macro variables are no longer available.

12

Controlling the Output

▪ Defining the Pages to be Output	254
▪ Defining the Paper Tray	256
▪ Blank Spaces	258
▪ Translating Characters	261
▪ Altering the Text Orientation	268
▪ Problem Handling	270

As a rule, the term "output" refers to a formatted version of a document which is printed on paper, displayed on the screen or filed as a new document.

This chapter covers the following topics:

Defining the Pages to be Output

.OP NPG - Number of Pages to be Output

```
.OP NPG=number
```

You can specify the number of formatted pages you want to output.

For example, if you want to output the first 5 pages of a document, you must specify the following instruction at the beginning of the source text:

```
.OP NPG=5
```

The index and table of contents are always output after the document text. They are not affected by this instruction.

.OP STA - Start Output at a Specific Page

```
.OP STA=number
```

You can start the output of the formatted document at a specific page number. All pages before the specified page number are not considered.

For example, to start the output at page 11, you must specify the following instruction at the beginning of your source text:

```
.OP STA=11
```



Note: When you specify the above instruction, for example, on page 5, all pages up to the instruction on page 5 are also printed.

When the pages that are output contain titles and index entries, a table of contents and an index are output for these pages only.

.OP STO - Stop Output at a Specific Page

```
.OP STO=number
```

You can stop the output of the formatted document at a specific page number. All pages after the specified page number are not considered.

For example, to stop the output at page 20, you must specify the following instruction at the beginning of your source text or before page 20 occurs:

```
.OP STO=20
```

When the pages that are output contain titles and index entries, a table of contents and an index are output for these pages only.

Defining the Paper Tray

When your printer has two paper trays, you can define from which paper tray the sheets are to be taken.

The instructions `.OP BIN` and `.OP SSF` must be used in conjunction with a printer profile.

.OP BIN - Paper Bin

```
.OP BIN=1  
.OP BIN=2
```

You can select one of two paper trays.

To select paper tray 1, you must specify:

```
.OP BIN=1
```

To select paper tray 2, you must specify:

```
.OP BIN=2
```

When the `.OP BIN` instruction occurs in the source text, Con-form internally creates the following symbols:

F1	Form feed from paper tray 1 (for <code>.OP BIN=1</code>).
F2	Form feed from paper tray 2 (for <code>.OP BIN=2</code>).

When you do not specify a paper tray at all, Con-form internally creates the following symbol:

FF	Form feed.
----	------------

For each symbol, you must define the appropriate printer command sequence (which is documented in your printer manual) in your printer profile. See the *Con-nect User's Guide*, section *Printer Profiles* for further information.

.OP SSF - Single Sheet Feeder

```
.OP SSF=ON  
.OP SSF=OFF
```

If the first sheet is to be taken from tray 1 and the following sheets are to be taken from tray 2, you must specify:

```
.OP SSF=ON
```

This is useful, when you want to print the first page of a document with thicker, headed or colored paper.



Note: When .OP SSF=ON is specified, any conflicting .OP BIN settings are ignored.

Initially, all sheets are taken from the same paper tray. This corresponds to the following:

```
.OP SSF=OFF
```

Blank Spaces

.CB - Compress Blanks

```
.CB ON  
.CB OFF
```

You can replace all multiple blanks in your source document with single blanks. If you want to use this feature, filling must be switched on (.FI ON).

To replace all multiple blanks with single blanks, you must specify:

```
.CB ON
```

When both filling and justification are switched on (.FI ON; .JU ON), single blanks may subsequently be "stretched" (i.e. additional blanks are added between words) to create an even right-hand margin.

The initial value of this instruction is:

```
.CB OFF
```

.SB - Skip Blanks at the Beginning of a Line

```
.SB ON  
.SB OFF
```

One or more blank spaces at the beginning of a line cause a break in line filling. To ignore the blank spaces at the beginning of a line, you must specify:

```
.SB ON
```

The initial value of this instruction is:

```
.SB OFF
```

Unlike the `.CB` instruction, you can also use the `.SB` instruction when filling is switched off (`.FI OFF`). In this case, the line starts at the left margin - without the leading blanks.

You can also create "hard spaces" (see the [.TR instruction](#)).

Example

This example illustrates the instructions `.CB` and `.SB`.

Source Text

```
.LM 0;.RM 65
.FI ON;.JU OFF
.CB ON
You can replace all occurrences of multiple blanks by single
blanks.
.IL 1
.FI ON;.JU ON
However, when justification has been switched on, single
blanks may be 'stretched' again to create an even right-hand
margin.
.CB OFF
.IL 1
A space character at the beginning of a line causes
a break in line filling. However, the formatted output also
has the space in the beginning of the line.
.IL 1
.SB ON
When the .SB instruction is switched on, a blank space
at the beginning of a line is ignored.
.IL 1
.FI OFF
Unlike the .CB instruction, you can also use the .SB instruction
when filling has been switched off.
```

Formatted Output

You can replace all occurrences of multiple blanks by single blanks.

However, when justification has been switched on, single blanks may be 'stretched' again to create an even right-hand margin.

A space character at the beginning of a line causes a break in line filling. However, the formatted output also has the space in the beginning of the line.

When the `.SB` instruction is switched on, a blank space at the beginning of a line is ignored.

Unlike the `.CB` instruction, you can also use the `.SB` instruction when filling has been switched off.

Translating Characters

You can specify that a specific character in your source text is to be replaced with another character or string in the formatted output.

.TR - Translate Character to Another Character

```
.TR input-character output-character  
.TR
```

You use the .TR instruction to output a specific character in your source text as a different character in the formatted output.

The parameters of the .TR instruction must be either single characters or two-digit hexadecimal representations of the characters.

The first parameter (input character) defines the character that you want to convert, and the second parameter (output character) defines how the character is to be output. There must always be a space between the two parameters.

You can specify as many different .TR instructions as required.

Single Characters

For example, to convert all occurrences of the lower-case character "a" to the upper-case character "A", you must specify:

```
.TR a A
```

If you want to use the instruction separator character (initially, this is the semicolon) as a parameter, you must repeat it (";;").

Hexadecimal Representations

For example, when you work on a mainframe and want to convert all occurrences of the lower-case character "a" to the upper-case character "A", you must specify the .TR instruction with the appropriate EBCDIC character codes:

```
.TR 81 C1
```

If Con-form is installed on a mainframe, the following character codes should not be used with the .TR instruction (where X is the character to be translated):

```
.TR X 00  
.TR X 03  
.TR X 04  
.TR X 0E  
.TR X 0F  
.TR X 27  
.TR X 3F  
.TR X 40  
.TR X FE
```

Creating a "Hard Space"

You can define hard spaces in your source text when you want to ensure that no additional blanks are added between two words as a result of the justification process. Furthermore, a hard space is not considered as a possible position for a line break.

To define a hard space, you must specify a hexadecimal output character.

The EBCDIC character code for the hard space is 40. For example:

```
.TR ^ 40  
See Chapter^7 for additional information.
```

In the above example, the defined input character (^) has been inserted between the word "Chapter" and the chapter number "7". In the formatted version, these are output in the same line with exactly one space in between.

Canceling a Single Translation

To cancel a single translation, both parameters of the .TR instruction must be the same. For example, to cancel the translation caused by the instruction .TR a A, you must specify:

```
.TR a a
```

Canceling All Translations

To cancel all translations, you must specify the `.TR` instruction without parameters:

```
.TR
```

All translations are processed *after* a source text line has been formatted. Thus, the position of the `.TR` instruction (which cancels all translations) is important, especially when filling is switched on (`.FI ON`). This is illustrated by the following example:

```
.LM 0;.RM 72
.FI ON;.JU OFF
.TR a A
Each lower-case a is to be translated into an upper-case a.
.TR
However, the formatted output produces an unexpected result since
the .TR instruction (which cancels all translations) occurs at a position
in the source text, where the whole line has not yet been formatted
(filled).
.IL
.TR a A
If you want to replace a specific character in a paragraph with another
character, the .TR instruction (which cancels all translations) should
be included after the next break in filling.
.IL;.TR
```

The above instructions cause the following formatted output:

```
Each lower-case a is to be translated into an upper-case a. However, the
formatted output produces an unexpected result since the .TR instruction
(which cancels all translations) occurs at a position in the source
text, where the whole line has not yet been formatted (filled).
```

```
If you wAnt to replAce A specific chArActer in A pArAgrAph with Another
chArActer, the .TR instruction (which cAncels All trAnslAtions) should
be included After the next breAk in filling.
```

.TS - Translate Character to String

```
.TS input-character output-string
.TS input-character
.TS
```

You use the .TS instruction to translate a specific character into a character string.

The first parameter (input character) of the .TS instruction defines the character that you want to convert, and the second parameter (output string) defines how the character is to be output. There must always be a space between the two parameters.

As with the .TR instruction, the input character must either be a single character or a two-digit hexadecimal representation of the character. The output string can be several characters long and can contain blank spaces; it must *not* contain hexadecimal values.

You can specify as many different .TS instructions as required.



Tip: If you want to translate a single character into another single character, it is recommended that you use the .TR instruction.

For example, to convert all occurrences of the character "/" to the string "Slash", you must specify:

```
.TS / Slash
```

When the input character re-appears in the output string, it is not translated once more. If you want to use the instruction separator character (initially, this is the semicolon) as a parameter, you must repeat it (";;").

Unlike the .TR instruction, all translations are processed *before* a source text line is formatted.



Note: All characters in the source text are processed, including those which were introduced as a result of the variable substitution process. See the [.SV instruction](#) for further information.

Canceling a Single Translation

To cancel a single translation, you must specify the character which has previously been defined as the input character. You must not specify the output string.

For example, to cancel the translation caused by the instruction ".TS / Slash" (see above), you must specify:

```
.TS /
```

Canceling All Translations

To cancel all translations, you must specify the .TS instruction without parameters:

```
.TS
```

Example

This example illustrates the instructions .TR and .TS.

Source Text

```
.TR - =
All occurrences of the hyphen are replaced by an equal sign.
-----
.TR - -;.** Cancel the previous translation.
Now, the hyphen (-) is not translated in the formatted output.
.LM 0;.RM 45
.FI ON;.JU ON
.IL 1
.** Use the underscore as the hard space.
.** This example works in a mainframe environment. Thus, 40
.** is the hexadecimal representation for a space character.
.TR _ 40
When you define hard_spaces in your source_text, no additional
spaces are included in the formatted_output. Furthermore, a line_break
does not occur at the position of a hard space.
.TR;.IL 1
.LM 0;.RM 70
.TS * This is the output string caused by the asterisk (*).
.TS / This is another output string caused by the slash (/).
When the asterisk appears in the source text, the following is output: *
When the slash appears in the source text, the following is output: /
.IL 1
.TS;.** Cancel all translations.
As you can see in the previous example, you can include the input
character (* and /) in the output string. In this case, the character
is not translated once more.
```

Formatted Output

All occurrences of the hyphen are replaced by an equal sign.

Now, the hyphen (-) is not translated in the formatted output.

When you define hard spaces in your source text, no additional spaces are included in the formatted output. Furthermore, a line break does not occur at the position of a hard space.

When the asterisk appears in the source text, the following is output: This is the output string caused by the asterisk (*). When the slash appears in the source text, the following is output: This is another output string caused by the slash (/).

As you can see in the previous example, you can include the input character (* and /) in the output string. In this case, the character is not translated once more.

Altering the Text Orientation

.TO - Text Orientation

```
.TO L  
.TO R
```

This instruction is useful, when you want to include text in your document which is to be processed from right to left during the formatting process (for example, when you want to include a Hebrew quote in your English text).

This instruction does not affect rendition or line formatting since Con-form assumes that the respective text is processed by dedicated hardware. It simply passes the value for the predominant text orientation to the routine which is used to convert RFT-DCA documents into Con-form documents and vice versa.

Initially, text is processed from left to right. This corresponds to the following setting:

```
.TO L
```

To process text from right to left, you must specify the following instruction:

```
.TO R
```

The meaning of "left" and "right" as used in instructions such as .LM and .RM must now be interpreted differently. For example, the left margin is the side where you start writing (which is physically the right side) and the right margin is the side where you stop writing (which is physically the left side).

Even when text is processed from right to left, Arabic page numbers (as defined using the page number character) are always processed from left to right. Initially, the page number character is the hash (#).



Note: See [Text Functions](#) for information regarding the text function I which inverts the rendition of a variable's contents.

Using Escape Sequences to Alter the Text Orientation

You can also use escape sequences to alter the direction in which your text is to be processed. Before you can issue an escape sequence, you must define the escape character (see [.OP ESC - Escape Character](#)).

When your text is predominantly processed from left to right, you can include the following symbols in your source text to change the orientation for the marked text:

Z1	Switches right-to-left orientation on.
Z0	Switches right-to-left orientation off.

When your text is predominantly processed from right to left, you can include the following symbols in your source text to change the orientation for the marked text:

A1	Switches left-to-right orientation on.
A0	Switches left-to-right orientation off.

The above symbols are processed during line formatting. You can also use them with other symbols (such as M1 and M0); in this case they must be properly nested.

You must not use the tab character (see [.TB - Set Tab Stops](#)) for text containing the above symbols.

Problem Handling

.OP ECH - Echo Instructions

```
.OP ECH=ON  
.OP ECH=OFF
```

You can echo (list) all Con-form instructions. This is useful, if your formatted output does not show the desired results and you want to check whether you used the Con-form instructions correctly in your source text.

To echo the Con-form instructions, you must specify:

```
.OP ECH=ON
```

Initially, this feature is switched off. This corresponds to the following:

```
.OP ECH=OFF
```

When you want to check only part of your source document, you can include the text to be echoed between the above two instructions.



Note: The Con-form instructions are not echoed in all environments.

.DU - Dump Workspace

```
.DU
```

This instruction is a diagnostic aid and should only be used when requested by the Software AG support. You insert the .DU instruction in your source text, just before the point at which the program terminated abnormally and then print a formatted version of the source document.

13

Formatting a Document

- The FORMAT Command 272
- Printer Profiles 272
- Formatting Profiles 273

When you create a document in Con-nect, the document format for a document containing Con-form instructions is Cnf.

This chapter covers the following topics:

The FORMAT Command

To process all Con-form instructions in your document text, you must issue the FORMAT command in conjunction with the document. As a result, you access the "Format *Document-name*" screen in which you can specify whether you want the formatted version of the document to be displayed, printed or filed. You can also specify that the formatted version of the document is to be placed in the Con-nect editor so that you can further edit it. For further information, see the description of the FORMAT command in the *Con-nect User's Guide*, section *Documents*.

When you display a formatted version, the display on your screen is dependent on the environment in which you are working. For example, in a mainframe environment it is possible that emphasized text (such as boldface or underscoring) is highlighted on your screen, but you cannot distinguish the format type. However, when you print the formatted document, the printout shows the desired results - provided that you use the correct printer profile.

You can TRANSLATE an RFT document into the Cnf format. For further information, see the *Con-nect User's Guide*, section *Translating Documents*.

Printer Profiles

Certain features of Con-form require a printer profile. The printer profile is dependent upon the environment in which you are working. For detailed information on how to create a printer profile, see the *Con-nect User's Guide*, section *Printer Profiles*.

Con-form does not support proportional fonts.

The printer name and the name of the printer profile are defined in your Con-nect user profile (see the *Con-nect User's Guide*, section *Print Defaults*).

If you do not use a printer profile, or if there are wrong or missing entries in your printer profile, the printed output may produce unexpected results.

Formatting Profiles

You can define a formatting profile that is to be processed before each of your source documents. The formatting profile can contain, for example, default settings for the page layout. Macros should also be contained in the formatting profile so that they are always available to you (otherwise they are only valid in the document in which they have been defined).

The name FPROFILE is automatically defined as the formatting profile in your user profile (see the *Con-nect User's Guide*, section *Print Defaults*). However, a document called FPROFILE is *not* automatically delivered with Con-nect. It is recommended that your administrator creates a document with this name in the cabinet SYSCNT and makes it thus available for all cabinets.

You can also specify another formatting profile (i.e. the name of another document) in your user profile. In this case, Con-form first processes the system formatting profile in cabinet SYSCNT and then your own. This is useful, for example, when you want to replace a macro in the system formatting profile with an enhanced macro of your own.

When you FORMAT your source document, you must ensure that the "Formatting Profile" field in the "Format *Document-name*" screen is marked. For further information on the FORMAT command, see the *Con-nect User's Guide*, section *Documents*.

14 Special Characters

▪ Ampersand (&)	276
▪ Apostrophe (')	276
▪ Circumflex (^)	276
▪ Comma (,)	277
▪ Commercial-at (@)	277
▪ Dollar (\$)	277
▪ Hash (#)	279
▪ Period (.)	279
▪ Semicolon (;)	279
▪ Slash (/)	280
▪ Space	281

Certain characters have a special meaning in Con-form. In some cases, you can assign the special meaning to a different character. The special characters and their meanings are described in this chapter.

The following topics are covered below:

Ampersand (&)

Initially, the ampersand is used as the variable character.

You can define a different character. For example, to define the paragraph sign as the variable character, you must specify:

```
.OP VSG=$
```

To cancel the special effect of the variable character, you must specify:

```
.SU OFF
```

Apostrophe (')

If a parameter contains spaces or commas, you must enclose it within apostrophes.

If a constant of a .IF or .WH instruction includes commas or blank spaces, you must enclose it within apostrophes.

Circumflex (^)

Initially, the circumflex is used as the shift character. It is only effective when lower-case is switched on (.LO ON).

You can also define a different character. For example, to define the asterisk as the new shift character, you must specify:


```
.OP SHI=*
```

Comma (,)

The comma is used to separate the parameters of an instruction.

If a macro has more than one parameter, you must insert a comma or space between two parameters.

If a constant of a .IF or .WH instruction includes commas or blank spaces, you must enclose it within apostrophes.

Commercial-at (@)

Initially, the commercial-at sign is used as the shiftlock character. It is only effective when lowercase is switched on (.LO ON).

You can also define another shiftlock character. For example, to define the hash as the new shiftlock character, you must specify:

```
.OP LOC=#
```

Dollar (\$)

Initially, the dollar sign is used as the end-of-line character.

The dollar sign is only interpreted as the end-of-line character when it is the last character in a source text line.

You can define a different character. For example, to define the percent sign as the end-of-line character, you can specify one of the following instructions:

```
.OP END=%
```

. EC %

Hash (#)

Initially, the hash is used as the page-number character. You can use it in top and bottom titles and with the `.SV` instruction.

The hash always represents the number of the current page. There is one exception: when you use the hash in the bottom title, it represents either the number of the current page (`.OP PAG=EQU`) or the number of the next page (`.OP PAG=DIF`).

You can define a different character. For example, to define the exclamation mark as the page-number character, you must specify:

```
.OP PNS=!
```

Period (.)

A period in the first column of a source text line is used to indicate a Con-form instruction.

To cancel the special effect of the period, so that a period in the first column is not interpreted as a Con-form instruction, you must specify:

```
.IC ON
```

You can also use a period to output another variable or text after the (first) variable without a blank in between (for example `"&var1.&var2"`).

Semicolon (;)

Initially, the semicolon is used as the instruction separator character.

You can define a different character. For example, to define the colon as the instruction separator character, you must specify:

.OP CSE=:

Slash (/)

The slash is used as the separator character in top and bottom titles. It separates the three different areas for left-justified, centered and right-justified text. The text you define for each area must not include the slash.

Space

If you insert one or more blank spaces at the beginning of a line in your source text, you cause a break in filling. However, the formatted text also has the blank spaces in the beginning of the line.

If an instruction or macro has one or more parameters, you must insert a space between the instruction or macro and the first parameter.

If a macro has more than one parameter, you must separate the parameters by inserting a space or comma between them.

If a parameter includes spaces or commas, you must enclose it within apostrophes.

If you use the `.IF` or `.WH` instruction, you must include at least one blank space before and after the relational operator.

If a constant of a `.IF` or `.WH` instruction includes blank spaces or commas, you must enclose it within apostrophes.

15 Form Letters

■ Using the COMPOSE Statement	284
■ Testing Your Form Letter	286

With Con-form you are able to extract data from Adabas files and thus to automatically create documents which contain the extracted data. To do so, you must create a Natural program which makes use of the COMPOSE statement (see the Natural documentation for further information).

This chapter covers the following topics:

Using the COMPOSE Statement

The following example shows one of several possible ways to perform mass mailing. It is assumed that you work with Con-nect on a mainframe (see the *Con-nect User's Guide* for information regarding procedures and documents).

To create a form letter, you can proceed as follows:

1. Create a Natural program to extract data from the database.

The Natural program must contain the name of the Adabas file and the fields to be used from that file. Furthermore, the Natural program must contain the COMPOSE ASSIGNING statement; for example:

```
COMPOSE ASSIGNING
  'SALUT' = SALUTATION
  'NAME' = LASTNAME
  'STREET' = STREET
  'TOWN' = CITY
```

The ASSIGNING clause assigns values to the Con-form variables being used in your document. The operands on the left side of the equal sign (enclosed in single quotes) are the variable names that you use in your document. The operands on the right side of the equal sign are the fields being extracted from Adabas.

2. Create a document in Con-nect using the command sequence "ADD Document". This is the skeleton for your form letter containing the text and the variables to be substituted with information from the Adabas file.

```
.LM 0;.RM 60
&salut &name
&street
&town
.SL
Dear &salut &name.,
.SL
.LM 0;.RM 60
Your subscription with MAGNIFICENT WILDLIFE magazine will soon expire.
If you act now and renew your subscription for one full year, you will receive
```



```
a 40% discount - a savings of $25.00 off the newsstand price!  
.SINCERELY
```



Note: The macro .SINCERELY in the above example has been defined in the FPROFILE.

3. Use the command sequence "ADD Procedure" to define the Natural program to Con-nect.
4. Use the command sequence "INFO Document *document-name*" to link the procedure to the Con-nect document. Define the procedure as a pre-formatting procedure.
5. FORMAT the document.



Caution: When you use the FORMAT command in Con-nect, the procedure must not contain COMPOSE statements with clauses other than ASSIGNING or EXTRACTING.

Testing Your Form Letter

The following is an example of a form letter which produces different results depending on whether the addressee is female or male. The example also illustrates the advantage of the `.SV` instruction. It is helpful when you want to test whether your form letter produces the desired results before you have the variables replaced with data from the database.

```
.LM 0;.RM 60
.FI ON;.JU ON
.SV sex=F
.SV salut=Ms.
.SV name=Smith
.SV street=451 Sundown Drive
.SV town=Reston, VA 22091
.SV magname=MAGNIFICIENT WILDLIFE
.** The above variables are helpful, when you want to test
.** whether your form letter produces the desired results.
.** To replace the variables with information from your database, you
.** must create a Natural program which uses the COMPOSE statement.
.** When you replace the variables with information from your database,
.** you must erase the above .SV instructions.
.IF &sex = M
.TH
.SV newmag=FOOTBALL HIGHLIGHTS
.SV text1=highlights from the most recent games
.SV text2=the latest news of the hottest players of the season
.SV phone=FOOT
.EL
.SV newmag=HEALTHY LIVING
.SV text1=new discoveries from the research laboratories, diet tips
.SV text2=recipes which incorporate the latest health findings
.SV phone=HEAL
.EI
&salut &name.$
&street.$
&town.$
.SL 3
.RA ON
&$MN &$DA., &$CN.&$YE
.RA OFF
.SL 3
Dear &salut &name.,
.SL
Your subscription with &magname magazine will soon expire. If you
act now and renew your subscription for one full year, you will receive
a 40% discount - a savings of $25.00 off the newsstand price!
You are a valued customer of ours and we would hate to have you miss
this opportunity.
```

.SL

Additionally, we are introducing a new magazine called &newmag which offers &text1 and &text2..

If you are interested in our newest magazine, simply fill out the form below along with your renewal information or call our toll-free number 800-766-&phone..

.SL

Sincerely,

.SL 4

J. Baker\$

Vice President of Sales

The above instructions cause the following formatted output:

Ms. Smith
451 Sundown Drive
Reston, VA 22091

April 21, 2000

Dear Ms. Smith,

Your subscription with MAGNIFICENT WILDLIFE magazine will soon expire. If you act now and renew your subscription for one full year, you will receive a 40% discount - a savings of \$25.00 off the newsstand price! You are a valued customer of ours and we would hate to have you miss this opportunity.

Additionally, we are introducing a new magazine called HEALTHY LIVING which offers new discoveries from the research laboratories, diet tips and recipes which incorporate the latest health findings. If you are interested in our newest magazine, simply fill out the form below along with your renewal information or call our toll-free number 800-766-HEAL.

Sincerely,

J. Baker
Vice President of Sales

When the variables "sex" and "salut" of the above Con-form document contain the values below, the formatted output contains a different version:

```
.SV sex=M  
.SV salut=Mr.
```

In this case, the formatted output looks as follows:

Mr. Smith
451 Sundown Drive
Reston, VA 22091

April 21, 2000

Dear Mr. Smith,

Your subscription with MAGNIFICENT WILDLIFE magazine will soon expire. If you act now and renew your subscription for one full year, you will receive a 40% discount - a savings of \$25.00 off the newsstand price! You are a valued customer of ours and we would hate to have you miss this opportunity.

Additionally, we are introducing a new magazine called FOOTBALL HIGHLIGHTS which offers highlights from the most recent games and the latest news of the hottest players of the season. If you are interested in our newest magazine, simply fill out the form below along with your renewal information or call our toll-free number 800-766-FOOT.

Sincerely,

J. Baker
Vice President of Sales

16

Deprecated Instructions

▪ .>> - Label	291
▪ .GO - Go to the Specified Label	291
▪ .LO - Lower-Case	292
▪ .OP LOC - Shiftlock Character	292
▪ .OP SHI - Shift Character	294
▪ .PN - Page Number	294
▪ .TY - Type Information	294
▪ .UP - Upper-Case	295

This chapter lists all Con-form instructions, in alphabetical sequence, which are still retained for compatibility with older Con-form versions. However, it is recommended that you do *not* use these instructions.

The following topics are covered below:

.>> - Label

```
.>> label
```

The .>> instruction defines the label which is referenced by the .GO instruction. You must define a unique label within each While-loop or macro.

For example, to define a label called "end", you must specify:

```
.>> end
```

.GO - Go to the Specified Label

```
.GO label
```

The instructions .GO and .>> form a pair. They can only be used within a macro or a While-loop.

The .GO instruction is used to jump to a label which has been defined with the .>> instruction. You can jump forwards or backwards.

For example, to jump to the label called "end", you must specify:

```
.GO end
```

.LO - Lower-Case

```
.LO ON
.LO OFF
```

Initially, lower-case is switched off (i.e. all characters which have been entered in upper-case are also output in upper-case). This corresponds to the following:

```
.LO OFF
```

When your terminal does not provide lower-case letters so that you can only enter upper-case letters, you can define the sections of text which you want to output in lower-case. To do so, you must switch lower-case simulation mode on:

```
.LO ON
```

As a result, all letters are converted to their lower-case equivalents.

The following exceptions apply for the .LO instruction:

- The letter after the shift character (see [.OP SHI - Shift Character](#)) is not converted to lower-case.
- All letters enclosed between two shiftlock characters (see below) are not converted to lower-case.

.OP LOC - Shiftlock Character

```
.OP LOC=character
```

Normally, this instruction is used with .LO ON when your source text is entirely in upper-case. You can put letters between shiftlock characters, so that they are output in upper-case (i.e. they are not converted to lower-case).

The shiftlock character must always appear pairwise in each line of the source text. In order to output larger sections of text in upper-case, it may be more convenient to use the instruction .LO OFF.

Initially, the shiftlock character is the commercial-at sign (@). You can also define a different shiftlock character. For example, to define the hash (#) as the new shiftlock character, you must specify:

`.OP LOC=#`

.OP SHI - Shift Character

```
.OP SHI=character
```

Normally, this instruction is used with .LO ON when your source text is entirely in upper-case. You can put the shift character before a letter, so that it is output in upper-case (i.e. it is not converted to lower-case).

Initially, the shift character is the circumflex (^). You can also define a different shift character. For example, to define the asterisk (*) as the new shift character, you must specify:

```
.OP SHI=*
```

.PN - Page Number

```
.PN number
```

The .PN instruction defines a different page number for the next page. However, it does not cause a form feed.

It is recommended that you use the [.NP instruction](#) instead.

.TY - Type Information

```
.TY text
```

You can write information into the Con-form message area. To do so, you must specify the information as a parameter of the .TY instruction. For example:

```
.TY You are currently on page 5 of the source document.
```

If you want to use the instruction separator character (initially, this is the semicolon) within the text to be output, you must repeat it (";;").

If you do not specify a parameter, a blank line is output.

Variable substitution (see the [.SV instruction](#)), character translations (see the [.TR instruction](#) and the [.TS instruction](#)) and upper-case/lower-case conversion (see the [.UP instruction](#) and the [.LO instruction](#)) are performed on the parameter before the defined text is output.

.UP - Upper-Case

```
.UP ON  
.UP OFF
```

If you have already entered text in lower-case, you can convert it to upper-case.

To print a section of text in upper-case, you must enclose it between the following two instructions:

```
.UP ON
```

```
.UP OFF
```

The .UP instruction does not cause a break in filling.

