

Controlling the Output

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
 - Defining the Paper Tray
 - Blank Spaces
 - Translating Characters
 - Altering the Text Orientation
 - Problem Handling
-

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.