Applications Series
January 29, 1985

THE COMPOSE TEXT FORMATTER

Compose is a powerful text-formatter that allows precise control over all aspects of a printed document's appearance. It also lets you take full advantage of the Information Systems laser printer's many different printing styles. This memo introduces the most commonly used compose instructions and gives examples of what they do. It also describes the Multics command for printing formatted documents. The fonts available on the laser printer are listed and examples of what they look like are given in an appendix.

The following documents may be useful for additional information:

VENDOR DOCUMENTATION

New Users' Introduction to Multics--Part I (CH24)
New Users' Introduction to Multics--Part II (CH25)
Multics WORDPRO Reference Guide (AZ98)
Guide to Multics WORDPRO for New Users (DJ18)
Emacs Text Editor Users' Guide (CH27)
qedx Text Editor Users' Guide (CG40)
Multics Commands and Active Functions Quick Reference Guide (AW17)

INFORMATION SERVICES MEMOS

Guide to Information Services
Multics at MIT (MS-1)
Essential Multics (MS-6)

This memo has been extensively revised and does not contain marginal change indicators.

CONTENTS

I. INTRODUCTION

Word processing is the use of computer programs to help produce memos, reports, letters, and other documents. On mainframe computers like Multics, word processing is handled by two separate facilities rather than one software package. You enter, store, and edit text with a facility called a *text editor*. Once you are through writing and revising your text in the editor, another facility called a *text formatter* gets the text ready for printing (formats it). On Multics, you can choose between several different text editors and several text formatters as well.

Compose is a versatile text formatter that lets you tailor most aspects of your document's appearance on as specific or as general a level as you wish. It gives you control over the way standard text is laid out on the page and handles tedious tasks like fitting in repeating headings and footnotes. You can quickly learn enough basic functions to print out simple documents on the Information Systems printers. By using a fuller range of functions, you can pull together a full-scale sectioned document like this one, including a table of contents and an index.

This is an introductory memo; no prior experience with text formatting is assumed. This first chapter introduces the different computer facilities used for word processing on Multics and describes how they all work together in the production of a printed document. The chapters on basic formatting, font switching and printing provide enough information for the production of most simple documents, while the fourth chapter introduces some of compose's more advanced capabilities.

This memo is not intended as an introduction to either the Multics operating system or its text editors. If you are not familiar with Multics or have not chosen a text editor yet, consult the appropriate manuals from those mentioned in the "Documentation" section of this chapter.

PRODUCING A DOCUMENT WITH COMPOSE

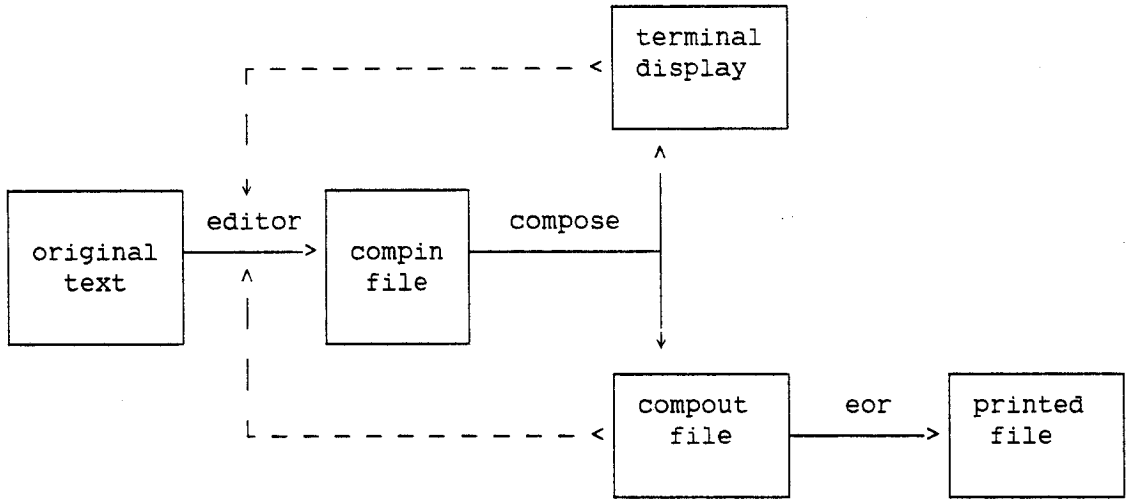
Word processing on Multics can be divided into three different actions: editing, formatting, and printing. Though these actions are all logically distinct, they tend to overlap in practice. The following sections provide an introductory overview of how these different steps fit together when you're actually writing and revising a document (summarized in the diagram on the next page). The rest of this memo will then describe each of these steps in detail.

Creating an Input File with a Text Editor

When creating a document on line, you actually spend a substantial part of your time in a text editor. Unlike text formatters, which are designed specifically for word processing purposes, text editors are general-purpose tools. They can be used whenever you want to enter information and store it on line in a segment of memory (i.e., for writing programs or entering data as well as writing prose.) By typing into a computer editor instead of directly onto paper, you create an on-line copy of your text that can be reworked indefinitely. The basic text is only typed once. In subsequent revisions you can move existing text around at will; only changes or additions require more typing.

To produce a document with compose, use the text editor to create an *input file* (also called a *compin* file) for processing by the text formatter. This input file includes not only your basic text, but instructions to compose on how you want the text to look when it's printed.

The formatting instructions that you add to the input file are called *compose controls*, because they control the way compose formats your text. When adding compose controls you need to think only about page layout. Decisions about how to print the file can be postponed until the next stage, when you format the file. While still learning, it may be easier to add controls as a separate step, after you have finished entering text. As you become increasingly familiar with compose, you will probably type most routine formatting instructions at the same time as your text.



The Document Production Process Using the Compose Text Formatter

Formatting the Input File

Once you have written text that satisfies you and have saved the input file, you are ready to format the text by issuing the "compose" command. Compose processes the specified input file by applying any inserted formatting instructions to the text and producing formatted output.

The resulting output may be placed in an *output file* for later printing or sent directly to your terminal. Examining your formatted text at the terminal is a convenient way of checking for errors before actually printing a document. If you find that the formatted text does not look the way you want it to, re-enter the text editor and take another look at the compose controls you inserted. Once you've corrected the input file, you can run the text formatter again to see the new improved version. Producing a document of any complexity usually requires several trial runs before you are satisfied with the final output. (See above for a diagram of this process.)

You issue two different types of instructions to the compose text formatter at two different stages in the document production process. The formatting instructions described above, those placed inside the input file, control the appearance of specific parts of the text. You can also add control arguments to the "compose" command line when formatting the input file. These control arguments modify the way the document as a whole is formatted.

Most users ultimately want to produce a high-quality printed document on an Information Systems printer. When formatting a compin file for one of these printers, you need to add control arguments to the "compose" command line. The printers feature a variety of lettering

styles; you specify the style of your choice with a control argument. This option can be changed each time you format the file, thus changing the appearance of the resulting output without any modification of the input file. You also use a control argument to direct the formatted output to an on-line segment that becomes your output file (also called a *compout file*).

Printing the Output File

The two Information Systems *laser printers* work most effectively with compose's many functions and are the most frequently used printing option, though compout files can be sent to other printers. Laser printers produce high-quality output on regular 8.5 by 11 inch paper. A Xerox 8700 is located on the second floor of Building 11 and is used for most output requests. Extremely large printing jobs (over 500 pages) are routed to the Xerox 9700 in Building W91. Except for greater speed, the 9700's capabilities are identical to those of the 8700. Any file will be printed in exactly the same way by either machine.

The Multics "eor" command sends the output file to the laser printer. Pick up the printed copy at the Dispatch Area (Room 11-226). The printing process, the "eor" command and its control arguments are described in the chapter "Formatting and Printing your File".

Font Switching

The Information Systems laser printers can print text in a variety of lettering styles, or *fonts.*[1] These fonts fall into two general categories: *fixed-width* and *proportional*. With fixed-width fonts, every letter takes up the same amount of space. Most typewriters and terminals use fixed-width fonts. The laser printers also feature *proportional* fonts, where spacing is determined by individual letter size. Proportional fonts minimize unattractive gaps within words and lines, resulting in output that resembles professionally printed books. For example, the basic font used throughout this memo is a proportional font called "press roman". Currently, compose is the only text formatter supported on Multics that can take advantage of the proportional fonts.

In addition to choosing between a variety of fonts for the document as a whole, you can change fonts within a document. The overall printing style that you choose when formatting your file includes a pre-defined set of fonts in addition to the basic font. For example, you can change to a mathematical symbol font for equations or use a boldface font for chapter and section titles (e.g. the titles on this page). Descriptions of the specific fonts available on the laser printers and instructions on how to use them are given in the chapter "Fonts".

---

[1] For the purposes of this memo, the term *font* is very broadly defined. It can apply both to a general letter style (i.e., press roman, univers) and to specific versions or intensities of that style (i.e., bold, italic, small).

DOCUMENTATION

If you are not familiar with the Multics operating system, it is introduced in a Honeywell document, *New Users' Introduction to Multics--Part I* (CH24) and an Information Services memo, *Essential Multics* (MS-6). On Multics, the two most popular text editors are Emacs and qedx. If you are not familiar with either, consult *Emacs Text Editor Users' Guide* (CH27), or *Multics qedx Text Editor Users' Guide* (CG40). If you want to use Multics only for word processing, the Honeywell *Guide to Multics WORDPRO for New Users* combines a brief introduction to compose with enough information about the operating system and text editing to get you started.

This memo introduces a large subset of the compose text formatter's functions and controls, and describes how to use the laser printers to produce formatted documents. For information on compose's more advanced capabilities, see *WORDPRO Reference Guide* (Honeywell manual AZ98), *the* definitive document on compose. It defines *all* compose controls, and is the only manual to describe the "compose" command and its control arguments.[1] Users truly interested in pushing compose to the limits of its extensive capabilities should refer to this manual. However, it assumes considerable experience with computers and text formatting, and is not a convenient resource for beginners.

Several on-line help files provide information on the current status of the local implementation of compose. Type "help compose.controls" to print out a summary of control words, and "help compose.builtins" to see a list of compose's built-in variables. For information on the latest version of compose, type "help new_compose". Though there are several other compose-related help files, most of them are either out-of-date or intended for fairly advanced users. The Multics forum meeting facility has an entire meeting (called "Compose") devoted to compose questions and issues. This can be a quick way of getting questions answered and finding out about otherwise unpublicized bugs.

SUPPORT

All compose software, except the laser printer support modules, is maintained by Honeywell Information Systems. MIT's Information Systems staff forwards bug reports to Honeywell. The laser printer support software is maintained by Information Systems. We will grant credit for wasted computing or printing charges incurred because of unpublicized errors in software or documentation, or incorrect advice from authorized Information Services consultants. See Memo RT-5, *Credit: Policies and Procedures*, for credit policies. Information Services provides basic assistance in understanding documentation and solving basic problems through the Computing Assistance Office. Assistance in planning applications or solving difficult problems is also available for a fee. However, since Information Systems is currently re-evaluating its support policy for compose and other text formatters, support levels may change in the future.

---

[1] Since "compose" is classified as a WORDPRO (Multics Word Processing system) command, it is not covered in any of the standard Honeywell documentation for Multics commands.

CAUTION . . .

Compose is a powerful and complex text formatter, but it is not entirely free of bugs. Some of the more complicated formats can turn out to be more trouble than they're worth. Most of the controls presented in Chapter II work in a predictable fashion; Chapter IV includes some more erratic functions, but warns you about most of their known limitations. For major documents like theses and grant applications, allow yourself plenty of time for preparation, so that you can try different approaches to a problem if necessary. Information Services will try to help if you encounter a bug, but we cannot guarantee that compose will work properly for all applications.

II. BASIC COMPOSE CONTROLS

The compose text formatter is a computer facility that translates instructions about how you want your text to look into commands for a local printer. Instead of having to learn the strange and difficult language of laser printers,[1] you need to master only a set of abbreviated commands called *compose controls*. The complete set of these controls is extensive, with functions ranging from the simple to the very specialized.

Though this wide range of functions makes compose a powerful formatting tool, it also makes learning the formatter in its entirety a fairly lengthy process. Fortunately, compose can be learned in stages; this chapter introduces enough compose controls to create simple documents made up largely of text.

NAMING AN INPUT FILE

Distinguish a compose input file from the other segments in your directory by adding a period and the letters "compin" as a suffix to its name, for example:

```
My_epic_novel.compin
```

If you do not include this suffix, compose cannot locate your file for processing.

FORMATTING WITHOUT CONTROLS

Before introducing any compose controls, it should be pointed out that you don't always need them. Compose formats input files according to certain *defaults*, a set of pre-defined formatting standards that are adhered to unless you explicitly request something different. These defaults take care of the basic tasks of page layout: the number of spaces between lines, size of margins, dividing the text into pages. You need controls to change this default format or to add extra features to your text.

Basic Text Layout

The following examples show what the printed output would look like if you sent a compin file containing only a few paragraphs of text directly to the formatter. The resulting document is printed in the font specified on the "compose" command line, in this case the "press roman" font. Here, and in examples throughout the memo, what you type at the terminal is labeled "Input File:", while the printed output it produces is labeled "Resulting Output:".

---

[1] If you are curious to know what the actual printer instructions look like, print a compout file at your terminal screen sometime.

Input File:

```
Something smelled horrible, like a dead skunk.
I groaned.
My last chance for a decent night's rest was gone for good.
Should I open my eyes?
No, that might be a mistake (with a job like mine you can never be
too cautious).  What if I'm still lying in the middle of Main St?

I rolled over and hit the floor like a stack of soggy pancakes.
Either I stop sleeping at the office or I get a wider couch.
```

Resulting Output:

Something smelled horrible, like a dead skunk. I groaned. My last chance for a decent night's rest was gone for good. Should I open my eyes? No, that might be a mistake (with a job like mine you can never be too cautious.) What if I'm still lying in the middle of Main St?

I rolled over and hit the floor like a stack of soggy pancakes. Either I stop sleeping at the office or I get a wider couch.

The formatted text has been *justified*: the uneven right margin of the compin file is completely smoothed out. The text is single-spaced, and new paragraphs are indicated by a blank line, as they were in the compin file. If the text were longer, compose would divide it up into pages, but would not add any kind of page numbering.

In the previous example, all input lines began at the left margin. If you start a line with one or more spaces instead, this causes a *break* in the printed text. The previous line is broken off, and compose adds as many blanks at the beginning of the new output line as you left when typing the input line. You can use this feature to indent lines within the body of the text or for indenting new paragraphs.

Input File:

```
What is that sickening smell? My razor-sharp brain begins
carving out explanations:
    a dead skunk (my previous hunch)
    a poison gas so innovative even I don't recognize it
    a liverwurst sandwich
```

Resulting Output:

What is that sickening smell? My razor-sharp brain begins carving out explanations:

a dead skunk (my previous hunch)
a poison gas so innovative even I don't recognize it
a liverwurst sandwich

Default Page Measurements

Since the previous examples were not only brief but also indented to distinguish them from surrounding text, they do not illustrate the default compose page measurements very clearly. The standard margins used throughout this memo *do* represent unmodified default settings. The introductory chapter of this memo provides a good example of how plain text is laid out over several pages.

Unless you specify otherwise, compose ensures a vertical margin of at least an inch at the top and bottom of the page. Since page numbers have been added to this memo, the text has been moved down two lines, and it is the page numbers that start an inch from the top of the page. Bottom margins can vary somewhat. Compose may break text early to avoid splitting a few lines off from the rest of its paragraph. For example, both pages 3 and 4 were shortened in order to move the title of the following section to the top of the next page.

You determine the side margins. Compose has set the text line width to six and a half inches,[1] which leaves two inches for right and left margins. By default, the printer starts text at the left edge of the paper, creating a two-inch margin to the right of the text. You can change this at formatting time by specifying a left indentation when issuing the "compose" command. The width of the right margin is determined indirectly by how much extra space remains after the left indentation and the width of the text. See "Producing an Output File" on page 58 for how to specify left indentation.

ADDING COMPOSE CONTROLS

The rest of this chapter is devoted to controls that modify the default format described above. Though you type formatting controls into your compin file like the rest of your text, they do not appear in the finished document except through their effects on surrounding text. Control words are distinguished from plain text lines by their own particular syntax.

In the input file, compose controls are placed on lines by themselves, at the very beginning of the line. They always begin with a period, and consist of several lowercase letters (usually two or three.) You can modify the action of some controls by giving them numeric arguments or by specifying certain options with letters or words. These options or arguments must always be separated from the control proper by a space. For example, the formatting control used to insert one blank line is ".spb", while to insert ten blank lines you would type ".spb 10". The control word together with all its options or arguments is called a *control line*.

Control lines must follow this specific syntax if they are to be recognized and properly interpreted by the formatter. When entering text lines you are much freer. You can make lines as long or as short as you like, since compose arranges your text to fit within the page margins you request. However, you should not split words across lines with hyphens. Compose has its own hyphenation facility which you can request, but interprets any hyphens you add as the final character of the preceding word fragment. Finally, never begin a text line with a period. If you should happen to do so, compose will try to process it as a control line!

---

[1] Compose measures line width in terms of ten-pitch characters (i.e., ten characters to an inch), so this default setting equals 65 characters.

FILLING AND ALIGNMENT

As shown in the previous examples, compose automatically justifies text. This default mode of operation involves two separate actions: *filling* and *alignment*. First, compose fills text lines to approximately the same length by shuffling words back and forth between lines. Then it pads the lines with blanks to align them at both right and left margins.

Both these default operations can be changed. If you choose to turn filling off, then compose processes each input line as one line of output. You have the further option of aligning your text (whether filled or not) in the center of the page or at one margin only.

The following compose controls adjust filling and alignment. Their longer, more descriptive names are included in parentheses. You do not type the full names into the input file; they are added here only to remind you of the function associated with their particular controls.

.fif (fill-off)
Turns off fill mode. Your text lines are printed out exactly as you entered them. If you have typed an input line that is very long, it is allowed to wander into the margin and even off the page. This mode is especially useful for tables and charts.

.fin (fill-on)
Turns fill mode on after a ".fif" control. This is the default.

.all (align-left)
Aligns each output line at the left margin, but the right margin is not aligned. This is the default when filling is turned off, so it has a noticeable effect only in fill mode.

.alr (align-right)
Aligns text at the right margin and leaves left ragged.

.alb (align-both)
Creates perfectly smooth right and left margins. This is the default in fill mode, and works only when fill mode is on. Used to return to default alignment after an ".all" or similar control.

.alc (align-center)
Centers text. Turns fill off if you want a series of input lines centered independently.

When Fill Mode is On

While in fill mode, only ".alb" and ".all" produce alignments that are generally useful. Examples of text aligned at both margins were shown in the previous section on compose's default operation, so none are given here.

If you turn off right alignment with an ".all" control, the resulting output looks as if it were typed with a typewriter. Input lines are filled to approximately the same length, but no extra spaces are added to create a perfectly even right margin.

Input File:

```
.all
I opened one eye cautiously.
Waking up on the floor has its advantages.
There was a Lucky Strike about ten inches from my nose.
Had some subtle intuition made me drop it there last night?
Sometimes I'm so intuitive I scare myself.
.alb
```

Resulting Output:

I opened one eye cautiously. Waking up on the floor has its advantages. There was a Lucky Strike about ten inches from my nose. Had some subtle intuition made me drop it there last night? Sometimes I'm so intuitive I scare myself.

If you feel an uncontrollable urge to defy convention, you can try out different alignments for your filled text with ".alr" or ".alc". In either case, the output lines will be filled to exactly the same length as in the example above. Only the placement of each line on the page will be different.

Turning Fill Mode Off

Once you have turned fill mode off with the ".fif" control, your alignment options are considerably different. The ".alb" control no longer functions, since aligning unfilled input lines at both margins doesn't make much sense. Short lines would have to be padded with many extra blanks, while very long input lines would not work at all. Compose avoids such a situation by automatically changing the alignment to ".all" when filling is turned off. If you do add an ".alb" control, compose does not send you an error message, but politely treats it as equivalent to ".all".

Input File:

```
.fif
Happiness is:
A bottle of non rot-gut bourbon
Blonde clients who pay in advance and fall for me when the case is over
Knowing where my next pack of Luckies is coming from
```

Resulting Output:

Happiness is:
A bottle of non rot-gut bourbon
Blonde clients who pay in advance and fall for me when the case is over
Knowing where my next pack of Luckies is coming from

As in fill mode, compose respects any blank spaces at the beginning of an input line. The ".all" alignment does not force such a line over to the left margin; each output line is positioned at the same place on the page as the line of text you typed at the terminal.

Input File:

```
.fif
Unhappiness is:
  Paying bills
  Waking up before the neighborhood bar opens
  A secretary who eats liverwurst sandwiches
```

Resulting Output:

Unhappiness is:
  Paying bills
  Waking up before the neighborhood bar opens
  A secretary who eats liverwurst sandwiches

As already described, adding either ".all" or ".alb" to the previous input file would not have changed the resulting output in any way. On the other hand, the ".alc" control word is much more useful once fill mode has been turned off. You generally want text lines to be centered separately without being all run together first.

Input File:

```
.fif
.alc
The Case of the Hairless Dachshund
or
How I Spent Five Glorious Days in a San Francisco Opium Den
on Police Department Money
A True Story
by
Maxwell K. Swill
.fin
.alb
```

Resulting Output:

The Case of the Hairless Dachshund
or
How I Spent Five Glorious Days in a San Francisco Opium Den
on Police Department Money
A True Story
by
Maxwell K. Swill

If you hadn't turned filling off, this series of titles would look like this:

The Case of the Hairless Dachshund or How I Spent Five Glorious Days in a San Francisco Opium Den on Police Department Money A True Story by Maxwell K. Swill

When fill mode has been turned off, the ".alr" control may help create charts or tables of numbers.

Input File:

```
.alr
.fif
 3 hours at $30.00/hour =   $90.00
72 hours at $20.00/hour = $1440.00

          Total          $1530.00
.fin
.alb
```

Resulting Output:

```
 3 hours at $30.00/hour =   $90.00
72 hours at $20.00/hour = $1440.00

          Total          $1530.00
```

LINE SPACING

By default, compose prints text single-spaced (as in this memo). You can change this default at two different points in the document production process: from inside the compin file or later, while formatting. Control lines inside the compin file let you modify line-spacing on a very local level.

Unlike those controls previously introduced, the linespace control word takes an argument. In the chart below (and in all similar charts throughout the memo) an italic character following the control word represents an argument. The *n* can be replaced by either an absolute positive number (otherwise known as an *unsigned* number) or by a *signed* number (i.e., +3 or -3).

.ls *n* (linespace)
: Changes amount of white space between text lines. You can specify quarter-line increments with decimal arguments (i.e., 0.25, 1.50 . . .). The default is 1. Each of these vertical spacing units is equal to approximately one-sixth of an inch, so an 11-inch page is 66 lines long.

You can use this control to change the spacing for a certain section of the text, and then restore the default setting by issuing a control line with no argument or an argument of "1". An absolute argument changes the line spacing from 1 to the number of lines specified. A signed number increases or decreases the present line spacing by the specified amount.

This control is especially useful when switching between different sized fonts. The default line spacing is designed for 10 or 12 character-per-inch fonts. When switching to smaller or larger characters, you may want to change the linespace value to keep the distance between lines attractive. For instance, a 14 character-per-inch font looks better with the line-space value set to 0.75.

Input File:

```
.ls 0.75
.fnt small_typ
Inspired by the promise of nicotine, my brain was working furiously.
Was this cigarette only another hoax?  What did it have to do with
the hypothetical dead skunk?  What was I doing last night?
Why had I been lying face down in the middle of Main St.?
```

Resulting Output:

```
Inspired by the promise of nicotine, my brain was working furiously.  Was
this cigarette only another hoax?  What did it have to do with the
hypothetical dead skunk?  What was I doing last night?  Why had I been
lying face down in the middle of Main St.?
```

NOTE: the ".fnt" control appearing in the example above is used to change the font and is introduced in the next chapter, "Fonts".

You don't have to commit yourself to a particular line-spacing while writing the input file. When issuing the "compose" command line, you can add a control argument requesting that the entire compin file be formatted with a different spacing. For a complete description of this control argument see Chapter V, "Formatting and Printing Your Text".

INSERTING BLANK LINES

The actual control words for adding blank lines and creating text divisions are introduced in the second part of this section. The first part of this section provides some background information that you may not need until after you've actually tried out the control words. If you find that these controls don't always work as you expect, read this first part--it will help explain what's going on and how to make the controls act more predictably.

Basic Text Divisions

When compose processes a compin file, it divides your text up in various ways. From a user perspective, these divisions fall into two different categories: *page breaks* and *text breaks*. If your text is long enough, compose breaks it into separate pages. You need to add controls only when you want to change this default page handling. Text breaks, on the other hand, are requested by the user. Within a page, you can add blank lines or control lines to divide the text up into units such as paragraphs.

Compose adds page and text breaks to conform to human conventions of readability. It also divides the text up into units according to its own conventions in order to process the compin file. These units are called *text blocks*, and they do not necessarily coincide with the divisions

that you see in the printed output. As a result, compose sometimes moves your text around in rather unexpected or unattractive ways. You can minimize the chances of this happening, or control it when it does, if you are familiar with some of the basic ways compose processes text blocks.

Many of the controls which you use to request text breaks also signal the end of a text block as well. These controls are said to cause *block breaks*. You can also choose formatting controls that temporarily alter the appearance of the formatted text within text blocks without signalling the end of the block. Such control words cause *format breaks*.

When you come to the logical end of a chunk of text and want to have compose add a certain number of spaces or start formatting your text in a new way, it is generally a good idea to use controls that also signal the end of a text block. Whenever possible, compose avoids splitting text blocks across pages, so this practice will minimize the occurrence of illogical page breaks.

Adding block breaks at every text division will also keep you from exceeding the maximum allowable block size. Text blocks are limited to 1000 output lines. If you pass compose an input file with a text block that exceeds this length, compose breaks off in the middle of the formatting process and sends you an error message.

Breaking Text and Adding White Space

The section on compose defaults showed ways of adding blank space and text breaks without any control words. This section introduces controls that perform these same functions in a more reliable manner.

Compose controls can take two basic types of numeric arguments. The more versatile type, represented by an italicized *n*, was introduced in the previous section on line-spacing. Here, a # (pound sign or number symbol) stands for an argument that can only be a positive number.

.brf (break-format)
Causes the preceding text line to be printed without filling in words from the next text line. This does not signal the end of a text block, but only causes a break in format.

.spb # (space-block)
Prints # blank lines and signals a block break. Should be used whenever you want to add white space between two units of text. The default argument is 1.

.spf # (space-format)
Prints # blank lines. The default argument is 1. Does not signal a block break. Use only when you want to keep compose from splitting the surrounding text across pages.

.spd *n* (space-to-depth)
With an absolute number, this control adds blank lines until you reach the line *n* lines down from the top of the page. A control line with a *+n* argument adds *n* blank lines after the current line. This amount of white space is always added, even if it has to be split across a page break.

The ".brf" control is generally used to interrupt fill mode for a specific output line within a text block. The preceding line is broken off even if it doesn't extend to the right margin. The following line begins at the left margin and is filled. This control lets you add a text break without beginning the next input line with a space (as described in the "Basic Text Layout" section) or leaving it blank.

Input File:

```
The dead skunk hypothesis was fading fast as the noxious fumes peculiar
to liverwurst enveloped me.  Only one person in the world has the
nerve to do this to me:  Gertrude was at it again.
.brf
"Gertrude!"  I groaned.  "You've got to choose . . . me or the liverwurst."
.brf
No reply, just more loud chewing.  Maybe if I paid her a salary she'd
respect me more. . .
```

Resulting Output:

The dead skunk hypothesis was fading fast as the noxious fumes peculiar to liverwurst enveloped me. Only one person in the world has the nerve to do this to me: Gertrude was at it again.
"Gertrude!" I groaned. "You've got to choose. . . me or the liverwurst."
No reply, just more loud chewing. Maybe if I paid her a salary she'd respect me more. . .

An ".spb" control line is equivalent to a blank line. Both signal the beginning of a new text block by adding one empty line. Though blank lines usually work fine, it is a good habit to always add ".spb" or ".spf" instead. When using a full-screen editor like emacs, it's easy to confuse the purposeful with the mistaken blanks as you're skimming through the input file.

Input File:

```
Her gold stiletto heels clicked past. . .
.spb 2
A match landed next to the cigarette.
She may have filthy eating habits, but she knows me like nobody
else does.
```

Resulting Output:

Her gold stiletto heels clicked past. . .

A match landed next to the cigarette. She may have filthy eating habits, but she knows me like nobody else does.

The ".spf" has a more specialized function than the ".spb" control. Adding ".spf" control lines whenever you want to insert white space, you risk creating a text block that's too long to process. Use ".spf" mainly to insert blank lines within a text unit that you don't want split by

a page break (like tables or charts). Compose tries to make page breaks coincide with text breaks whenever possible, and is more likely to start a new page when it encounters an ".spb" than an ".spf".

The ".sp" controls do not guarantee that the amount of white space you specify always appears on the printed page. They were designed primarily for marking off sections of text with white space, and the formatter regards a page break as a satisfactory substitute for such a division. So, if an ".sp" control coincides with a page break, the formatter *trims* the requested space.

To add a certain amount of space that can't be discarded during page breaks, use the ".spd" command instead. This control functions in two different ways. If you give it an unsigned number *n* as an argument, blank space is added until a point *n* lines down from the top edge of the page. With a positive argument *n*, the control adds *n* blank lines, starting from your current location on the page. Either case guarantees that the requested number of blank lines appears in the output, even if they are distributed at the top and bottom of two separate pages. Use this control whenever you want to add blank space at the top of a page.

PAGINATION

Compose normally fills each page with text before starting the next one. Though the formatter tries to keep the bottom margins even, it never leaves a single line of a text block by itself at either the bottom or top of the page. Two is the minimum number of lines that can be split off.[1]

Controlling Page Breaks

In spite of all its built-in flexibility, compose may still break your text in unattractive or illogical ways. The following controls can be used to change where pages are broken or to force a new page before the previous one is full.

.brp (break-page)
Causes an immediate page break. Compose finishes processing the current page, even if it isn't full, and starts printing at the top of the next page.

.brn # (break-need)
Causes a *conditional* page break: if the argument given is greater than the number of lines still left on the page, then compose breaks the page at the current point in the text. Otherwise, the page is not broken.

.bbk # (block-begin-keep)
Marks a unit of text which cannot be split across pages. If you supply an argument, the next # lines will be kept as a unit. Otherwise, this control's action must be ended by a ".bek" control.

.bek (block-end-keep)
Indicates the end of a ".bbk" unit, and allows page breaks to operate again.

---

[1] These lonely groups of lines are called *widows*. You can change the minimum allowable widow size if you like. See the later section "Changing Defaults".

At certain points in your text, you may want to force compose to start a new page whether or not the previous one is full. This is especially convenient when starting a new section in a long document. For example, the input file for the first page of Chapter IV looks like this:

```
Italics version of Xerox standard printer typestyle.
.brp
.alc
IV. COMPOSE CONTROLS CONTINUED
.alb
.spb 2
This chapter introduces several more specific . . .
```

If compose has split some text across pages in a way you don't like, you can fix this by adding a ".brp" in front of that text. However, the ".brn" control provides a more flexible way of controlling page breaks. It tells compose to start a new page only if a specified number of lines can't fit on the current page.

```
.brn 4
How could I ever have trusted someone with a name like Shirley Love?
Still, I never argue with redheads, especially when they're sticking
a rifle in my ribs. Women are like that.  One minute they're admiring
your tie, the next they want to blow a hole in you.
```

The ".brn" control line prevents this short paragraph from being broken into two chunks of two lines each. If there are not at least four lines left at the bottom of the current page, compose starts a new one. Otherwise, this control is ignored. This way, you don't have to keep checking on how the pages break, adding and deleting ".brp" controls each time you reformat. You can also use ".brn" within a very large text block to indicate where it can be broken if it won't all fit on one page.

The ".brn" control helps minimize unattractive stretches of white space and illogical snippets of text. If you are more concerned with keeping a unit of text together than the appearance of the page as a whole, choose the ".bbk" and ".bek" controls. They create a specially protected text block. If there is enough room left, the block is added to the current page. Otherwise, compose moves it as a unit to the next page no matter how much intervening white space that leaves. For example, the first page of this chapter breaks off early because ".bbk" and ".bek" were used to prevent the "Input File" example from being split across a page. If you mark off a unit larger than a page, compose simply ignores these commands and breaks it up without sending you any error messages.

Compose features more sophisticated controls, specifically for inserting charts and tables. These controls also mark off protected text blocks, but are designed to avoid adding large sections of blank space in front of the inserted text block. See the "Tables and Charts" section in Chapter IV for a description of these controls.

Simple Page Numbering

Compose does not number pages for you by default. The control line for inserting page numbers involves several concepts that have not yet been introduced, and will be fully described in a later section, "Page Numbering Revisited", page 52. For the time being, here are two different control lines you can use to add numbers to your documents.

For centered numbers at the top of the page, add the following line to the beginning of your input file:

```
.phl ||%PageNo%||
```

For centered numbers at the bottom of the page, use the following control line instead:

```
.pfl ||%PageNo%||
```

By adding options and arguments to the simple ".brp" control line described above, you can change the page number count, or the numbering format (i.e., from roman numerals to arabic). This more advanced use of the ".brp" control is also described in "Page Numbering Revisited".

INDENTING TEXT

Compose features a series of controls that let you temporarily shorten the printed text line by moving the margins in toward the center of the page. You can indent the right or left margin individually, or both at once. Indentations remain in effect until turned off by a following control.

.inl *n* (indent-left)
Indents *n* tenths of an inch from the current left margin. The default value is 0, so an ".inl" without an argument cancels any previous left indentation and restores the default margin setting.

.inr *n* (indent-right)
Indents *n* tenths of an inch from the current right margin. The default value is 0, and turns off previous right-indents.

.inb *n* (indent-both)
Indents *n* tenths of an inch from both right and left margins. The default value is 0, and cancels any previous indentation for both left and right margins.

Note that the horizontal measurements for these controls are given in tenths of an inch. If you are using a 10-pitch font (10 characters per inch), then these units correspond to letter spaces. If you're using a different size font (as in this memo, for instance), then you'll have to develop an instinct for the ratio between the 10-pitch units and your letters.

You can use these controls in two different ways, depending on the type of argument. An unsigned numeric argument specifies an absolute indentation, measured from the default margin; any previous indentations are cancelled.

Input File:

```
The note was short, nasty, and to the point:
.inb 5
.spb
If you ever want to see your dachshund alive again, its going to
cost you 500,000 big ones.  Put the cash in that hideous alligator
suitcase you inherited from your mother and leave it with your
favorite bartender by 5:00 this afternoon.
.spb
.inb
Reeling from this fiendish blow, I reached blindly for one of
Gertrude's sandwiches.
```

Resulting Output:

The note was short, nasty, and to the point:

> If you ever want to see your dachshund alive again, its going to cost you 500,000 big ones. Put the cash in that hideous alligator suitcase you inherited from your mother and leave it with your favorite bartender by 5:00 this afternoon.

Reeling from this fiendish blow, I reached blindly for one of Gertrude's sandwiches.

By supplying a positive or negative argument instead, you can add to or subtract from the previous margin setting. This is particularly convenient when you want to change the indentation for a stretch of text and then reverse it, without keeping track of what the previous indentation was.

Input File:

```
.fif
.inb 5
Oh where, oh where
.inl +2
has my little dog gone?
.inl -2
Oh where, oh where can he be?
With his tail
.inl +2
cut short
and his hair all gone?
.inl -2
Oh where, oh where can he be?
.inb
```

Resulting Output:

Oh where, oh where
  has my little dog gone?
Oh where, oh where can he be?
With his tail
  cut short
  and his hair all gone?
Oh where, oh where can he be?

UNDENTING SINGLE LINES

In addition to the standard indentation function, compose features a related function called *undentation*. Undentation controls change the current margin setting for one input line only. Since they are frequently used to reverse the direction of a previous indentation, their arguments work in the opposite direction from those applied to the indentation controls. Positive arguments move the line towards the margin, negative arguments move it towards the center of the page.

For example, the following chart was produced by indenting the descriptions on the left and right, while undenting the line naming the control.

.unl *n* (undent-left)
Changes left indentation for the next input line only according to the value *n*. Default value of *n* is the left margin.

.unr *n* (undent-right)
Changes right indentation according to value of *n* for next input line only. Default value of *n* is the right margin.

.unh *n* (undent-hanging)
Changes the left indentation of the next input line by *n* units. Used primarily for offsetting phrases or numbers from paragraphs of filled text. See examples in following section for a more complete description of how this control works.

The ".unl" and ".unr" undentation controls are fairly straight-forward variations on the indentation controls. Since they both work in the same way, the following examples demonstrate only the more generally useful ".unl" control word.

If filling is on, the undented line is joined with following input for justification.

Input File:

```
.inl 2
.spb
.unl
MISS BLISS
My favorite client ever.  Unspeakably blonde and decadent,
with an unspeakably rich and indulgent father.
.spb
.inl
```

Resulting Output:

MISS BLISS My favorite client ever. Unspeakably blonde and decadent, with an unspeakably rich and indulgent father.

Used with a negative argument, this control provides a convenient way of indenting paragraphs or one-line examples.

Input File:

```
.unl -5
Eustace, my inoffensive dachshund!  The last reminder (besides that
beautiful alligator suitcase) of my dear departed mom.
What kind of sick and perverted mind could have dreamed up such a
nefarious scheme?
.spb
.unl -5
A really tough guy is not afraid to show his
emotions.  I often wax sentimental, especially for
appreciative audiences.
```

Resulting Output:

Eustace, my inoffensive dachshund! The last reminder (besides that beautiful alligator suitcase) of my dear departed mom. What kind of sick and perverted mind could have dreamed up such a nefarious scheme?

A really tough guy is not afraid to show his emotions. I often wax sentimental, especially for appreciative audiences.

When filling is turned off, the input line following the the undented phrase is always placed on its own output line.

Input File:

```
.fif
.inl 7
.unl 2
Urgent:
Blackmail Veronica for rent money
Dazzle landlady with my boyish charm
Find out where Miss Bliss goes when she isn't where she ought to be
.spb
.unl 2
Amusing Side Issues:
Tell Joey's wife it's over
Send anonymous note to vice squad denouncing Fred's non-middle class morality
Threaten crude brat who delivers for the corner grocery
.inb
.fin
```

Resulting Output:

Urgent:
Blackmail Veronica for rent money
Dazzle landlady with my boyish charm
Find out where Miss Bliss goes when she isn't where she ought to be

Amusing Side Issues:
Tell Joey's wife it's over
Send anonymous note to vice squad denouncing Fred's non-middle class morality
Threaten crude brat who delivers for the corner grocery

Of the three controls, ".unh" is the most useful as well as the most complicated to explain. It is designed to help you offset numbers or phrases from paragraphs of *filled* indented text, and combines several unexpected features in order to accomplish this. As a result, it is best introduced by an immediate example.

Input File:

```
.inl 15
.inr 5
.unh 15
Veronica Sludge
A girl who's no better than she should be.
Built like a tank, with a face like processed cheese.
A night with her is like going through a cement mixer.
.spb
.unh
Gertrude Glumley-FitzHumbert
One of THE Glumley-FitzHumberts.  By night, the bored darling
of high society; my secretary by day.  Petulant and spoiled,
constantly haunted by tall dark strangers and her murky past.
.inb
```

Resulting Output:

Veronica Sludge A girl who's no better than she should be. Built like a tank, with a face like processed cheese. A night with her is like going through a cement mixer.

Gertrude Glumley-FitzHumbert
One of THE Glumley-FitzHumberts. By night, the bored darling of high society; my secretary by day. Petulant and spoiled, constantly haunted by tall dark strangers and her murky past.

The undented line is always treated as a separate unit from the rest of the input lines, and is never joined with them for filling. The placement of the following text line depends on the length of the undented phrase. In the example above, "Veronica Sludge" takes up less than the 15-space indentation, so her character description starts on the same output line as her name. Since "Gertrude Glumley-FitzHumbert" is too aristocratic a name to squeeze into such a constricted space, her description is automatically moved down a line.

Note the different ways of specifying how much to offset the numbers. A ".unh" control line without any argument always moves the undented text out to the left margin. An unsigned argument is equivalent to a positive argument, so a ".unh +15" control line could have been used instead of either of the given control lines.

Though ".unh" also works with fill mode turned off, it is not as flexible and easy to use. The undented phrase and the following input line are *always* placed on the same output line. If you accidentally include an indented phrase that's too long, overprinting results.

For example, if you turned filling off in the previous compin file, the resulting output would look like this:

Veronica Sludge A girl who's no better than she should be.
Built like a tank, with a face like processed cheese.
A night with her is like going through a cement mixer.

Gertrude Glumley-FitzHumbert THE Glumley-FitzHumberts. By night, the bored darling of high society; my secretary by day. Petulant and spoiled, constantly haunted by tall dark strangers and her murky past.

To force the character description down one line, you *must* use an ".spf" or ".spb" control. When inserted between a line that is to be undented and its following line of text, these controls do not *not* add a blank line. A ".brf" control inserted here would have no effect. This is true whether fill mode is on or off.

The ".unh" control is also very useful for creating numbered lists, when you want the text of all the numbered items to be evenly aligned at the left. In the following example, ".unh" is used to create two different indentation levels.

Input File:

```
.spb
.inl 3
.unh
1.
Suspects
.spb
.inl +3
.inr 5
.unh +3
a.
Sammy the Slouch.  Always my favorite suspect.  A generally
sleazy character, jealous of my debonaire appearance and wild
success with expensive dames.
.spb
.unh 3
b.
Joey Dribble.  I hate his aftershave.
.spb
.unh +3
c.
Thelma la Louche.  She hates dogs.  She hates dachshunds.  She
looks suspicious.  She needs the money.
.spb 2
.inb
```

Resulting Output:

1. Suspects

    a. Sammy the Slouch. Always my favorite suspect. A generally sleazy character, jealous of my debonaire appearance and wild success with expensive dames.

    b. Joey Dribble. I hate his aftershave.

    c. Thelma la Louche. She hates dogs. She hates dachshunds. She looks suspicious. She needs the money.

You are not restricted to undenting only one phrase at a time. By using a series of ".unh" controls, you can place several undented phrases on the same output line, if you want.

Input File:

```
.inl 35
.unh
Veronica Sludge
.unh 15
double rate
Doing business with her is like walking on jello.
.spb
.unh
Shirley Love
.unh 15
standard rate
A tough customer, but she's brought in too much business to ignore.
.spb
.unh
Otis Gloub
.unh 15
paid!
Pale and slimey with sunken eyes and suspicious nocturnal habits, but
very polite and accepted in all the best circles.
.spb
.unh
Miss Bliss
.unh 15
discount rate
Her name says it all . . .
.inl
```

Resulting Output:

Veronica Sludge	double rate	Doing business with her is like walking on jello.
Shirley Love	standard rate	A tough customer, but she's brought in too much business to ignore.
Otis Gloub	paid!	Pale and slimey with sunken eyes and suspicious nocturnal habits, but very polite and accepted in all the best circles.
Miss Bliss	discount rate	Her name says it all . . .

III. FONTS

The Information Systems laser printers can print in a variety of fonts, and compose lets you take full advantage of this, offering you a choice of fonts at two different levels. When choosing one of the available printing styles, you commit yourself to a standard font that is used by default throughout the document. Each printing style includes a group of additional fonts, so you have the further option of switching to any of these other fonts within the document itself.

This chapter describes the current choice of fonts and the controls for changing them. Fonts are introduced here because many of the more advanced formatting features described in the next chapter are most effective when combined with font switching.

DEVICE TYPES

In order to format your compin file for the laser printer, you have to select one of several *device types* (often referred to as "devices"). Device types are not actually different physical printing devices, but simply the different printing formats available. Each format defines a certain group of fonts to be used for that printing job and the orientation of the text on the page. You specify a device type on the "compose" command line. When creating the output file, compose adds printer instructions about the appropriate fonts and page orientation.

Currently, the Information Systems laser printers support eight different device types. Five produce output in portrait mode, where the text runs across the width of the page (as in this memo). Three devices print in landscape mode, where the output runs along the length of the page. Each device is named after the default font associated with it, i.e. an 11-point press roman for x9700_pr. The device types currently supported are:

x9700_pr	portrait mode, press roman font, 11 characters per inch, proportional-width letters; used for this memo
x9700_un	portrait mode, univers (sans-serif) font, 11 characters per inch, proportional-width letters
x9700_p10x6	portrait mode, ten characters per inch, fixed-width letters; like pica typewriter output in appearance
x9700_p12x6	portrait mode, 12 characters per inch, fixed-width letters; like elite typewriter output in appearance
x9700_p14x8	portrait mode, 14 characters per inch, fixed-width letters; very small!
x9700_pr_land	landscape mode, press roman font, proportional-width letters
x9700_un_land	landscape mode, univers font, proportional-width letters
x9700_l14x8	landscape mode, 14 characters per inch, fixed-width letters

Most users interested in producing attractive documents prefer one of the two portrait-mode, proportional-font devices. They offer the largest selection of fonts, and work very well in fill mode. Fixed-width fonts make it easier to format simple tables or equations, but tend to create awkward gaps in filled text lines.

A document is printed in the default font of the chosen device unless you specifically request font changes within the compin file. By adding control lines at appropriate points in your text, you can switch to any of the other fonts defined for the device type. Descriptions of the font groups for each of the eight device types are given in the section below, "List of Available Fonts". Examples of what the different fonts actually look like are included in Appendix C.

Aside from special circumstances, you do not usually have to decide on a device before adding font switching control lines to your text.[1] Several of the most frequently used fonts are common to all device types, and font names are usually generic. For example, every device has a default font which is always named "ascii", as well as the "sub", "super" and "italic" fonts.

CHANGING FONTS

You can employ two different methods of indicating font changes in your compin file. The first alternative is simple but not foolproof. The second is more complicated, but much more likely to work.

If you want the change to affect an entire unit of text, like a sentence or a paragraph, a ".fnt" control line generally works. Find the name of the new font you want to use in the following "List of Available Fonts" section, and add it to the ".fnt" control as an argument. Put this control line in front of the appropriate text.

Input File:

```
You've certainly got something there, Otis.
If you can get a good write-up in the
.fnt italic
Journal of Vampire Medicine,
.fnt
your reputation is assured.
```

Resulting Output:

You've certainly got something there, Otis. If you can get a good write-up in the *Journal of Vampire Medicine,* your reputation is assured.

Compose remembers all the font changes you make, and stores them in a *font stack*. You can travel backwards through this series of stored fonts by using a ".fnt" control with no arguments to revert to the *previous* font on the stack. This stack can hold only a limited

---

[1] If you know that you need to employ an unusual font (like one with umlauts) then you must find out ahead of time which devices include it.

number of font changes. If you pile up too many, you may end up with output printed in a completely unexpected font. If you're lucky, compose warns you during formatting by sending you a "Font stack exhausted." error message. To avoid this, you can always empty the font stack with a ".fnt reset" control line. However, the safest method of all is to always switch back to the "ascii" font with a ".fnt" control after every font change, as in the following example.

Input File:

```
.fnt small
Ants are clearly the most intelligent and clean-living species on earth.
.fnt
.fnt bold
Lions are clearly the laziest and most slovenly species on earth.
.fnt
.fnt italic
The fact remains, however, that lions can easily squash ants, while
.fnt
exhaustive planning, perfect teamwork, and at least a million ants
would be required for
.fnt bold
successful lion-squashing.
.fnt
```

Resulting Output:

Ants are clearly the most intelligent and clean-living species on earth.
Lions are clearly the laziest and most slovenly species on earth. *The fact remains, however, that lions can easily squash ants, while* exhaustive planning, perfect teamwork, and at least a million ants would be required for **successful lion-squashing.**

This technique has limitations. It is usually adequate if you want the font change to take effect with the next line, or, if fill mode is on, with the next word. It was not designed for changing fonts in the middle of a line when fill mode is off, or in the middle of a word. In addition to these predictable limitations, the control may sometimes add extra blank lines or spaces to your output file for no apparent reason (as at the end of the first line in "Resulting Output" above.) If you encounter either of these situations, use the more complicated set of controls described next.

This second font-changing technique requires a fairly advanced control word that is not introduced until the next chapter, so a complete explanation is not provided here. However, a brief example is given below, since this method is much more reliable than the simpler one. You do not add the control on a line by itself, but insert the control words into the appropriate line of text.

Input File:

```
.ur forty-two goats, %.fnt italic%fifty-two fowl%.fnt%, and no pigs
```

Resulting Output:

forty-two goats, *fifty-two fowl*, and no pigs

You must always add a space between the ".ur" control and the input line, and type both percent signs and ".fnt" controls exactly as shown. Forgetting a percent sign or a period makes the control line unrecognizable to the formatter, and you will probably get an error message to that effect. For a complete explanation of what is going on here, see "Defining and Using Variables" in the next chapter.

LIST OF AVAILABLE FONTS

This section describes the actual fonts available for each of the device types introduced earlier in this chapter. The listed names are added as arguments to a ".fnt" control when switching fonts, as described above. For ease of typing, only a short name is listed here (and an abbreviation if one exists). However, there are many alternate names for each font that work equally well. If you want a complete list of all possible font names, type "help new_compose".

Font groups for similar device types tend to have a strong family resemblance, making it easy to format the same file for either device without changing any font names.

The two proportional-width, portrait-mode device types have the largest font groups. The x9700_pr and x9700_un devices have 20 and 19 fonts respectively; 16 of them are common to both devices. The two landscape mode proportional devices have *identical* groups of 12 fonts each.

The fixed-width device types generally feature a more limited range of fonts than the proportional devices. The x9700_p10x6 and x9700_p12x6 devices have 14 and 11 fonts respectively, 9 of which are common to both. These two fonts are available only in portrait mode. Fixed-width, 14-pitch fonts come in both portrait and landscape modes. These two device types offer the smallest choice; both feature an identical set of five fonts.

If you have very specific needs, these may determine which device type you should choose. For instance, umlauts are defined only in the x9700_pr device. On the other hand, if you need to use mathematical symbols, all the portrait mode devices have identical "symbol" fonts. See page 64 in Chapter VI for an example of using the "sym" font to format equations.[1]

To see what all these different fonts actually look like, go to Appendix C.

---

[1] Underlining is not currently included in any of these font groups; italics or bold-face generally provide an adequate substitute. If you must underline, switch into a fixed-width font, and type an underscore and a backspace in for each character you want underlined. Since the underscore is of a fixed length, this method does not work very well in a proportional font.

Device x9700_pr:

These are proportionally spaced fonts in portrait orientation. Most of them are in the "press roman" family, a serifed typestyle. This memo was printed using this device type.

ascii, m	11-point press roman medium body font.
super, sp	The "ascii" characters raised one-half line.
sub, sb	The "ascii" characters lowered one-half line.
ASCII, M	The "ascii" character set, but completely upper case.
bold, b	11-point press roman medium bold.
BOLD, B	Same as "bold", but completely upper case.
italic, i	10-point univers medium italics.
small, sm	10-point press roman medium body font.
head14, L	14-point press roman capitals, numbers and a few special characters.
head18, VL	18-point press roman capitals and numbers (plus ,-./).
typ, t	12-character-per-inch, fixed-pitch, "letter-gothic" typewriter font.
small_typ, tsm	approximately 14-character-per-inch, fixed-pitch font in Xerox standard printer typestyle.
typ_super, tsp	The "typ" character set, raised one-half line.
typ_sub, tsb	The "typ" character set, lowered one-half line.
footref	Default font used for footnote references.
sym, s	A set of symbols: including greek letters, mathematical symbols, line and corner elements for simple graphics--like boxes.
sym_super, ssp	The "sym" characters raised for superscripting. Includes a set of small numbers.
sym_sub, ssb	The "sym" characters, lowered for subscripting.
umlaut, um	Upper- and lower-case vowels with umlauts.
umlaut_italic, umi	The "um" characters in italics.

Device x9700_pr_land:

These are the fonts for the x9700_pr device that are also available in landscape mode.

ascii, m	11-point press roman medium body font.
super, sp	The "ascii" characters raised one-half line.
sub, sb	The "ascii" characters lowered one-half line.
bold, b	11-point press roman medium bold.
italic, i	10-point univers medium italics.
small, sm	10-point press roman medium body font.
head14, L	14-point press roman capitals, numbers, and a few special characters.
head18, VL	18-point press roman capitals and numbers (plus ,-./).
typ, t	12-character-per-inch fixed-pitch, "letter-gothic" typewriter font.
small_typ, tsm	approximately 14-character-per-inch fixed-pitch font in Xerox standard printer typestyle.
typ_super, tsp	The "typ" character set, raised one-half line.
typ_sub, tsb	The "typ" character set, lowered one-half line.

Device x9700_un:

These are proportionally spaced fonts in portrait orientation. Most of them are in the "univers" family, a sans-serif typestyle.

ascii, m	11-point univers medium body font.
super, sp	The "ascii" characters raised one-half line.
sub, sb	The "ascii" characters lowered one-half line.
ASCII, M	The "ascii" character set, but completely upper case.
bold, b	11-point univers medium bold.
BOLD, B	The "bold" character set, but completely upper case.
italic, i	10-point univers medium italics.
small, sm	10-point univers medium body font.
head14, L	14-point univers capitals, numbers and a few special characters.
head18, VL	18-point univers capitals and numbers (plus ,-./)
typ, t	12-character-per-inch, fixed-pitch, "letter-gothic" typewriter font
small_typ, tsm	approximately 14-character-per-inch, fixed-pitch font in Xerox standard printer typestyle.
un311, cd	A condensed 11-point font.
un411, cdb	A condensed bold font.
un611, cdi	A condensed italic font.
footref	Default font used for footnote references.
sym, s	A set of symbols: including greek letters, mathematical symbols, line and corner elements for simple graphics--like boxes.
sym_super, ssp	The "sym" characters raised for superscripting. Includes a set of small numbers.
sym_sub, ssb	The "sym" characters, lowered for subscripting.

Device x9700_un_land:

These are the x9700_un fonts for landscape mode.

ascii, m	11-point univers medium body font.
super, sp	The "ascii" characters raised one-half line.
sub, sb	The "ascii" characters lowered one-half line.
bold, b	11-point univers medium bold.
italic, i	10-point univers medium italics.
small, sm	10-point univers medium body font.
head14, L	14-point univers capitals, numbers and a few special characters.
head18, VL	18-point univers capitals an numbers (plus ,-./).
typ, t	12-character-per-inch, fixed-pitch, "letter-gothic" typewriter font
small_typ, tsm	Approximately 14-character-per-inch, fixed-pitch font in Xerox standard printer typestyle.
typ_super, tsp	The "typ" characters raised one-half line.
typ_sub, tsb	The "typ" characters lowered one-half line.

Device x9700__p12x6:

All these are fixed-pitch fonts in portrait orientation, 12 characters per horizontal inch, 6 lines per vertical inch. Note that this device does NOT include a bold-face font.

ascii, m	Similar to "letter-gothic" typewriter font.
super, sp	The "ascii" characters, raised one-half line.
sub, sb	The "ascii" characters, lowered one-half line.
italic, i	Italics.
script, scr	Script.
tcc	Similar to "prestige" typewriter font.
tya	Similar to "courier12" typewriter font.
sym, s	A set of symbols: including greek letters, mathematical symbols, line and corner elements for simple graphics--like boxes.
sym_super, ssp	The "sym" characters raised for superscripting. Includes a set of small numbers.
sym_sub, ssb	The "sym" characters, lowered for subscripting.

Device x9700__p10x6:

All these are fixed-pitch fonts in portrait orientation, 10 characters per horizontal inch, 6 lines per vertical inch.

ascii, m	Pica-style typewriter font.
super, sp	The "ascii" characters raised one-half line.
sub, sb	The "ascii" characters lowered one-half line.
bold, b	Boldface.
italic, i	Italics.
script, scr	Script.
grk, g	Greek symbols.
oaa	Optical character-reader font, type A.
oba	Optical character-reader font, type B.
tya	Similar to "pica" or "courier" typewriter font.
x1200, x	Xerox standard printer typestyle.
sym, s	A set of symbols: including greek letters, mathematical symbols, line and corner elements for simple graphics--like boxes.
sym_super, ssp	The "sym" characters raised for superscripting. Includes a set of small numbers.
sym_sub, ssb	The "sym" characters, lowered for subscripting.

Device x9700_p14x8:

All these are fixed-pitch fonts, in portrait orientation, approximately 14 characters per horizontal inch, 8 lines per vertical inch. Add a ".ls .75" control line to the top of your compin file because less space is needed between lines.

ascii, m	Xerox standard printer typestyle.
super, sp	The "ascii" characters raised one-half line.
sub, sb	The "ascii" characters lowered one-half line.
bold, b	Bold version of Xerox standard printer typestyle.
italic, i	Italics version of Xerox standard printer typestyle.

Device x9700_l14x8:

These are the fonts from the x9700_14x8 device type, for landscape orientation. In the device name, the two characters following the underscore are the letter "l" and the number "1" in that order.

ascii, m	Xerox standard printer typestyle.
super, sp	The "ascii" characters, raised one-half line.
sub, sb	The "ascii" characters, lowered one-half line.
bold, b	Boldface version of Xerox standard printer typestyle.
italic, i	Italics version of Xerox standard printer typestyle.

IV. COMPOSE CONTROLS CONTINUED

This chapter introduces several more specific and complicated formatting functions than those covered in Chapter II. If you still want more after reading this, turn to the *Honeywell WORDPRO Reference Guide* (AZ98) for a complete description of compose's capabilities. If you're still not satisfied, then you can write your own formatting controls (*macros*). The *Reference Guide* includes many controls designed specifically for macros.

FOOTNOTES

Compose makes the process of adding footnotes very simple if you are content with the default method of formatting and positioning them. You type in the actual footnote text at the place in the text where you want a reference inserted, and compose keeps track of numbering the footnotes and fitting them into the page layout. If you don't like the standard method, you can change it. However, these changes usually require additional work on your part, and the controls involved are not always reliable.[1]

The following controls perform two different functions: the first two delimit the text of the footnote, while the others are used to change how footnotes are formatted.

.bbf (block-begin-footnote)
Indicates that all following lines of text will be part of a footnote.

.bef (block-end-footnote)
Ends the block of footnote text begun with ".bbf".

.ftp (footnote-paged)
Changes format of all succeeding footnotes to "paged": footnotes are inserted at the bottom of the page where they are referenced and are renumbered from 1 for each new page. Since this is the default format, this control is used primarily to turn off an ".ftu" control.

.ftu (footnote-unreferenced)
All following footnotes will be unreferenced: no footnote reference is inserted in the text. Footnotes are inserted as in the "paged" format, but are left unnumbered.

.fth (footnote-hold)
Holds all succeeding footnotes, and inserts them at the end of the document. Footnotes are numbered continuously instead of being renumbered from 1 for each page.

In the default style of formatting, footnotes are said to be *paged*: they are added at the foot of the page on which they are referenced, separated from the text by a line. Numbering starts

---

[1] *Multics WORDPRO Reference Manual* (AZ98) includes many more controls, and describes many more footnote formats, than are introduced in this section. Most are too unpredictable to be included here, but might prove useful if you are willing to engage in a certain amount of experimentation.

over from one with each new page of footnotes. If there is not enough room on the page for all the footnotes that are waiting to be printed, they are continued at the bottom of the next page. If one of the final two text lines on a page includes a footnote reference, compose automatically ejects that page and begins a new one before printing the line that generates the footnote.

To add footnotes according to this default mode, simply insert a ".bbf" control line at the point in the text where you want the footnote reference to be located and start typing in the footnote text.[1] For example, the previous footnote was generated with the following input file:

```
start typing in the footnote text.
.bbf
This is a gratuitous footnote!
.bef
For example, the previous footnote . . .
```

In this default format, compose numbers footnotes automatically; you do *not* enter a number yourself. To eliminate this standard format, add an ".ftu" control line before any footnote or footnotes you want to change. Unreferenced footnotes are placed at the bottom of the page and separated from the text by a line, but no numbers are added in the text or in front of the footnotes themselves. This allows you to specify a new format by hand.

For instance, if you object to numbered footnotes, you can switch to asterisks instead.* This footnote was added with the following series of control lines:

```
you can switch to asterisks instead.*
.ftu
.bbf
.inl 2
.unh
*
Another gratuitous footnote!!
.bef
.ftp
```

In this example the final ".ftp" control line restores the default format used throughout the rest of the memo. The standard method may not work when inserting footnotes into the middle of text blocks that are formatted in some unusual way--like tables or charts. In this case, you have to add the footnote by hand with an ".ftu" control. See the examples on page 66 in Chapter VI.

When the ".fth" control is added to your input file, all succeeding footnotes are held until the end of the document. A reasonable amount of space is inserted after the concluding paragraph,

---

[1] This is a gratuitous footnote!

* Another gratuitous footnote!!

a line is drawn across the page, and all the pending footnotes are inserted, numbered continuously from one.[1]

CHARTS & TABLES

The chapter on basic formatting showed how the ".unh" and ".unl" controls could be used to set up simple tables. The following sections introduce ways of producing charts and tables that are more time-consuming but also more flexible.

Formatting by Hand

Sometimes you don't need controls to create columns or tables. If you don't mind using fixed-width characters, you can simply turn off fill mode and line up columns by hand.

Input File:

```
.fnt typ
bats      boxes     boats     screaming
cats      foxes     coats     yellow
hats      oxes      goats     ZONKERS
knats     soxes     gloats    green
spats     taxes     moats     MEANIES
```

Resulting Output:

```
bats      boxes     boats     screaming
cats      foxes     coats     yellow
hats      oxes      goats     ZONKERS
knats     soxes     gloats    green
spats     taxes     moats     MEANIES
```

However, if you want to use a proportional font for your table, the different-sized spaces can play havoc with your carefully aligned columns. Without the ".fnt typ" control line, the previous input file would have produced output that looked like this:

bats boxes boats screaming
cats foxes coats yellow
hats oxes goats ZONKERS
knats soxes gloats green
spats taxes moats MEANIES

The longer the columns get, the more scraggly they look. To avoid this effect, you have to use formatting controls. The following two sections introduce two different sets of controls for creating tables and charts.

---

[1] *Multics WORDPRO Reference Manual* (AZ98) includes an ".ift" control for inserting these pending footnotes at any point within the document (e.g., at the end of chapters). Unfortunately, this control does *not* renumber from one for the following set of footnotes. And when all the footnotes won't fit onto the last page of a chapter, ".ift" is not very good at breaking pages to accommodate the leftovers.

Defining Tabs with Compose

This first series of controls is designed to produce tab settings that work even with proportional fonts. You define tab stops according to the 10-pitch spacing by which compose measures line width, instead of adding variable-sized spaces by hand. When you specify a tab setting ten units from the left margin, the first characters of any words added at that tab setting will be evenly aligned an inch (i.e., 10 tenths of an inch) from the margin, regardless of the font size you are using.

.htd *name* #,#,# . . . (horizontal-tab-define)
Defines a series of tab stops # tenths of an inch across the page, and identifies this pattern by associating it with *name*.

.htn *c name* (horizontal-tab-on)
Turns on a previously defined series of tab stops identified by *name*. Chooses the character *c* to mark off tab stops within the input lines for the table. Turns fill mode off.

.htf c^1,c^2. . . (horizontal-tab-off)
Turns off any set of tab stops that uses the characters c^n as the tab stop marker. If no argument is given then all tabs are turned off.

Define a series of tab settings with the ".htd" control. The numbers must be separated by commas only; no spaces can be inserted. Turn the pattern on by using the ".htn" control line to designate a character for marking off this set of tab stops. Choose a character that does not occur any place in the text you will be adding to the chart. Activating tabs automatically turns filling off. When you have entered all the input lines for the table, turn the tab settings off with ".htf". If filling was on before the tabs were activated, then it is automatically restored.

Here is the example from the previous section, using tab stops defined with ".htd".

Input File:

```
.htd tableA 10,20,30
.htn $ tableA
bats$boxes$boats$screaming
cats$foxes$coats$yellow
hats$oxes$goats$ZONKERS
knats$soxes$gloats$green
spats$taxes$moats$MEANIES
.htf
```

Resulting Output:

bats	boxes	boats	screaming
cats	foxes	coats	yellow
hats	oxes	goats	ZONKERS
knats	soxes	gloats	green
spats	taxes	moats	MEANIES

Since the actual tab stops are specified only in the ".htd" control line, you can experiment with the spacing of the table by changing just that one line and formatting successive versions. Also, once a series of tab settings is defined with a unique name, it can be turned on again later in the document without being redefined. So, if you want to use these settings in a subsequent table, the ".htd" control line is not necessary. Just turn the table on again with an ".htn" control line and the same name as an argument. You are free to choose a new character with each new ".htn" control line.

This method of defining tab settings can accommodate input lines that are far less symmetrical than those shown above. You can skip tab stops by adding two tab stop characters in a row without any text in between. If you want to skip one or more tab stops at the end of the line, you do not need to insert any characters in their place.

Since filling is turned off, compose automatically begins a new series of tab settings for each input line, whether the final settings were used on the previous line or not. You can add spaces by hand to indent from tab stops instead of creating a new setting, and have ".spf" or ".spb" controls insert blank lines within the table.

Input File:

```
.htd  tableB  12,26,28,40,42
.htn  $  tableB
NAME$RESIDENCE$OCCUPATION$HOBBY
.spb
Joe Blow$Skokie$Carpet Sweeper$Being an all-round
$$$$$nice guy
Mary Normal$New York$Dental Hygienist$Sneezing
Peter Pan$Neverneverland$Kidnapping bored$Flying around in
$$  children$$green tights
```

Resulting Output:

NAME	RESIDENCE	OCCUPATION	HOBBY
Joe Blow	Skokie	Carpet Sweeper	Being an all-round nice guy
Mary Normal	New York	Dental Hygienist	Sneezing
Peter Pan	Neverneverland	Kidnapping bored children	Flying around in green tights

Table Mode

This second set of table formatting controls is more time-consuming than setting tabs, but it is also far more versatile. Each column is a separate formatting environment. The options chosen inside one column do not have to be explicitly turned off before switching to another column or returning to a standard text block. Unlike the ".htd" controls that automatically turn filling off, in table mode you have the full range of alignment and filling options available.

.tab *name column_definition:column_definition. . .* (table-define)
Defines a table of up to 20 columns and names it. Each column definition is divided from the next by a colon.

Each column definition must include: the column starting position, then a comma followed by a measure of the column width. These characters may be immediately followed by a two-character sequence specifying filling and alignment. By default the columns are filled. To turn filling off, you must specify an alignment with one of the following character sequences:

nc	unfilled and aligned in the center of the column
nl	unfilled and aligned at left column margin
nr	unfilled and aligned at right column margin

.tan *name* (table-on)
Turns on table mode according to the parameters defined for *name* in a previous ".tab" line.

.tac # (table-column)
Makes all following input lines into table column number #. To begin a new column, issue the same control line with the next column number.

.taf (table-off)
Turns off table mode.

When all these control lines are combined to create a table, they look something like this:

```
.tab tableA 5,25nl:30,25
.tan tableA
.tac 1
input lines for first column . . .
.tac 2
input lines for second column . . .
.taf
```

When defining columns on the ".tab" control line, you don't have to specify a width for the final column if you want it to extend all the way to the right margin. The comma must still be added after the number specifying the column starting position, however. If you want the first column to start at the left page margin, specify this setting as "1" rather than "0". A setting of zero returns an error message.

The following example demonstrates using the ".tab" controls to create columns with different fonts and varying alignments.

Input File:

```
.tab A 1,11nl:12,11nr:23,11nc:34,11nl:45,11nr
.tan A
.tac 1
Forty
Battling
Butlers
Crawled
Across
the
Floor.
.spb
.tac 2
.fnt italic
While
Thirty
Shrewish
Houseflies
Shouted
Out
For
More!
.tac 3
.fnt bold
More
WHAT?
Screamed
the
Butlers,
distracted
from the
fray.
.tac 4
.fnt typ
And
PROMPTLY
trampled
the
Houseflies
underfoot.
.tac 5
And
THAT
was the
end
of
THAT.
.taf
```

Resulting Output:

Forty Battling Butlers Crawled Across the Floor.	*While Thirty Shrewish Houseflies Shouted Out For More!*	**More WHAT? Screamed the Butlers, distracted from the fray.**	And PROMPTLY trampled the Houseflies underfoot.	And THAT was the end of THAT.

Note that none of the font changes added to any of the previous columns were turned off. Formatting controls affect only the column in which they occur; the fifth column is printed in the "ascii" font though the "typ" font of the previous column was not turned off. However, a font change in the final column of table mode carries over into the following text when not turned off with a ".fnt" control line. If this table is turned back on again, the font changes made here will still be in effect for each column.

The ".tab" controls handles font changes much more gracefully than the ".htd" controls. See the examples on pages 70 and 72 of Chapter VI for a comparison of these two different methods for constructing charts with internal font changes.

Though table mode tolerates columns of extremely varying lengths, trying to add blank space inside the table can cause problems. Using ".spb" or ".spf" controls to insert blank lines within columns sometimes adds a blank line across the entire table. In this case, try adding the appropriate number of blank lines instead. However, to guarantee a blank space that cuts across the entire table, you have to turn table mode off and then back on again.

Input File:

```
.tab resume 1,19nl:20,
.tan resume
.tac 1
.fnt bold
INTERESTS, SKILLS
AND HOBBIES
.tac 2
Extensive experience with dangerous and wealthy women, specializing
in trigger-happy redheads.  Will drink all comers under the table or
die trying.  Nerves of steel, thrives on suspense, intrigue and
frequent run-ins with the police.
.taf
.spb
.tan resume
.tac 1
ADDITIONAL
INFORMATION
.tac 2
Will go anywhere, do anything if the price is right and the case
is appropriately violent and stylish.  Results never guaranteed.
.taf
```

Resulting Output:

INTERESTS, SKILLS AND HOBBIES	Extensive experience with dangerous and wealthy women, specializing in trigger-happy redheads. Will drink all comers under the table or die trying. Nerves of steel, thrives on suspense, intrigue and frequent run-ins with the police.
ADDITIONAL INFORMATION	Will go anywhere, do anything if the price is right and the case is appropriately violent and stylish. Results never guaranteed.

Inserting Charts and Tables into Your Text

The following controls are especially suited for controlling the way tables and charts are fitted onto the page.

.bbp # (block-begin-picture)
Adds # lines of blank space as a picture block. If no argument is given, then all following input lines are processed as a picture block until a ".bep" control line is encountered. A picture block is broken across pages only if it is more than a page long. If a picture block does not fit on the current page, text lines following the block are added to fill up the current page and the picture is inserted on the following page.

.bep (block-end-picture)
Ends the picture mode begun by previous ".bbp". Ignored if there is no previous ".bbp".

These controls are more sophisticated and flexible versions of the ".bbk" and ".bek" controls introduced in an earlier chapter. The ".bbp" control creates a protected block of input lines called a *picture block* that cannot be split across pages. When used with an argument, this is a handy way of reserving sufficient white space for pasting in figures or illustrations. When combined with the ".bep" control, it is a convenient delimiter for units of formatted text like tables or charts.

To improve the appearance of overall page layout, any gap created by moving a picture block to a later page is filled with the text that follows the picture block in the input file. If you like, you can define a series of up to ten picture blocks in a row. Compose then inserts the blocks in the order they were defined and as space becomes available.

DEFINING AND USING VARIABLES

Keeping track of changing values is something computers do very well. It is this ability which makes text formatters so handy for gruelling tasks like cross references, indexes, and the numbering and renumbering of footnotes and pages during successive revisions. Compose stores information useful for such tasks in a series of pre-defined variables, and lets you create your own variables as well.

The following control lines allow you to define variables and insert them in text. In most formatting contexts, they have three primary uses: page numbering, cross referencing, and changing fonts within an input line.

.ur *%name%* (use-reference)
Scans input line for delimiters indicating a variable with a previously defined *name*. The default delimiter is the percent sign (%). This control evaluates the variable, and inserts the resulting value in the text. It applies to one input line. *Unlike other controls, it is separated from the text it applies to by a space, not a carriage return.* (See examples of syntax below.)

.srv *name expression* (set-reference-value)
Defines *name* as a user variable whose value is set equal to the value of the *expression* specified.

.csd *character* (change-symbol-delimiter)
Changes the % used to begin and end user and built-in variables to another character.

Inserting Pre-defined Variables

Compose automatically records and updates information about various characteristics of any input file it is processing: e.g., type of device it's being formatted for, size of bottom margin, whether fill mode is on. The changing values are stored in a set of *built-in variables* ("builtins"). These variables are available to users, and are especially helpful for writing your own formatting controls or *macros*. A list of all built-in variable names and a description of what information they store is included in *Multics Wordpro Reference Manual* (AZ98); you can view the list on line by typing "help compose.builtins".

For most routine formatting, the only builtins you are likely to need are "Date", "Time", and "PageNo". Of these, PageNo is by far the most useful. It is demonstrated in both sections on page numbering as well the following section on "User-defined Variables". Unfortunately, Date and Time both express their information in a very condensed format of fairly limited usefulness.

To insert the current value of a compose builtin into the printed text, add a ".ur" control *in front* of the input line containing the variable you want interpreted. Do not place the ".ur" on a line by itself, simply add a space between the control word and the text line that contains the variable. Remember to enclose the variable name with percent signs or it will be treated as plain text.

Input File:

```
.ur At the sound of the tone the time will be exactly %Time%.
```

Resulting Output:

At the sound of the tone the time will be exactly 1916.4.

Input File:

```
.ur On the morning of %Date% Miss Bliss was, as usual, not where
she should be.
```

Resulting Output:

On the morning of 01/28/85 Miss Bliss was, as usual, not where she should be.

You can use ".ur" in much the same way to interpret any of the Multics active functions. If, for example, you want to insert a longer version of the current date, you can use the [long_date] active function instead of the Date builtin.

Input File:

```
.ur On the morning of %[long_date]% Miss Bliss was, as usual, not
where she should be.
```

Resulting Output:

On the morning of January 28, 1985 Miss Bliss was, as usual, not where she should be.

The open- and closed-brackets are part of an active function's syntax and must be included for it to be recognized and properly interpreted. For basic formatting purposes, [long_date] is probably the single most useful active function. If you want to know more about active functions and how to use them, see *New Users' Introduction to Multics--Part II* (CH25). For a list of the most commonly used active functions, see *Multics Commands and Active Functions Quick Reference Guide* (AW17).

User-defined Variables

One of the most common uses for your own variables is to let you refer back and forth between pages in your document. Using a variable in a cross reference means that you don't need to know exactly what page number you're referring to or keep track of page numbers for passages that changes places during editing. The formatter inserts an updated page number for the reference each time it produces a new output file.

First, define a variable to hold the relevant page number with an ".srv" control line. Once the variable has been defined, its value can be inserted anywhere else in the file with a ".ur" control line. For example, to allow future references (like this one!) to an example on page 17, an ".srv" control line was inserted after that example in the input file:

```
your tie, the next they want to blow a hole in you.
.srv Redhead %PageNo%
```

This control line creates a variable called "Redhead" to store the current value of the PageNo builtin for that particular place in the text.[1] The names you give your variables can be up to 32 characters long and can include both upper- and lower-case letters as well as numbers and underscores. The main factor limiting name choice is your own ability to remember long names.

The variable "Redhead" can now be used whenever you want to refer to the page on which it was defined.

Input File:

```
I was suddenly overwhelmed with nostalgia for a previous
case. . . for a sweltering summer night and a sultry redhead.
For that very same Shirley Love who was sticking a gun into my ribcage
.ur back on page %Redhead%!
```

Resulting Output:

I was suddenly overwhelmed with nostalgia for a previous case. . . for a sweltering summer night and a sultry redhead. For that very same Shirley Love who was sticking a gun into my ribcage back on page 17!

Since the ".srv" control line defining "Redhead" occurs on a previous page of this memo, compose already knows the value of "Redhead" by the time it encounters the cross reference above and can immediately insert the appropriate page number. If a variable refers ahead to some value that has not yet been defined, you must add a "-pass 2" control argument to the compose command when formatting your document. This tells the formatter to make two passes over the input file. On the first pass, it notes the undefined variable reference and then locates its subsequent definition and value. On the second pass, it inserts the value. See "Producing an Output File" in the next chapter for a description of the "-pass" control argument.

You can also use the ".srv" control to define an abbreviation for a long or difficult group of words or letters (called *character strings*) that you find yourself having to type frequently. For example, to insert the greek letter "pi" without switching into the symbol font, you have to use a ".ur" control and the character sequence "*c#360". You can avoid typing this unmemorable combination over and over again by inserting the following control line near the beginning of your file:

```
.srv pi "*c#360"
```

Note that the character string is enclosed in quotation marks. You still have to use a ".ur", but now you represent a π simply as %pi%.

---

[1] The ".srv" control can be used to set variables equal to almost any meaningful value. For cross referencing purposes, however, always set it equal to the PageNo builtin.

Interpreting Other Character Sequences with ".ur"

The ".ur" control line can be used to interpret more than variables and active functions. You may sometimes find that you want a control to take effect in the middle of an input line, instead of being placed on a line by itself. To do this, add a ".ur" control in front of the input line, and enclose the control word between percent sign delimiters. When ".ur" scans the line and comes to a pair of percent signs, it interprets the control word as a formatting request rather than a series of letters to be treated literally. The control most commonly embedded in this way is ".fnt", for reasons given in the previous chapter.

Beginning an input line with a ".ur" control means that certain characters within that line will not be interpreted literally. The percent sign is one of these characters. If you consistently need a literal percent sign, you can change the default variable delimiter with the ".csd" control line introduced above. If your need for literal percent signs is intermittent, you can always insert one by prefacing it with an asterisk (*). In these control lines, the asterisk functions as an *escape character* and protects the following character from being interpreted as something else. To have one literal asterisk appear in the printed text, you must add two asterisks to the compin file.

TITLE LINES

The next few sections describe a series of controls designed to help you label the various sections of a long document. All of these controls make use of a special entity called a *title line* that combines a number of formatting features into a very useful package.

Filling is automatically turned off for title lines; they are never joined with the following or the preceding text. However, their most visible and distinctive feature is that they can be split into three separate units, each with its own alignment. These units are marked off by vertical bars (|). The syntax of a title line looks like this:

|. . . . text . . .. | . . . text . . . .| . . .text . . . |

Each of these units represents a separate alignment. Any text inserted between the first pair of delimiters is aligned at the left margin. Text inserted between the two middle delimiters is centered between the two margins, while text enclosed between the two right-hand delimiters is aligned at the right margin. For example, the following title line:

```
|aligned at left margin|centered|aligned at right margin|
```

would produce output like this:

aligned at left margin centered aligned at right margin

You do not need to add text in all three positions at once. To align text only at the right margin, you would use the following title line:

```
|||This title is flush right.|
```

which would be formatted like this:

This title is flush right.

All four vertical bars do not have to be included every time, only enough to specify an unambiguous alignment for the title. In the preceding example, the final vertical bar could have been left out with no ill effects. However, all title lines must begin with a vertical bar. If you accidentally forget to do so, compose sends you an error message, and the entire input line is ignored.

Title lines have another less visible feature that can be extremely useful. They automatically interpret embedded controls as well as built-in and user defined variables. You do not have to add a ".ur" at the beginning of the line. The implications of this will become clearer as the different uses for title lines are introduced.[1]

When processing title lines, compose is in an entirely different formatting mode from that used for standard text. You *cannot* insert title lines into your text simply by adding the appropriate number of vertical bars at the beginning of an input line. To have compose start formatting input as title lines, you have to add one of a series of possible controls. These controls allow you to add title lines to your text in many different ways and are introduced in the following sections.

Text Headers and Captions

This section describes controls for placing title lines in front of a block of text (as a text header) or following one (as a text caption). Although you can create such headers and captions by combining controls from previous chapters, these new controls accomplish the same thing more efficiently.

Both of the following control lines use a syntax similar to that of the ".ur" control. Instead of being placed on a separate line ahead of the title line, the control and any arguments are separated from the title line only by a space.

.tlh *# n title_line* (title-line-header)
Inserts a title line and associates it with the following text. The first variable indicates the number of lines to be inserted between the header and the text; the default is 0. The next number is the amount you want the line to be indented; the default is 0. If you choose to indent the title line using an unsigned number, you must also specify the first variable to keep the formatter from confusing the two values.

.tcl *# n title_line* (title-caption-line)
Inserts a title line and associates it with the previous text. This control is the same as ".tlh", except that the # argument indicates the number of spaces inserted *in front of* the caption line.

---

[1] The asterisk functions as an escape character within title lines as it does for ".ur" control lines. To insert a vertical bar (|) inside a title line, preface it with an asterisk. You can also change from a vertical bar to a different character with the "change-title-delimiter" control line summarized in the following section, "Inserting Blocks of Title Lines".

Compose's default method of operation is designed to make it easy to format paragraphs of filled text, since such paragraphs constitute the bulk of most documents. There are times, however, when these default conditions force you to pile up a series of control lines to counteract them. Section and chapter titles are a good example of this. If you tried to add a chapter title simply by modifying a standard text block, your input file might end up looking like this:

Input File:

```
Mary was skipping gaily along, thinking how glad
she was that her a teacher was allergic to lambs, when
suddenly she bumped right into:
.spb 2
.alc
.fnt bold
.brn 4
THE NEXT CHAPTER
.fnt
.alb
.spf 2
"Oh, no," she shrieked, "I thought I still had five pages left
to go before having to deal with that slimy green ogre lurking
in the bushes!"
```

Resulting Output:

Mary was skipping gaily along, thinking how glad she was that her a teacher was allergic to lambs, when suddenly she bumped right into:

THE NEXT CHAPTER

"Oh, no," she shrieked, "I thought I still had five pages left to go before having to deal with that slimy green ogre lurking in the bushes!"

The ".tlh" control[1] was designed for precisely this situation. The title line allows you to center the line easily without having to change the general alignment for the rest of the text, and lets you embed ".fnt" controls without needing a ".ur" to interpret them. The ".brn" control line is no longer necessary, because another ".tlh" function keeps the title line from being split away from its following text. Finally, you can specify the number of lines between the title line and its text on the ".tlh" control line without having to add a following ".spf". To produce exactly the same effect as that in the previous "Resulting Output", you could use the following input file:

---

[1] If you're looking through *Multics WORDPRO Reference Manual* (AZ98), you will notice that the latest version of this control is called ".thl" instead. Since that one is less reliable, the older version is documented here.

```
Mary was skipping gaily along, thinking how glad
she was that her a teacher was allergic to lambs, when
suddenly she bumped right into:
.spb 2
.tlh 2 ||%.fnt bold%THE NEXT CHAPTER%.fnt%
"Oh, no," she shrieked, "I thought I still had five pages left
to go before having to deal with that slimy green ogre lurking
in the bushes!"
```

Officially, the ".tlh" control is sufficient to interpret embedded controls inside the following title line. Unfortunately, it sometimes doesn't interpret font changes correctly. In this case, you can change fonts outside of the title line or add a ".ur" in front of the ".tlh" control line.

Make sure to always end one text block before adding a title for the next. Compose regards all ".tlh" controls within a text block as creating titles to be added at the beginning of that block. If a ".spf 2" had been used instead of a ".spb 2" to separate Mary from the chapter title, the output would look like this:

THE NEXT CHAPTER

Mary was skipping gaily along, thinking how glad she was that her a teacher was allergic to lambs, when suddenly she bumped right into:

"Oh, no," she shrieked, "I thought I still had five pages left to go before having to deal with that slimy green ogre lurking in the bushes!"

The ".tcl" control works in much the same way; it is used to add captions to charts, tables or inserts. Both header and caption lines are treated as part of the preceding or following text blocks respectively and are not split from them across page breaks. However, since compose allows widows of two lines or more, you may end up with the title or caption and one accompanying line of text on a separate page from the rest of the paragraph or figure. You might need a ".brn" or ".bbk" control to correct this.

Inserting Blocks of Title Lines

In contrast with controls like ".tlh" and ".tcl" which insert individual title lines, the following controls can add entire blocks of title lines. They conform to the more common compose syntax and get the input line to themselves.

.bbt # (block-begin-title)
Signals the beginning of a title block: the following input can include title lines. # indicates how many input lines to accept. If no argument is added, this control's action must be terminated by a ".bet".

.bet (block-end-title)
Signals the end of a title block. When the ".bbt" control is issued without an argument, this control is used to turn it off.

.ctd *character* (change-title-delimiter)
Changes the | used to begin title lines to another character.

These controls are particularly useful for creating simple tables, since title lines provide the simplest way of aligning parts of a single input line at both the left and the right margin. Other control lines can be inserted within a title block.

Input File:

```
.bbt
||%.fnt bold%TIME ESTIMATES FOR THE BLISS JOB%.fnt%
.spf 2
|Following sleazy blonde to a series of cheap hotels||15 hours|
.spf
|Getting pounded to pieces by a series of her goonish boyfriends||3 hours|
.spf
|Forgetting my wounds at shady joint in Chinatown||3 hours|
.spf
|Drugged unconscious by random and devious|||
| nightclub owner in Chinatown||48 hours|
.spf
|Total||69 gruelling hours|
.bet
```

Resulting Output:

TIME ESTIMATES FOR THE BLISS JOB

Following sleazy blonde to a series of cheap hotels	15 hours
Getting pounded to pieces by a series of goonish boyfriends	3 hours
Forgetting my wounds at shady joint in Chinatown	3 hours
Drugged unconscious by random and devious nightclub owner in Chinatown	48 hours
Total	69 gruelling hours

Because of their built-in ability to interpret variables, blocks of title lines can even be used to construct a table of contents. See example on page 68 in Chapter VI.

Page Headers and Footers

The following controls insert title lines above or below the body of the text on a series of pages as headers or footers. Note that the optional "e" and "o" are not variables and therefore are not italicized. Like Multics control arguments, these letters must be typed exactly as shown. They are separated from the other parts of the control by at least one space. Two types of controls are summarized below. One group is analogous to the ".tlh" and ".tcl" controls, inserting individual header or footer lines. The other set adds blocks of title lines (like ".bbt") as headers or footers.

.phl e,o *title_line* (page-header-line)
Adds the given title line as a header to the following pages: with an added "e", the title is inserted only on even pages; with an added "o" only on odd pages. With no added option, the title is placed on all pages.

.pfl e,o *title_line* (page-footer-line)
Adds the given title line as a footer to the following pages. The options function the same as for ".phl".

.bph e,o (begin-page-header)
Formats following input lines as a multiple-line page header until an ".eph" is encountered.

.eph (end-page-header)
Stops formatting input lines as a page header.

.bpf e,o (begin-page-footer)
Formats following input lines as a multiple-line page footer until an ".epf" is encountered.

.epf (end-page-footer)
Stops formatting input lines as a page footer.

The "e" and "o" options place the most pertinent information towards the more visible outside edge of the paper. For example, the following input lines were used to produce the footers added to the last few pages:

```
.pfl o |More Controls||Chapter 4|
.pfl e |Chapter 4||More Controls|
```

If you want a ".phl" control to affect the current page, you must to add it to your input file before any of the text lines that occur on that page. Otherwise, the change won't take effect until the next page. To change footers, however, the information only has to be inserted before the *end* of the page they are to appear on. A ."phl" or ".pfl" control continues to function until its action is turned off or changed by a new control.

You can use the ".bpf" and ".bph" controls to create footers or headers of several lines. The new footer at the bottom of this page was created by adding the following lines to this memo's input file:

```
.bpf
||DRAFT||
.spf
||All information subject to sudden, arbitrary changes||
.spf
|This is an ostentatious example||of a footer line!|
.epf
```

This new set of controls cancels the previous footer line, and inserts a block of three title lines instead. No "e" or "o" option is specified, so this new footer will be added on all pages. Since our pages are threatening to get rather bottom-heavy, these extraneous footers can be turned off with the following control line:

```
.pfl
```

By default, compose leaves four lines between the top or bottom of the page and the header or footer lines, and two lines between the header and footers and the text. All the page numbers in this memo were inserted in the default header position. The one-line footers on some of the previous pages of this chapter are also in the standard footer position. It is possible to change the position of these lines on the page. See "Page Layout", below, for more details on how.

PAGE NUMBERING REVISITED

The page-numbering method that compose offers you is extremely flexible. You can number your pages in just about any format and change the page count at any point in your document. Because of all these options, however, even the insertion of the plainest possible numbering system is not intuitively obvious. A complete explanation of this basic function comes rather late in the memo since pagination requires familiarity with built-in variables as well as header or footer lines.

Because you insert page numbers with the same ".phl" or ".pfl" controls that were introduced in the previous section, they are not summarized again in the chart below. The controls included here change the number of the next page or switch numbering styles.

.brp o,e,*n mode_type* (break-page)
Breaks to a new page, and changes the new page number according to the given option. An "o" or "e" sets it to the next odd or even number. If *n* is specified, then the number is either reset to *n* (if an unsigned variable is supplied) or is changed by *n* units (if a signed variable is supplied).

DRAFT

All information subject to sudden, arbitrary changes

This is an ostentatious example of a footer line!

If *mode_type* is included on the command line, the style of numbering is changed. Modes are given as two-character combinations:

ar Arabic (the default)
al lowercase alphabetic
au uppercase alphabetic
rl lowercase roman
ru uppercase roman

.brs # *"text" "header" "footer"* (break-skip)
Breaks the current page and adds # blank ones. The default argument is 1. You can add a line of text to be centered on the page, or a header or footer line; they must be specified in the order shown. Always type in the quotation marks, whether they enclose anything or not. These pages are not numbered unless you use the header or footer line to add them by hand. (See example, below).

The simplest way of adding page numbers was shown earlier in "Simple Page Numbering", page 18. Instead of centering the numbers as shown there, you could decide to position them at the outside margin of each page, using the "o" and "e" control-line options. In this case, two different control lines are needed to specify different alignments for odd and even pages. For example, the page numbers in this manual were inserted with the following control lines:

```
.phl o |AP-43-1||Page %PageNo%|
.phl e |Page %PageNo%||AP-43-1|
```

If you do not want the first page of text to be numbered, don't add any control lines until after the first page. The ".phl" control line must be the first item of information on the new page, while the ".pfl" control line can be added anywhere during the second page. In either case, the number added is a 2. The PageNo variable starts counting pages at the beginning of the document even if the numbers aren't printed.

By default, the PageNo builtin produces arabic numerals. You can change this default numbering mode to any of those listed in the chart above by using one of the ".brp" advanced functions.

When issued at the very beginning of a page, the ".brp" control does not cause a page break, but can be used to reset the PageNo variable's value or the kind of numbering used. To have the introductory matter in this memo numbered with lower-case roman numerals centered at the bottom of the page, the following control lines were added at the beginning of the input file:

```
.brp 1 rl
.pfl ||%PageNo%||
```

To change to a different numbering format for text and to start numbering from 1 again, the following control lines were used:

```
.pfl
.brp 1 ar
.phl o |AP-43-1||Page %PageNo%|
.phl e |Page %PageNo%||AP-43|
```

Though ".brp" can skip page numbers, it can't actually generate a blank page. To do this, you need the ".brs" control. If you want the blank page to be numbered as part of the text, specify an appropriate header or footer line as well as the number of blank pages to be inserted. If you don't want to use all three text lines, you must add pairs of quotation marks in their place. To create one blank page with an arabic numeral centered at the bottom of the page, type:

```
.brs "" "" "||%PageNo%||"
```

STILL MORE CONTROLS

Although hard to categorize, the following controls can be very useful.

..*pathname* or .ifi *pathname* (insert-file)
Stops processing the current input file, goes to named file and reads input from there. Both ".." and ".ifi" are equivalent, and can be used interchangeably.

.trn *abababab*. . . (translate)
For every *a* that occurs in the input file, produces *b* in the output. You can add as many *ab* combinations as you like to the control line. Issuing a new control line with *aa* cancels the translation process for that character.

.* *input line* (comment)
Any input line following the ".*" control is treated as a comment line and does not appear in the printed output or affect it in any way. This input line must be separated from the ".*" by a space.

The insert-file control permits you to divide a very long input file up into smaller units. This can help save formatting time (and therefore money), since you can format each of these smaller files separately while revising them. In this case, the ".ifi" control line makes it clear that the insertions are separate files. The pathname can refer to files outside your home directory as well as inside it.

You can choose to write your own formatting commands or *macros* to make frequent or complicated formatting tasks easier. If you do, add them to your compin files with the ".." version of this control. It makes them look more like regular compose controls.

The ".trn" control line sets one character equal to another. One of its most common uses is to set an infrequently used character equal to a space:

```
.trn #
```

In fill mode, compose collapses all blank space added within input lines to one space, regardless of how many you may have typed. If you insert the "#" as translated by the ".trn" command line above, the resulting white space is not condensed in this way. Compose automatically treats any period followed by a space as the end of a sentence, and inserts two spaces. The ".trn" control can be used to prevent this from happening as well:

```
.trn #
I said to Mr.#Jones, "Wipe that silly cigar off your face!"
```

CHANGING DEFAULTS

Page Layout

The formatting controls presented throughout this memo change the way text and blank space are laid out on a page, but so far these changes have taken place within the default page size described in the an earlier section, "Default Page Measurements". Though these settings are usually satisfactory for most purposes, compose allows you the option of changing them all: vertical margins, and length and width of text.

.vm *t,h,f,b* (vertical-margin-all)
Resets vertical margins. The variables can be signed or unsigned numbers. They must be separated by commas exactly as shown. If you are not changing all of the margins, you don't have to include a value for each variable, but you must still type in the surrounding commas. The variables are:

- *t* page top margin; the default value is 4 lines from top edge of paper.
- *h* page header margin; the default value is 2 lines between header and text.
- *f* page footer margin; the default value is 2 lines between text and footer.
- *b* page bottom margin; the default value is 4 lines to bottom edge of paper.

.vmt *n* (vertical-margin-top)
Resets page top margin to *n* or changes it by *n*; default setting is 4 lines from top edge of page.

.vmh *n* (vertical-margin-header)
Resets page header margin to *n* or changes it by *n*; default setting is 2 lines between header and text.

.vmf *n* (vertical-margin-footer)
Resets page footer margin to *n* or changes it by *n*; default setting is 2 lines between text and footer.

.vmb *n* (vertical-margin-bottom)
Resets page bottom margin to *n* or changes it by *n*; default setting is 4 lines between footer and bottom edge of page.

.pdl *n* (page-define-length)
Changes number of lines on a page. The default is 66 10-pitch lines.

.pdw *n* (page-define-width)
Changes width of the text lines. The default is 65 10-pitch characters per line.

If you are using a smaller font than the 10-pitch standard, you may have to approximate. For example, a 12-pitch font generally fits 78 characters per default line rather than 65.

Widowing

A *widow* is a piece of text broken off from the rest of its paragraph by a page break. Aesthetically, a widow is any number of lines that looks abandoned and lonely enough to deserve this name. Compose enforces a more precise definition: two is the minimum number of lines that it will widow for the sake of even bottom margins. If you disapprove of this policy, you can change it with the following control. A negative value or one greater than the current page length both cause errors.

.wit *n* (widow-text)
Changes the widow size. The default is 2.

V. FORMATTING AND PRINTING YOUR FILE

This chapter describes the two commands used to format and print your text. To actually have the formatting controls that you added to your input file applied to the text, issue the "compose" command. You can choose to format your file directly at your terminal or to translate your input file into instructions that a local printer can understand.

Formatted output cannot be immediately directed to a printer as it can to your terminal. Instead, you have the formatter create an output file that can then be sent off to the printer with the Multics "enter_output_request" ("eor") command. This is a standard command for sending any kind of text file to the printer. When sending compout files, however, a few specific control arguments are needed, as described later in this chapter.

COMPOSING AT YOUR TERMINAL

Screen Terminals

If you are working on a display terminal, the simplest and fastest way to get an idea of what your formatted document will look like is to have compose format it at your terminal screen. To do this, invoke compose by typing the command's name ("compose") followed by the name of the file to be formatted.

To format a segment of yours called "goodgrief.compin", type:

```
compose  goodgrief
```

Note that you can omit the ".compin" part of the segment name; compose automatically searches for a file name ending in ".compin".

Compose formats your text according to the formatting instructions in the input file, and prints each output line at the terminal. If you have given compose a control line it cannot understand, an error message about it is printed after the document is finished (error messages are explained more fully in a following section).

The output that appears on your screen shows how aspects of page layout (indentations, undentations, charts, etc.) are working. However, what you see on the screen differs from printed output in two major ways. Since a terminal's page length and line length do not generally coincide with the pages and lines produced by the printer, this technique does not let you predict how pages will break, or when an unfilled output line will be too long. Also, most font changes will not be represented. Sometimes underscores can be used in place of italics, but most font switching requests simply generate an error message.

The compose command includes an option to display only part of the specified file, which is especially convenient when dealing with larger documents. To request a certain number of formatted pages, add the control argument "-pages" (or "-pgs") to your compose command line. You can specify individual pages (as few as one, and as many as 100) by listing all their numbers with spaces in between:

```
compose ziggity  -pages 6 11 13 57
```

You can also request a range of pages by listing the first and last numbers, separated by a comma:

```
compose ziggity -pgs 8,16
```

Printing Terminals

Sometimes, you may feel like using a printing terminal to bypass the longer process of formatting and printing an output file. For more attractive output, you can print onto separate sheets of paper instead of continuous-form terminal paper. In this case, tell compose to stop before each page (including the first) so that you can insert a single sheet of paper in the terminal. You do this by adding the "-stop" option when you issue the compose command:

```
compose fatmemo -stop
```

Once the sheet is aligned, type a single carriage return; compose prints one page of output and then stops. At this point you can go on to the next page. You can also ask compose to re-do the page it just typed by entering "reprint" followed by a carriage return. This is useful if your terminal ribbon runs out or you misalign your paper. When it gets to the last page, compose stops. Type one more carriage return to get a ready message.

PRODUCING AN OUTPUT FILE

Getting formatted output from an Information Systems printer is a slightly more complicated process. It requires a longer "compose" command line than the examples shown above. You must add a control argument to specify which of the supported device types you want the file to be formatted for. You must also add a second control argument telling compose to store the formatted output in an on-line segment instead of sending it to your terminal.

Control Arguments

When having compose produce an output file, the following are some of the most useful control arguments. For a complete list of all possible control arguments, see *Multics WORDPRO Reference Manual* (AZ98). Since "compose" is categorized as a WORDPRO command rather than a Multics command, this information is not included in the documentation for standard commands and active functions.

Of the arguments listed below, *the first three are necessary* to ensure the effectiveness of the formatting controls you inserted.

-output_file, -of
: Directs formatted output to an on-line segment instead of the terminal. The name of the output file must include a ".compout" suffix.

-device *name*, -dv *name*
: Specifies which device type the file is going to be formatted for. This choice determines page orientation and the group of fonts available. The *name* argument can be any of the eight device type names listed in Chapter III, "Fonts".

-indent *n*, -ind *n*, -in *n*
Specifies the size of the left margin. *n* is measured in tenths of an inch, so a value of 10 produces a one-inch margin regardless of the font size actually being used. If you don't add this control argument, your text will start at the left edge of the paper.

-pass *n* Specifies the number of formatting passes compose makes over your input file. The default is 1. Two passes are needed if you ever refer to a variable *before* defining it, or if you are adding a table of contents or an index.

-hyphenate, -hyph
Turns on compose's hyphenation facility. If not used, then words are not broken across lines. This option costs more to use, but it may improve the appearance of your document by reducing the amount of extra blank space appearing in filled text.[1]

-linespace *n*, -ls *n*
Changes the default line spacing for the document to *n* spaces. This allows you to print out single-spaced and double-spaced versions of the same file without making any internal changes.

-pages *page_list*, -pgs *page_list*
Selects only a section of your compin file for formatting. This option allows you to format sections of your text. The page numbers which you specify refer to the numbers which appear on the pages you want. Also described in the section "Composing at Your Terminal", above.

To compose part of your "fatmemo.compin" segment with the univers proportional-width device and have it double-spaced, type:

```
compose fatmemo -of -dv x9700_un -ind 10 -pgs 5,10 -ls 2
```

Once you have polished your prose to a radiant precision, and you decide to produce the most professional of all possible final versions, you might change to the following options:

```
compose fatmemo -of -dv x9700_un -ind 10 -hyph -pass 2
```

Multics prints a ready message when the compose command above has finished formatting. At this point, you might use the WORDPRO "create_wordlist" command to check for possible typographical errors. For full information on using dictionaries, for both hyphenation and spelling checking, see *Multics WORDPRO Reference Manual* (AZ98).

When the system is heavily loaded, composing an input file can take a long time, especially if you request more than one formatting pass. In this case, you might want to compose and print

---

[1] Compose hyphenates by looking up words in a computer "dictionary". You can create other dictionaries to override the standard hyphenation or add words specific to your project (which may not be in the standard dictionary). See *Multics WORDPRO Reference Manual* (AZ98) for further information.

the file as an absentee job. See Information Services Memo MS-1, *Multics at MIT* on how to enter absentee requests.

COMPOSE ERROR MESSAGES

If at some point in your input file you have issued a combination of compose controls that leads to an error, compose generally registers this by sending you an error message. When output is directed to a file rather than the terminal, these messages are displayed as the file is being processed and before the ready message signalling the end of formatting.

If you get any such error messages, go back and repair the input file (using a text editor). Compose makes this easier by including line numbers with the error messages.[1] Then give the compose command again. Normally you should *not* print the compout file until your compose command executes with no errors.

Sometimes (not very often) compose responds to an error by stopping and leaving you at a new command level instead of just sending you a message. When this happens, you can give the "pi" command to identify the line being processed when the problem occurred. This may help you determine how to get around the problem next time.

Some messages are more in the nature of warnings, letting you know that compose ignored a command that didn't make sense.

Anatomy of an Error Message

The following example represents a set of two error messages returned while compose was formatting the "fatmemo.compin" segment.

```
compose error list: (Vers. 9.17k)
fatmemo; 46:
    Unknown control request.
    .unb
fatmemo; 509: index; 3:
     Variable "IndexOff" undefined for this reference.
     .ts "%IndexOff%" = "1"
r 15:08 14.226 842
```

The list of errors is prefaced by an introductory line including the version of compose that is formatting your file. Here, the error list ends with a ready message, since the problems encountered were not serious enough to keep compose from processing the entire file.

The actual error messages are composed of three lines each. The first line begins with the name of the compin file being formatted, and then lists the line number on which the error occurs. The second line tells you what the problem is in more or less comprehensible terms. The final line is a verbatim copy of the actual input line responsible for the problem. Though

---

[1] In Emacs, use the "ESC-G *line_number*" request to go directly to the lines causing the problems.

the formatting error may not be immediately apparent from this information, examining the relevant section of the compin file often shows you what went wrong.

In the second of the error messages shown above, the problem did not occur on one of "fatmemo"'s lines, but within a file called "index" that was inserted with a ".." or ".ifi" control. In this case, the "fatmemo" line number refers to the ".ifi index" control line. The extra information on this line lists the name of the inserted file and the line number within *that* file which is causing the error.

THE eor COMMAND

Once you've got a compout file that seems to be error-free, use the standard Multics "enter_output_request" ("eor") command to send it to the laser printer. If you are not familiar with the basic functions of this command, see Memo MS-6, *Essential Multics*.

You must use some extra control arguments to have the printer process the file according to the compose format.

-forms control,*printing options*
: Tells the laser printer that compose has inserted font instructions into the compout file. You can request specific printing options with additional arguments. The "1s" argument specifies one-sided printing, while "^h" requests unpunched paper. For thesis-quality unpunched paper, use the request "bond" instead of "^h". The arguments may be given in any order, and must be separated with commas.

-no_endpage, -nep
: Prevents the "eor" command from inserting page breaks into your output file--which compose has already broken into pages. (If you get back output that has a few lines from the bottom of each page moved to the next page, followed immediately by another page eject, you may have forgotten to use "-nep".)

-delete, -dl
: Deletes the compout file after it has been printed. Compout files are very good candidates for automatic deletion, since a new one must be generated for each revision of the compin file. Adding a "-delete" ("-dl") control argument minimizes the number of useless compout files taking up storage space in your directory.

To print your "fatmemo" compout file on unpunched paper, and delete it after it's been sent to the printer, type:

```
eor fatmemo.compout -forms control,^h -nep -dl
```

To have the same file printed one-sided on archival bond (suitable for MIT theses), type:

```
eor fatmemo.compout -nep -dl -forms 1s,control,bond
```

You must be sure to add the ".compout" suffix. If you specify only the first part of the name, "eor" will send back a message saying it can't find the segment. And if you accidentally

request printing of the compin segment, Multics makes no comments but sends it off to be printed. The resulting output looks exotic, but is generally pointless.

NOTE: *Never* use the "-indent" control argument with "eor" when specifying "-forms control". If you do, the spaces added by "eor" displace the printer control instructions in the output so that they are not interpreted.

These lengthy "eor" command lines quickly become a nuisance to type correctly each time. You can save yourself time and typing by creating your own abbreviations for your most commonly used versions of this command line. See the *New Users' Introduction to Multics -- Part II* (CH25) for a description of the Multics "abbrev" command.

If you want more information about the "eor" command or the Information Services printing facilities, type "help x9700" and "help x9700_forms" or see Memo MS-1, *Multics at MIT*, which includes a more complete description of the "eor" command.

VI. EXAMPLES OF INPUT AND RESULTING OUTPUT

The following series of extended examples shows how different controls are combined in actual formatting tasks.

Equations

Input File:

NOTE: The first line beginning with ".ur" is meant to be typed in as one input line. Carriage returns were inserted here for clarity.

```
.trn !
.fif
      n
.ur %.fnt bold%S%.fnt%(%.fnt bold%b%.fnt%) = %.fnt sym%S%.fnt% e%.fnt sub%i%.fnt%
 (%.fnt bold%b%.fnt%)%.fnt sym%A%.fnt%%.fnt super%2%.fnt%e%.fnt sub%i%.fnt%
 (%.fnt bold%b%.fnt%)
     i=1
.spb 6
.tab equation 10,5nl:15,4nl:19,20nl:28,6nl:33,2nl
.tan equation
.tac 1
!
!
.ur %.fnt sym%@%.fnt%
.ur |%.fnt sym%s%.fnt%%.fnt super%(k+1)%.fnt%
.ur %.fnt sym%**%.fnt%
.tac 2
!
!
.ur %.fnt sym%#%.fnt%%.fnt sub%2%.fnt%
| =
.ur %.fnt sym%+%.fnt%
.tac 3
 n
.ur  %.fnt sym%S%.fnt%  XWTS%.fnt sub%i%.fnt%
i=1
-------------------------
    1  n
.ur  d%.fnt sub%H%.fnt% -  %.fnt sym%S%.fnt% XWTS%.fnt sub%i%.fnt%%.fnt super%(k)%.fnt%
    n i=1
.tac 4
.ur  %.fnt sym%@%.fnt%
.ur %.fnt sym%²%.fnt%|z%.fnt sub%i%.fnt%.fnt super%(k)%.fnt%
.ur  %.fnt sym%**%.fnt%
.tac 5
.ur %.fnt sym%#%.fnt%2
|
.ur %.fnt sym%+%.fnt%
.taf
```

Resulting Output:

NOTE: If you want to use a proportional font, and plan to include many equations, the spacing for the "univers" font generally works better than for "press roman".

$$S(b) = \sum_{i=1}^{n} e_i(b) \nabla^2 e_i(b)$$

$$\left[\sigma(k+1)\right]^2 = \frac{\sum_{i=1}^{n} XWTS_i \cdot \left[z_i^{(k)}\right]^2}{d_H - \frac{1}{n}\sum_{i=1}^{n} XWTS_i^{(k)}}$$

Inserting a Footnote into a Chart

Input File:

```
.htd Q 20,35,55
.htn ! Q
.spb 2
SLOBOVIAN KINGS!DATES!RURITANIAN KINGS!DATES
.spb
.ur Rhoemetalces I%.fnt symbol%1%.fnt%!66 - 2 B.C.!Gotarzes II!44 - 28 B.C.
.ftu
.bbf
.inl 2
.unh
.ur %.fnt symbol%1%.fnt%
A posthumous child, his rule began three months before he was born.
His strong-minded mother, Galla Draconia, was regent from 66-48 B.C.
and effective ruler until her death in 23 B.C.  After that his favorite
concubine Roxanna controlled affairs of state until 2 B.C., when she
poisoned Rhoemetalces to make way for her son.
.bef
.brf
Rhescuporis!2 B.C. - 18 A.D.!Ariobarzanes I!28 B.C. - 3 A.D.
Artaxis the Short!18 - 19 A.D.!Demonax the Satanic!3 - 23 A.D.
Retroaxis!19 - 22 A.D.!Volpone the Vulture!24 - 29 A.D.
Extraxis!22 - 36 A.D.!Phraates III!29 - 38 A.D.
Urgothorp the Goon!36 - 54 A.D.!Xargofax the Dim!38 - 56 A.D.
.htf
```

Resulting Output:

SLOBOVIAN KINGS	DATES	RURITANIAN KINGS	DATES
Rhoemetalces I[1]	66-2 B.C.	Gotarzes II	44-28 B.C.
Rhescuporis	2 B.C. - 18 A.D.	Ariobarzanes I	28 B.C. - 3 A.D.
Artaxis the Short	18 - 19 A.D.	Demonax the Satanic	3 - 23 A.D.
Retroaxis	19 - 22 A.D.	Volpone the Vulture	24 - 29 A.D.
Extraxis	22 - 36 A.D.	Phraates III	29 - 38 A.D.
Urgothorp the Goon	36 - 54 A.D.	Xargofax the Dim	38 - 56 A.D.

[1] A posthumous child, his rule began three months before he was born. His strong-minded mother, Galla Draconia, was regent from 66-48 B.C. and effective ruler until her death in 23 B.C. After that his favorite concubine Roxanna controlled affairs of state until 2 B.C., when she poisoned Rhoemetalces to make way for her son.

Table of Contents

Input File:

```
.brp 1 rl
.pfl ||%PageNo%
.spb 2
.tlh ||%.fnt bold%CONTENTS%.fnt%
.spb 3
.bbt
|%.fnt bold%GETTING STARTED%.fnt%
.spb
.inl 5
|INTRODUCTION||1
|LOGGING IN TO MULTICS||2
|CORRECTING TYPING MISTAKES||3
|WHEN YOUR TERMINAL SCREEN FILLS UP||4
|INTERRUPTING EXECUTION||4
|LOGGING OUT||4
.spb 3
.unl
|%.fnt bold%BASIC COMMANDS%.fnt%
.spb
|MULTICS COMMAND SYNTAX||5
|SEGMENTS, DIRECTORIES, PATHNAMES||6
|COMMAND EXAMPLES||8
|GETTING PRINTED OUTPUT||10
.spb 3
.unl
|%.fnt bold%SUBSYSTEMS%.fnt%
.spb
|SUBSYSTEM FEATURES||13
|EMACS TEXT EDITOR||14
|QEDX TEXT EDITOR||16
|SENDING AND RECEIVING ELECTRONIC MAIL||18
|FORUM||21
.spb 3
.unl
|%.fnt bold%FURTHER INFORMATION%.fnt%
.spb
.ur |MULTICS HELP SYSTEM||23
.ur |DOCUMENTATION||24
.bet
.inl
```

Resulting Output:

CONTENTS

Using "htd" to Construct a Chart

Input File:

```
.tlh 4 ||%.fnt bold%FLY FISHING CLASS%.fnt%
.ur %.fnt italic%ATTENDEES%.fnt%
.htd tableA 3,22,41,56
.htn $ tableA
.ur %.fnt bold%$NAME$DEPARTMENT$ROOM$TEL%.fnt ascii%
.spb
$Bill Beetlebrow$Music$14N-430$2826
$Fifi Laroux$Athl Dept$W32-121$7947
$Prof. Snort$Music$14N-433$3671
$Daffy Duck$Biology$16-512$5054
$Dudley Doright$Police$NE40-408$3175
$Tweety Bird$Entomology$12-090$8278
$Ezekial Frump$Chem E$66-350$4561
$Mary Lamb$Physics$6-107$4851
.htf
.spb 3
.fnt italic
CLASSROOMS:
.fnt
.htd tableB 3,18,31,49
.htn $ tableB
.ur %.fnt bold%$DATES$DAY$LOCATION$TIME%.fnt%
.brf
$June 9$Wed$11-401$9:00 to 12:00
$June 11$Fri$11-401$9:00 to 12:00
$June 16$Wed$11-401$9:00 to 12:00
$June 18$Fri$11-401$9:00 to 12:00
.htf
.spb 3
.htd tableC 3,31
.htn $ tableC
.ur %.fnt bold%$CLASS$CLASS
.ur $INSTRUCTORS$MANAGER%.fnt%
.spf
$E. Hemingway$  G. Stein
$Nick Adams$  11-316
$396-9494$  3-4290
.htf
```

Resulting Output:

FLY FISHING CLASS

ATTENDEES

NAME	DEPARTMENT	ROOM	TEL
Bill Beetlebrow	Music	14N-430	2826
Fifi Laroux	Athl Dept	W32-121	7947
Prof. Snort	Music	14N-433	3671
Daffy Duck	Biology	16-512	5054
Dudley Doright	Police	NE40-408	3175
Tweety Bird	Entomology	12-090	8278
Ezekial Frump	Chem E	66-350	4561
Mary Lamb	Physics	6-107	4851

CLASSROOMS:

DATES	DAY	LOCATION	TIME
June 9	Wed	11-401	9:00 to 12:00
June 11	Fri	Charles River	9:00 to 12:00
June 16	Wed	11-401	9:00 to 12:00
June 18	Fri	Idaho	9:00 to 5:00

CLASS INSTRUCTORS	CLASS MANAGER
E. Hemingway	G. Stein
Nick Adams	11-316
396-9494	3-4290

If you want to change fonts within an ".htd" format, it is best to separate the different fonts with a block break. The first set of columns (Attendees) is more evenly aligned than the following two because of the ".spb" inserted between the bold face line (see input line 5) and the rest of the columns. Aligning numbers and letters can also cause problems in an ".htd" format. Compare this output with the following example of ".tab".

Same chart with ".tab" controls

Input File:

NOTE: the following input was divided up into two columns to conserve space, but would actually be typed in one continuous column. Here, the abbreviated font names from the "List of Available Fonts" section are used.

```
.tlh 4 ||%.fnt b%FLY FISHING CLASS%.fnt%
.ur %.fnt i%ATTENDEES%.fnt%
.brf
.tab fly 3,19nl:22,19nl:41,15nl:56,5nl
.tan fly
.tac 1
.ur %.fnt b%NAME%.fnt%
Bill Beetlebrow
Fifi Laroux
Prof. Snort
Daffy Duck
Dudley Doright
Tweety Bird
Ezekial Frump
Mary Lamb
.tac 2
.ur %.fnt b%DEPARTMENT%.fnt%
Music
Athl Dept
Music
Biology
Police
Entomology
Chem E
Physics
.tac 3
.ur %.fnt b%ROOM%.fnt%
14N-430
W32-121
14N-433
16-512
NE40-408
12-090
66-350
6-107
.tac 4
.ur %.fnt b%TEL%.fnt%
2826
7947
3671
5054
3175
8278
4561
4851
.taf
.spb 3
.ur %.fnt i%CLASSROOMS:%.fnt%
.tab fish 3,15nl:18,13nl:31,18nl:49,10nl
.tan fish
.tac 1
.ur %.fnt b%DATES%.fnt%
June 9
June 11
June 16
June 18
.tac 2
.ur %.fnt b%DAY%.fnt%
Wed
Fri
Wed
Fri
.tac 3
.ur %.fnt b%LOCATION%.fnt%
11-401
11-401
11-401
11-401
.tac 4
.ur %.fnt b%TIME%.fnt%
9:00 to 12:00
9:00 to 12:00
9:00 to 12:00
9:00 to 12:00
.taf
.spb 3
.tab class 3,28nl:31,10nl
.tan class
.tac 1
.fnt b
CLASS INSTRUCTOR
.fnt
E. Hemingway
Nick Adams
396-9494
.tac 2
.fnt b
CLASS MANAGER
.fnt
G. Stein
11-316
3-4290
.taf
```

Resulting Output:

FLY FISHING CLASS

ATTENDEES

NAME	DEPARTMENT	ROOM	TEL
Bill Beetlebrow	Music	14N-430	2826
Fifi Laroux	Athl Dept	W32-121	7947
Prof. Snort	Music	14N-433	3671
Daffy Duck	Biology	16-512	5054
Dudley Doright	Police	NE40-408	3175
Tweety Bird	Entomology	12-090	8278
Ezekial Frump	Chem E	66-350	4561
Mary Lamb	Physics	6-107	4851

CLASSROOMS:

DATES	DAY	LOCATION	TIME
June 9	Wed	11-401	9:00 to 12:00
June 11	Fri	11-401	9:00 to 12:00
June 16	Wed	11-401	9:00 to 12:00
June 18	Fri	11-401	9:00 to 12:00

CLASS INSTRUCTOR	CLASS MANAGER
E. Hemingway	G. Stein
Nick Adams	11-316
396-9494	3-4290

A Sample Resume

Input File:

```
.alc
.ur %.fnt bold%GERTRUDE GLUMLEY-FITZHUMBERT%.fnt%
.spb
.alb
.fin
.tlh |1028 1/2 Pine St.||Home:  (415) 472-1532|
.tlh |San Francisco, CA 94107||Business:  (415) 942-0700|
.spb
.fnt symbol
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
.spb 2
.fnt ascii
.inl 12
.unh
.ur %.fnt bold%EDUCATION%.fnt%
.ur %.fnt bold%Miss Katherine Quarentine's School for Young Ladies%.fnt%.
San Francisco, CA., June 1932.  Voted "Girl with the Most Exhausting Social
Life" for three years running.
.spb 2
.unh
.ur %.fnt bold%EXPERIENCE%.fnt%
.ur %.fnt bold%Secretary%.fnt%, September 1935 - present.
Maxwell K. Swill, Private Investigator
.spb
Screen everyone who walks through the door.  Get rid of creditors and
finance company representatives.  Estimate potential cases for
amount of violence, profit and sex they might provide.  Mix drinks and
keep the liquor cabinet stocked.  Entertain associates of Mr. Swill
when he is not available.  Eliminate enemies of Mr. Swill whenever possible.
Responsible for making sure that Mr. Swill is always awake by 6:00 P.M.
and locating him if he isn't asleep on the office sofa by 8:00 A.M.
As time permits, seduce attractive mobsters for timely information
on white slave market, drug traffic and pending massacres.
During evenings, monitor latest scandles among San Francisco's smart
young set for potentail blackmail victims or divorce cases.
.spb
.ur %.fnt bold%Glittering Young Socialite%.fnt%,  Fall 1933 - present.
San Francisco's High Society
.spb
Attend at least three parties a night on the weekends, carouse with inner
circle of intimate friends during the week.  Cultivate languid demeanour
denoting utter boredom with life in general and intimate friends in
particular.  Temper tantrums on a bi-weekly basis.  Firing personal
maid once a month.  Smash something worth more than $1000 whenever I feel
like it.  Making sure parents aren't aware of younger sister's drug habits,
oldest brother's three wives, or other brother's Mafia connections.
Paying off father's mistresses when he tires of them.  Keeping mother on
the wagon.  Ignore all eligible bachelors with appropriate social
connections and incomes.  Frequent older men of shady past and
dubious continental titles.
.spb
.ur %.fnt bold%World Traveler%.fnt%,  Summer 1932 - Summer 1933.
.spb 2
.unh
.ur %.fnt bold%INTERESTS%.fnt%
.ur %.fnt bold%Guns.%.fnt%  Expert markswoman and collector of antique
firearms.
.spb
.ur %.fnt bold%Clothes.%.fnt%  Twenty hours per week shopping.  Extensive
knowledge of the more exclusive Bay Area stores.
.spb
.ur %.fnt bold%Gangsters.%.fnt%  If they're dark, not too fat, and don't
chew tobacco.
.inl
```

Resulting Output:

GERTRUDE GLUMLEY-FITZHUMBERT

1028 1/2 Pine St.
San Francisco, CA 94107

Home: (415) 472-1532
Business: (415) 942-0700

EDUCATION **Miss Katherine Quarentine's School for Young Ladies.** San Francisco, CA., June 1932. Voted "Girl with the Most Exhausting Social Life" for three years running.

EXPERIENCE **Secretary**, September 1935 - present. Maxwell K. Swill, Private Investigator

Screen everyone who walks through the door. Get rid of creditors and finance company representatives. Estimate potential cases for amount of violence, profit and sex they might provide. Mix drinks and keep the liquor cabinet stocked. Entertain associates of Mr. Swill when he is not available. Eliminate enemies of Mr. Swill whenever possible. Responsible for making sure that Mr. Swill is always awake by 6:00 P.M. and locating him if he isn't asleep on the office sofa by 8:00 A.M. As time permits, seduce attractive mobsters for timely information on white slave market, drug traffic and pending massacres. During evenings, monitor latest scandles among San Francisco's smart young set for potentail blackmail victims or divorce cases.

Glittering Young Socialite, Fall 1933 - present. San Francisco's High Society

Attend at least three parties a night on the weekends, carouse with inner circle of intimate friends during the week. Cultivate languid demeanour denoting utter boredom with life in general and intimate friends in particular. Temper tantrums on a bi-weekly basis. Firing personal maid once a month. Smash something worth more than $1000 whenever I feel like it. Making sure parents aren't aware of younger sister's drug habits, oldest brother's three wives, or other brother's Mafia connections. Paying off father's mistresses when he tires of them. Keeping mother on the wagon. Ignore all eligible bachelors with appropriate social connections and incomes. Frequent older men of shady past and dubious continental titles.

World Traveler, Summer 1932 - Summer 1933.

INTERESTS **Guns.** Expert markswoman and collector of antique firearms.

Clothes. Twenty hours per week shopping. Extensive knowledge of the more exclusive Bay Area stores.

Gangsters. If they're dark, not too fat, and don't chew tobacco.

APPENDIX A
SYMBOL FONT CHARACTERS

The following chart lists the symbols defined for the symbol font, and their keyboard equivalents. This font is available on all portrait mode device types. If you prefer to represent the character with its octal code instead of a keyboard character, you must preface the three digit listing with a backslash "\".

To insert the small version of the greek letter "pi" you must type its octal code "\037". There is no keyboard equivalent. See page 45 for another way of inserting "pi".

SYMBOL	NAME	OCTAL CODE	KEYBOARD EQUIVALENT
×	multiply	166	v
÷	divide	047	'
≤	less than or equal	074	<
≥	greater than or equal	076	>
≠	not equal	073	;
≃	approximately equal	132	Z
~	equivalence	057	/
∝	proportional to	126	V
▽	del	101	A
∞	infinity	102	B
≡	identity	130	X
∫	integral	077	?
√	radical	140	`
‖	parallel	072	:
∂	partial differential	115	M
=	double overline	137	_
←	left arrow	160	p
↓	down arrow	041	!
→	right arrow	120	P
┌	upper left box corner	100	@
├	left middle box side	050	(
└	lower right box corner	053	+
│	box side	174	|
┐	upper right box corner	043	#
┴	middle box bottom	046	&
─	horizontal box	044	$
╰	lower left box corner	052	*
┤	right middle box side	051	)
┼	plus intersect	045	%
┬	middle box top	136	^
-	required hyphen	055	-
/	backslash	054	,
[	left bracket	133	[
]	right bracket	135	]
{	left brace	105	E
}	right brace	124	T
‹	left angle bracket superscript	175	}
›	right angle bracket superscript	173	{
α	alpha small	141	a
β	beta small	142	b

γ	gamma small	161	q
δ	delta small	167	w
ϵ	epsilon small	145	e
ζ	zeta small	172	z
η	eta small	150	h
ϑ	script theta small	152	j
θ	theta small	162	r
ι	iota small	151	i
κ	kappa small	153	k
λ	lambda small	147	g
μ	mu small	155	m
ν	nu small	156	n
ϕ	phi small	144	d
ψ	psi small	143	c
o	omicron small	111	I
ξ	xi small	165	u
ρ	rho small	157	o
σ	sigma small	163	s
τ	tau small	164	t
υ	upsilon small	171	y
Φ	phi capital	104	D
χ	chi small	170	x
Ψ	psi capital	103	C
ω	omega small	154	l
Γ	gamma capital	121	Q
Δ	delta capital	127	W
Θ	theta capital	122	R
Λ	lambda capital	107	G
Π	pi capital	134	\
Σ	sigma capital	123	S
Υ	upsilon capital	131	Y
Ξ	xi capital	125	U
Ω	omega capital	114	L
^{0}	zero superscript	060	0
^{1}	one superscript	061	1
^{2}	two superscript	062	2
^{3}	three superscript	063	3
^{4}	four superscript	064	4
^{5}	five superscript	065	5
^{6}	six superscript	066	6
^{7}	seven superscript	067	7
^{8}	eight superscript	070	8
^{9}	nine superscript	071	9
¤	international currency	176	~
ƀ	substitute blank	116	N
ø	zero slash	056	.
□	open box	075	=
©	copyright	112	J
®	registered trade mark	113	K
ℓ	liter	117	O
R_x	prescription	106	F
™	trade mark	110	H
•	bullet	042	"
£	pound	146	f

APPENDIX B
SUMMARY OF COMPOSE CONTROLS

An italic character following the control word represents an argument. The *n* can be replaced by either an absolute positive number (otherwise known as an *unsigned* number). A # (pound sign or number symbol) stands for an argument that can only be a positive number.

..*pathname* or .ifi *pathname*
: Stops processing the current input file, goes to named file, and reads input from there. Both ".." and ".ifi" are equivalent, and can be used interchangeably.

.* *input line*
: Any input line following the ".*" control is treated as a comment line and does not appear in the printed output or affect it in any way. This input line must be separated from the ".*" by a space.

.alb
: Creates two perfectly smooth right and left margins. This is the default in fill mode, and works only when fill mode is on. Used to return to default alignment after an ".all" or similar control.

.alc
: Centers text. Turns fill off if you want a series of input lines centered independently.

.all
: Aligns each output line at the left margin, but the right margin is not aligned. This is the default when filling is turned off, so it has a noticeable effect only in fill mode.

.alr
: Aligns text at the right margin and leaves left ragged.

.bbf
: Indicates that all following lines of text will be part of a footnote.

.bbk #
: Marks a unit of text which cannot be split across pages. If you supply an argument, the next # lines will be kept as a unit. Otherwise, this control's action must be ended by a ".bek" control.

.bbp #
: Adds # lines of blank space as a picture block. If no argument is given, then all following input lines are processed as a picture block until a ".bep" control line is encountered. A picture block is broken across pages only if it is more than a page long. If a picture block does not fit on the current page, text lines following the picture block are added to fill up the current page and the picture is inserted on the following page.

.bbt #
: Signals the beginning of a title block: the following input can include title lines. # indicates how many input lines to accept. If no argument is added, this control's action must be terminated by a ".bet".

.bef
: Ends the block of footnote text begun with .bbf

.bek
: Indicates the end of a ".bbk" unit, and allows page breaks to operate again.

.bep
: Ends the picture mode begun by previous ".bbp". This control is ignored if there is no previous ".bbp".

.bet
: Signals the end of a title block. When the ".bbt" control is issued without an argument, this control is used to turn it off.

.bpf e,o Formats following input lines as a multiple-line page footer until an ".epf" is encountered.

.bph e,o Formats following input lines as a multiple-line page header until an ".eph" is encountered.

.brf Causes the preceding text line to be printed without filling in words from the next text line. This does not signal the end of a text block, but only causes a break in format.

.brn # Causes a *conditional* page break: if the argument given is greater than the number of lines still left on the page, then compose breaks the page at the current point in the text. Otherwise, the page is not broken.

.brp Causes an immediate page break. Compose finishes processing the current page, even if it isn't full, and starts printing at the top of the next page.

.brp o,e,*n* *mode_type*
Breaks to a new page, and changes the new page number according to the given option. An "o" or "e" sets it to the next odd or even number. If *n* is specified, then the number is either reset to *n* (if an unsigned variable is supplied) or is changed by *n* units (if a signed variable is supplied).

If *mode_type* is included on the command line, the style of numbering is changed. Modes are given as two-character combinations:

ar Arabic (the default)
al lowercase alphabetic
au uppercase alphabetic
rl lowercase roman
ru uppercase roman

.brs # *"text" "header" "footer"*
Breaks the current page and adds # blank ones. The default argument is 1. You can add a line of text to be centered on the page, or a header or footer line; they must be specified in the order shown. Always type in the quotation marks, whether they enclose anything or not. These pages are not numbered unless you use the header or footer line to add them by hand.

.csd *character*
Changes the % used to begin and end user- and built-in variables to another character.

.ctd *character*
Changes the | used to begin title lines to another character.

.epf Stops formatting input lines as a page footer.

.eph Stops formatting input lines as a page header.

.fif Turns off fill mode. Your text lines are printed out exactly as you entered them. If you have typed an input line that is very long, it is allowed to wander into the margin and even off the page.

.fin Turns fill mode on again after a ".fif" control. This is the default.

.fnt *name* Succeeding output will be printed in the font specified by *name*. If no argument is given, changes back to to the last font specified before the current one. A ".fnt reset" control line empties the font stack of all previous font changes.

.fth Holds all succeeding footnotes, and inserts them all at the end of the document. Footnotes are numbered continuously instead of being renumbered from 1 for each page.

.ftp Changes format of all succeeding footnotes to "paged": footnotes are inserted at the bottom of the page where they are referenced and are renumbered from 1 for each new page. Since this is the default format, this control is used primarily to turn off an ".ftu" control.

.ftu All following footnotes will be unreferenced: no footnote reference is inserted in the text; footnotes are inserted as in the "paged" format, but are unnumbered.

.htd *name* #,#,# . . .
Defines a series of tab stops # tenths of an inch across the page, and identifies this pattern by associating it with *name*.

.htn *c name*
Turns on a previously defined series of tab stops identified by *name*. Chooses the character *c* to mark off tab stops within the input lines for the table. Turns fill mode off.

.htf c^1,c^2. . .
Turns off any set of tab stops that uses the characters c^n as the tab stop marker. If no argument is given then all tabs are turned off.

.inb *n* Indents *n* tenths of an inch from both right and left margins. The default value is 0, and cancels any previous indentation for both left and right margins.

.inl *n* Indents *n* tenths of an inch from the current left margin. The default value is 0, so an ".inl" without an argument cancels any previous left indentation and restores the default margin setting.

.inr *n* Indents *n* tenths of an inch from the current right margin. The default value is 0, and turns off previous right-indents.

.ls *n* Changes amount of white space between text lines. You can specify quarter-line increments with decimal arguments (i.e., 0.25, 1.50 . . .). The default is 1.

.pdl *n* Changes number of lines on a page. The default is 66 10-pitch lines.

.pdw *n* Changes width of the text lines. The default is 65 10-pitch characters per line.

.pfl e,o title_line
Adds the given title line as a footer to the following pages: with an added "e", the title is inserted only on even pages; with an added "o" only on odd pages. With no added option, the title is placed on all pages.

.phl e,o title_line
Adds the given title line as a header to the following pages: The options function the same as for ".pfl".

.spb # Prints # blank lines and signals a block break. Should be used whenever you want to add white space between two text units. The default argument is 1.

.spd *n* With an absolute number, this control adds blank lines until you reach the line *n* lines down from the top of the page. A control line with a *+n* argument adds *n* blank lines after the current line. This amount of white space is always added, even if it has to be split across a page break. Used to at white space at top of pages.

.spf # Prints # blank lines. The default argument is 1. Does not signal a block break. Use only when you want to keep compose from splitting the surrounding text across pages.

.srv *name expression*
Defines *name* as a user variable whose value is set equal to the value of the *expression* specified.

.tab *name column definition:column definition. . .*
Defines a table of up to 20 columns and names it. Each column definition is divided from the next by a colon.

Each column definition must include: its column starting position, then a comma followed by a measure of the column width. These characters may be immediately followed by a two-character sequence specifying filling and alignment. By default the columns are filled. To turn filling off, you must specify alignment with one of the following character sequences:

nc unfilled and aligned in the center of the column
nl unfilled and aligned at left column margin
nr unfilled and aligned at right column margin

.tac # Makes all following input lines into table column number #. To begin a new column, issue the same control line with the next column number.

.taf Turns off table mode.

.tan *name* Turns on table mode according to the parameters defined for *name* in a previous ".tab" line.

.tcl *# n title_line*
Inserts a title line and associates it with the previous text. The first variable indicates the number of lines to be inserted between the text and the caption; the default is 0. The next number is the amount you want the line to be indented; the default is 0. If you choose to indent the title line using an unsigned number, you must also specify the first variable to keep the formatter from confusing the two values.

.tlh *# n title_line*
Inserts a title line, and associates it with the following text. This control is the same as ".tcl", except that the # argument indicates the number of spaces inserted *after* the title line.

.trn *abababab. . .*
For every *a* that occurs in the input file, produces *b* in the output. You can add as many *ab* combinations as you like to the control line. Issuing a new control line with *aa* cancels the translation process for that character.

.unh *n* Changes the left indentation of the next input line by *n* units. The margin setting is the default. In fill mode, the text from the input line immediately following is aligned according to the current indentation and placed on the same output line as

the undented text if there is room. If the undented text is too long, and would cause overlapping, then the new input line is moved down to a new output line. If fill mode is off, then these two input lines are always placed on the same output line, even if it causes overprinting.

.unl *n* Changes left indentation for the next input line only according to the value *n*. Default value of *n* is the left margin.

.unr *n* Changes right indentation according to value of *n* for next input line only. Default value of *n* is the right margin.

.ur *%name%*

Scans input line for delimiters indicating a variable with a previously defined *name*. The default delimiter is the percent sign (%). This controls evaluates the variable, and inserts the resulting value in the text. It applies to one input line. *Unlike other controls, it is separated from the text it applies to by a space, not a carriage return.*

.vm *t,h,f,b*

Resets vertical margins. The variables can be signed or unsigned numbers. They must be separated by commas exactly as shown. If you are not changing all of the margins, you don't have to include a value for each variable, but you must still type in the surrounding commas. The variables are:

- *t* page top margin; the default value is 4 lines from top edge of paper.
- *h* page header margin; the default value is 2 lines between header and text.
- *f* page footer margin; the default value is 2 lines between text and footer.
- *b* page bottom margin; the default value is 4 lines to bottom edge of paper.

.vmb *n* Resets page bottom margin to *n* or changes it by *n*; default setting is 4 lines between footer and bottom edge of page.

.vmf *n* Resets page footer margin to *n* or changes it by *n*; default setting is 2 lines between text and footer.

.vmh *n* Resets page header margin to *n* or changes it by *n*; default setting is 2 lines between header and text.

.vmt *n* Resets page top margin to *n* or changes it by *n*; default setting is 4 lines from top edge of page.

.wit *n* Changes the widow size. The default is 2.

APPENDIX C
FONT SAMPLES

This appendix shows the fonts available for each of the Xerox 9700 device types. Each row in a font table comprises eight characters; the octal codes corresponding to the first and last character in the row are given along the sides.

Devices x9700_pr and x9700_pr_land:

Font: ascii, m

	040	041	042	043	044	045	046	047	
040		!	"	#	$	%	&	'	047
050	(	)	*	+	,	-	.	/	057
060	0	1	2	3	4	5	6	7	067
070	8	9	:	;	<	=	>	?	077
100	@	A	B	C	D	E	F	G	107
110	H	I	J	K	L	M	N	O	117
120	P	Q	R	S	T	U	V	W	127
130	X	Y	Z	[	\	]	^	_	137
140		a	b	c	d	e	f	g	147
150	h	i	j	k	l	m	n	o	157
160	p	q	r	s	t	u	v	w	167
170	x	y	z	{	|	}	~		177
	170	171	172	173	174	175	176	177	

Font: super, sp

	040	041	042	043	044	045	046	047	
040		!	"	#	$	%	&	'	047
050	(	)	*	+	,	-	.	/	057
060	0	1	2	3	4	5	6	7	067
070	8	9	:	;	<	=	>	?	077
100	@	A	B	C	D	E	F	G	107
110	H	I	J	K	L	M	N	O	117
120	P	Q	R	S	T	U	V	W	127
130	X	Y	Z	[	\	]	^	_	137
140		a	b	c	d	e	f	g	147
150	h	i	j	k	l	m	n	o	157
160	p	q	r	s	t	u	v	w	167
170	x	y	z	{	|	}	~		177
	170	171	172	173	174	175	176	177	

Font: sub, sb

	040	041	042	043	044	045	046	047	
040		!	"	#	$	%	&	'	047
050	(	)	*	+	,	-	.	/	057
060	0	1	2	3	4	5	6	7	067
070	8	9	:	;	<	=	>	?	077
100	@	A	B	C	D	E	F	G	107
110	H	I	J	K	L	M	N	O	117
120	P	Q	R	S	T	U	V	W	127
130	X	Y	Z	[	\	]	^	_	137
140		a	b	c	d	e	f	g	147
150	h	i	j	k	l	m	n	o	157
160	p	q	r	s	t	u	v	w	167
170	x	y	z	{	|	}	~		177
	170	171	172	173	174	175	176	177	

Font: ASCII, M

	040	041	042	043	044	045	046	047	
040		!	"	#	$	%	&	'	047
050	(	)	*	+	,	-	.	/	057
060	0	1	2	3	4	5	6	7	067
070	8	9	:	;	<	=	>	?	077
100	@	A	B	C	D	E	F	G	107
110	H	I	J	K	L	M	N	O	117
120	P	Q	R	S	T	U	V	W	127
130	X	Y	Z	[	\	]	^	_	137
140		A	B	C	D	E	F	G	147
150	H	I	J	K	L	M	N	O	157
160	P	Q	R	S	T	U	V	W	167
170	X	Y	Z	{	|	}	~		177
	170	171	172	173	174	175	176	177	

Font: bold, b

	040	041	042	043	044	045	046	047	
040		!	"	#	$	%	&	'	047
050	(	)	*	+	,	-	.	/	057
060	0	1	2	3	4	5	6	7	067
070	8	9	:	;	<	=	>	?	077
100	@	A	B	C	D	E	F	G	107
110	H	I	J	K	L	M	N	O	117
120	P	Q	R	S	T	U	V	W	127
130	X	Y	Z	[	\	]	^	_	137
140		a	b	c	d	e	f	g	147
150	h	i	j	k	l	m	n	o	157
160	p	q	r	s	t	u	v	w	167
170	x	y	z	{	|	}	~		177
	170	171	172	173	174	175	176	177	

Font: BOLD, B

	040	041	042	043	044	045	046	047	
040		!	"	#	$	%	&	'	047
050	(	)	*	+	,	-	.	/	057
060	0	1	2	3	4	5	6	7	067
070	8	9	:	;		=		?	.077
100	@	A	B	C	D	E	F	G	107
110	H	I	J	K	L	M	N	O	117
120	P	Q	R	S	T	U	V	W	127
130	X	Y	Z					_	137
140		A	B	C	D	E	F	G	147
150	H	I	J	K	L	M	N	O	157
160	P	Q	R	S	T	U	V	W	167
170	X	Y	Z						177
	170	171	172	173	174	175	176	177	

Font: italic, i

	040	041	042	043	044	045	046	047	
040		*!*	*"*	*#*	*$*	*%*	*&*	*'*	047
050	*(*	*)*	*	+	*,*	-	.	*/*	057
060	*0*	*1*	*2*	*3*	*4*	*5*	*6*	*7*	067
070	*8*	*9*	*:*	*;*	<	=	>	*?*	077
100	@	*A*	*B*	*C*	*D*	*E*	*F*	*G*	107
110	*H*	*I*	*J*	*K*	*L*	*M*	*N*	*O*	117
120	*P*	*Q*	*R*	*S*	*T*	*U*	*V*	*W*	127
130	*X*	*Y*	*Z*	[	\	]	^	_	137
140		*a*	*b*	*c*	*d*	*e*	*f*	*g*	147
150	*h*	*i*	*j*	*k*	*l*	*m*	*n*	*o*	157
160	*p*	*q*	*r*	*s*	*t*	*u*	*v*	*w*	167
170	*x*	*y*	*z*	{	|	}	~		177
	170	171	172	173	174	175	176	177	

Font: small, sm

	040	041	042	043	044	045	046	047	
040		!	"	#	$	%	&	'	047
050	(	)	*	+	,	-	.	/	057
060	0	1	2	3	4	5	6	7	067
070	8	9	:	;	<	=	>	?	077
100	@	A	B	C	D	E	F	G	107
110	H	I	J	K	L	M	N	O	117
120	P	Q	R	S	T	U	V	W	127
130	X	Y	Z	[	\	]	^	_	137
140		a	b	c	d	e	f	g	147
150	h	i	j	k	l	m	n	o	157
160	p	q	r	s	t	u	v	w	167
170	x	y	z	{	|	}	~		177
	170	171	172	173	174	175	176	177	

Font: head14, L

	040	041	042	043	044	045	046	047	
040		!	"				&	'	047
050	(	)			,	-	.	/	057
060	0	1	2	3	4	5	6	7	067
070	8	9						?	077
100		A	B	C	D	E	F	G	107
110	H	I	J	K	L	M	N	O	117
120	P	Q	R	S	T	U	V	W	127
130	X	Y	Z						137
140		A	B	C	D	E	F	G	147
150	H	I	J	K	L	M	N	O	157
160	P	Q	R	S	T	U	V	W	167
170	X	Y	Z						177
	170	171	172	173	174	175	176	177	

Font: head18, VL

	040	041	042	043	044	045	046	047	
040									047
050					,	-	.	/	057
060	0	1	2	3	4	5	6	7	067
070	8	9							077
100		A	B	C	D	E	F	G	107
110	H	I	J	K	L	M	N	O	117
120	P	Q	R	S	T	U	V	W	127
130	X	Y	Z						137
140									147
150									157
160									167
170									177
	170	171	172	173	174	175	176	177	

Font: typ, t

	040	041	042	043	044	045	046	047	
040		!	"	#	$	%	&	'	047
050	(	)	*	+	,	-	.	/	057
060	0	1	2	3	4	5	6	7	067
070	8	9	:	;	<	=	>	?	077
100	@	A	B	C	D	E	F	G	107
110	H	I	J	K	L	M	N	O	117
120	P	Q	R	S	T	U	V	W	127
130	X	Y	Z	[	\	]	^	_	137
140		a	b	c	d	e	f	g	147
150	h	i	j	k	l	m	n	o	157
160	p	q	r	s	t	u	v	w	167
170	x	y	z	{	|	}	~		177
	170	171	172	173	174	175	176	177	

Font: small_typ, tsm

	040	041	042	043	044	045	046	047	
040		!	"	#	$	%	&	'	047
050	(	)	*	+	,	-	.	/	057
060	0	1	2	3	4	5	6	7	067
070	8	9	:	;	<	=	>	?	077
100	@	A	B	C	D	E	F	G	107
110	H	I	J	K	L	M	N	O	117
120	P	Q	R	S	T	U	V	W	127
130	X	Y	Z	[	\	]	^	_	137
140		a	b	c	d	e	f	g	147
150	h	i	j	k	l	m	n	o	157
160	p	q	r	s	t	u	v	w	167
170	x	y	z	{	|	}	~		177
	170	171	172	173	174	175	176	177	

Font: typ_super, tsp

	040	041	042	043	044	045	046	047	
040		!	"	#	$	%	&	'	047
050	(	)	*	+	,	-	.	/	057
060	0	1	2	3	4	5	6	7	067
070	8	9	:	;	<	=	>	?	077
100	@	A	B	C	D	E	F	G	107
110	H	I	J	K	L	M	N	O	117
120	P	Q	R	S	T	U	V	W	127
130	X	Y	Z	[	\	]	^	_	137
140		a	b	c	d	e	f	g	147
150	h	i	j	k	l	m	n	o	157
160	p	q	r	s	t	u	v	w	167
170	x	y	z	{	|	}	~		177
	170	171	172	173	174	175	176	177	

Font: typ_sub, tsb

	040	041	042	043	044	045	046	047	
040		!	"	#	$	%	&	'	047
050	(	)	*	+	,	-	.	/	057
060	0	1	2	3	4	5	6	7	067
070	8	9	:	;	<	=	>	?	077
100	@	A	B	C	D	E	F	G	107
110	H	I	J	K	L	M	N	O	117
120	P	Q	R	S	T	U	V	W	127
130	X	Y	Z	[	\	]	^	_	137
140		a	b	c	d	e	f	g	147
150	h	i	j	k	l	m	n	o	157
160	p	q	r	s	t	u	v	w	167
170	x	y	z	{	|	}	~		177
	170	171	172	173	174	175	176	177	

Font: footref

	040	041	042	043	044	045	046	047	
040									047
050	(	)		+	,	-			057
060	0	1	2	3	4	5	6	7	067
070	8	9							077
100									107
110									117
120									127
130									137
140									147
150									157
160									167
170									177
	170	171	172	173	174	175	176	177	

Font: sym, s

	040	041	042	043	044	045	046	047	
040		↓	•	⌉	−	∔	⊥	÷	047
050	⊢	⊣	⌊	⌋	/	⁻	⌀	~	057
060	0	1	2	3	4	5	6	7	067
070	8	9	‖	≠	≤	□	≥	∫	077
100	⌜	▽	∞	Ψ	Φ	{	℞	Λ	107
110	™	o	©	®	Ω	∂	ƀ	ℓ	117
120	→	Γ	Θ	Σ	}	Ξ	∝	Δ	127
130	≡	Υ	≃	[	Π	]	⊤	⁼	137
140		α	β	ψ	φ	ϵ	£	λ	147
150	η	ι	ϑ	κ	ω	μ	ν	ρ	157
160	←	ϒ	θ	σ	τ	ξ	×	δ	167
170	χ	υ	ζ	⟩	|	⟨	¤		177
	170	171	172	173	174	175	176	177	

Font: sym_super, ssp

	040	041	042	043	044	045	046	047	
040		↓	•	]	−	+	⊥	÷	047
050	⊢	⊣	∟		/	−	∅	~	057
060	0	1	2	3	4	5	6	7	067
070	8	9	‖	≠	≤	□	≥	∫	077
100		▽	∞	Ψ	Φ	{	R_x	Λ	107
110	™	*o*	©	®	Ω	∂	ƀ	*ℓ*	117
120	→	Γ	Θ	Σ	}	Ξ	∝	Δ	127
130	≡	Υ	≃	[	Π	]	⊤	=	137
140		*α*	*β*	*ψ*	*φ*	*ϵ*	£	λ	147
150	*η*	*ι*	*ϑ*	*κ*	*ω*	*μ*	*ν*	*ρ*	157
160	←	*ϒ*	*θ*	*σ*	*τ*	*ξ*	×	*δ*	167
170	*χ*	*υ*	*ζ*	>	|	<	¤		177
	170	171	172	173	174	175	176	177	

Font: sym_sub, ssb

	040	041	042	043	044	045	046	047	
040		↓	•	]	−	+	⊥	÷	047
050	⊢	⊣	∟		/	−	∅	~	057
060	0	1	2	3	4	5	6	7	067
070	8	9	‖	≠	≤	□	≥	∫	077
100		▽	∞	Ψ	Φ	{	R_x	Λ	107
110	™	*o*	©	®	Ω	∂	ƀ	*ℓ*	117
120	→	Γ	Θ	Σ	}	Ξ	∝	Δ	127
130	≡	Υ	≃	[	Π	]	⊤	=	137
140		*α*	*β*	*ψ*	*φ*	*ϵ*	£	λ	147
150	*η*	*ι*	*ϑ*	*κ*	*ω*	*μ*	*ν*	*ρ*	157
160	←	*ϒ*	*θ*	*σ*	*τ*	*ξ*	×	*δ*	167
170	*χ*	*υ*	*ζ*	>	|	<	¤		177
	170	171	172	173	174	175	176	177	

Font: umlaut, um

	040	041	042	043	044	045	046	047	
040									047
050									057
060									067
070									077
100		Ä				Ë			107
110		Ï						Ö	117
120						Ü			127
130									137
140		ä				ë			147
150		ï						ö	157
160						ü			167
170									177
	170	171	172	173	174	175	176	177	

Font: umlaut_italic, umi

	040	041	042	043	044	045	046	047	
040									047
050									057
060									067
070									077
100		*Ä*				*Ë*			107
110		*Ï*						*Ö*	117
120						*Ü*			127
130									137
140		*ä*				*ë*			147
150		*ï*						*ö*	157
160						*ü*			167
170									177
	170	171	172	173	174	175	176	17	

Devices x9700_un and x9700_un_land:

Font: ascii, m

	040	041	042	043	044	045	046	047	
040		!	"	#	$	%	&	'	047
050	(	)	*	+	,	-	.	/	057
060	0	1	2	3	4	5	6	7	067
070	8	9	:	;	<	=	>	?	077
100	@	A	B	C	D	E	F	G	107
110	H	I	J	K	L	M	N	O	117
120	P	Q	R	S	T	U	V	W	127
130	X	Y	Z	[	\	]	^	_	137
140		a	b	c	d	e	f	g	147
150	h	i	j	k	l	m	n	o	157
160	p	q	r	s	t	u	v	w	167
170	x	y	z	{	|	}	~		177
	170	171	172	173	174	175	176	177	

Font: super, sp

	040	041	042	043	044	045	046	047	
040		!	"	#	$	%	&	'	047
050	(	)	*	+	,	-	.	/	057
060	0	1	2	3	4	5	6	7	067
070	8	9	:	;	<	=	>	?	077
100	@	A	B	C	D	E	F	G	107
110	H	I	J	K	L	M	N	O	117
120	P	Q	R	S	T	U	V	W	127
130	X	Y	Z	[	\	]	^	_	137
140		a	b	c	d	e	f	g	147
150	h	i	j	k	l	m	n	o	157
160	p	q	r	s	t	u	v	w	167
170	x	y	z	{	|	}	~		177
	170	171	172	173	174	175	176	177	

Font: sub, sb

	040	041	042	043	044	045	046	047	
040		!	"	#	$	%	&	'	047
050	(	)	*	+	,	-	.	/	057
060	0	1	2	3	4	5	6	7	067
070	8	9	:	;	<	=	>	?	077
100	@	A	B	C	D	E	F	G	107
110	H	I	J	K	L	M	N	O	117
120	P	Q	R	S	T	U	V	W	127
130	X	Y	Z	[	\	]	^	_	137
140		a	b	c	d	e	f	g	147
150	h	i	j	k	l	m	n	o	157
160	p	q	r	s	t	u	v	w	167
170	x	y	z	{	|	}	~		177
	170	171	172	173	174	175	176	177	

Font: ASCII, M

	040	041	042	043	044	045	046	047	
040		!	"	#	$	%	&	'	047
050	(	)	*	+	,	-	.	/	057
060	0	1	2	3	4	5	6	7	067
070	8	9	:	;	<	=	>	?	077
100	@	A	B	C	D	E	F	G	107
110	H	I	J	K	L	M	N	O	117
120	P	Q	R	S	T	U	V	W	127
130	X	Y	Z	[	\	]	^	_	137
140		A	B	C	D	E	F	G	147
150	H	I	J	K	L	M	N	O	157
160	P	Q	R	S	T	U	V	W	167
170	X	Y	Z	{	|	}	~		177
	170	171	172	173	174	175	176	177	

Font: bold, b

	040	041	042	043	044	045	046	047	
040		!	"	#	$	%	&	'	047
050	(	)	*	+	,	-	.	/	057
060	0	1	2	3	4	5	6	7	067
070	8	9	:	;	<	=	>	?	077
100	@	A	B	C	D	E	F	G	107
110	H	I	J	K	L	M	N	O	117
120	P	Q	R	S	T	U	V	W	127
130	X	Y	Z	[	\	]	^	_	137
140		a	b	c	d	e	f	g	147
150	h	i	j	k	l	m	n	o	157
160	p	q	r	s	t	u	v	w	167
170	x	y	z	{	|	}	~		177
	170	171	172	173	174	175	176	177	

Font: BOLD, B

	040	041	042	043	044	045	046	047	
040		!	"	#	$	%	&	'	047
050	(	)	*	+	,	-	.	/	057
060	0	1	2	3	4	5	6	7	067
070	8	9	:	;		=		?	077
100	@	A	B	C	D	E	F	G	107
110	H	I	J	K	L	M	N	O	117
120	P	Q	R	S	T	U	V	W	127
130	X	Y	Z						137
140		A	B	C	D	E	F	G	147
150	H	I	J	K	L	M	N	O	157
160	P	Q	R	S	T	U	V	W	167
170	X	Y	Z						177
	170	171	172	173	174	175	176	177	

Font: italic, i

	040	041	042	043	044	045	046	047	
040		*!*	*"*	*#*	*$*	*%*	*&*	*'*	047
050	*(*	*)*	*	*+*	*,*	*-*	*.*	*/*	057
060	*0*	*1*	*2*	*3*	*4*	*5*	*6*	*7*	067
070	*8*	*9*	*:*	*;*	*<*	*=*	*>*	*?*	077
100	@	*A*	*B*	*C*	*D*	*E*	*F*	*G*	107
110	*H*	*I*	*J*	*K*	*L*	*M*	*N*	*O*	117
120	*P*	*Q*	*R*	*S*	*T*	*U*	*V*	*W*	127
130	*X*	*Y*	*Z*	[	\	]	^	_	137
140		*a*	*b*	*c*	*d*	*e*	*f*	*g*	147
150	*h*	*i*	*j*	*k*	*l*	*m*	*n*	*o*	157
160	*p*	*q*	*r*	*s*	*t*	*u*	*v*	*w*	167
170	*x*	*y*	*z*	{	|	}	~		177
	170	171	172	173	174	175	176	177	

Font: small, sm

	040	041	042	043	044	045	046	047	
040		!	"	#	$	%	&	'	047
050	(	)	*	+	,	-	.	/	057
060	0	1	2	3	4	5	6	7	067
070	8	9	:	;	<	=	>	?	077
100	@	A	B	C	D	E	F	G	107
110	H	I	J	K	L	M	N	O	117
120	P	Q	R	S	T	U	V	W	127
130	X	Y	Z	[	\	]	^	_	137
140		a	b	c	d	e	f	g	147
150	h	i	j	k	l	m	n	o	157
160	p	q	r	s	t	u	v	w	167
170	x	y	z	{	|	}	~		177
	170	171	172	173	174	175	176	177	

Font: head14, L

	040	041	042	043	044	045	046	047	
040		!	"				&	'	047
050	(	)			,	-	.	/	057
060	0	1	2	3	4	5	6	7	067
070	8	9						?	077
100		A	B	C	D	E	F	G	107
110	H	I	J	K	L	M	N	O	117
120	P	Q	R	S	T	U	V	W	127
130	X	Y	Z						137
140		A	B	C	D	E	F	G	147
150	H	I	J	K	L	M	N	O	157
160	P	Q	R	S	T	U	V	W	167
170	X	Y	Z						177
	170	171	172	173	174	175	176	177	

Font: head18, VL

	040	041	042	043	044	045	046	047	
040									047
050					,	-	.	/	057
060	0	1	2	3	4	5	6	7	067
070	8	9							077
100		A	B	C	D	E	F	G	107
110	H	I	J	K	L	M	N	O	117
120	P	Q	R	S	T	U	V	W	127
130	X	Y	Z						137
140									147
150									157
160									167
170									177
	170	171	172	173	174	175	176	177	

Font: typ, t

	040	041	042	043	044	045	046	047	
040		!	"	#	$	%	&	'	047
050	(	)	*	+	,	-	.	/	057
060	0	1	2	3	4	5	6	7	067
070	8	9	:	;	<	=	>	?	077
100	@	A	B	C	D	E	F	G	107
110	H	I	J	K	L	M	N	O	117
120	P	Q	R	S	T	U	V	W	127
130	X	Y	Z	[	\	]	^	_	137
140		a	b	c	d	e	f	g	147
150	h	i	j	k	l	m	n	o	157
160	p	q	r	s	t	u	v	w	167
170	x	y	z	{	|	}	~		177
	170	171	172	173	174	175	176	177	

Font: small_typ, tsm

	040	041	042	043	044	045	046	047	
040		!	"	#	$	%	&	'	047
050	(	)	*	+	,	-	.	/	057
060	0	1	2	3	4	5	6	7	067
070	8	9	:	;	<	=	>	?	077
100	@	A	B	C	D	E	F	G	107
110	H	I	J	K	L	M	N	O	117
120	P	Q	R	S	T	U	V	W	127
130	X	Y	Z	[	\	]	^	_	137
140		a	b	c	d	e	f	g	147
150	h	i	j	k	l	m	n	o	157
160	p	q	r	s	t	u	v	w	167
170	x	y	z	{	|	}	~		177
	170	171	172	173	174	175	176	177	

Font: un311, cd

	040	041	042	043	044	045	046	047	
040		!	"	#	$	%	&	'	047
050	(	)	*	+	,	−	.	/	057
060	0	1	2	3	4	5	6	7	067
070	8	9	:	;	<	=	>	?	077
100	@	A	B	C	D	E	F	G	107
110	H	I	J	K	L	M	N	O	117
120	P	Q	R	S	T	U	V	W	127
130	X	Y	Z	[	\	]	^	_	137
140		a	b	c	d	e	f	g	147
150	h	i	j	k	l	m	n	o	157
160	p	q	r	s	t	u	v	w	167
170	x	y	z	{	|	}	~		177
	170	171	172	173	174	175	176	177	

Font: un411, cdb

	040	041	042	043	044	045	046	047	
040		!	"	#	$	%	&	'	047
050	(	)	*	+	,	−	.	/	057
060	0	1	2	3	4	5	6	7	067
070	8	9	:	;	<	=	>	?	077
100	@	A	B	C	D	E	F	G	107
110	H	I	J	K	L	M	N	O	117
120	P	Q	R	S	T	U	V	W	127
130	X	Y	Z	[	\	]	^	_	137
140		a	b	c	d	e	f	g	147
150	h	i	j	k	l	m	n	o	157
160	p	q	r	s	t	u	v	w	167
170	x	y	z	{	|	}	~		177
	170	171	172	173	174	175	176	177	

Font: un611, cdi

	040	041	042	043	044	045	046	047	
040		!	"	#	$	%	&	'	047
050	(	)	*	+	,	-	.	/	057
060	0	1	2	3	4	5	6	7	067
070	8	9	:	;	<	=	>	?	077
100	@	A	B	C	D	E	F	G	107
110	H	I	J	K	L	M	N	O	117
120	P	Q	R	S	T	U	V	W	127
130	X	Y	Z	[	\	]	^	_	137
140		a	b	c	d	e	f	g	147
150	h	i	j	k	l	m	n	o	157
160	p	q	r	s	t	u	v	w	167
170	x	y	z	{	|	}	~		177
	170	171	172	173	174	175	176	177	

Font: footref

	040	041	042	043	044	045	046	047	
040									047
050	(	)		+	,	-			057
060	0	1	2	3	4	5	6	7	067
070	8	9							077
100									107
110									117
120									127
130									137
140									147
150									157
160									167
170									177
	170	171	172	173	174	175	176	177	

Font: sym, s

	040	041	042	043	044	045	046	047	
040		↓	•	]	−	∔	⊥	÷	047
050	├	┤	└	]	/	−	∅	~	057
060	0	1	2	3	4	5	6	7	067
070	8	9	‖	≠	≤	□	≥	∫	077
100	┌	▽	∞	Ψ	Φ	{	R_x	Λ	107
110	™	*o*	©	®	Ω	∂	ƀ	*ℓ*	117
120	→	Γ	Θ	Σ	}	Ξ	∝	Δ	127
130	≡	Υ	≃	[	Π	]	⊤	=	137
140		*a*	*β*	*ψ*	*ϕ*	∈	£	λ	147
150	*η*	*ι*	*ϑ*	*κ*	*ω*	*μ*	*ν*	*ρ*	157
160	←	*γ*	*θ*	*σ*	*τ*	*ξ*	×	*δ*	167
170	*χ*	*υ*	*ζ*	>	|	<	¤		177
	170	171	172	173	174	175	176	177	

Font: sym_super, ssp

	040	041	042	043	044	045	046	047	
040		↓	•	]	−	∔	⊥	÷	047
050	├	┤	└	]	/	−	∅	~	057
060	0	1	2	3	4	5	6	7	067
070	8	9	‖	≠	≤	□	≥	∫	077
100	┌	▽	∞	Ψ	Φ	{	R_x	Λ	107
110	™	*o*	©	®	Ω	∂	ƀ	*ℓ*	117
120	→	Γ	Θ	Σ	}	Ξ	∝	Δ	127
130	≡	Υ	≃	[	Π	]	⊤	=	137
140		*a*	*β*	*ψ*	*ϕ*	∈	£	λ	147
150	*η*	*ι*	*ϑ*	*κ*	*ω*	*μ*	*ν*	*ρ*	157
160	←	*γ*	*θ*	*σ*	*τ*	*ξ*	×	*δ*	167
170	*χ*	*υ*	*ζ*	>	|	<	¤		177
	170	171	172	173	174	175	176	177	

Font: sym_sub, ssb

	040	041	042	043	044	045	046	047	
040		↓	•	]	−	∔	⊥	÷	047
050	⊦	⊣	L	]	/	⁻	⌀	~	057
060	0	1	2	3	4	5	6	7	067
070	8	9	‖	≠	≤	□	≥	∫	077
100		▽	∞	Ψ	Φ	{	℞	Λ	107
110	™	*o*	©	®	Ω	∂	ƀ	*ℓ*	117
120	→	Γ	Θ	Σ	}	Ξ	∝	Δ	127
130	≡	Υ	≃	[	Π	]	⊤	=	137
140		*α*	*β*	*ψ*	*ϕ*	∈	£	λ	147
150	*η*	*ι*	*ϑ*	*κ*	*ω*	*μ*	*ν*	*ρ*	157
160	←	*χ*	*θ*	*σ*	*τ*	*ξ*	×	*δ*	167
170	*χ*	*υ*	*ζ*	›	|	‹	¤		177
	170	171	172	173	174	175	176	177	

Device x9700_pl0x6:

Font: ascii, m

	040	041	042	043	044	045	046	047	
040		!	"	#	$	%	&	'	047
050	(	)	*	+	,	-	.	/	057
060	0	1	2	3	4	5	6	7	067
070	8	9	:	;	<	=	>	?	077
100	@	A	B	C	D	E	F	G	107
110	H	I	J	K	L	M	N	O	117
120	P	Q	R	S	T	U	V	W	127
130	X	Y	Z	[	\	]	^	_	137
140		a	b	c	d	e	f	g	147
150	h	i	j	k	l	m	n	o	157
160	p	q	r	s	t	u	v	w	167
170	x	y	z	{	|	}	~		177
	170	171	172	173	174	175	176	177	

Font: super, sp

	040	041	042	043	044	045	046	047	
040		!	"	#	$	%	&	'	047
050	(	)	*	+	,	-	.	/	057
060	0	1	2	3	4	5	6	7	067
070	8	9	:	;	<	=	>	?	077
100	@	A	B	C	D	E	F	G	107
110	H	I	J	K	L	M	N	O	117
120	P	Q	R	S	T	U	V	W	127
130	X	Y	Z	[	\	]	^	_	137
140		a	b	c	d	e	f	g	147
150	h	i	j	k	l	m	n	o	157
160	p	q	r	s	t	u	v	w	167
170	x	y	z	{	|	}	~		177
	170	171	172	173	174	175	176	177	

Font: sub, sb

	040	041	042	043	044	045	046	047	
040		!	"	#	$	%	&	'	047
050	(	)	*	+	,	-	.	/	057
060	0	1	2	3	4	5	6	7	067
070	8	9	:	;	<	=	>	?	077
100	@	A	B	C	D	E	F	G	107
110	H	I	J	K	L	M	N	O	117
120	P	Q	R	S	T	U	V	W	127
130	X	Y	Z	[	\	]	^	_	137
140		a	b	c	d	e	f	g	147
150	h	i	j	k	l	m	n	o	157
160	p	q	r	s	t	u	v	w	167
170	x	y	z	{	|	}	~		177
	170	171	172	173	174	175	176	177	

Font: bold, b

	040	041	042	043	044	045	046	047	
040		!	"	#	$	%	&	'	047
050	(	)	*	+	,	-	.	/	057
060	0	1	2	3	4	5	6	7	067
070	8	9	:	;	<	=	>	?	077
100	@	A	B	C	D	E	F	G	107
110	H	I	J	K	L	M	N	O	117
120	P	Q	R	S	T	U	V	W	127
130	X	Y	Z	[	\	]	↑	_	137
140		a	b	c	d	e	f	g	147
150	h	i	j	k	l	m	n	o	157
160	p	q	r	s	t	u	v	w	167
170	x	y	z	{	|	}	¬		177
	170	171	172	173	174	175	176	177	

Font: italic, i

	040	041	042	043	044	045	046	047	
040		!	"	#	$	%	&	'	047
050	(	)	*	+	,	-	.	/	057
060	0	1	2	3	4	5	6	7	067
070	8	9	:	;		=		?	077
100	@	A	B	C	D	E	F	G	107
110	H	I	J	K	L	M	N	O	117
120	P	Q	R	S	T	U	V	W	127
130	X	Y	Z						137
140		a	b	c	d	e	f	g	147
150	h	i	j	k	l	m	n	o	157
160	p	q	r	s	t	u	v	w	167
170	x	y	z						177
	170	171	172	173	174	175	176	177	

Font: script, scr

	040	041	042	043	044	045	046	047	
040		!	"	#	$	%	&	'	047
050	(	)	*	+	,	-	.	/	057
060	0	1	2	3	4	5	6	7	067
070	8	9	:	;		=		?	077
100	@	A	B	C	D	E	F	G	107
110	H	I	J	K	L	M	N	O	117
120	P	Q	R	S	T	U	V	W	127
130	X	Y	Z						137
140		a	b	c	d	e	f	g	147
150	h	i	j	k	l	m	n	o	157
160	p	q	r	s	t	u	v	w	167
170	x	y	z						177
	170	171	172	173	174	175	176	177	

Font: grk, g

	040	041	042	043	044	045	046	047	
040		!	"	#	$	%	&	'	047
050	(	)	*	+	,	-	.	/	057
060	0	1	2	3	4	5	6	7	067
070	8	9	:	;	<	=	>	?	077
100	@	A	B	X	Δ	E	Φ	Γ	107
110	H	I	J	K	Λ	M	N	O	117
120	Π	Θ	P	Σ	T	Υ	V	Ω	127
130	Ξ	Ψ	Z	[	\	]	^	_	137
140		α	β	χ	δ	ϵ	ϕ	γ	147
150	η	ι	$\jmath$	κ	λ	μ	ν	o	157
160	π	θ	ρ	σ	τ	υ	v	ω	167
170	ξ	ψ	ζ	{	|	}	~		177
	170	171	172	173	174	175	176	177	

Font: oaa

	040	041	042	043	044	045	046	047	
040					≒		&		047
050			*		⌝	-	.	/	057
060	0	1	2	3	4	5	6	7	067
070	8	9			.				077
100		A	B	C	D	E	F	G	107
110	H	I	J	K	L	M	N	O	117
120	P	Q	R	S	T	U	V	W	127
130	X	Y	Z					_	137
140									147
150									157
160									167
170									177
	170	171	172	173	174	175	176	177	

Font: oba

	040	041	042	043	044	045	046	047	
040					$		&	'	047
050			*	+	,	-	.	/	057
060	0	1	2	3	4	5	6	7	067
070	8	9				=			077
100		A	B	C	D	E	F	G	107
110	H	I	J	K	L	M	N	O	117
120	P	Q	R	S	T	U	V	W	127
130	X	Y	Z					_	137
140									147
150									157
160									167
170									177
	170	171	172	173	174	175	176	177	

Font: tya

	040	041	042	043	044	045	046	047	
040		!	"	#	$	%	&	'	047
050	(	)	*	+	,	-	.	/	057
060	0	1	2	3	4	5	6	7	067
070	8	9	:	;		=		?	077
100	@	A	B	C	D	E	F	G	107
110	H	I	J	K	L	M	N	O	117
120	P	Q	R	S	T	U	V	W	127
130	X	Y	Z					_	137
140		a	b	c	d	e	f	g	147
150	h	i	j	k	l	m	n	o	157
160	p	q	r	s	t	u	v	w	167
170	x	y	z						177
	170	171	172	173	174	175	176	177	

Font: x1200, x

	040	041	042	043	044	045	046	047	
040		!	"	#	$	%	&	'	047
050	(	)	*	+	,	-	.	/	057
060	0	1	2	3	4	5	6	7	067
070	8	9	:	;	<	=	>	?	077
100	@	A	B	C	D	E	F	G	107
110	H	I	J	K	L	M	N	O	117
120	P	Q	R	S	T	U	V	W	127
130	X	Y	Z	[	\	]	↑	_	137
140		a	b	c	d	e	f	g	147
150	h	i	j	k	l	m	n	o	157
160	p	q	r	s	t	u	v	w	167
170	x	y	z	{	|	}	¬		177
	170	171	172	173	174	175	176	177	

Font: sym, s

	040	041	042	043	044	045	046	047	
040		↓	•	⌉	−	┼	⊥	÷	047
050	├	┤	└	⌋	/	⁻	∅	~	057
060	₀	₁	₂	₃	₄	₅	₆	₇	067
070	₈	₉	‖	≠	≤	□	≥	∫	077
100	⌈	▽	∞	Ψ	Φ	{	℞	Λ	107
110	™	₀	©	®	Ω	∂	ƀ	ℓ	117
120	→	Γ	Θ	Σ	}	Ξ	∝	Δ	127
130	≡	Υ	≃	[	Π	]	⊤	⁼	137
140		α	β	ψ	φ	ϵ	£	λ	147
150	η	ι	ϑ	κ	ω	μ	ν	ρ	157
160	←	χ	θ	σ	τ	ξ	×	δ	167
170	χ	υ	ζ	⟩	|	⟨	¤		177
	170	171	172	173	174	175	176	177	

Font: sym_super, ssp

	040	041	042	043	044	045	046	047	
040		↓	•	⌉	−	∔	⊥	÷	047
050	⊦	⊣	∟	⌋	/	⁻	∅	~	057
060	0	1	2	3	4	5	6	7	067
070	8	9	‖	≠	≤	□	≥	∫	077
100	⌜	▽	∞	Ψ	Φ	{	R_x	Λ	107
110	™	*o*	©	®	Ω	∂	ℬ	ℓ	117
120	→	Γ	Θ	Σ	}	Ξ	∝	Δ	127
130	≡	Υ	≃	[	Π	]	⊤	=	137
140		*a*	*β*	*ψ*	*ϕ*	*ϵ*	£	λ	147
150	*η*	*ι*	*ϑ*	*κ*	*ω*	*μ*	*ν*	*ρ*	157
160	←	*χ*	*θ*	*σ*	*τ*	*ξ*	×	*δ*	167
170	*χ*	*υ*	*ζ*	›	|	‹	¤		177
	170	171	172	173	174	175	176	177	

Font: sym_sub, ssb

	040	041	042	043	044	045	046	047	
040		↓	•	⌉	−	∔	⊥	÷	047
050	⊦	⊣	∟	⌋	/	⁻	∅	~	057
060	0	1	2	3	4	5	6	7	067
070	8	9	‖	≠	≤	□	≥	∫	077
100	⌜	▽	∞	Ψ	Φ	{	R_x	Λ	107
110	™	*o*	©	®	Ω	∂	ℬ	ℓ	117
120	→	Γ	Θ	Σ	}	Ξ	∝	Δ	127
130	≡	Υ	≃	[	Π	]	⊤	=	137
140		*a*	*β*	*ψ*	*ϕ*	*ϵ*	£	λ	147
150	*η*	*ι*	*ϑ*	*κ*	*ω*	*μ*	*ν*	*ρ*	157
160	←	*χ*	*θ*	*σ*	*τ*	*ξ*	×	*δ*	167
170	*χ*	*υ*	*ζ*	›	|	‹	¤		177
	170	171	172	173	174	175	176	177	

Device x9700_p12x6:

Font: ascii, m

	040	041	042	043	044	045	046	047	
040		!	"	#	$	%	&	'	047
050	(	)	*	+	,	-	.	/	057
060	0	1	2	3	4	5	6	7	067
070	8	9	:	;	<	=	>	?	077
100	@	A	B	C	D	E	F	G	107
110	H	I	J	K	L	M	N	O	117
120	P	Q	R	S	T	U	V	W	127
130	X	Y	Z	[	\	]	^	_	137
140		a	b	c	d	e	f	g	147
150	h	i	j	k	l	m	n	o	157
160	p	q	r	s	t	u	v	w	167
170	x	y	z	{	|	}	~		177
	170	171	172	173	174	175	176	177	

Font: super, sp

	040	041	042	043	044	045	046	047	
040		!	"	#	$	%	&	'	047
050	(	)	*	+	,	-	.	/	057
060	0	1	2	3	4	5	6	7	067
070	8	9	:	;	<	=	>	?	077
100	@	A	B	C	D	E	F	G	107
110	H	I	J	K	L	M	N	O	117
120	P	Q	R	S	T	U	V	W	127
130	X	Y	Z	[	\	]	^	_	137
140		a	b	c	d	e	f	g	147
150	h	i	j	k	l	m	n	o	157
160	p	q	r	s	t	u	v	w	167
170	x	y	z	{	|	}	~		177
	170	171	172	173	174	175	176	177	

Font: sub, sb

	040	041	042	043	044	045	046	047	
040		!	"	#	$	%	&	'	047
050	(	)	*	+	,	-	.	/	057
060	0	1	2	3	4	5	6	7	067
070	8	9	:	;	<	=	>	?	077
100	@	A	B	C	D	E	F	G	107
110	H	I	J	K	L	M	N	O	117
120	P	Q	R	S	T	U	V	W	127
130	X	Y	Z	[	\	]	^	_	137
140		a	b	c	d	e	f	g	147
150	h	i	j	k	l	m	n	o	157
160	p	q	r	s	t	u	v	w	167
170	x	y	z	{	|	}	~		177
	170	171	172	173	174	175	176	177	

Font: italic, i

	040	041	042	043	044	045	046	047	
040		*!*	*"*	*#*	*$*	*%*	*&*	*'*	047
050	*(*	*)*	***	*+*	*,*	*-*	*.*	*/*	057
060	*0*	*1*	*2*	*3*	*4*	*5*	*6*	*7*	067
070	*8*	*9*	*:*	*;*		*=*		*?*	077
100	*@*	*A*	*B*	*C*	*D*	*E*	*F*	*G*	107
110	*H*	*I*	*J*	*K*	*L*	*M*	*N*	*O*	117
120	*P*	*Q*	*R*	*S*	*T*	*U*	*V*	*W*	127
130	*X*	*Y*	*Z*					*_*	137
140		*a*	*b*	*c*	*d*	*e*	*f*	*g*	147
150	*h*	*i*	*j*	*k*	*l*	*m*	*n*	*o*	157
160	*p*	*q*	*r*	*s*	*t*	*u*	*v*	*w*	167
170	*x*	*y*	*z*						177
	170	171	172	173	174	175	176	177	

Font: script, scr

	040	041	042	043	044	045	046	047	
040		!	"	#	$	%	&	'	047
050	(	)	*	+	,	-	.	/	057
060	0	1	2	3	4	5	6	7	067
070	8	9	:	;		=		?	077
100	@	A	B	C	D	E	F	G	107
110	H	I	J	K	L	M	N	O	117
120	P	Q	R	S	T	U	V	W	127
130	X	Y	Z					_	137
140		a	b	c	d	e	f	g	147
150	h	i	j	k	l	m	n	o	157
160	p	q	r	s	t	u	v	w	167
170	x	y	z						177
	170	171	172	173	174	175	176	177	

Font: tcc

	040	041	042	043	044	045	046	047	
040		!	"	#	$	%	&	'	047
050	(	)	*	+	,	-	.	/	057
060	0	1	2	3	4	5	6	7	067
070	8	9	:	;	<	=	>	?	077
100	@	A	B	C	D	E	F	G	107
110	H	I	J	K	L	M	N	O	117
120	P	Q	R	S	T	U	V	W	127
130	X	Y	Z	[	\	]	↑	_	137
140		a	b	c	d	e	f	g	147
150	h	i	j	k	l	m	n	o	157
160	p	q	r	s	t	u	v	w	167
170	x	y	z	{	|	}	¬		177
	170	171	172	173	174	175	176	177	

Font: tya

	040	041	042	043	044	045	046	047	
040		!	"	#	$	%	&	'	047
050	(	)	*	+	,	-	.	/	057
060	0	1	2	3	4	5	6	7	067
070	8	9	:	;	<	=	>	?	077
100	@	A	B	C	D	E	F	G	107
110	H	I	J	K	L	M	N	O	117
120	P	Q	R	S	T	U	V	W	127
130	X	Y	Z	[	\	]	^	_	137
140		a	b	c	d	e	f	g	147
150	h	i	j	k	l	m	n	o	157
160	p	q	r	s	t	u	v	w	167
170	x	y	z	{	|	}	~		177
	170	171	172	173	174	175	176	177	

Font: sym, s

	040	041	042	043	044	045	046	047	
040		↓	•	⌉	−	┼	⊥	÷	047
050	├	┤	⌊	⌋	/	¯	∅	~	057
060	0	1	2	3	4	5	6	7	067
070	8	9	‖	≠	≤	□	≥	∫	077
100	⌐	▽	∞	Ψ	Φ	{	℞	Λ	107
110	™	∂	©	®	Ω	∂	ƀ	ℓ	117
120	→	Γ	Θ	Σ	}	Ξ	∝	Δ	127
130	≡	Υ	≃	[	Π	]	⊤	≈	137
140		α	β	ψ	φ	∈	£	λ	147
150	η	ι	ϑ	κ	ω	μ	ν	ρ	157
160	←	χ	θ	σ	τ	ξ	×	δ	167
170	χ	υ	ζ	⟩	|	⟨	¤		177
	170	171	172	173	174	175	176	177	

Font: sym_super, ssp

	040	041	042	043	044	045	046	047	
040		↓	•	⌉	−	┼	⊥	÷	047
050	⊢	┤	⌊	⌋	/	¯	ø	~	057
060	0	1	2	3	4	5	6	7	067
070	8	9	‖	≠	≤	□	≥	∫	077
100	Γ	▽	∞	Ψ	Φ	{	R_x	Λ	107
110	™	*o*	©	®	Ω	∂	ƀ	*l*	117
120	→	Γ	Θ	Σ	}	Ξ	∝	Δ	127
130	≡	Υ	≃	[	Π	]	⊤	=	137
140		*a*	*β*	*ψ*	*φ*	∈	£	λ	147
150	*η*	*ι*	*ϑ*	*κ*	*ω*	*μ*	*ν*	*ρ*	157
160	←	*ϒ*	*θ*	*σ*	*τ*	*ξ*	×	*δ*	167
170	*χ*	*υ*	*ζ*	>	|	<	¤		177
	170	171	172	173	174	175	176	177	

Font: sym_sub, ssb

	040	041	042	043	044	045	046	047	
040		↓	•	⌉	−	┼	⊥	÷	047
050	⊢	┤	⌊	⌋	/	¯	ø	~	057
060	0	1	2	3	4	5	6	7	067
070	8	9	‖	≠	≤	□	≥	∫	077
100	Γ	▽	∞	Ψ	Φ	{	R_x	Λ	107
110	™	*o*	©	®	Ω	∂	ƀ	*l*	117
120	→	Γ	Θ	Σ	}	Ξ	∝	Δ	127
130	≡	Υ	≃	[	Π	]	⊤	=	137
140		*a*	*β*	*ψ*	*φ*	∈	£	λ	147
150	*η*	*ι*	*ϑ*	*κ*	*ω*	*μ*	*ν*	*ρ*	157
160	←	*ϒ*	*θ*	*σ*	*τ*	*ξ*	×	*δ*	167
170	*χ*	*υ*	*ζ*	>	|	<	¤		177
	170	171	172	173	174	175	176	177	

Devices x9700_p14x8 and x9700_114x8:

Font: ascii, m

	040	041	042	043	044	045	046	047	
040		!	"	#	$	%	&	'	047
050	(	)	*	+	,	-	.	/	057
060	0	1	2	3	4	5	6	7	067
070	8	9	:	;	<	=	>	?	077
100	@	A	B	C	D	E	F	G	107
110	H	I	J	K	L	M	N	O	117
120	P	Q	R	S	T	U	V	W	127
130	X	Y	Z	[	\	]	↑	_	137
140		a	b	c	d	e	f	g	147
150	h	i	j	k	l	m	n	o	157
160	p	q	r	s	t	u	v	w	167
170	x	y	z	{	|	}	¬		177
	170	171	172	173	174	175	176	177	

Font: super, sp

	040	041	042	043	044	045	046	047	
040		!	"	#	$	%	&	'	047
050	(	)	*	+	,	-	.	/	057
060	0	1	2	3	4	5	6	7	067
070	8	9	:	;	<	=	>	?	077
100	@	A	B	C	D	E	F	G	107
110	H	I	J	K	L	M	N	O	117
120	P	Q	R	S	T	U	V	W	127
130	X	Y	Z	[	\	]	↑	_	137
140		a	b	c	d	e	f	g	147
150	h	i	j	k	l	m	n	o	157
160	p	q	r	s	t	u	v	w	167
170	x	y	z	{	|	}	¬		177
	170	171	172	173	174	175	176	177	

Font: sub, sb

	040	041	042	043	044	045	046	047	
040		!	"	#	$	%	&	'	047
050	(	)	*	+	,	-	.	/	057
060	0	1	2	3	4	5	6	7	067
070	8	9	:	;	<	=	>	?	077
100	@	A	B	C	D	E	F	G	107
110	H	I	J	K	L	M	N	O	117
120	P	Q	R	S	T	U	V	W	127
130	X	Y	Z	[	\	]	↑	_	137
140		a	b	c	d	e	f	g	147
150	h	i	j	k	l	m	n	o	157
160	p	q	r	s	t	u	v	w	167
170	x	y	z	{	|	}	¬		177
	170	171	172	173	174	175	176	177	

Font: bold, b

	040	041	042	043	044	045	046	047	
040		!	"	#	$	%	&	'	047
050	(	)	*	+	,	-	.	/	057
060	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	067
070	**8**	**9**	:	;	<	=	>	?	077
100	**@**	**A**	**B**	**C**	**D**	**E**	**F**	**G**	107
110	**H**	**I**	**J**	**K**	**L**	**M**	**N**	**O**	117
120	**P**	**Q**	**R**	**S**	**T**	**U**	**V**	**W**	127
130	**X**	**Y**	**Z**	[	\	]	^	_	137
140		**a**	**b**	**c**	**d**	**e**	**f**	**g**	147
150	**h**	**i**	**j**	**k**	**l**	**m**	**n**	**o**	157
160	**p**	**q**	**r**	**s**	**t**	**u**	**v**	**w**	167
170	**x**	**y**	**z**						177
	170	171	172	173	174	175	176	177	

Font: italic, i

	040	041	042	043	044	045	046	047	
040		!	"	#	$	%	&	'	047
050	(	)	*	+	,	-	.	/	057
060	0	1	2	3	4	5	6	7	067
070	8	9	:	;	<	=	>	?	077
100	@	A	B	C	D	E	F	G	107
110	H	I	J	K	L	M	N	O	117
120	P	Q	R	S	T	U	V	W	127
130	X	Y	Z	[	\	]	^	_	137
140		a	b	c	d	e	f	g	147
150	h	i	j	k	l	m	n	o	157
160	p	q	r	s	t	u	v	w	167
170	x	y	z						177
	170	171	172	173	174	175	176	177	

INDEX