bcc	title	prefix/class-number.revision
	M1 TEXT FILE STANDARD	TEXTF/S-27

checked	authors	approval date / revision date	pages
Bobbie W Kampen	L. Peter Deutsch	1/28/70	11
checked: Larry L Barnes		classification: Specification	
approved: Mel		distribution: Company Private	

ABSTRACT and CONTENTS

This document specifies the format of text files in the M1 system, including provisions for line-numbered files and several kinds of "marginal notes."

I. File Organization

A text file has a header which specifies some global information about the file and gives the order of data pages in the file. Each data page contains some optional non-textual information, followed by a contiguous block of text and some empty space. This situation is summarized in figure 1. This organization allows insertion and deletion of pages by moving around the index in the header, and insertion and deletion within a page without affecting other pages (except the header). Figure 2 depicts the header.

Line numbers have their uses: for example, non-interactive editing from a listing, or in BASIC. Consequently, there is a bit (LN) in the header to indicate that a file is line-numbered. Furthermore, if the line numbers are in order, then one need not search the entire file to find the Nth line if one knows the first line number on each page. Hence the LNS bit to indicate whether the file is sorted by line number.

The line number attached to a line is in a special word in the prefix that may precede the line: if there is a need for additional information attached to a line, there may be more such prefix words. The PF field in the header gives the size of the prefix: if LN is on, PF must be at least 1. Files produced by the standard

editor and library routines will have PF = 1 if LN = 1 and PF = Ø if LN = Ø, but one can think of other cases: for example, a system which kept track of the date on which a line was last changed.

In the index itself, which is preceded by a convenient hardware array descriptor, each entry basically contains a page number and a count of textual information on that page. Here again, one can imagine uses for additional information about a page which one would like to have access to without touching the page itself. In line-numbered files, the number of the first line on each page appears in the entry for that page. The ability of a hardware descriptor to multiply a subscript by any factor up to 64 is very convenient here.

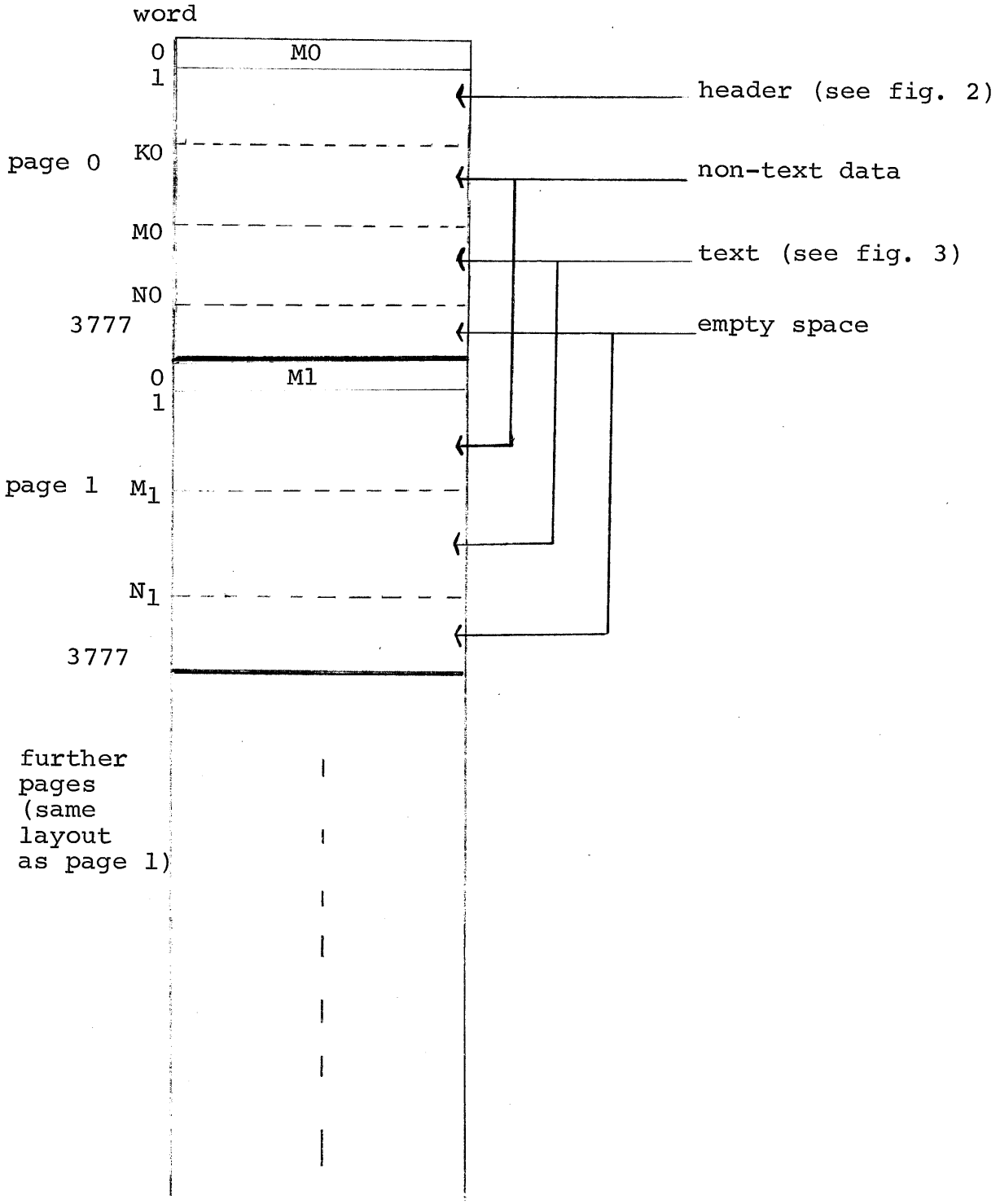
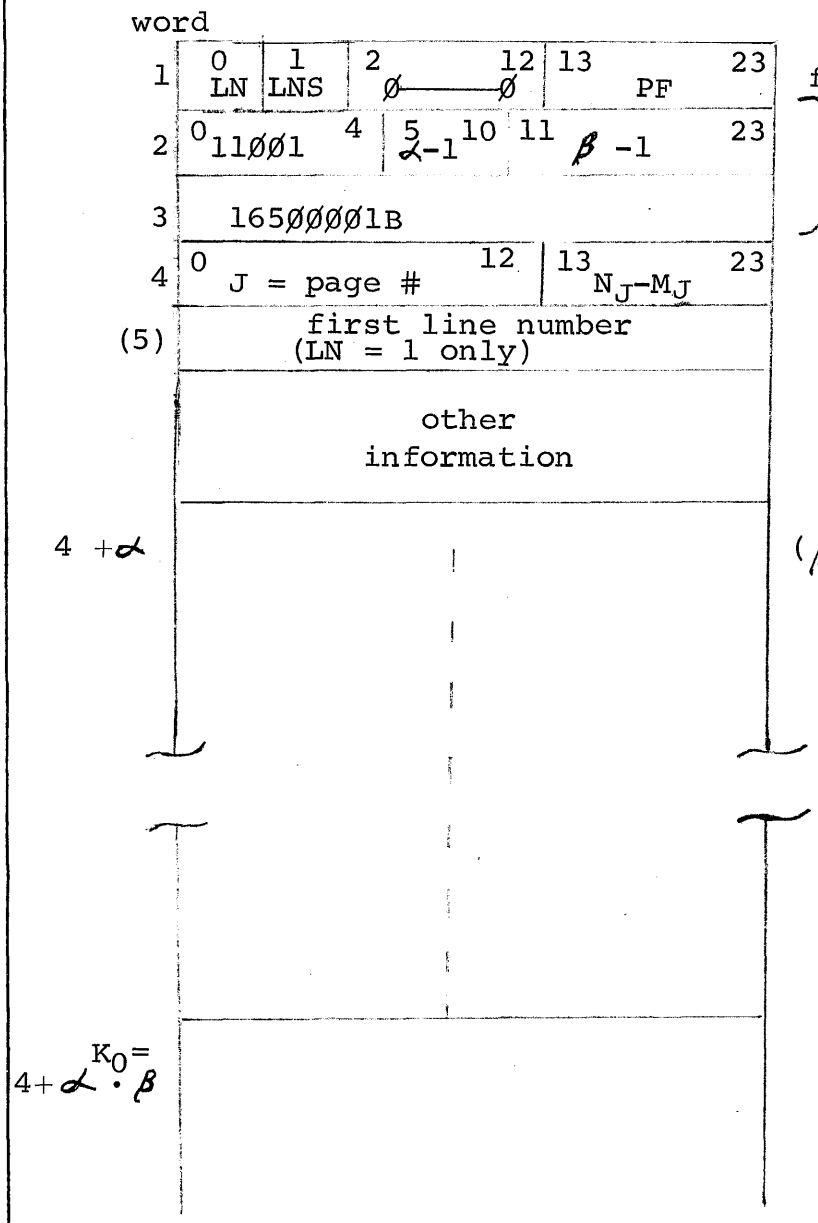


Fig. 1: format of a text file



LN = file is line-numbered
LNS = file is line-numbered and sorted
PF = number of prefix words on each line

The array descriptor has LB = ATRAP = Ø, LEB = 1, MULT = α -1, UB = β -1, and address = source + 1 (i.e. word 4)

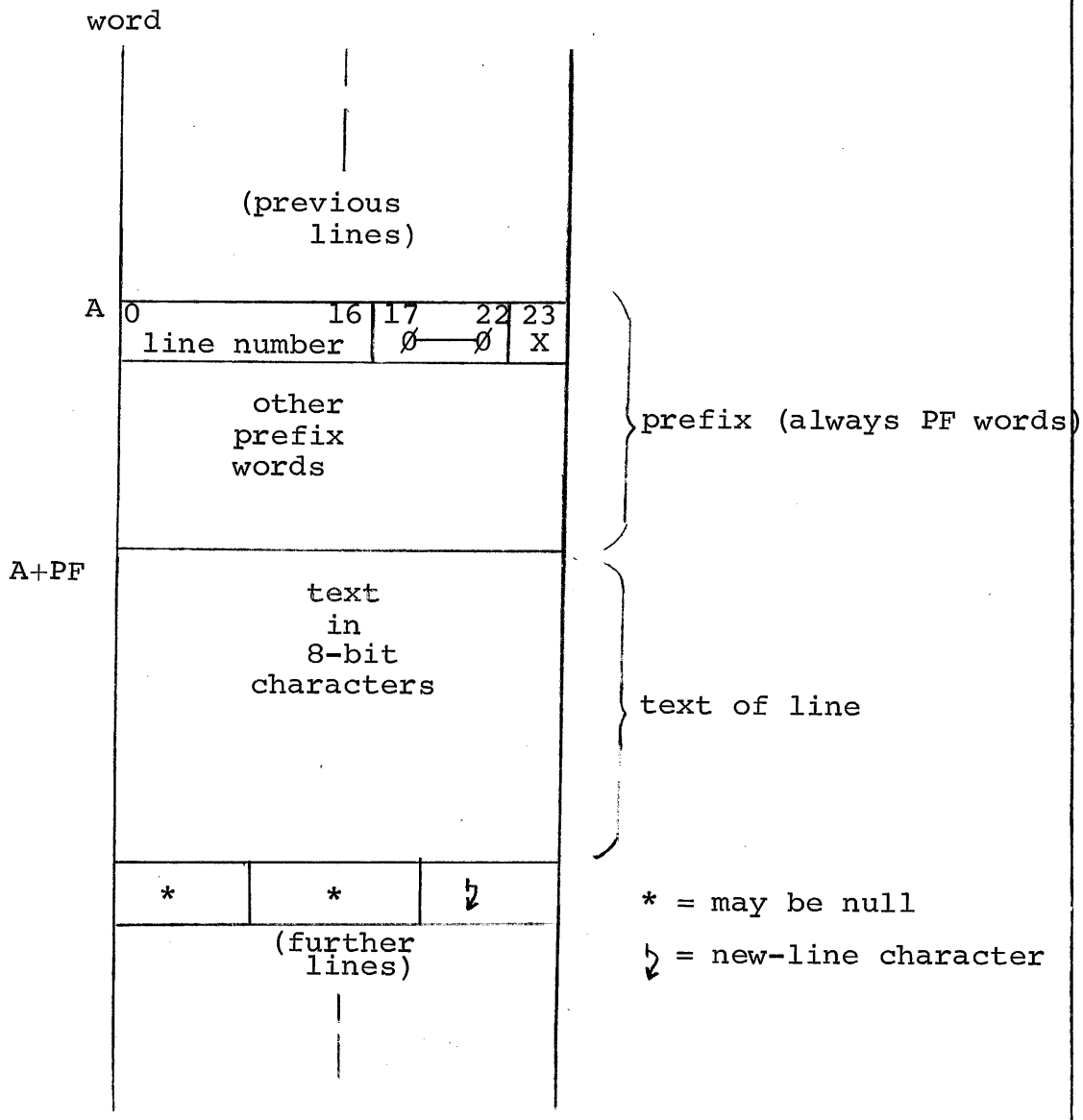
Fig. 2: format of header

II. Text Page Organization

For applications such as a document preparation system, which might require keeping layout or structural information with the text, a text page may start with non-textual information, headed by a word count (see figure 3). The text itself is formatted in lines. A line always begins with PF words of non-textual information. If the file is line-numbered (LN = 1), the top 17 bits of the first word of the prefix contain a binary line number between 0 and 99999 inclusive. If LNS = 1, the line numbers in the file must be increasing order with no duplications. The bottom bit of the line-number word is defined as the complement of the corresponding bit in the end-of-line character to allow sequencing through the file backward.

The text itself consists of 8-bit characters in the M1 standard character set. The only restriction is that the end-of-line character must only appear as the last character of the line. In fact, it must appear in the last character position of the last word occupied by the line: this may require inserting one or two null characters just before it. Otherwise, the contents of the line are unrestricted as far as this standard is concerned: it may contain nulls, characters not defined by the character set standard, etc. The convention of terminating the line with an end-of-line character was chosen in preference to a prefixed character

count because it is less vulnerable to local hardware error, it allows sequencing through the file backward, and it enables an editor to use binary (logarithmic) rather than linear search in a sorted line-numbered file. (These last two properties depend on all words of the prefix being distinguishable from end-of-line words.) The binary search proceeds by first going to the middle of the right page, then scanning to the next line boundary and comparing the line number found there against the desired line number; this indicates which half of the page holds the desired line. The process then repeats on the half-page, quarter-page, etc., until the block being searched only contains a few lines, which can then be searched linearly.



The first word of the prefix only has the format shown if the file is line-numbered. X is the complement of the bottom bit of the ↵ character.

Fig. 3: format of text

III. Property List

Every text file begins with a property list, which consists of all the text up to the first blank line. Each line of the property list begins with a property name, which is any string of non-blank characters, followed by either the end-of-line or else at least one blank and then further information associated with that property. The intent is that this area be used for communication between programs (e.g. a language processor recording the number of symbols for use by a cross-reference program), from a program to a user (e.g. a background program recording the time it took to process the file), or from a user to a program (e.g. a user informing an editor about layout conventions). All such applications have the property that ill-advised tinkering by the user can only result in either misleading reporting or less than optimum performance by a program: a mangled property list, unlike a mangled index or prefix, leaves the file perfectly readable and processable. (To avoid duplication of property names, we might want to set up a registry).

IV. Discussion

An adequate standard for text files should enjoy the following properties:

1) It should have low overhead for ordinary text;

2) It should provide for efficient editing;

3) It should not be fragile, i.e. the effects of local hardware errors should be local;

4) It should be open-ended, allowing for additional structural or summary information in a systematic way;

5) It should only specify those aspects of text storage actually necessary for communicability of text files among all parts of the software.

This M1 text standard deals with these desiderata as follows:

1) The overhead is the page index, the nulls used to fill out lines to a word boundary, the empty space at the end of each page, and a few fixed words at the front of the file. Only the second and third of these are significant: for large files, where they are not dominated by the fact that space cannot be allocated in chunks smaller than 2K, they should add up to about 20% for lines of reasonable length.

2) Insertion and deletion of small amounts of text involve only the affected page. Moving 2K words takes 2 milliseconds in the phase 2 CPU, a comparatively short time.

Inserting and deleting pages involve a similar operation on the index. Finding the Nth line involves going to the correct page, a simple task since line counts are kept in the index, and counting end-of-line characters, which can be done a word at a time. Finding line N in an unsorted line-numbered file involves searching the entire file; however, in a sorted file the index contains the initial line number on each page, and a form of binary search can be used within the page.

3) A hardware error anywhere except the index or the first word of a page affects only the line or lines it mutilates. An error in the index may result in not knowing the correct order of pages in the file, or the amount of text on a page; the latter problem could be solved by adopting the convention of keeping the empty space filled with nulls. An error in the first word of a page may result in not knowing where the text of that page starts. These two irreducible errors still leave the file in a state where it can be fixed up by hand after running it through a program that converts all 2048 words of each page to readable form.

4) Three places have already been discussed for insertion of non-textual information in a text file: in index entries, at the head of a page, and in the line prefix. We assume that in general, such information is only useful to the program that created it in the first place. This does not vitiate the standard: the purpose of the standard is to ensure that the textual information in a text file

written by any program can be read (but not necessarily manipulated) by any other program. The property list provides a "marginal note" facility that is guaranteed manipulable by all programs, as well as by the user.

5) Designing and describing a good standard is not simple (as anyone who has ever read the USASI FORTRAN and COBOL standard knows.) Hopefully this document answers all interesting questions within its scope of discussion.