Computer-aided typesetting with a CRT

By Marya S. Goldman*

An experimental program written for a computer-aided typesetting project provides scope editing facilities as well as typesetting functions. The configuration used was a PDP/7 and 340 Display. The program accepts text from paper tape, and arranges the characters into a page of justified lines which is displayed on the CRT. The user is allowed to edit the text on the display with a light pen and commands from an on-line console. The text is always maintained in justified lines and can be punched onto paper tape at any time. The characters on the display can appear in upper and lower case roman and italic in two sizes. The processing of footnotes has been allowed for.

(Received January 1969)

This paper describes a program which was written as the initial part of a computer-aided typesetting project for a book printer with access to a small computer and a CRT display. The printer desired to use a computer as an aid in producing books but did not want to let the computer change the methods or output of the printing house.

Book printing

The process of printing a book in the traditional way was analysed and the procedure divided into stages:

1. The manuscript is annotated with marks to indicate size and style of type, special characters or instructions.
2. The manuscript is keyboarded onto a paper tape which drives a typesetting machine (e.g. a casting machine which produces lead type) which produces right justified text.
3. Footnotes and other type not of the same size as the text type are set separately and added to the text during make-up into pages by a hand craftsman.
4. Proofs are printed, the errors marked by proof readers, and the type is corrected manually.
5. Proofs are printed, sent to the author for his alterations, and the type corrected.
6. When steps 4 and 5 have been repeated to the customer's satisfaction, the type is put on the presses and sheets of pages are printed.
7. The sheets are folded, the edges of the pages are cut, and the book is bound.

The application of a computer in this process is at the keyboarding, make-up and correction stages, steps (2, 3, 4 and 5). The typewritten manuscript must be turned into a code on paper tape which specifies the characters in the text as well as information to justify the right hand margin of each line. The corrections involve changing characters in the text and often respacing the lines in which the text has been altered so the lines will be justified. Large changes in the text can also result in upsetting the layout of the page, causing several pages to be made-up again with the proper number of lines on each page. Traditionally, the craftsman does correction, respacing, and make-up by picking up the individual characters of lead type and moving them about.

Employing a computer to justify text has been accomplished many times in recent years, (e.g. in the UK: Elliott Bros. for Thomson Newspapers at Reading and Hemel Hempstead, ICT for Southwark Offset in London, Digital Equipment for various newspapers, IBM for newspapers and general printing) (Balzer, 1967), but the problem of correcting justified text is more complex. Text stored on magnetic or paper tape can be updated with corrections by line-numbering the text and then specifying which lines are to be corrected on an additional input tape. This procedure can lead to confusion in the text when the paper tape with the corrections has been punched with errors.

It is believed that a more satisfactory way to amend text is to use a CRT display on which justified text can be corrected and the new version of the text seen immediately. The visual response afforded by the CRT method is of importance to the printer of high quality, high accuracy work because, besides changing the particular characters in the text, the craftsman who corrects the type also makes sure that the pages are aesthetically satisfactory in their make-up and that there are no awkward breaks in lines or uneven spacing that make reading difficult. Furthermore, since the skill of the printer is still at hand, and can be used readily, a general facility for page make-up can be included in the system without very complex programming.

Thus, it was the aim of the project to simulate the operations with metal type on a CRT and to remove certain tedious operations from the craftsman's responsibility. A constraint on the project was the equipment to be used. A small computer was desired to demonstrate that a system of relatively high sophistication could be achieved with a low capital expenditure. Experience at the University Mathematical Laboratory at Cambridge showed that text editing could be successfully programmed on the PDP-7 and 340 Display there (Wiseman, 1966). The configuration consists of paper tape punch and reader, on-line console, and 8K of 18 bit words. Access to this computer was obtained for the development of an experimental program, CUP-TEXT, which is the subject of this paper.

CUP–TEXT

CUP–TEXT reads text from paper tape, justifies the text, displays a page on the screen, accepts editing

* *Cambridge University Press.* Now at *School of Architecture, University of Cambridge*

instructions or more text from an on-line keyboard and rejustifies the text on the screen, and punches out a paper tape with a corrected version of the text in a code suitable for a typesetting machine.

The program is designed in a modular fashion and the modules may be changed to accommodate differing operational requirements. The input/output routines are independent of the main data structure; the instructions which answer peripheral interrupts and organise the I/O housekeeping are not contained in the main program; all display information is contained outside the main routines; and routines for justifying, editing and specifying type faces are self-contained. The project made use of this flexibility during development when it was necessary to change many of the parts to accommodate different pieces of printing equipment and try different methods of displaying or correcting the text.

For example, the standard input routine accepts either unjustified tape with a modified Flexowriter code or justified tape in TTS code, but routines have been written to accept unjustified tape produced in ASCII and ISO codes. The output routine also exists in several versions for various filmsetting machines.

At this stage in the programming specific modules are built into various programs and are not movable. The method of assembling programs in the standard software does not permit relocateable addresses in routines and no independent assembly system has been written. It is envisaged that an extended suite of typesetting modules would eventually be implemented which could be used in various combinations.

User facilities

To the user, CUP–TEXT is straightforward. From the on-line console he can either type text directly on the screen or enter a phase of the program, Altmode, which interprets subsequent characters typed in as editing, format, or control characters. On the display a marker in the form of an underline can be moved under any character with a light pen or the edit instructions. This marker identifies the character in the text affected by edit operations. If a character is typed in directly from the console it will be inserted in the text on the screen after the underlined character and the underline will move under the new character. As each character is entered in this way, the text on the screen is rejustified. The inserted characters can be anywhere on the screen either in the middle or at the end of text already present or as new text on a blank screen.

Single characters can be erased if the RUBOUT key is depressed instead of an alphanumeric key on the on-line console. The character which is erased is the one with the marker under it. The marker moves to the left after the deletion and the text is rejustified.

A special non-printing character typed on the console will allow the operator to use the Altmode instructions. They are of three kinds:

1. Edit instructions which move the underline, erase and insert blocks of text, transpose characters, and make-up the page;
2. Format instructions which change the type size or style and the line width;
3. Control instructions which read in text and punch out text.

A summary of these can be found in **Table 1** and are explained in detail in the following section.

Altmode instructions

When the on-line operator begins a job, he will first load his text tape into the reader and set the switches on the computer to indicate which code is on the tape. The Altmode instruction M will read in enough lines of text to make a page. The lines will be justified and the page will appear on the screen with the footnotes at the bottom and the reference numbers in order (see **Fig. 1**). When he makes a correction in the text, the lines will be rejustified but the page make-up will be undone and each footnote will appear immediately after the word referencing it (see **Fig. 2**). There will probably be more lines on the screen than are required for one page since all the text in the computer that can fit on the screen is being displayed. The extra lines arise because the text is read into a 128 word buffer area in the store and not made up into lines until the buffer is filled and because some text may be added during editing. When the corrections are completed, the operator will indicate with the underline which line is to end the page and use the Altmode instruction P to make-up the page again with the footnotes at the bottom and the surplus text removed from the screen (though not from the store). If he then finds another error he can correct it and use the make-up instruction again until the page is satisfactory. Only when the page is made-up can he use the Altmode instruction O to punch the text out onto paper tape. The Altmode command M to read more text will remove the old text down to the end of the page and add the new text after the surplus lines at the end of the previous page. If there were no end of page marker (i.e. the page had not been made-up) the new text would be added to whatever text was present. In either case, the computer would read enough text to put a complete page on the screen.

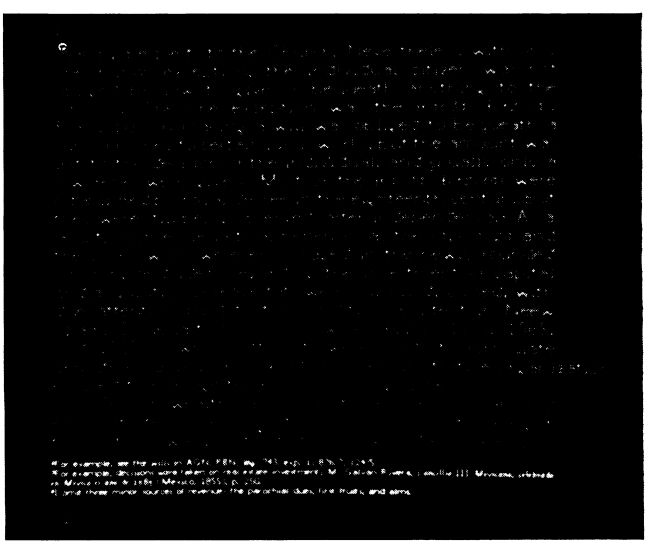

Fig. 1. Display screen with a page of text that has been read in and made-up. The word 'organization' hanging over the right hand margin needs to be hyphenated.

The restriction on punching out only when there is a made-up page is unique in the Altmode commands. Otherwise, any of the instructions can be used in any order and corrections can be made in any order in the text presently on the screen.

The editing instructions in Altmode include, besides instructions to move the underline instructions to create a secondary marker in the form of a double underline which is used with the movable underline to isolate a block of text. Using Altmode commands X, I, E, the block of text (anything from one character to several lines) can be extracted into an invisible buffer and inserted at a later time into another place in the text or can be erased. Erased text disappears from the screen after the Altmode command E.

Blocks of text enclosed by the underline and double underline can be changed to italic or roman upper and lower case with the Altmode commands ←, ↑, —, =. The Altmode command T is used to transpose the characters over the single underline with the one to its right.

Character widths

To obtain the variety of characters necessary for printing, four fonts are available in any one job. In this program, a font is defined as 256 characters usually consisting of upper and lower case roman and italic letters, numbers, and punctuation. Associated with each font is a table of unit widths for all the characters. The program assumes that the width of a character is the same in roman and italic and contains two widths values for each character, one for upper and one for lower case (e.g. if A = 13 and a = 9, then italic $A = 13$ and $a = 9$ also). This arrangement is satisfactory for most filmsetters but for lead casting machines a different version of the width look-up routine exists which obtains separate widths for roman and italic.

The program also assumes that the user might wish to mix characters of different type sizes in one line. For this reason, there must be a separate width table for each type size used, even if the type face, or style, is the same. When the user indicates a change of font in Altmode, the program adds the widths of characters of both the old and new fonts in the line until the width of the line equals the preset line measure. The alternate method of changing type size is to hold the widths for all sizes of the alphabet of one type face as one set of numbers and to vary the line measure proportionally to the size change, but this does not permit mixing of sizes in a single line.

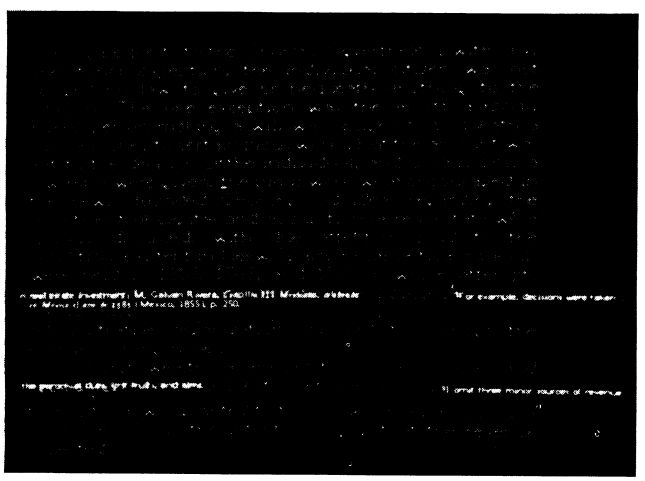

Fig. 2. Display screen with page of text during editing. The page make-up has been undone.

Each font contains four alphabets, where an alphabet is defined as 64 characters. The usual practice is for a font to have upper and lower case roman and italic alphabets, but the width tables for the alphabets could be changed to produce a font with other combinations such as bold type instead of italic, or Greek instead of roman, or small and large upper case in roman and italic without any lower case, etc. The only problem with such changes at the moment is that there is only one set of characters on the display (see **CRT character generator** for details of the CRT). For the user to change alphabets in one font there is a shift key for upper and lower case and Altmode instructions for roman and italic.

Format instructions

The format instructions in the Altmode commands are available to the off-line keypunch operator. In this experiment, a Friden Flexowriter was used. The keyboard layout was modified to include some function keys. A format instruction is recognised by the program if it is proceeded by an Altmode code, which appears on the Flexowriter hardcopy as a′, (see **Fig. 3**). For example, a change to font 1 involves the three keystrokes Altmode, F, 1, and appears on the hard copy as 'F1. If an error is made in an Altmode instruction, a special character appears on the display when the tape is processed.

Beside the Altmode instructions, there is the footnote signal, which is represented on the hard copy by ∼. When this is keyed, all the text following it up to the next occurrence of the footnote code will be treated in the computer as a footnote and set in a smaller type size with a reference number inserted in the text and at the beginning of the footnote. Each footnote is keyed as it occurs in the text, immediately after the word which references it.

Although on the Flexowriter there is a new-line key which positions the hard copy at the beginning of the next line, this code is ignored by the computer when read in on the tape and can be used by the keypunch operator to create a legible copy from the Flexowriter typewriter. The program inserts new line characters in the justification routine as necessary. At the places in the text where the line is not justified, as at the end of a paragraph, a special new line character which prints as a § on the hard copy is punched on the tape. The tab key is used as on a typewriter, the location of the tab having been set in the program. The space bar is used for the variable space between words.

Data structure

The program, CUP–TEXT, is constructed on simple principles which could be translated to another computer with a suitable configuration. For a concise description of the PDP-7 and 340 display the reader is referred to Appendices 1 and 2 of Cross (1967).

making bequests to the Church. Nevertheless, with only one minor exception, the individual citizen was not obliged by law to give or bequeath anything to the Church. The one exception was the '→mandas forzosas'X. Every person making a will was obliged to bequeath a sum of money to certain pious works, but the amount was left to the decision of the individual, and usually only a few '→reales'X were given. Most of the '→mandas forzosas'X were established by royal decree in the eighteenth century but they were still being observed after independence.~For example, see the wills in AGN, PBN, leg. 743, exp. 1, 876/7, 124/5.~ As a result, such ecclesiastical benefices as the '→capillanza'X and the pious work were encouraged in the newly founded Church, and ideas regarding the investment of capital and acquisition of real estate were quickly discussed, with due attention paid to the conditions existent in New Spain. Ecclesiastical councils were held in 1555, 1565, and 1585 in which attempts were made to formulate definite fiscal policies.~For example, decisions were taken on real estate investment; M. Galvan Rivera, '→Concilio 'XIII'→ Mexicano, celebrado en Mexico el ano de 1585'X (Mexico, 1855), p. 250.~ Gradually, administrative organizations evolved to manage the ever-increasing riches.§

Church wealth came from three main sources and each affected a specific sector of the economy.~I omit three minor sources of revenue: the parochial dues, first fruits, and alms.~ The right to levy these was granted to the Crown in a papal bull of 1 January 1501, the intention being that the revenue collected should be used by the monarch towards financing the establishment of the Church, and towards providing a stipend for the clergy. Spaniards and '→mestizos'', and to a lesser extent the Indians, were expected to pay, and virtually every item of produce from cows and sheep to eggs and milk was taxable, and not even the parish priests themselves were exempt. The distribution of the money raised from the sale of the produce was made in accordance with royal decrees issued on 3 Dc· was given to the bishop, and one quarter to the Cathedral chapter; the remaining half was divided into ninths, four of which went to the parishes, three to the maintentance of ecclesiastical buildings and hospitals and the remaining two ninths were reserved for the monarch. This theoretical division remained without alteration until the year 1804 when an additional ninth was ordered to be given to the monarch.§

In many respects it was inevitable that the Church would acquire great wealth in Mexico and that the State would eventually have to intervene and nationalize clerical holdings of real estate and capital. The main purpose of the conquest and colonization of New Spain was in theory the conversion of the indigenous population to the Christian fiath, which meant that the clergy were looked upon from the very beginning as key members of the colonial society. The privileges enjoyed by the Church in Spain, for exaple the ecclesiastical '→fuero'X, were therefore naturally transferred to the newly established Church in Mexico. In addition, apart from religious conviction, the Spanish monarchs [ere well aware of of the value of an established Church in maintaining their own influence and control over the distant colony. I

Fig. 3. Hard copy produced on Flexowriter when unjustified tape is punched

In CUP–TEXT, characters in the text are stored in a list, one character per word, in an internal code. The code accommodates the characters in a basic font and some auxiliary characters in 9 bits (half a PDP-7 word) leaving enough room in each word for change of font and measure information. The characters without format changes could be packed two characters per word but the extra program locations and instruction time required would probably offset the advantage of computer store gained.

Justification

As text is read into the program the characters are tested for special codes. If the character is one in the basic alphabet, its width is obtained and added to the width of the word presently being accumulated. When a variable space code is encountered, the width of the current word plus the width of the space plus the width of the rest of the line is compared to the line measure. If the line measure has not been exceeded, the line width is updated and the new word width is zeroed. If the line measure is shorter than the complete line, the variable spaces are tested to see if they can be contracted to include the last word without becoming smaller than the minimum allowed space. If the word cannot fit in, it is carried over to the next line and the variable spaces are expanded to fill the line, unless their increased width makes them larger than the maximum allowed space. A system line ending is put after the appropriate word in the place of the space coded there. If the word cannot be carried over or squeezed into the line, it must be hyphenated. At this stage in its development the program does not have a hyphenation routine, but leaves the word at the end of the line for the user to divide it.

Rejustification

When a section of the text in the computer needs to be rejustified, a routine scans the list of internal code characters, interprets them as part of the basic font or as spaces, and uses the justifying routine described above to adjust the spaces and reinsert system line endings. All system line endings are treated as spaces before the line is rejustified. Besides identifying characters, spaces and system line endings, the scanning routine recognises user line endings and format codes. When it comes to a user line ending (e.g. at the end of a paragraph), it will zero the line width for the start of the next line and leave the variable spaces in the current line with the standard number of units.

In the text list, format codes are tagged by a bit set in the left hand half of the word. When this bit = 1, the routine examines the other bits in that half word for changes of font and measure and makes the necessary adjustments to the working parameters. For example, a font change implies that the size of part of the text on the screen may be changed and that the address of the beginning of a new width table will be needed.

Footnotes

The routine also recognises footnotes. A footnote always begins with a special word that has two bits set in the left half and contains a computer address in the remaining bits. The address points to the place in the list of text where the main body text is resumed. At the end of the footnote is another special word which contains the address of the beginning of the next footnote. When the page is made-up, these links allow the footnotes to be placed at the bottom of the page. During the time when the routine scans the text for rejustification, the footnotes are still attached to words in the text and the link words contain the address of the subsequent word in the text list. This forces the routine to justify the footnotes within the text, as is done when text is initially read into the computer, and to number them in correct order.

When the routine sees the first special word, it saves the width of the line being processed, zeros the present line and word widths, and begins justifying the characters in the footnote. The routine restores the width parameters for the line in the body of the text when it comes to the second special word at the end of the footnote.

The text in the computer is rejustified by this routine after the insertion or deletion of a character directly from the on-line keyboard or of several lines of text in a block edit.

Page make-up

Page make-up is achieved by connecting the link words in the footnotes, if any, and placing a page number, or folio, at the bottom of the page. No attempt to make-up the page according to any printer's rules is made. It is left to the on-line user to rearrange the lines on the page when he finds clumsy page endings. Such instances might be a hyphen in the last word on the last line or the last line of a paragraph falling at the top of a new page.

In the first instance of make-up, at the tape input stage, the height of each line is accumulated into a total page length. The height of a line is defined as the height of the type plus any additional space between lines. It is kept with the font size information. If a line contains a footnote, the height of the line plus the height of the footnote lines is held until the text line is complete before adding it to the page length, much as the way in which the width of a word is kept until a space code. As in the justification routine, the accumulated page length and page make-up is effected when the desired length of the page is reached. In the instance that the last line on the page contains a footnote that would make the page too long, that line is taken over to the next page and the page is left short.

A more sophisticated make-up routine has been tried but it was found to be unnecessary since the first attempt at make-up after the tape is read in is usually disturbed by correction and subsequent make-up is under the control of a skilled operator at the CRT. Several editing instructions are desirable to aid him in perfecting the make-up of a page, such as one which tightens up the spaces throughout the page, thereby saving a line.

Vertical spacing, which can be entered from the console or paper tape, is treated in the same manner as a footnote during the calculation for make-up, the height of the space being associated with the height of the line which precedes it.

CRT character generator

Text appears on the display in two sizes. There is no hardware character generator on the 340 Display at

the University Mathematical Laboratory, but character generation by software is relatively easy. The ability to program the shape of a character is advantageous in this application since the characters used by printers are of variable width and may include special symbols and non-roman alphabets. There are two drawbacks on this computer. The instructions out of which the characters are constructed must be stored in the core of the computer, taking up programming and text space. The time taken to draw a character on the screen is much longer than it would be if the characters were hardware generated, and when there are many characters on the screen, the accumulation of these times is often longer than the bright time of the phosphor on the face of the CRT since the beam moves at a constant velocity. The resulting flicker makes the text very difficult to read. These two problems are out of the programmer's control, but it is now obvious that more suitable hardware must be chosen before the system can be used by the printer.

It was necessary to generate all the characters in a basic font to make the text on the screen meaningful to the user. The characters are made of vectors which are grouped into subroutines, one subroutine per character. The display is digital and contains on its $9\frac{3}{4} \times 9\frac{3}{4}$ inch screen a grid of 1024×1024 locations on which the CRT beam can rest. In a printer's font each character has a unique width expressed in units. The linear measure of each unit depends on the size of the type used. The unit widths of letters in printer's type vary slightly among type faces but the proportions of letters follow a general pattern. For example, an M is usually about twice the width of an I. For this exercise, one printer's unit was chosen to correspond to two spots on the CRT grid and a well-known type face, Baskerville, was chosen as being representative of traditional book type. An on-line system written at the Mathematical Laboratory (Grant, 1967), which lets the user examine and change on the screen the effect of display instructions as they are being composed, was employed to develop a set of characters which have the same unit widths as the letters in Baskerville. The complete character generator for upper and lower case roman and italic letters, numbers and punctuation takes up about 1,500 words in the store.

The size of the characters on the display depends on the scale setting in the display commands. The settings 0, 1, 2, 3 double successively the length of the vectors drawn on the screen. The characters were designed to be used in scale 1 (an upper case M appears $\frac{1}{3}$ inch wide). Because the changes of scale are in multiples of two, it was decided to represent the text on the screen in only two sizes. The body of the text is in scale 1 and both the footnotes and small type in half the text size. The use of three sizes would have meant that the smallest characters would have been illegible or the largest so big that a whole page could not fit on the screen.

CRT workings

Movement of the CRT beam in the 340 Display is controlled from instructions interpreted by the display hardware. These instructions may be located anywhere in the computer store. After the display has decoded the instructions and drawn on the screen, an interrupt is sent to the central processor. Control is transferred from the main program to a routine which restarts the display.

In CUP–TEXT the text is stored in internal code and only one character at a time is displayed. The character is translated into a display subroutine with a table containing subroutine calls arranged in internal code order for the four alphabets in a basic font. It is then sent to the CRT data channel, prefaced by a control word specifying the scale setting, and control is returned to the main program. This process is repeated until all the text has been displayed and then restarted to regenerate the page.

Besides translating internal code characters, the display routine checks for changes in font which are expressed as changes of scale and inserts double and single underlines. It also recognises jumps in the text around the footnotes and executes these on the screen.

Although it might seem that this method of displaying text would slow down the CRT, in fact the time taken to draw a character is much greater than the time expended translating a character from internal code. There is no additional flickering on the screen this way than exists when the CRT ranges down a list of display subroutines without being stopped. The extra instructions which are obeyed after each interrupt take considerable time from the main program, but in a reactive system most of the main program time is not spent in processing, but in waiting for the operator to type on the on-line console or use the light pen. Thus the user does not notice that the central processor is busy updating the CRT.

The advantage of this method is that the text is stored in a code that is not dependent on the CRT.

Text file manipulation

The manipulation by the program following an instruction from the on-line console is achieved by adjusting the text file. List processing techniques were considered for inserting or erasing blocks of text and for providing a structure in the text to be used during justification, but the extra programming instructions and store in the lists required made these techniques seem impractical. Instead, a system of pointers and word-shifting routines was developed.

The main pointer keeps track of the character which is underlined on the screen; that is, it indicates which character is currently being processed during input or editing. When a new character is added to the text file, the content of the pointer is updated to hold the address of the new character. If the character is at the end of the file, the value of the pointer will be equal to the contents of the end pointer which has in it the address of the last word in the file. If the new character is to be put in the middle of the file, all the text from the pointer to the end is moved down the store by one location, so the new character can fit in. A reverse motion occurs if a character is erased. Thus, at any time the file does not contain characters which are not desired in the text. After the text has been shifted, all of the file is rejustified. This occurs because the link words before and after footnotes must be updated and several lines may need to be respaced after an edit.

Another pointer holds the address of the word marked

by the double underline on the screen during block edits. In this case, erasure of a block between the double and single pointers is affected by moving up the text from the higher pointer to the lower one and eventually removing the double underline by zeroing its pointer in store.

Further development

Further possible extensions of the program are numerous. The obvious ones are a hyphenation routine, a routine for setting tabular matter, and edit commands for page make-up. The first of these, hyphenation, has been ignored because much work has been done elsewhere on this complex problem and the development of another hyphenation routine seemed to be redundant. In this particular system, the need for hyphenation by the computer is not crucial because the operator can insert hyphens manually at the CRT with ease.

The other two have been considered in detail but omitted for the present time for the sake of space in the computer. The problem of setting right justified tabular matter can be reduced to setting several columns of matter with very narrow measures without inserting a line ending after each column. Since multiple column formats also involve setting narrow measures across a page, the program might also be extended to cater for this layout. Multicolumn text could be held in the data file as one long single column but appear on the display and output tapes as if continuous across the width of the page. The editing commands for page make-up have been mentioned previously. An extended study of page make-up requirements would be valuable.

More elaborate systems programming is also needed to allow easy input of width tables and parameters. The present method relies on using DDT, the Digital Equipment software routine for program alteration, for input of line width, page length, etc.

Extension of the character generator must be contemplated before very complex setting is undertaken. Without extending the computer configuration to include backing store, the only way to access a larger range of characters is to read from paper tape before each job only the subset of character subroutines needed for the particular text.

An important next stage in the development of a computer aided composition system is the composition of mathematical equations. It is particularly advantageous to be able to edit on the screen in this application because of the ease of singling out specific characters on the screen with the light pen. In the CRT system the equation would be made-up again automatically after a correction and displayed. The user could see immediately if the equation was set awkwardly and could make adjustments until it was typographically sound as well as accurate in its contents.

Results

The program in its present state is still rather crude and not up to the standard required for a working system, but it has shown that CRT editing is indeed a realisable component in a computer-aided composition project. Sufficient facilities have been implemented to enable operational tests to be carried out and a number of experiments have been made. Time studies took place at the printing house and at the computer. Comparisons were based on the number of minutes taken to alter an 8-page section of a book with normal first proof house corrections: transpositions, mis-spelt words, keyboard errors, and casting machine errors. In the computer project conditions were far from ideal. A light pen was used rather than a tracker ball located on the keyboard console, there was considerable flicker on the display, the keyboard console was in an awkward position, and no provision was made for holding or lighting the marked proof from which the operator took instructions.

Nevertheless, it was found that the display method of correction was about three times faster than the traditional way in lead type. The ratio is much larger for certain kinds of corrections, such as deletion or insertion of several lines or words. It was observed that most of the time at the computer was taken in locating the word or letter to be corrected on the proof and on the display. Further studies have been recommended to reduce this time.

Conclusion

This exploratory programming has been useful in demonstrating that the concept of computer-aided typesetting with scope editing facilities is a viable proposition and in determining the nature of the equipment that would be suitable in such a system.

It is amusing to speculate about a giant multi-access computer with a CRT in every reader's box, on-line input keyboards or OCR input directly from the manuscript, and output signals to high speed filmsetters or electronic printing machines. A more sensible approach, economically and technically, is the one adopted here. Two or three displays used for correction and make-up can handle the composing room load for a medium sized printing house. Complex computer networks are not involved. Most importantly, changes in the printers' craft are not needed, since the reader can mark the proof and the compositor can alter it on the CRT with techniques used traditionally.

Acknowledgements

I would like to express my gratitude to the Cambridge University Press for employing me to research into this subject and for allowing me to present this paper. In particular I would like to thank Mr. L. A. Gray for his patience and unfailing printing guidance in directing my work. I would also like to thank the staff of the University Mathematical Laboratory at Cambridge for their co-operation, for the use of their equipment and in particular Mr. N. E. Wiseman for his guidance throughout the project.

Table 1

Altmode instructions

Format instructions:

↑ text entered after the underline will be in roman
← text entered after the underline will be in italic
F, followed by 0, 1, 2, or 3
all text after the underline will be in font 0, 1, 2, or 3
W, followed by 1, 2, or 3
all text following the underline will be justified to a measure specified to the computer by the 1, 2, or 3.

The program returns to normal mode after the format instruction has been interpreted by Altmode.

Editing instructions:

E erase the text between the single and double underlines; the page is rejustified
X extract the text enclosed by the single and double underlines and put it into the buffer
I insert the contents of the buffer after the pointer in the text and rejustify the page
P make-up the page with the footnotes and folio in the right place, the end of the page is indicated by the underline position
U move underline up one line
D move underline down one line
R move underline to the right one character
L move underline to the left one character

CNTRL with U, D, L, or R
create double underline at current single underline position if double one is not already present, move single underline as instructed by U, D, L, R
T transpose the character over the underline with the one to its right

RUBOUT erase the character over the underline and move the underline to the left
↑ change characters between the double and single underline to roman
← change characters between double and single underline to italic
— change characters between double and single underline to upper case
= change characters between double and single underline to lower case

Control instructions:

M read in paper tape until there are enough lines to make a page; if a made-up page is on the display when the instruction is typed, the page is deleted and a new one read in after any surplus text at the end of the page; if there is not a made-up page, the new text is added to the lines already in the store
O punch out a made-up page onto paper tape

ALT return to normal mode
' summon Altmode
→ italic to follow
× roman to follow
§ line ending
~ footnote beginning or ending
▨ erase (ignored by program)

References

BALZER, F. (1967). Survey of Computer Aided Typesetting in Europe, *Print in Britain*, Vol. 15, No. 4, pp. 18–23.
BARNETT, M. P. (1967). *Computer Typesetting, Experiments and Prospects*, MIT Press, Cambridge and London.
COMPUTER TYPESETTING CONFERENCE (1965). Report of Proceedings, Institute of Printing Ltd., London.
COMPUTER TYPESETTING CONFERENCE (1966). Preprints, Institute of Printing Ltd., London.
COOPER, D. L., and NIELD, C. D. (1968). *An Analysis of Computer Typesetting Systems in the UK*, Pira, Leatherhead, Surrey.
CROSS, P. (1967). A Logical Drawing Board for the PDP–7/340, *The Computer Bulletin*, Vol. 7, No. 3, pp. 243–245.
GRANT, J. S., and HORTON, J. (1967). Display Assembly Language, private communication.
STEVENS, M. E., and LITTLE, J. L. (1967). *Automatic Typographic-Quality Typesetting Techniques: A State-of-the-Art Review*, U.S. Government Printing Office, Washington, D.C.
WISEMAN, N. E. (1966). A Scope Text Editor for the PDP–7/340. DECUS Proceedings of 1966 European Spring Seminar of DEC Users' Society.

The Computer Journal—Honorary Editors

Because of pressure of company business, Mr. H. W. Gearing is to relinquish his position as joint honorary editor, business application papers, with effect from the end of Volume 12. Mr. Gearing will remain a member of the Editorial Board, available for consultation and refereeing.

Business applications papers will be dealt with by Mr. A. McG. Leckie with effect from Volume 13, February 1970. His address for correspondence is:

24 Beverley Road, Sunbury-on-Thames, Middx.