Secrets of the Exidy Word Processor

The best reason for owning an Exidy Sorceror is its word processing capability. An excellent machine-language program sold by Exidy (in the form of a read-only-memory cartridge called a Rom Pac) provides valuable text editing features such as block deletion or insertion, block moves, and selective search-and-replace. The user is given complete control of tabs, line length, and page length.

Ordinary user commands such as "print" or "save" can be entered on a separate line at the top of the screen. (That mode of operation is called the "command mode," to distinguish it from normal full-screen text entry in the "edit mode.") Less ordinary commands can be imbedded within the text, for dynamic operations such as changing print parameters on the fly or ejecting the page. These imbedded commands show up as peculiar graphic shapes on the screen, and are referred to as graphic commands.

Yet another feature, and one that isn't widely known, is user extendability. The value of that will be seen throughout this article. (If you'd like more background information than I've given here, see the article "Do the Job for a Lot Less" in the March 1980 issue of Microcomputing.)

The Bad News

I recently disassembled a large portion of the Word Processor Pac, trying to understand how it handled proportional spacing, boldfacing, subscripting, and all the other fancy operations mentioned in the User's Manual from Exidy. Like any other computer owner, I wanted my system to have every possible option; I thought its inabilty to do those things was a result of mere ignorance. After all, my printer (an NEC Spinwriter) was capable of microscopic carriage control with the best of them.

The most surprising discovery I made is that the Rom Pac will not do boldfacing, proportional spacing, subscripting, or superscripting. Don't believe everything the User's Manual tells you! You must add extra software of your own to implement those features; the best that can be said of the Rom Pac is that it is extensible. I will be outlining in this article the basis for adding your own features in software.

Another way to get the extra features is to add smart hardware. A Diablo printer, for example, can be educated by adding specially programmed read-only memory. Other smart printers are appearing on the market now, such as the Xymec and the Centronics 737. One advantage to the hardware solution is speed -- your computer doesn't have to send a multitude of control characters to the printer. The disadvantage is cost, especially if you already own a semi-smart printer, as I did.

More bad news. Some of the graphic commands won't work within a line of text, but must be on a line all by themselves; the formfeed (GRAPHIC-1) and reformat (GRAPHIC-5) are examples.

The mark (GRAPHIC-9) is supposed to serve as a place marker, to automatically halt the execution of large-scale commands such as forward, backward, delete, and print. It does its job for the first three, but it does not halt printing. Fortunately, several of the commands _will_ stop printing: GRAPHIC-8 designates the end of text, and GRAPHIC-2, 3, and 4 are treated as errors. When the Pac encounters one of the latter commands during printing, it pauses to ask your judgment on the error: to continue printing (hit RETURN) or to abort (hit the ESCape key).

This makes possible a kluge method for subscripting: insert a "wrong" graphic command, then the subscript, then another command. When the printer pauses at each command, position the paper manually, and then hit RETURN to resume. If you have a Centronics-style printer which buffers a line at a time, your procedure will be more involved; at the pause, the preceding characters on the line are still sitting in the line buffer waiting for a carriage return.

My solution to this subscripting problem will, in a moment, serve as an illustration of how to add your own features. But first I need to cover a little more background.

Canned Output Routines

The Word Processor Rom Pac contains two ready-made printer drivers, one using the Sorceror's serial port and the other the parallel port. The serial printer driver starts at hexadecimal address DE90. The parallel driver, at address DE70, is designed for a Centronics-like printer (such as my Spinwriter).

A characteristic of the latter driver is that it filters out and discards line feeds, because Centronics printers usually supply their own line feeds after receiving a carriage return. Since, however, we *want* to produce line feeds for the purpose of subscripting, we must use the following short modification in Z-80 assembly language. It jumps into the Centronics driver immediately after the line-feed filter.

```
F5             PUSH AF
C3 75 DE       JP   CENTRX+5
```

Now, the normal way to access one of those output routines is the selection of a value in a table, the so-called Y-table. Choosing Print Device 1 specifies the serial driver and Print Device 2 the Centronics one. The Rom Pac will then send its stream of individual characters to the chosen output port.

Neither choice is really right for our purposes, though. It's difficult to make large-scale format changes when we're catching one character at a time. We would rather get our hands on a whole line at a time, in some sort of print buffer. And we can do that, by selecting Print Device 0.

Print Device 0 does nothing. (If it seems only natural to you that Device 0 does nothing, then you haven't read the Exidy Manual. The secret of this non-device is very well hidden.) The Pac carefully avoids doing anything to the line of text, to make sure it doesn't interfere with whatever fancy driver routine you're using for a daisy-wheel printer. Device 0 doesn't even send out the individual characters as the other devices do.

Here's how it works. Location 07DC is reserved for a jump to a printer driver. The default is C9, i.e., a do-nothing return, but you can change it to C3 70 DE for a Centronics, or to C3 90 DE for serial output, or to your own jump. But the only characters sent to that location are spaces, vertical tabs, and carriage returns, for indenting, tabbing, ejecting the page, etc. I suspect they're handled separately like this in case you have a Diablo-style printer with separate platen control lines.

What about the rest of the characters? A print buffer at 06B1 is filled with a line at a time. Nothing is done with the line: no justifying, no acting upon graphic commands. The Pac then hops to address 07E9, where you can put a jump to your own buffer-handler. The default content of 07E9 is a simple return; the characters are sent nowhere.

The Solution

Enough background. We now know how to access the print buffer and where to send the characters after we're through with them. We want to write a routine that does the following (outlined here in "structured English"):

```
REPEAT until the end of the buffer:
     Get a character from the print buffer.
     If it's a subscript command:
          OUTPUT the sequence of characters
          for a half-line-feed.
     If it's a superscript command:
          OUTPUT the sequence of characters
          for a negative half-line-feed.
```

```
If it's a normal character:
    OUTPUT it.
```

Listing 1 shows this routine coded into Z-80 assembly language. Some fine points to note are:

1. The imbedded commands that stand for subscript and superscript are the hexadecimal values 12 and 13. (See also Table 1.)
2. The seven-byte sequence that produces a half-line-feed on a Spinwriter is:

   ESC-]-R   to select half spacing,

   LF   to do it, and

   ESC-]-W   to resume normal spacing.

   For a negative line feed, change the LF to ESC-9.
3. The output routine we use for those special escape sequences is the one we saw earlier: Centronics with line feeds. For normal text output, however, we still use the canned driver, so that we don't get doubly-spaced lines.

4. A carriage return is what marks the end of the print buffer.
5. We don't send a return at the end of the line; that's handled separately for Device 0.

We store the code in the unused memory starting at 0000, and we put a jump to it at 07E9. We also need to put at 07EC (that's where the spaces and returns are sent) a jump to the Centronics driver. To clarify:

```
At 07E9:  C3 00 00
At 07EC:  C3 70 DE
```

For your system you might need to alter the escape sequences for your printer, or the output routine if yours is a serial device.

Other Solutions

It is relatively easy to expand the method to handle boldfacing, shadow printing, automatic centering, formfeeds, and vertical tabs. Bidirectional printing is another natural extension, since a one-line buffer is already set up; just send to the printer the proper byte sequence to initiate right-to-left carriage motion, then output the buffer in reverse. If you feel really ambitious, you can try adding true proportional spacing.

Text Storage Formats

The text as you enter it is stored in a buffer which begins at 0800 hex. At the head of the buffer is a string of fourteen 0E bytes, followed by a 02 (ASCII for start-of-text, STX). Your text is stored from 080F up. At the end comes an end-of-text character (ETX, 03) and a trailer of fifteen 0E bytes.

Text is stored in memory essentially as ASCII characters. The non-alphanumeric ASCII codes (less than 32 or greater than 127 decimal) signify special operations, as shown in Table 1. You'll see in the table the familiar tokens for the sub- and super-script commands.

Notice the efficiency of the text and command storage. Indentation of an entire subparagraph requires an overhead of only three bytes. Line feeds are not stored. Space filling for right justification does not take up any extra room, nor does underlining. Very compact.

The one-line print buffer also uses many of the codes in Table 1. It occupies the space from 06B1 through 072F hex. Location 06B0 is a justification flag; it contains a 1 if the line needs space-filling for right justification. The print line in the buffer always ends with a carriage return.

The memory area between 0730 and 07FF is used as a scratch pad for all the operating parameters. Table 2 lists the secrets of the work area, as far as I've been able to unravel them. The major functional areas are:

a storage area for buffer pointers,
a tab table,
a table of print parameters, and
a series of jump instructions for user-definable print vectors.

Figure 1 is a memory map, showing those functional areas and others in the Sorceror.

Exploring Further

Several other nice features can be added to the Word Processor, beyond the print-formatting extensions discussed above. You can, for example, write your own global commands. When the Pac receives one of the four undefined letters (G,J,N,O) or a non-letter, it jumps to 07EF. Since that's in user memory (RAM), it's modifiable. You can insert a jump to your own execution routine. You could install a help function (display a list of legal commands), or a word-counting function (if you get paid by the word). The structured-English foundation for your command processor might look like:

```
Examine the command, using the command buffer
     and its pointer. (See Figure 1 and Table
     2.)
If it's a '?':
     Go do the HELP function.
If it's an 'N':
     Go COUNT the words.
If it's anything else:
     Return to the INVALID message in the Pac.
```

Another idea. Now that you know where the text is stored in memory, it is relatively easy (and I've done it) to write a modem transmission routine; take bytes beginning at 0800 hex and send them out the Sorceror's serial port one at a time, until the end-of-text byte is encountered. (For a similar technique, see the article "Use Your Exidy as a Smart Terminal" in the July issue of Microcomputing.) Put someone with another Sorceror and Word Processor Pac at the other end of the phone line, and presto -- electronic mail! Or a distributed word-processing business, with all your employees working in their own homes.

If you have ideas of your own, here are some more canned routines in the Rom Pac that might come in handy (all addresses in hexadecimal):

CC0A Sets up reverse-video (black-on-white) characters.

DE4E Keyboard input. (This is the part of the Pac contributed by Exidy. The rest was written for Exidy by Testan Scientific.)

CF52 Beginning of command execution table.

If you want to dig deeper into the Pac on your own, the table at CF52 contains the execution addresses for all the commands. For instance, the first two bytes (at CF52) are B1 D4, so the routine to handle the "A" command starts at D4B1. The two bytes at CF54 form the address for the "B" command, and so forth.

Table 1. The meanings of non-alphanumeric codes. These are stored in the text and print buffers along with the normal ASCII characters to signify formats and special operations. All the codes are given in hexadecimal form.

Table 2. An index to the working and control area of memory, with known functions and their locations.

Figure 1. Memory map showing the partitioning of RAM by the Word Processor Pac. A Sorceror with 32 kilobytes of memory is assumed for illustration; the top three addresses will be different for other systems. The top half of the available 64K is not shown; it includes the Rom Pac itself, video RAM, the Power-On Monitor, and character generators.

(NO CAPTION FOR LISTING 1.)

Table 1

Byte	Significance
01 - 0B	Number of spaces to print between two words. Used in the print buffer if extra spaces are needed for justification.
0C	Hard hyphen occurring at end of a line.
0D	Carriage return. A line feed is not stored along with it, as is the case with some other editors (such as CP/M's).
0E	Soft carriage return for lines longer than the specified page width. End of the line on the video screen.
10 - 19	Imbedded graphic commands. GRAPHIC-1 is 10, GR-2 is 11, and so on. GR-0 is 19.
1D	Soft hyphen.
1F	Indentation marker. An indented block of text begins with a three-byte code: 1F <number of spaces to indent> 1F.
7F	Deleted character. All 7F's are erased when the user presses the CLEAR key.
80 - FE	Underlined characters. If the high bit is one (that is, 80H), the remaining 7 bits are an ASCII character to be underlined.

Locations	Function
0730-073A	Miscellaneous controls and flags.
073B	Page title working byte. Loaded with page title value (from 07D4) at start of each page.
073C-073F	?
0740-0741	Cursor location in video RAM, from F080 to F7FF.
0742-0743	Address of top of text buffer and bottom of holding buffer.
0744-0745	Address of top of holding buffer.
0746-0747	Text pointer, to start of present line.
0748-0749	Pointer to start of next line.
074A-074B	Pointer to end of text.
074C	Post-command parameter, for example 55 in the command "P55" to print 55 lines.
074D	?
074E	Cursor location. (074E) + (0751) = position of cursor within present line.
074F	A print parameter. (?)
0750	?
0751	Cursor location. See 074E.
0752-0755	Indentation values. (?)

0756-0757	?
0758-0759	Print buffer pointer, from 06B0 to 072F.
075A-0762	?
0763-0764	Command buffer pointer, to next command in a series.
0765-0766	Pointer to origin of command buffer, 0600.
0767	Pre-command parameter: number of times to execute a command line.
0768-076D	?
076E-077A	Tab table. The default tabs are 10, 20, ..., 120 (in decimal), so this table in memory initially contains 0A, 14, ..., 78. It ends with the byte FF as a delimiter.
077B-07CF	?
07D0-07DE	Y-table. The table of print values such as page length, margins, and line spacing.
07DF	Print flag. If this is zero, characters aren't sent to the printer (for verifying).
07E0	A print parameter. (?)
07E1	Line length. Default 63 decimal = 3F hex.
07E2	Cassette baud rate. Default = 40 hex for 1200 Baud. 0 means 300 Baud. No effect on serial printer baud rate.

07E3-07E4	Flags indicating whether a cassette write or read file is still open.
07E5	Mode flag, to indicate Command or Edit Mode. (?)
07E6-07E8	Output vector for Print Device 1. Default is C3 90 DE for serial printer.
07E9-07EB	Print vector for Device 0. Does not receive a character stream at all, as discussed in the text.
07EC-07EE	Output vector for Print Device 0, but normally receives only spaces and carriage returns.
07EF-07F1	User-definable vector for unused commands. Default = C3 86 CF = a jump to "INVALID ENTRY" message.
07F2-07F4	A jump vector called during cassette operations.
07F5-07F7	A jump vector called during cassette operations.
07F8-07F9	Initial value for the text pointer, 0800.
07FA-07FF	Unused. (?)

Bryan Lewis Figure 1

7FFF 7F00	Monitor work area and stack.
7EFF	Holding buffer.
yyyy	Free space for text and holding buffers.
xxxx 0800	Text buffer: previously entered text.
07FF 07E6	Series of jump vectors.
07E5 07DF	Operating mode flags.
07DE 07D0	Y-table of printing parameters.
07CF 077B	Scratch-pad area.(?)
077A 076E	Tab table.
076D 0730	Scratch-pad area for various buffer pointers.
072F 06B0	One-line print buffer.
06AF 0640	Word Processing Pac's stack area.
063F 0600	Command buffer.
05FF 0500	Cassette read buffer.
04FF 0400	Cassette write buffer.
03FF 0300	Macro-programming buffer.
02FF 0000	Unused, free for user's additions.

```
EXIDY Z-80 ASSEMBLER
   ADDR  OBJECT    ST #

                   0001 ;      W.P. PAC ENHANCEMENT     BY BRYAN LEWIS     9/23/80
                   0002 ;
                   0003 ;      An output routine for Print Device 0.  Retrieves
                   0004 ;      characters from the print buffer and sends them to the
                   0005 ;      printer, except subscript and superscript command
                   0006 ;      tokens are converted to the proper escape sequences
                   0007 ;      for carriage control.  Written for a Spinwriter, but
                   0008 ;      easily modifiable.
                   0009 ;
                   0010 ;---------------------------------------------------------
                   0011 ;
                   0012 ;      Define a few characters:
                   0013 ;
   >000D           0014 CR     EQU    0DH     ;Carriage return.
   >000A           0015 LF     EQU    0AH     ;Line feed.
   >001B           0016 ESC    EQU    1BH     ;Escape.
   >0012           0017 SUB    EQU    12H     ;The token for the subscript command.
   >0013           0018 SUPER  EQU    13H     ;The token for superscript.
                   0019 ;
                   0020 ;      And a few addresses:
                   0021 ;
   >DE70           0022 CENTRX EQU    0DE70H  ;Centronics driver in the WP Pac.
   >06B1           0023 PBUFFR EQU    006B1H  ;Origin of the one-line print buffer.
                   0024 ;
                   0025        ORG    0000H   ;Put in free memory.
                   0026 ;
                   0027 ;      --------   The main loop   --------
                   0028 ;
  '0000  D5        0029        PUSH   DE               ;Preserve the registers we're
  '0001  E5        0030        PUSH   HL               ;  going to wipe out.
  '0002  21B106    0031        LD     HL,PBUFFR        ;Start at buffer start.
  '0005  7E        0032 REPEAT LD     A,(HL)           ;Get the character pointed to.
  '0006  FE0D      0033        CP     CR               ;If it's a CR, that's the
  '0008  CA1C00'   0034        JP     Z,DONE           ;  end of the buffer.  Done.
  '000B  FE12      0035        CP     SUB              ;If it's a subscript token,
  '000D  CA2000'   0036        JP     Z,DOWNSH         ;  go do a downshift.
  '0010  FE13      0037        CP     SUPER            ;If it's a superscript token,
  '0012  CA2900'   0038        JP     Z,UPSH           ;  go do an upshift.
  '0015  CD70DE    0039 NORMAL CALL   CENTRX           ;Anything else, normal output.
  '0018  23        0040 NEXT   INC    HL               ;Increment pointer to next.
  '0019  C30500'   0041        JP     REPEAT           ;And continue.
                   0042 ;
                   0043 ;      --------   End of main loop   -----
                   0044 ;
  '001C  AF        0045 DONE   XOR    A                ;Clear the flags to make sure.
  '001D  E1        0046        POP    HL               ;Restore.
  '001E  D1        0047        POP    DE
```