<table>
<thead>
<tr>
<th>Author(s)</th>
<th>Griffith, Webster</th>
</tr>
</thead>
<tbody>
<tr>
<td>Title</td>
<td>Programming a remote monitor and display system for a time sharing computer.</td>
</tr>
<tr>
<td>Publisher</td>
<td>Monterey, California: U.S. Naval Postgraduate School</td>
</tr>
<tr>
<td>Issue Date</td>
<td>1963</td>
</tr>
<tr>
<td>URL</td>
<td><a href="http://hdl.handle.net/10945/12900">http://hdl.handle.net/10945/12900</a></td>
</tr>
</tbody>
</table>
Programming a Remote Monitor and Display System for a Time Sharing Computer

Webster Griffith
PROGRAMMING
A REMOTE MONITOR AND DISPLAY SYSTEM
FOR A
TIME SHARING COMPUTER

* * * * *

Webster Griffith
PROGRAMMING
A REMOTE MONITOR AND DISPLAY SYSTEM
FOR A
TIME SHARING COMPUTER

by
Webster Griffith

Submitted in partial fulfillment of
the requirements for the degree of
MASTER OF SCIENCE
IN
ENGINEERING ELECTRONICS

United States Naval Postgraduate School
Monterey, California

1963
PROGRAMMING
A REMOTE MONITOR AND DISPLAY SYSTEM
FOR A
TIME SHARING COMPUTER
by
Webster Griffith
Lieutenant, United States Navy
This work is accepted as fulfilling
the thesis requirements for the degree of
MASTER OF SCIENCE
IN
ENGINEERING ELECTRONICS
from the
United States Naval Postgraduate School
ABSTRACT

The optimization of computer usage from both the users and the computer point of view is discussed. A system wherein several users simultaneously operate separate consoles is explained and specific requirements are laid down for a monitor and display system for a time sharing computer patterned to fit operations at the U. S. Naval Postgraduate School, Computer Center and particularly the CDC 1604 computer and the DD 65 display unit. A more stringent and less flexible system working on the same principle is described and programmed with explanations and instructions for expansion.
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Introduction</td>
<td>1</td>
</tr>
<tr>
<td>2.</td>
<td>System requirements</td>
<td>3</td>
</tr>
<tr>
<td>3.</td>
<td>Description of the general system</td>
<td>5</td>
</tr>
<tr>
<td>4.</td>
<td>Functions of the system</td>
<td>17</td>
</tr>
<tr>
<td>5.</td>
<td>Description of the example system</td>
<td>20</td>
</tr>
<tr>
<td>6.</td>
<td>Conclusion</td>
<td>30</td>
</tr>
<tr>
<td>7.</td>
<td>Illustrations</td>
<td>31-38</td>
</tr>
<tr>
<td></td>
<td>Appendix-A MACHINE DD65A</td>
<td>39</td>
</tr>
</tbody>
</table>
# LIST OF ILLUSTRATIONS

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Interrupt handler</td>
<td>31</td>
</tr>
<tr>
<td>2.</td>
<td>Idle loop</td>
<td>32</td>
</tr>
<tr>
<td>3.</td>
<td>DD65 keyboards</td>
<td>33</td>
</tr>
<tr>
<td>4.</td>
<td>Flow chart of DD65A</td>
<td>34</td>
</tr>
<tr>
<td>5.</td>
<td>Flow chart of KYBORD</td>
<td>35</td>
</tr>
<tr>
<td>6.</td>
<td>Flow chart of MODE</td>
<td>36</td>
</tr>
<tr>
<td>7.</td>
<td>Flow chart of PICGEN</td>
<td>37</td>
</tr>
<tr>
<td>8.</td>
<td>Flow chart of PACK</td>
<td>38</td>
</tr>
</tbody>
</table>
1. Introduction.

The conflict between the large, fast and flexible digital computer and its slow human master has led to the loss of efficiency by having multi-million dollar computers idle as the human operators think.

One answer, and the subject of this thesis, is to let many users operate the computer simultaneously at separate and remote locations to combat the user inefficiency in the computer installation at the U. S. Naval Postgraduate School.

A control system which would permit the user to inspect, alter or compose a program from a remote console was desired. Available equipment included a CDC 1604 Computer and a DD 65 Display Unit, the latter is a device with two cathode ray tubes, character generator, a random access memory for storage (reading is sequential to the character generator and associated logic circuits only), two keyboards for input and the necessary equipment to allow compatibility with the CDC 1604, CDC 160 and assorted radars. Direct linkage is not provided between the keyboards, the display and the user.

General specifications for this type of a system are difficult unless exact requirements are prescribed. The example used is therefore a limited case, however it is hoped that it can lead to a larger, fuller system.

1Control Data Corporation, Computer Division, Control Data 1604 Computer Programming Manual, Publication 167, Undated.

2Instruction Manual for the DD 65 Display Unit.
The system should be capable of utilizing as much of the flexibility of the DD 65 as possible and still allow the user an easy to operate console.

Since the "Monitor" system in use at present is more or less adequate for normal stacked job operation, this system is aimed for the user requiring immediate access to his program or to its input or output. This requirement is such that little modification or addition is needed to expand the functions of the system to those mentioned in section 5.

The principle effort was directed to producing a system which would allow symbol manipulation in such a way as to achieve compatibility between not only the equipment used but also the equipment and the users.

The example program is designed for use with the FORTRAN 60 Compiler\(^1\) although the general system could handle as many compilers/assemblers as needed.

Besides the control system there are two principle subsystems:

a. A keyboard to intermediate code compiler called KYBORD.

b. A intermediate code to display unit compiler called PICGEN.

Other transcoders are of a more minor nature in design if not in importance.

\(^1\)Control Data Corporation, Computer Division, FORTRAN System for the Control Data 1604 Computer, Publication 087A, 1961.
2. System Requirements.

There will be two systems considered:

a. The general system.

b. The example system.

The example system is a lesser included part of the general system which is designed to demonstrate the practicability of the general system. The program of the example system and an explanation is given later. An "*" denotes requirements for the example system.

The system requirements are as follows:

a. Computers
   - CDC 1604 and two CDC 160's.
   *- CDC 1604.

b. Consoles
   - CDC 1604, two CDC 160's and DD 65.
   *- CDC 1604 and DD 65.

c. Input/output (on-line)
   - 9 magnetic tape units, 3 flex punches, one of 3 radar's, experimental devices via a D/A converter, graph plotter, typewriter, DD 65 typewriter, and card reader.

 d. Compilers and Assemblers
    - FORTRAN 60, COOP MONITOR, JOVIAL, NELIAC, SCRAP, AR, and octial machine inputs.
    *- FORTRAN 60.

e. Libraries
   - Any proven programs in the above form.
e. Libraries (Cont'd)* One integrated program as an example.

f. Delays permitted - Only those delays normally encountered in FORTRAN 60 plus actual run times.

g. Edit features - Insertions, deletions and exchanges can be made on programs on tape, in memory, and on the display tube.

* - No changes in tapes.

h. Location of control - Control is in resident.

* - Control is in Program DD 65A.
3. Description of the general system.

a. Resident program.

Under present conditions at the U. S. Naval Postgraduate School Computer Center, the fastest input/output medium is binary magnetic tape. This is a most significant factor in the operation of the system control.

A single resident and control program is required since the time delay in loading and unloading individual resident programs for individual consoles or special purpose usage would not permit the fulfillment of the requirement of instantaneous (for humans) response to DD 65 keyboard hits. This places a fundamental limitation on the system as a single resident program must be large for flexibility of control and display and small for flexibility of the input programs of the users.

The selection of the exact bias point is a delicate decision and the expedious answer of providing all that might possibly be used or only as much as is presently used is not wise since, if the system is to be put in general use, the users must be told a firm number of memory cells which they have available. If the bias is altered the user's previously checked out programs may either not run due to insufficient room or not run efficiently due to not utilizing all the memory available.

The bias point was calculated as follows:

(1) Basic FORTRAN 60 resident 5000 cells (octal)
(2) DD 65 processor (KYBORD) 2400 cells (octal)
(3) Buffer for DD 65 memory. 1000 cells (octal)
one buffer for each DD 65)

(4) Control system program 400 cells (octal)

Or 10,000 cells plus 1000 for each DD 65; over
1/8th of memory. Any further encroachment on the available memory space might easily limit the usefulness of the whole system and confine its use to only a portion of the user traffic.

b. Simultaneity and interrupt priority.

Since DD 65 keyboard hits must be processed without humanly discernible delay and since interrupting the keyboard in mid-routine requires the storage of many flag and switch settings and, also, since the keyboard routine requires less than one milli-second; interrupts actuated by a DD 65 not at present in control will be deferred if the console in control is processing a keyboard hit which does not directly activate a user program.

The interrupt priority list is as follows:

(1) A "normal" interrupt (e.g. "Interrupt on channel 5 active" or "Interrupt on Arithmetic Faults") will be processed as occurring.

(2) A DD 65 key hit interrupt can interrupt a running program but not another keyboard hit.

(3) A program run initiation can interrupt only when the control program is passive or idling. Once initiated a program can be interrupted by (b) or (a) above but not by another program which must wait until the present program is finished.
Priority Chart

<table>
<thead>
<tr>
<th>Program to be run</th>
<th>Idle</th>
<th>Program Being Run</th>
<th>Keyboard Hit Being Processed</th>
<th>Normal Int. In Process</th>
</tr>
</thead>
<tbody>
<tr>
<td>Do</td>
<td></td>
<td>Wait</td>
<td>Wait</td>
<td>Wait</td>
</tr>
<tr>
<td>Key hit to be processed</td>
<td>Do</td>
<td>Do</td>
<td>Wait</td>
<td>Wait</td>
</tr>
<tr>
<td>Normal interrupt to be processed</td>
<td>Do</td>
<td>Do</td>
<td>Do</td>
<td>Do</td>
</tr>
</tbody>
</table>

Table 1

A flow chart illustrating the interrupt scheme is shown in Figures 1 and 2.

c. Keyboard decoder.

The keyboard decoder will analyze the input from the designated input typewriter. It will be noted that each DD 65 unit has two keyboards; Keyboard #1 has a modified and enlarged standard keyboard, Keyboard #2 is a special keyboard which can be programmed as desired and consists of 34 keys each with selectable lights, in addition the meaning of the keys can be altered by using any one of eight overlays which transmits a three bit code along with the six bit code of the individual key. Even with the large number of keyboard #2 keys available, it is still inconvenient to govern all of the possible situations from Keyboard #2 only. Therefore, I have made Keyboard #2 the category selector and have let Keyboard #1 select subcategories as appropriate. For example, if "MODE" is selected by Keyboard #2, then "CHARACTER", "VECTOR", or "SPOT" is selected by hitting "C", "V", or "S" respectively on Keyboard #1.
In general then, the KEYBOARD DECODER analyzes the key hit in light of the key hit history (e.g. A "C" hit after "MODE" has been hit has an entirely different meaning than a "C" hit after "TYPE" has been hit) and sets switches and selects subroutines which will be used later in the KEYBOARD PROCESS portion of the program. The system of category selection on Keyboard #2 and subcategory selection on Keyboard #1 was chosen because:

(1) The number of Keyboard #2 keys under one overlay is limited to 34. Even the limited example system has 48 subcategories plus numerical selects.

(2) If multiple Keyboard #2 hits were used to select subcategories, a problem as to labeling would arise. One can label the category key easily but the subcategory code could not be labeled and would require the user to either remember or make continued reference to a code sheet.

(3) By allowing Keyboard #1 to denote subcategory selections, easily remembered codes can be used (e.g. "S" for small, "R" for right, "P" for pipper, etc.). In addition the number section and sign keys are handy (e.g. a right margin 63 spaces to the right of center is selected as follows: hit "RIGHT MARGIN" key on Keyboard #2 and then hit "+", "6", and "3").

d. Keyboard processor.

Once the KEYBOARD DECODER has completed its operations the KEYBOARD PROCESSOR is called and carries out the execution of the steps required by the particular hits.
e. The intermediate code.

Since information in the DD 65 memory is not retrievable except to be displayed on the DD 65 scopes, some additional storage is required. With our speed requirements only CDC core will serve. This information could be in any of three forms:

(1) Raw input as received in the computer.

(2) Processed input in a form directly transferable to the DD 65 memory.

(3) Processed input of a uniform nature but not necessarily in the form suitable for DD 65 memory.

The first form requires that every type of input be reprocessed by its own peculiar coder/decoder whenever updating or retrieval are required. The second form requires multiple input coders but only one decoder, however the decoder must unpack bits in highly condensed data words and would be slower than the third form which uses a special coder to place each type of input into a simple intermediate code. This code is stored in a buffer area and is identical for all inputs. This is the system used. Each form of input (e.g. DD 65 typewriter, BCD magnetic tape, flex tape) has its own coder which stores the transformed input in an area called PICBUF. Each of these coders are reciprocal and one can go from flex tape to PICBUF and back without loss. However, since all of the input forms differ in some respect, going from the DD 65 typewriter to PICBUF, to magnetic tape, back to PICBUF, and finally to the DD 65 display could, but would not necessarily, involve some loss.
f. Picture generator.

The picture generator (PICGEN) transforms the intermediate coded PICBUF into the form suitable for the DD 65 and transmits this code to the DD 65. There are several ways to approach the use of PICGEN for updating. One, and normally the fastest, is to only process or reprocess the last word sent to the DD 65 memory. This method has the disadvantage of being difficult to edit and some forms of editing are particularly cumbersome. Another method is to update only the last line of display on the DD 65. This allows easy editing of the last line but has the same difficulty encountered in the first method in deeper editing. The method chosen is to update the entire DD 65 memory each time a character is added to PICBUF. This is by far the slowest method but does allow easy and complete editing features. Since PICGEN returns to the control program as soon as an "end of file" symbol is reached, only when PICBUF is nearly full will the time required to decode be of any consequence. The time from entry to exit under worst case conditions is .36 seconds and under normal conditions would be about .10 seconds. For example, a man typing at a rate of 50 words a minute averages .20 seconds per key hit, therefore only a fast typist typing 128 characters per line at the bottom of the page would notice any reduction in his normal typing speed.
THE PIGGEN CODE

00 - Begin string. All bytes after this are transmitted directly to the DD 65 until a 77 byte is used.

01 - Disregard all bytes until this one is processed.

10 - Set for character mode.

11 - Set for vector mode.

12 - Set for spot mode. The next three bytes represent the X (1st 9 bits) and Y (last 9 bits) location of a ".". Exit from the string is automatic. If more than one spot is to be sent, each must be proceeded by a begin string code (00).

20 - Set X location equal to the value of the next byte.

21 - Set Y location equal to the value of the next byte.

22 - Set X location equal to the value of the next byte.

23 - Set Y location equal to the value of the next byte.

30 - Set for small size characters.

31 - Set for medium size characters.

32 - Set for large size characters.

40 - Set for display on right tube.

41 - Set for display on left tube.

50 - Set for normal intensity.

51 - Set for bright intensity.

60 - Set for auto-incrementation right.

61 - Set for auto-incrementation down.

76 - End of file, exit PIGGEN.

77 - Disregard this byte.
<table>
<thead>
<tr>
<th>Character Code</th>
<th>Character</th>
<th>Character Code</th>
<th>Character</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>Blank</td>
<td>26</td>
<td>W.</td>
</tr>
<tr>
<td>01</td>
<td>1.</td>
<td>27</td>
<td>X.</td>
</tr>
<tr>
<td>02</td>
<td>2.</td>
<td>30</td>
<td>Y.</td>
</tr>
<tr>
<td>03</td>
<td>3.</td>
<td>31</td>
<td>Z.</td>
</tr>
<tr>
<td>04</td>
<td>4.</td>
<td>32</td>
<td>J.</td>
</tr>
<tr>
<td>05</td>
<td>5.</td>
<td>33</td>
<td>,</td>
</tr>
<tr>
<td>06</td>
<td>6.</td>
<td>34</td>
<td>(</td>
</tr>
<tr>
<td>07</td>
<td>7.</td>
<td>35</td>
<td>\rightarrow</td>
</tr>
<tr>
<td>10</td>
<td>8.</td>
<td>36</td>
<td>Tab.</td>
</tr>
<tr>
<td>11</td>
<td>9.</td>
<td>37</td>
<td>\geq</td>
</tr>
<tr>
<td>12</td>
<td>\emptyset</td>
<td>40</td>
<td>-</td>
</tr>
<tr>
<td>13</td>
<td>=</td>
<td>41</td>
<td>J.</td>
</tr>
<tr>
<td>14</td>
<td>\notin</td>
<td>42</td>
<td>K.</td>
</tr>
<tr>
<td>15</td>
<td>::</td>
<td>43</td>
<td>L.</td>
</tr>
<tr>
<td>16</td>
<td>'</td>
<td>44</td>
<td>M.</td>
</tr>
<tr>
<td>17</td>
<td>[</td>
<td>45</td>
<td>N.</td>
</tr>
<tr>
<td>20</td>
<td>Space</td>
<td>46</td>
<td>O.</td>
</tr>
<tr>
<td>21</td>
<td>/</td>
<td>47</td>
<td>P.</td>
</tr>
<tr>
<td>22</td>
<td>S.</td>
<td>50</td>
<td>Q.</td>
</tr>
<tr>
<td>23</td>
<td>T.</td>
<td>51</td>
<td>R.</td>
</tr>
<tr>
<td>24</td>
<td>U.</td>
<td>52</td>
<td>J.</td>
</tr>
<tr>
<td>25</td>
<td>V.</td>
<td>53</td>
<td>$\ $.</td>
</tr>
<tr>
<td>54</td>
<td>$\ $.</td>
<td>55</td>
<td>$\ $.</td>
</tr>
<tr>
<td>56</td>
<td>$\ $.</td>
<td>57</td>
<td>$\ $.</td>
</tr>
<tr>
<td>58</td>
<td>$\ $.</td>
<td>59</td>
<td>$\ $.</td>
</tr>
<tr>
<td>60</td>
<td>$\ $.</td>
<td>61</td>
<td>$\ $.</td>
</tr>
<tr>
<td>62</td>
<td>$\ $.</td>
<td>63</td>
<td>$\ $.</td>
</tr>
<tr>
<td>64</td>
<td>$\ $.</td>
<td>65</td>
<td>$\ $.</td>
</tr>
<tr>
<td>66</td>
<td>$\ $.</td>
<td>67</td>
<td>$\ $.</td>
</tr>
<tr>
<td>68</td>
<td>$\ $.</td>
<td>69</td>
<td>$\ $.</td>
</tr>
<tr>
<td>70</td>
<td>$\ $.</td>
<td>71</td>
<td>$\ $.</td>
</tr>
<tr>
<td>72</td>
<td>$\ $.</td>
<td>73</td>
<td>$\ $.</td>
</tr>
<tr>
<td>74</td>
<td>$\ $.</td>
<td>75</td>
<td>$\ $.</td>
</tr>
<tr>
<td>76</td>
<td>$\ $.</td>
<td>77</td>
<td>End of string</td>
</tr>
</tbody>
</table>
THE VECTOR CODE

The vector code describes the size, direction and "on" condition of the vector in six bits as follows:

OX - Blank, small vector.
1X - Blank, large vector.
4X - Unblank, small vector.
5X - Unblank, large vector.
X0 - To the right.
X1 - To the right and up (elevated 45° to the right).
X2 - Up.
X3 - To the left and up (elevated 45° to the left).
X4 - To the left.
X5 - To the left and down (depressed 45° to the left).
X6 - Down.
X7 - To the right and down (depressed 45° to the right).
77 - End of vector string.

THE SPOT CODE

The spot code consists of 18 bits, 3 bytes or 6 octal numbers. The first 9 bits represent the X location and the last 9 represents the Y location. The tube is divided into 512 (1000 octal) lengths in each direction with the (0,0) location at the center. The location code is in one's complement notation.

Examples: (1) 377000 is right center
(2) 400400 is the lower left corner.
(3) 000000, 777777, 000777, and 777000 are all dead center.
An example: If the first few words in PICBUF looked like this:

10 10 31 40 50 60 23 77
21 77 00 62 65 67 71 45
20 70 65 51 65 77 77 76

It would mean to start the file (01), use character mode (10) of medium size (31) on the right tube (40) of normal intensity (50) and increment to the right (60). Start at the left (23 77) upper (21 77) corner. String follows (00). "BEGIN HERE". Stop the string (77) and exit (76).

g. Other decoder or transcoders.

Other media must be translateable to and from the PICBUF code. In this case; Flex paper tape, binary magnetic tape, BCD magnetic tape, binary core and BCD core. The actual source of the code is not important to the code since all will be delivered to a buffer area in core and separate routines will load and dump into these buffers. Therefore, we need three decoders and their reciprocals; Flex, BCD, and binary to PICBUF code and PICBUF code to Flex, BCD, and binary in addition of course, DD 65 keyboard to PICBUF and PICBUF to DD 65 display.

(1) Flex to PICBUF.

(a) Upper case to medium size and lower case to small size characters. The reverse in numerals.

(b) Change color of ribbon works a toggle switch on the intensity code.
(c) Characters not present to "$\$".
(d) "&" to plus sign.
(e) "*" to times sign.

(2) PICBUF to Flex.

The reverse of (1) above except large to upper case and all brackets to "(" and ")". It might be pointed out here that since PICBUF code is the most flexible, care must be taken in forcing some transcriptions like "Increment Down". One can force other media to increment down, however, one can not make them backspace. Impossible transcriptions like "Left Tube" are ignored.

(3) BCD to and from PICBUF

This is straight forward since the FORTRAN 60 BCD characters are relatively few and non-character selects absent.

(4) Binary to and from PICBUF.

The octal, 16 number representation is used.

(5) Vector to and from Graph Plot.

This transcription in either direction is within the capabilities of the system but has not been investigated further.

h. A call to run a user's program.

If the size and complexity of the system are to be retained within bounds, and automatic super-master program which could rebootstrap, control time length of runs under various priority conditions, permit check out program runs, interpretively run non-checked out programs, etc.; can not
be included nor do I feel it would be desirable. Since four users are being kept busy, the expense of one operator would be well worthwhile especially since monitor programs could be done whenever the computer was between runs. Therefore, all programs will be allowed to run if space permits. If a bad program clobbers the system the operator can easily bootstrap and he can cause premature exits from programs which take too long. Here, it might be pointed out is another advantage of the human operator since the users through the intercom can make special requirements known to the operator who can allow or not extra long programs and advise the other users if long delays are expected. The operator could also load special libraries and activate on-line equipment if desired.
4. Functions of the system.

Once the control system is in operation, it can be used for several functions; the diversity of which is determined by the sophistication of the program and the equipment available.

a. Student’s computer.

This mode of operation is designed for use by one not familiar with all intricacies of computer operation. In fact only previously checked out programs on a specific library will be permitted to run. The only option of the user will be to pick up the desired program and insert data. In most cases the output will be to the DD 65 display with an option for a permanent record when desired. Suitable error exits must be made for input data which is not appropriate to the program and likewise all output data must be tested for its fit on the display and printer (or graph plot).

Consequently only programs of a general and continuing nature would warrant the programming of a problem. As an example an instructor could ask a class studying Fourier analysis to input wave forms and observe the output. The student in this case would place vectors on the display to represent the wave form and a bar graph representation could represent the amplitude of the sin and cos coefficients.

In addition, if the results of a program under many varying parameters is to be studied at great length, this mode could allow untrained observers to procure data with little effort.
b. Check-out computer.

This mode of operation would allow the user to write, edit, compile and run FORTRAN programs.

One often is faced with a long time delay from the completion of coding to the commencement of useful runs due to the time consumed debugging their programs. In the check-out computer, a program to be tested is loaded on a tape unit or fed from the satellite 160. The user then requests from the DD 65 a compilation of the program. If compilation errors result they will appear on the display. The user calls in his program to the display tube, makes the necessary corrections, and reloads the tape unit. The process is repeated until all errors of compilation are eliminated. Then the program is run and all errors in running are likewise corrected. When the program is completely error free the user can ask for an on or off-line punch of new cards containing the revised program.

Previously a user had to either wait for the next monitor run or; if operating the computer personally, the computer was in an idle state for long periods while cards were changed, etc.

This computer is not as simple to use as the students' but would require little more knowledge than is needed for FORTRAN programming.

This computer would also be used for parameter variation problems as in the student's computer but would not
require the rigorous testing and "libing" of the latter.

   c. Experimental display computer.

   Using this mode, on-line experimental data is fed through the satellite CDC 160 to the CDC 1604 and the processed information is displayed on the DD 65.

   This offers a unique opportunity to adjust, not only the experimental equipment but also the processor parameters in a rapid sequence which would allow much quicker optimization of the equipment.

   d. Teaching machine.

   Although the student's computer is a teaching machine some such devices are of such a special nature as to require individual libraries and would have to be programmed as needed.

   e. Learning machine.

   As in d. above, special programs would be required.
5. Description of the example program.

As previously mentioned, this program is a lesser included program of the general system designed to show the latters practicability by providing a working system and to form a basis upon which to build the general system.

a. Basic program (DD 65A).

This program is the executive or control program and determines when, what and how soon a subroutine is to be called and what the inputs to the subroutine will be. The Idle Loop and Interrupt Handler would be inserted here in the general system but they are not necessary as such in this limited system. Note that the status reports are sent to DD 65A via the Q-register.

b. Keyboard handler (KYBORD).

This subroutine incorporates both the Keyboard Decoder and the Keyboard Processor of the general system. It has two arguments: 1. The input byte, 6 bits, right justified in the A-register, 2. The keyboard from which it came in the Q-register. The output, also in the Q-register, indicates where the next keyboard hit should originate, however, this is of information value only to the DD 65A since the latter can over-ride any recommendations of KYBORD.

In this and other subroutines routing is used whereby information in one hit alters the processing route of a future hit(s).

The keys are grouped as follows:

(1) Keys which set the characteristics of the
display when Keyboard #1 is subsequently hit.

(a) "Mode"

When the "Mode" key (see figure 6) is hit the routing is adjusted for the following possibilities:

1. "C" is hit next.
2. "V" is hit next.
3. "S" is hit next.
4. A Keyboard #1 key not "C", "V", or "S" is hit next.
5. A Keyboard #2 key is hit next.

If case 5, action is taken in accordance with the new Keyboard #2 hit and the "MODE" hit is ignored. In case 4, an error exit results which displays a flashing "KBI ERROR" on the typing tube and the control program goes into idle. In cases 1., 2. and 3. "CHARACTER", "VECTOR", or "SPOT" is displayed on the status tube and appropriate initiating action is taken in the PICBUF region. At the completion of the action DD 65 awaits the next Keyboard #2 hit.

(b) "SIZE"

The remainder of the characteristic Keyboard #2 keys are processed in a similar fashion to "MODE" and inappropriate hits are handled exactly the same.

Appropriate Keyboard #1 hits are:

1. "S" for "SMALL 126"
2. "M" for "MEDIUM 63"
3. "L" for "LARGE 31"
The indications of size and maximum number of characters per line are displayed on the status tube. In addition and in conjunction with the selection for the margins the actual number of characters per line ("CHAR/LINE XXX") is calculated and displayed on the status tube.

(c) "TUBE"

Appropriate Keyboard #1 hits are:
1. "R" for "RIGHT TUBE".
2. "L" for "LEFT TUBE".

It is well to mention that all of the characteristics selection displays are stored in a separate section from PICBUFF and therefore are not included in any dump or decoding of PICBUFF.

(d) "INT" (intensity)

Appropriate Keyboard #1 hits are:
1. "N" for "NORMAL".
2. "B" for "BRIGHT".

(e) "INCR" (increment).

1. "R" for "INCR RIGHT".
2. "D" for "INCR DOWN".

(f) "SPACING"

Appropriate Keyboard #1 hits are:
1. "S" for "SINGLE" spacing.
2. "D" for "DOUBLE" spacing.
2. "T" for "TRIPLE" spacing.

(g) "PIP"

The "PIP" key is an on/off switch. When
on the "PIP" key is lighted an "X" appears on the typing tube in the position of the pip. The pip's position is controlled by the four keys "UP", "RIGHT", "DOWN" and "LEFT" which are called direction keys and operate as follows: If the direction key hit is not the same as the last direction key hit, the pip is not moved, but if the direction key is the same the pip is moved in the direction indicated by an amount equal to one space more than it moved the last time. For example if the "UP" key was hit and then the "RIGHT" key was hit ten times in succession, the pip would move as follows:

"RIGHT" hits

Spaces moved to the right
1st 2nd 3rd 4th 5th 6th 7th 8th 9th 10th
0 1 2 3 4 5 6 7 8 9

Relative pip position
0 1 3 6 10 15 21 28 36 45

The pip has end around action, that is if one continues to hit the "RIGHT" key after the pip is on the right extreme of the tube, the pip will appear on the extreme left of the tube.

(h) "LEFT MARGIN".

Appropriate Keyboard #1 hits.

1. "P" for "PIP" where the left margin is taken from the horizontal position of the pip.

2. "+" or a "-" followed by two octal numbers where the left margin is taken from the number given considering the tube center to be 00 and to extend to
77 (oct) spaces in either direction. An indication of the margins is displayed on the typing tube by an upward pointing arrow and the octal position of the arrow displayed underneath.

(i) "RIGHT MARGIN" (same as left margin).

(j) "TAB SET".

This hit does not wait for a Keyboard #1 hit but uses the horizontal position of the pip as the desired tab setting. A tab is displayed as a caret on the typing tube (top). There are 16 tabs available and until all 16 are used elsewhere a tab will always be set at the right extreme of the tube.

(k) "TAB CLEAR"

The pip is again used as a position reference. If no tab is in the position indicated, the "TAB CLEAR" key will blank thrice and the program will return to idle.

(l) "MARKER"

The marker indicates where the next character will be placed or where the end of the last vector is (an aid in using blank vectors) as appropriate. The particular symbol to be used as a marker is determined by the Keyboard #1 key (including the space bar) immediately after the "MARKER" key is hit.

(m) "START"

The "START" key initially locates the marker and operates as follows:
1. "T" - Start at the top of the tube and at the left margin.

2. "P" - Start at the position indicated by the pip.

3. "Y" - Start at the left margin and vertically as indicated by the pip.

(n) "TRACK BALL"

The "TRACK BALL" key is an alternate way to position the PIP.

(o) General

All of the preceding keys affect the character mode but in the vector mode all indications are ignored except Mode, Tube, Marker and Start and in the spot only Mode, Tube, Pip, and Start are recognized. However, one may switch from mode to mode without reselecting. For example if Size is set for Large and one then transfers to Vector mode using small vectors, a transfer back to character Mode will result in using Large size characters unless altered.

The light associated with a key will remain on as long as additional information is required from Keyboard #1. A blinking light indicates that the function of the key could not be completed and when the blinking stops control is returned to the executive program.

(2) Set to type

The "TYPE" key activates Keyboard #1 and until another Keyboard #2 key is hit the Keyboard #1 keys will
operate as previously selected.

(a) In the "CHARACTER" mode normal typing procedures are used and, in addition, an automatic carriage return will result when the last character or space which will fit between the margins has been typed. The caret of the marker will always indicate where the next character will go. Tabs operate in a normal fashion except tabing past the left margin will result in an automatic carriage return.

(b) In the "VECTOR" mode only part of the keyboard is activated, that is the numerals 0 through 9 and the letters "S", "M", "L", and "E". The letters and "0" and "5" do not cause a vector to be formed but modify subsequent vectors as follows:

2. "M" - A large vector.
3. "L" - Two large vectors.
4. "E" - Four large vectors.
5. "5" - Unblank vectors.
6. "0" - Blank vectors.

The remaining numerals from vectors and their direction can be determined by the number key's relative position from the "5" key. They are as follows:

1. "1" - Southwest (The top of the tube is North
2. "2" - South
3. "3" - Southeast
4. "4" - West
5. "6" - East
6. "7" - Northwest
7. "8" - North
8. "9" - Northeast

(c) In the "SPOT" mode Keyboard #1 is not activated. While in this mode, a spot is generated each time the "TYPE" key is hit and the position of the spot is taken from the pip.

(2) Edit features
(a) The Keyboard #1 "blank" key acts as a combination backspace and delete.

(b) "CHAR EDIT" locates the character indicated by the pip and will then proceed as follows:

1. If indicated character can not be found the "CHAR EDIT" key will blink and exit the routine.

2. If "D" on Keyboard #1 is hit, the indicated character is deleted and all characters to the right on that line are moved to the left one space.

3. If "R" on Keyboard #1 is hit, the indicated character is deleted and replaced by the next hit on Keyboard #1.

4. If "I" on Keyboard #1 is hit, the next hit on Keyboard #1 is inserted just to the left of the pip. This key operates only in "CHARACTER" mode.

(c) "LINE EDIT" locates the line and character indicated by the pip and proceeds as follows:
1. If the line and character indicated can not be found, the "LINE EDIT" key will blink and exit the routine.

2. If "D" is hit the entire line indicated will be deleted and the following lines moved up to fill in.

2. If "R" is hit the indicated character and all to the right on the indicated line will be deleted. Keyboard #1 hits will then be inserted in sequence starting with the location of the pip and continue until either the end of the line or until "CR" is hit at which time the routine will exit. "TAB" will be ignored.

(d) "RESET" will extinguish all Keyboard #2 lights except its own and end the current string.

(e) "CLEAR" clears PICBUF and prepares it for a refill (starts file). It also clears the displays which are no longer appropriate.

c. Picture generator (PICGEN)

PICGEN is a transcoder which converts the intermediate PICGEN code to the machine code acceptable to the DD 65 display. The PICGEN code is composed of two parts; one of which, the string code, is merely a modified BCD code for transmitting specific characters, vectors or spots. The other code establishes the characteristics of the display. Both codes are in six bit bytes, 8 bytes to a word and words in numerical increasing order starting at address PICBUF+1. The program PICGEN shuffles through PICBUF byte by byte in sequence forming the proper skeletal words for input to the
DD 65 display until the "00" code causes a shift in mode to string processing whence each byte is inserted in the proper location word by word until the "77" code is reached which terminates the string and the original mode of operation is reactivated.

Since all 64 codes in the string code are meaningful there are no error exits in that mode, however, impossible characteristic codes cause an exit and error display on the right tube.

The process is continued until a "76" code is processed at which time an exit from PICGEN is taken. If a "76" code is not processed when PICBUF is exhausted an exit will also occur.

d. Character locator (CHALOC)

CHALOC uses the pip location to find a specific character in PICBUF and relay the information back to the calling program. If no character is found at the designated location, a -1 is inserted in the A-register. This routine could be expanded to trace vectors and spots but at present this cannot be handled nor can the vertical incrementation be processed.
6. Conclusion.

A time sharing, remote monitor and display system is feasible with equipment presently available at the U. S. Naval Postgraduate School.

The use of a DD 65 Display Unit in conjunction with a CDC 1604 computer with a FORTRAN 60 compiler permits a user to monitor and view his program and its input and output with greater ease and at the same time not interfere or delay appreciably with the normal operations on the main console.

An intermediate code or language of some sort is necessary to allow compatibility of various input and output equipment. Its form will be determined by the specific use to be made of the system and the equipment peculiarities, however, the more general code will allow easier additions and modifications.
INTERRUPT HANDLER

Cell 7Li

Set interrupt disable

Is this an overflow fault interrupt?

Process and clear interrupt

IDLE

N

All normal interrupts are handled in a similar manner.

KEYBOARD INTERRUPT HANDLER

Log unit making request in the bottom of the Keyboard Priority List.

"Program Running Flag" is:

OFF

Enable interrupt

00001

ON

Log cell 7th in top of Program Priority List.

Set "Program Running Flag" to "OFF".

Enable interrupt

IDLE

"Program Running Flag" is:

Enable interrupt

Fig 1

31
IDLE LOOP

Set "Program Running Flag" to "OFF".

Is the Keyboard Priority List (KPL) empty?

Is the Program Priority List (PPL) empty?

Set KEYBOARD DECODER to input from top of KPL.

Delete top item from KPL and close up list.

KEYBOARD DECODER

Does this key activate a program run?

KEYBOARD PROCESSOR

Enter unit and program location at bottom of PPL.

Starting address of top program in PPL → N

Delete top item in PPL and close up list.

Set "Program Running Flag" to "ON".

Jump to N

Fig 2
Enter the branch previously selected and perform the indicated subroutine.

Enter table and jump to the indicated subroutine.

THE NEXT HIT SHOULD COME FROM KB:

Q ← φ

Q ← 2

EXIT
MODE

Φ1A → Route next byte to "φic" → ΦEXT

Φ1C

1EXT

other

Keyboard
#1 character
is a ??

C

V

s

Φ1S

SPOT CODE
→ A

PAKBUF

Display
"SPOT"

Log spot.

Φ1C

CHARACTER
CODE → A.

PAKBUF

Display
"CHARACTER"

Log char.

Φ1V

VECTOR
CODE → A.

PAKBUF

Display
"VECTOR"

Log vector

Φ1E

"MODE"
light off.

2EXT
Process according to the input byte and set the appropriate switches.

Set switch to "1DG."

Read next byte.

Jump to:

1ST
PRIMER
3ST

1RN

1BR

1CG

1ST

SPOT

3ST

End string
PACK

START

1.PAK

A + W → A

2.PAK

B^T: 0 ≠

E^n-2 = B^T

1.PAK

Process- ing n:

A_{n-1} → A_n
Char + DW → A

AG_{n+1} → AG_{n+6}
A_n → A_{n+6}
AG_{n+1} → AG_{n+6}
Char + SW → A

6 + B^T

1 + B^T → B^T

A → W

W + DD65 mem.

O → W

3.PAK → EXIT

38
MACHINE DD65A
LIB(KBORD=KB)
CON(MASK = 7777 7777 7777 7700B)

* PREP KEYBORD.
  
  **T QT**
  SLJ(N)           ENQ(2)          LOAD FLAG.
  EXF(7712B)      EXF(77111B)     RELEASE AND REMOVE INT. SELECTS.
  EXF(7712B)      SLJ(L+2)        IF KB1 HIT, EXIT.
  EXF(77140B)     INT(INWD)       SEL KB1.
  EXF(77175B)     SLJ(L+2)        IF KB2 HIT, EXIT.
  EXF(77120B)     INT(INWD)       SEL KB2.
  ENA(34B)        SLJ4(KYB)       CLEAR.

* INPUT CYCLE.
  
  **1ST**
  EXF(77175B)     SLJ(2ST)        KB2 HIT, EXIT.
  EXF(77120B)     INT(INWD)       SELECT AND INPUT FROM KB2.
  QJP1(L+2)       ENQ(2)          IF KB2 NOT UP; RESET.
  ENA(55B)        SLJ4(KYB)       LOAD INPUT.
  LDA(INWD)       SCL(MASK)       SET FLAG AND ENTER KBORD.
  ENQ(2)          SLJ4(KYB)       KB1 HIT, EXIT.
  
  **2ST**
  EXF(77172B)     SLJ(1ST)        SELECT KB1 FOR INPUT.
  INT(INWD)       QJP1(1ST)       KB1 HIT NOT PROPER, JUMP.
  LDA(INWD)       SLJ4(KYB)       LOAD AND ENTER KBORD.
  SLJ(1ST)        SLJ4(KYB)       RECYCLE.

END

MACHINE KBORD
LOC(RST=5000)
LIB(CHALOC=C1, PICGEN=PF, FLASH=FLASH)
RSV(PICBUF = 1000B).
CON(CH1 =4557 6370 1670 1220B, CHA2 =6151 6163 2365 00513, 
  VCT1 =4557 2565 1670 1220B, VCT2 =6323 4651 2020 00203,
  SPT1 =4557 2247 1670 1220B, SPT2 =4623 2020 2020 0020B,
  SML1 =4557 2244 1671 1160B, SML2 =6143 4320 0102 0006B,
  MEDI =4557 4465 1674 1160B, MEDI2 =6471 2444 2006 00033,
**DECODER**

1ST SLI\(\text{N}) = QJP1(L+2)

1STP SLI\(\text{N}) = EXF(77741B) = IIA(-1)

LRS(3) = SAU(L+1)

ENII(N) = LLS(3)

IJP1(L+1) = SLI\(\text{L}+2)

INA(-3) = SLI\(\text{L}-1)

**K82 INPUT, JUMP.**

**K81 BRANCH POINT.**

**RESET OFF.**

**SHORTEN TABLE BY EXTRACTING**

**DON'T CARES.**

MAK

MAK

MAK

MAK

MAK

MAK

MAK
INA(1TBL)
SLJ(N)
1TBL

EXF(7720A)
EXF(7721A)
EXF(7722A)
EXF(7723A)
EXF(7724A)
EXF(7732A)
EXF(7733A)
EXF(7734A)
EXF(7740A)
EXF(7741A)
EXF(7742A)
EXF(7742A)
EXF(7750A)
EXF(7760A)
SLJ(42A)
EXF(7761B)
EXF(7762B)
EXF(7770A)
EXF(7770B)
EXF(7771B)
EXF(7772B)
EXF(7774B)
SLJ(61A)
SLJ(62A)
SLJ(63A)
SLJ(64A)

* EXITS
OEXT ENQ(0)
1EXT ENQ(1)

EXF(7701B)
ENI3(2)

EXF(7701B)
OUT(ERR1B)
EXF(7701B)
OUT(ERR1C)
EXF(7701B)
ENI3(2)

SAU(L+1)

ENTER TABLE.
LIGHT MODE.
SIZE.
TUBE.
INTENSITY.
INCREMENT.
LEFT MARGIN.
RIGHT MARGIN.
TAB SET.
TAB RESET.
SPACING.

TYPE.
START.
MARKER.
CLEAR.
TEST LIGHTS.
PIP.
TRACK BALL.

RESTART MACHINE DD65A.

LINE EDIT.
RESET.
UP.
RIGHT.
DOWN.
LEFT.

SET FLAGIO = 0 (KB1).
KEYBOARD 1 ERROR EXIT.
SEL AND WAIT FOR DD65 MEM.

SEL AND WAIT FOR DD65 MEM.
SEL AND WAIT FOR DD65 MEM.
SEL AND WAIT FOR DD65 MEM.

SEL AND WAIT FOR DD65 MEM.
<table>
<thead>
<tr>
<th>Line</th>
<th>Instruction</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>01A</td>
<td>ENA(01C)</td>
<td>JCP2(1EXT+1) SLJ(0EXT) SET FLAG10 = 2 (KB2).</td>
</tr>
<tr>
<td>01AE</td>
<td>SAU(1STP)</td>
<td>SLJ(0EXT) ROUTE NEXT HIT.</td>
</tr>
<tr>
<td>01C</td>
<td>EN16(1)</td>
<td>SLJ(1ST) NO PICT.</td>
</tr>
<tr>
<td></td>
<td>AJP(01CH)</td>
<td>INA(-63B) CHARACTER.</td>
</tr>
<tr>
<td></td>
<td>AJP(01V)</td>
<td>INA(36B) VECTOR.</td>
</tr>
<tr>
<td></td>
<td>AJP(01S)</td>
<td>INA(3EXT) SPOT.</td>
</tr>
<tr>
<td>01CH</td>
<td>ENA(10B)</td>
<td>SLJ4(1PB) SEL AND WAIT FOR DD65 MEM.</td>
</tr>
<tr>
<td></td>
<td>EXF(77010B)</td>
<td>EXF7(77010B)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>OUT(CHA)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SEL AND WAIT FOR DD65 MEM.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MODE = 0-CA, 1-VCT, 2-SPOT.</td>
</tr>
<tr>
<td>01E</td>
<td>STA(MODE)</td>
<td>SLJ(2EXT) EXIT MODE.</td>
</tr>
<tr>
<td></td>
<td>SLJ4(1PB)</td>
<td>EXF(77020B)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SEL AND WAIT FOR DD65 MEM.</td>
</tr>
<tr>
<td></td>
<td>ENA(1)</td>
<td>SEL AND WAIT FOR DD65 MEM.</td>
</tr>
<tr>
<td>01V</td>
<td>ENA(11B)</td>
<td>EXF(77010B)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SEL AND WAIT FOR DD65 MEM.</td>
</tr>
<tr>
<td></td>
<td>ENA(1)</td>
<td>SEL AND WAIT FOR DD65 MEM.</td>
</tr>
<tr>
<td>01S</td>
<td>ENA(12B)</td>
<td>SLJ4(1PB)</td>
</tr>
<tr>
<td></td>
<td>EXF(77010B)</td>
<td>EXF7(77010B)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>OUT(SPT)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SEL AND WAIT FOR DD65 MEM.</td>
</tr>
<tr>
<td></td>
<td>ENA(1)</td>
<td>SEL AND WAIT FOR DD65 MEM.</td>
</tr>
<tr>
<td></td>
<td>SLJ(01E)</td>
<td>SLJ4(1PB)</td>
</tr>
<tr>
<td></td>
<td>EXF(77010B)</td>
<td>EXF7(77010B)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>OUT(SPT)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SEL AND WAIT FOR DD65 MEM.</td>
</tr>
<tr>
<td></td>
<td>ENA(2)</td>
<td>SLJ(01E)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>EXF7(77010B)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>OUT(SPT)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SEL AND WAIT FOR DD65 MEM.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>02A</td>
<td>ENA(02C)</td>
<td>SLJ(01AE)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ROUTE NEXT HIT.</td>
</tr>
<tr>
<td>02C</td>
<td>EN16(1)</td>
<td>SLJ(01AE)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SMALL.</td>
</tr>
<tr>
<td></td>
<td>AJP(02S)</td>
<td>INA(-22B)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MEDIUM.</td>
</tr>
<tr>
<td></td>
<td>AJP(02M)</td>
<td>INA(1)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LARGE.</td>
</tr>
<tr>
<td></td>
<td>AJP(02L)</td>
<td>SLJ(3EXT)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SIZE = 4-SMALL, 8-MED, 16-LARGE.</td>
</tr>
<tr>
<td>02S</td>
<td>ENA(30B)</td>
<td>SLJ4(1PB)</td>
</tr>
<tr>
<td></td>
<td>EXF(77010B)</td>
<td>EXF7(77010B)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>OUT(SML)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SEL AND WAIT FOR DD65 MEM.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>02E</td>
<td>STA(SIZE)</td>
<td>ARS(1)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SIZE = 2 = SIZE2.</td>
</tr>
<tr>
<td></td>
<td>STA(SIZE2)</td>
<td>SLJ4(15Z)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RECAL SPACING.</td>
</tr>
<tr>
<td></td>
<td>EXF(77205B)</td>
<td>SLJ4(15Z)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RECAL CHAR/LINE.</td>
</tr>
<tr>
<td></td>
<td>ENA(31B)</td>
<td>SLJ(2EXT)</td>
</tr>
<tr>
<td></td>
<td>EXF(77010B)</td>
<td>EXIT.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SEL AND WAIT FOR DD65 MEM.</td>
</tr>
<tr>
<td></td>
<td>EXF(77010B)</td>
<td>SEL AND WAIT FOR DD65 MEM.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SEL AND WAIT FOR DD65 MEM.</td>
</tr>
<tr>
<td></td>
<td>ENA(10B)</td>
<td>SLJ(02E)</td>
</tr>
</tbody>
</table>
02L ENA(32B) SLJ4(1PB) • SEL AND WAIT FOR DD65 MEM. 02K
  EXF(77010B) EXF7(77010B) • SEL AND WAIT FOR DD65 MEM. 02K
  OUT(LRG1) OUT(LRG2) • 02K
  EXF(77010B) EXF7(77010B) • 02K
  ENA(20B) SLJ0(02E) • 02K

* TUBE - R=RIGHT, L=LEFT.
03A ENA(03C) SLJ(01AE) • ROUTE NEXT HIT. 03K
  EN16(1) INA(-51B) • RIGHT. 03K
  AJP(03R) INA(6) • LEFT. 03K
  AJP(03L) SLJ(1EXT) • NOTA. 03K

03C ENA(03C) SLJ(01AE) • PLACE INDICATORS ON LEFT TUBE. 03K
  EN16(1) INA(-51B) • 03K
  AJP(03R) INA(6) • 03K
  AJP(03L) SLJ(1EXT) • 03K
  ENA(40B) SLJ4(1PB) • 03K
  ENI1(35B) • 03K
  LDA1(CHA1) SCL(MK11) • 03K
  STA1(CHA1) IJP1(L-1) • 03K
  STA1(RM2) • 03K
  LDA1(RM2) SST(MK11) • 03K
  STA1(RM2) • 03K
  EXF(77010B) • 03K
  EXF(77010B) • 03K
  OUT(RHT1) • 03K
  EXF7(77010B) • 03K
  OUT(RHT2) • 03K

03R ENA(40B) SLJ4(1PB) • LIGHT OFF AND EXIT. 03K
  ENI1(35B) • 03K
  LDA1(CHA1) SCL(MK11) • 03K
  STA1(CHA1) IJP1(L-1) • 03K
  STA1(RM2) • 03K
  LDA1(RM2) SST(MK11) • 03K
  STA1(RM2) • 03K
  EXF(77010B) • 03K
  EXF(77010B) • 03K
  OUT(RHT1) • 03K
  EXF7(77010B) • 03K
  OUT(RHT2) • 03K

03L ENA(41B) SLJ4(1PB) • LIGHT OFF AND EXIT. 03K
  ENI1(35B) • 03K
  LDA1(CHA1) SCL(MK11) • 03K
  STA1(CHA1) IJP1(L-1) • 03K
  STA1(RM2) • 03K
  LDA1(RM2) SST(MK11) • 03K
  STA1(RM2) • 03K
  EXF(77010B) • 03K
  EXF(77010B) • 03K
  OUT(RHT1) • 03K
  EXF7(77010B) • 03K
  OUT(RHT2) • 03K

* INTENSITY - N=NORMAL, B=BRIGHT.
03A ENA(04C) SLJ(01AE) • 04K
  EN16(1) INA(-45B) • 04K
  AJP(04N) INA(-15B) • 04K
  AJP(046) SLJ(1EXT) • 04K
  LDA1(CHA1) SCL(MK11) • 04K
  STA1(CHA1) IJP1(L-1) • 04K
  STA1(RM2) • 04K
  LDA1(RM2) SST(MK11) • 04K
  STA1(RM2) • 04K
  EXF(77010B) • 04K
  EXF(77010B) • 04K
  OUT(NOR1) • 04K
  EXF7(77010B) • 04K
  OUT(NOR2) • 04K
  EXF(77010B) • 04K
  EXF7(77010B) • 04K
  OUT(BRT1) • 04K

04A ENA(04C) SLJ(01AE) • ROUTE NEXT HIT. 04K
  EN16(1) INA(-45B) • NORMAL. 04K
  AJP(04N) INA(-15B) • BRIGHT. 04K
  AJP(046) SLJ(1EXT) • NOTA. 04K
  LDA1(CHA1) SCL(MK11) • 04K
  STA1(CHA1) IJP1(L-1) • 04K
  STA1(RM2) • 04K
  LDA1(RM2) SST(MK11) • 04K
  STA1(RM2) • 04K
  EXF(77010B) • 04K
  EXF(77010B) • 04K
  OUT(NOR1) • 04K
  EXF7(77010B) • 04K
  OUT(NOR2) • 04K
  EXF(77010B) • 04K
  EXF7(77010B) • 04K
  OUT(BRT1) • 04K

04C ENA(04C) SLJ(01AE) • SEL AND WAIT FOR DD65 MEM. 04K
  EN16(1) INA(-45B) • SEL AND WAIT FOR DD65 MEM. 04K
  AJP(04N) INA(-15B) • 04K
  AJP(046) SLJ(1EXT) • 04K
  LDA1(CHA1) SCL(MK11) • 04K
  STA1(CHA1) IJP1(L-1) • 04K
  STA1(RM2) • 04K
  LDA1(RM2) SST(MK11) • 04K
  STA1(RM2) • 04K
  EXF(77010B) • 04K
  EXF(77010B) • 04K
  OUT(NOR1) • 04K
  EXF7(77010B) • 04K
  OUT(NOR2) • 04K
  EXF(77010B) • 04K
  EXF7(77010B) • 04K
  OUT(BRT1) • 04K

04N ENA(50B) SLJ4(1PB) • INTEENSITY OFF. 04K
  EXF(77010B) EXF7(77010B) • SEL AND WAIT FOR DD65 MEM. 04K
  OUT(NOR1) OUT(NOR2) • 04K
  EXF(7721B) SLJ(2EXT) • 04K
  ENA(51B) SLJ4(1PB) • 04K
  EXF(77010B) EXF7(77010B) • SEL AND WAIT FOR DD65 MEM. 04K
  OUT(BRT1) OUT(BRT2) • 04K

04G ENA(51B) SLJ4(1PB) • 04K
  EXF(77010B) EXF7(77010B) • 04K
  OUT(BRT1) OUT(BRT2) • 04K

43
EXF(77010B)  EXF(77010B)  . SEL AND WAIT FOR DD65 MEM.  04K
EXF(77221B)  OUT(BAT2)  . INTENSITY OFF.  04K
* INCREMENT - R=RIGHT, D=DOWN.
C5A ENA(05C)  SLJ(01AE)  . ROUTE NEXT HIT.  05K
C5C ENA(16)  INA(-51B)  . RIGHT.  05K
C5C AJP(05R)  INA(-13B)  . DOWN.  05K
C5C AJP(05D)  SLJ(1EXT)  . NOTA.  05K
C5R ENA(60B)  EXF(77010B)  . SEL AND WAIT FOR DD65 MEM.  05K
EXF(77010B)  EXF(77010B)  . SEL AND WAIT FOR DD65 MEM.  05K
OUT(RIT2)  ENA(-1)  . INCR = -1 = RIGHT.  05K
C5E STA(INCR)  EXF(77241B)  . EXIT.  05K
C5E SLJ(2EXT)  .  05K
C5D ENA(61B)  SLJ(4(1PB)  SEL AND WAIT FOR DD65 MEM.  05K
EXF(77010B)  EXF(77010B)  SEL AND WAIT FOR DD65 MEM.  05K
OUT(RIT1)  .  05K
EXF(77010B)  .  05K
C5D ENA(61B)  .  05K
C5D ENA(1)  SLJ(05E)  .  05K
* LEFT MARGIN = P-USE PIPX FOR INPUT, + OR - AND 2 OCT DIGITS.
11A ENA(11C)  SLJ(01AE)  PIP.  11K
11C INA(-47B)  AJP(11PI)  NOT - SIGN, JUMP.  11K
11C INA(7)  AJP(1(L+2)  -  11K
11C ENA(11N)  SLJ(01AE)  NOT + SIGN, ERROR EXIT.  11K
11C INA(-20B)  AJP(1(EXT)  11K
11C ENA(11P)  SLJ(01AE)  11K
11P LDA(PIP)  ARS(2)  11K
11N ALS(2)  SLJ(11E)  PIPPER.  11K
11N SCM(SEV)  SLJ(11E)  USE ONLY POSITS / BY 4.  11K
11P SLJ(4(RET)  11K
11E STA(LMAR)  SLJ(4(1MAR)  NIG. INPUT.  11K
11E EXF(77303B)  SLJ(2EXT)  POS. INPUT.  12K
* RIGHT MARGIN (SEE LEFT MARGIN).
12A ENA(12C)  SLJ(01AE)  12K
12C INA(-47B)  AJP(12PI)  12K
12C INA(7)  AJP(1(L+2)  12K
12C ENA(12N)  SLJ(01AE)  12K
12C INA(-20B)  AJP(1(EXT)  12K
12C ENA(12P)  SLJ(01AE)  12K
12P LDA(PIP)  ARS(2)  12K
12N ALS(2)  SLJ(12E)  PIP.  12K
12N SLJ(4(1RET)  USE ONLY POSITS / BY 4.  12K
12N SCM(SEV)  SLJ(12E)  12K
12P SLJ(4(1RET)  12K
12E STA(RMAR)  SLJ(4(1MAX)  12K
12E EXF(77305B)  SLJ(2EXT)  EXIT.  12K

44
SLJ(3IC1)
* MAIN CHARACTER LOOP.

31CA AJP(18KS)
AJP(3ICT)
AJP(31CR)
AJP(10EXT)
EN16(0)

31CT EN11(0)
LDA(INCR)
LD1(1TL)
SUB1(XLOC)
IS1(17B)
SLJ(1EXT)
SUB3(SIZE)
EN3(20B)

31CT1 LDA(YLOC)
SUB1 (TL1)
IS1 (17B)
SLJ(1EXT)
AJP(3ICT7)

31CM SLJ(N)
LDQ(INCR)
RAD(XLOC)
AJP3(3ICM)
RSB(YLOC)
STA(XLOC)
EN16(2)
EN1(79B)

31CM1 RSB(YLOC)
AJP2(3ICM)
STA(YLOC)
ADD1(XLOC)

31CR LAC(INCR)
AJP3(3ICM+1)

* VECTOR - S=SMALL, M=MEDIUM, L=LARGE, E=EXTRA LARGE, O=BLANK, 5=UNBLANK
9=NE, 6=E, 3=SE, 2=S, 1=SW, 4=W, 7=NW, 8=NORTH.

31VA AJP(1EXT)
AJP(31VU)
AJP(31VB)
AJP(31VS)
AJP(31VL)

BACKSPACE, JUMP.
TAB, JUMP.
CR, JUMP.
SEMICOLON NOT ALLOWED, RESTORE CODE.
PRINT CHARACTER.
CAL MAR AND MARKER.
MARKER.
EXIT.
TAB.
TAB DOWN, JUMP.
TAB TOO FAR TO LEFT, JUMP.
ERROR EXIT.
JUST RIGHT, EXIT.
LOAD BLANK, TRY AGAIN.
PROCESS.
TAB DOWN.
ERROR EXIT.
INCR DOWN, JUMP.
INSIDE MAR, JUMP.
START NEW LINE.
CR EXIT.
INSIDE MAR, JUMP.
CR.
0, EXIT.
UNBLANK, JUMP.
BLANK.
SMALL.
LARGE.
<table>
<thead>
<tr>
<th>Instruction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AJP(31VM)</td>
<td>INA(-21B)</td>
</tr>
<tr>
<td>AJP(31XV)</td>
<td>INA(53B)</td>
</tr>
<tr>
<td>AJP2(1EXT)</td>
<td>INA(12B)</td>
</tr>
<tr>
<td>INA(L+1)</td>
<td>SAU(L+1)</td>
</tr>
<tr>
<td>SLJ(N)</td>
<td>SLJ(31VP)</td>
</tr>
<tr>
<td>SLJ(N)</td>
<td>SLJ(31VP)</td>
</tr>
<tr>
<td>SLJ(N)</td>
<td>SLJ(31VP)</td>
</tr>
<tr>
<td>SLJ(N)</td>
<td>SLJ(31VP)</td>
</tr>
<tr>
<td>SLJ(N)</td>
<td>SLJ(31VP)</td>
</tr>
<tr>
<td>SLJ(N)</td>
<td>SLJ(31VP)</td>
</tr>
<tr>
<td>SLJ(N)</td>
<td>SLJ(31VP)</td>
</tr>
<tr>
<td>SLJ(N)</td>
<td>SLJ(31VP)</td>
</tr>
</tbody>
</table>

*PRINT VECTOR*

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31VP STA(T)</td>
<td>STA(T) TEMP</td>
</tr>
<tr>
<td>INA(-20B)</td>
<td>AJP(L+4)</td>
</tr>
<tr>
<td>AJP2(L+7)</td>
<td>ENI6(0)</td>
</tr>
<tr>
<td>LDA(T) TEMP</td>
<td>SLJ4(1PB)</td>
</tr>
<tr>
<td>SLJ(31V9)</td>
<td>ENI6(1)</td>
</tr>
<tr>
<td>LDA(T) TEMP</td>
<td>SLJ4(1PB)</td>
</tr>
<tr>
<td>SLJ(31V9)</td>
<td>SLJ4(1PB)</td>
</tr>
<tr>
<td>SLJ(31V9)</td>
<td>SLJ4(1PB)</td>
</tr>
<tr>
<td>SLJ(31V9)</td>
<td>SLJ4(1PB)</td>
</tr>
<tr>
<td>SLJ(31V9)</td>
<td>SLJ4(1PB)</td>
</tr>
</tbody>
</table>

*TRACK VECTOR WITH MARKER*

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31V9 LDA(T)</td>
<td>INA(-5)</td>
</tr>
<tr>
<td>AJP6(31MY)</td>
<td>AJP(L+4)</td>
</tr>
<tr>
<td>AJP6(31PX)</td>
<td>AJP(L+3)</td>
</tr>
<tr>
<td>AJP7(31MK)</td>
<td>INA(-3)</td>
</tr>
<tr>
<td>AJP7(31PV)</td>
<td>AJP(L+2)</td>
</tr>
<tr>
<td>AJP7(31MK)</td>
<td>INA(-6)</td>
</tr>
<tr>
<td>AJP7(31MK)</td>
<td>AJP(L+2)</td>
</tr>
<tr>
<td>AJP7(31MK)</td>
<td>AJP(L+2)</td>
</tr>
<tr>
<td>AJP7(31MK)</td>
<td>SLJ4(31PX)</td>
</tr>
<tr>
<td>SLJ4(31PX)</td>
<td>SLJ4(31PX)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31VE SLJ(10EXT)</td>
<td>31VE SLJ(N)</td>
</tr>
<tr>
<td>SLJ(N)</td>
<td>LDA(VSIZE)</td>
</tr>
<tr>
<td>LDA(VSIZE)</td>
<td>SLJ(L-1)</td>
</tr>
<tr>
<td>SLJ(L-1)</td>
<td>LDA(VSIZE)</td>
</tr>
<tr>
<td>LDA(VSIZE)</td>
<td>SLJ(L-1)</td>
</tr>
<tr>
<td>SLJ(L-1)</td>
<td>+X</td>
</tr>
<tr>
<td>SLJ(L-1)</td>
<td>-X</td>
</tr>
</tbody>
</table>

47
31VY SLJ(N)  LDA(VSIZE)  +Y
31MV SLJ(N)  SLJ(L-1)  -Y
31VX SLJ(N)  LDA(VSIZE)  
* MODIFY VECTOR.
31Vu LDA(31VT)  SST(MK6)  UNBLANK.
31VB LDA(31VT)  SLJ(0EXT)  BLANK.
31VS LDA(31VT)  SCL(MK5)  SMALL.
31Vt SCL(MK4)  ENQ(14)  EXIT.
31Vh STQ(VSIZE)  SLJ(0EXT)  MEDIUM.
31Vl ENQ(40B)  SLJ(31VT)  LARGE.
31Vx ENQ(40B)  SLJ(31VT)  EXTRA LARGE.

31A ENI6(4)  SLJ4(1PB)  PRINT WHEN SPOT COMPLETE, START
31B LDA(PIPX)  ARS(3)  STRING.
31Ca LDA(PIPY)  SLJ4(1PB)  1ST 6 BITS.
31Cb LDA(PIPY)  LRS(11B)  
31Cc LDA(PIPY)  LLS(3)  
31Cd LDA(PIPI)  SLJ4(1PB)  2ND 6 BITS.
31Ce ENI6(10)  SLJ4(1PB)  3RD 6 BITS.
31Cf ENA(77B)  SLJ4(1PB)  PRINT.
31Cg EXF(77503B)  SLJ(2EXT)  END STRING.
31Ch EXF(77503B)  SLJ(2EXT)  FULL EXIT.

* START - PLACE THE NEXT CHAR. (XLOC, YLOC) AS INDICATED.
* T - TOP OF PAGE AT LEFT MARGIN, P - AT LOCATION OF PIPE.
* OR Y - AT PIPPER Y COORD. AND LEFT MARGIN.

32A ENA(32C)  SLJ(01AE)  T.
32C INA(-23B)  AJP(32T)  Y.
32D INA(-5)  AJP(32Y)  P.
32E INA(17B)  AJP(32P)  NDTA.
32F SLJ(1EXT)  SUB(SIZE2)  START AT TOP.
32G ENA(376B)  STA(YLOC)  
32H STA(YLOC)  STA(32H)  XLOC.
32I LDA(3LMAR)  STA(XLOC)  POSITION MARKER.
32J EXF(77503B)  SLJ4(33MK)  POSITION PICBUF.
32K EXF(77503B)  SLJ4(33MK)  START OFF.
32L LDA(PIPY)  ARS(2)  YLOC BY PIPE.
32M ALS(2)  STA(YLOC)  
32N SLJ(32T1)  SLJ(32T1)  XLOC AND YLOC BY PIPE.
LIU2(HERE) LDA1(PICBUF) CLEAR CHAR.
         SCL[8K77] INSERT NEW CHAR.
LRS2(0) LLS2(0) INSERT TEMP.
         ADD(TMP)
STA1(PICBUF) SAL{PF+1) DO PICGEN.
        SLJ4{2EXT} CHAR EDIT OFF.
44EX ENA1(PICBUF) SLJ{PF)
EXF(7721B) SLJ{2EXT)
* DELETE AND CLOSE UP.
44D LILI(HERE) LIU2(HERE) BAD BYTE IN A{RJ). BYTE A{RJ) IS EMPTY.
         LDA1(PICBUF) LRS2(0) DONT PROCESS MORE THAN NEEDED.
         ARS(6) LLS2{6) PASS EMPTY BYTE THRU PICBUF.
         STA1(PICBUF) LDQ1(PICBUF+1) UPDATE POS.
         LDA1(PICBUF) LDQ1{PICBUF+1) UPDATE POS.
         RSA(6) LLL{6) EXIT.
         STA1(PICBUF) STQ1(PICBUF+1) EXIT.
         KASL{11) LLL{11-3) EXIT.
         LII{POSIT) LNAI{603) EXIT.
         AJP1{L+2) RSAI{POSIT) EXIT.
         ENI{6) SII{POSIT) EXIT.
         IN1{6) EXIT.
SLJ{44EX) EXIT.
* INSERT NEXT KB1 CHAR
44I ENA1{4411) SLJ{01AE) ROUTE.
         SLJ{01AE) B2=8BIT WHICH DIVIDES OLD AND NEW.
         STA1(T) LIU2{HERE) EXIT.
         LIU2{HERE) LDA1(PICBUF) EXIT.
         LDA1(PICBUF) LDA1{PICBUF) EXIT.
         LRS2{16) AL{6) EXIT.
         SCL[8K77) ADD(TMP)
         LLS2{0) STA1(PICBUF) EXIT.
         STA1(PICBUF) INIAI{1) EXIT.
         LDA1(PICBUF) LDA1{PICBUF) EXIT.
         LLS{6) LDQ1{PICBUF) EXIT.
         LLS{52B) STA1(PICBUF) EXIT.
         ISK1{N) SLJ{IL-3) EXIT.
         LII{POSIT) ENA2{6) EXIT.
         AJP1{L+2) ENI{6) EXIT.
         RAOI{POSIT) ENI2{608) EXIT.
         ENI2{608) EXIT.
         ENI2(608) EXIT.
SLJ{44EX) EXIT.
* LINE EDIT - R = DELETE CHAR ON PIP AND ALL TO THE RIGHT. ANY STRING OF
         KBI CHAR CAN FOLLOW (EXCEPT TAB OR CR) UNTILL END OF
         STRING IS REACHED, WENCE AN AUTO EXIT OCCURES. TAB IS IGNORED AND
         CR CAUSES AN INMEDIATE EXIT. FLASHING KEY = LINE CAN'T BE FOUND.
         D = DELETE AND CLOSE UP. SILF EXIT.
         I = INSERT LINE (NOT PROGRAMED).
         54A ENA{54C) SAI{LSTP) ROUTE NEXT HIT.
         ENA{PICBUF) SAL{C1+1) ROUTE NEXT HIT.
         ENA{PICBUF) SAL{C1+2) ROUTE NEXT HIT.
         ENA{POSIT) SAL{C1+3) ROUTE NEXT HIT.
ENAI(P1PX)
ENAI(P1PY)
AJP2(0EXT)
EXEC(77720B)
EXEC(77721B)
IJP2(L-2)
INAI(-51B)
INAI(-13B)
SLJ1(EXT)
# REPLACE
54A
ENAI(L+1)
LILI(HERE)
AJP(0EXT)
AJP(0EXT)
AJP(54RX)
STA(T)
LRS2(6)
LLS(6)
LDA(T)
LDA(PICBUF)
LSS2(6)
ENAI(PICBUF)
IJP2(L+2)
RAO(HERE)
IN12(-5)
SLJ(0EXT)
54RX
EXEC(77721B)
# DELETE AND CLOSE UP
54D
LILI(HERE)
LDA1(PICBUF)
ENQ(77777B)
ENI3(7)
ENQ(0)
QJP(54DF)
IJPI(L+1)
LDA1(PICBUF)
54DF
ENI4(108)
IN14(-1)
LDQ1(PICBUF)
LDA4(KO)
AJP(54DA)
SST4(KO)
IJP4(L-3)
TSK1(100CR)
# MOVE EACH LINE UP ONE.
54DA
LIU2(HERE)
LDA1(PICBUF)
LRS2(0)

0
- DO CHALOC.
- NOT HERE, BLINK KEY.
- ON.
- OFF.
- DELETE AND REPLACE.
- DELETE AND CLOSE UP.
- NOTA.
- ROUTE NEXT HIT BELOW.
- RELOAD INDIGES.
- IGNORE BACKSPACE AND TAB.
- EXIT ON CR.
- OLD BYTE IN Q(LJ).
- END OF STRING, EXIT.
- NEW BYTE IN Q(LJ).
- NEW WORD FORMED AND STORED.
- PRINT.
- NOT LAST BYTE IN WORD, JUMP.
- CALL NEXT WORD.
- STORE INDIGES.
- LIGHT OFF AND EXIT.
- FIND THE START OF THE STRING(00).
- X BYTE IN Q(LJ).
- X BYTE AND ALL TO THE RIGHT = 77.
- ***
- *** EXAM BYTES FROM RT TO LF.
- *** 00 FOUND, JUMP.
- *** PICBUF EXAUERED, JUMP.
- ***
- B4 = 7 - B3.
- REPLACE ALL BYTES IN THE
- *** STRING WITH 77.
- *** END OF STRING, JUMP.
- ***
- ***
- ***
- ***
STQ(W)          ENA2(0)          .     54K
ENQ(0)          SAU(L+1)         .     54K
DVI(M6)         LDA(PNYP)         .     54K
ENI2(N)         AJP2(N+2)         .     54K
ARS(2)          AJP2(N+2)         .     54K
SCM(5EV)        INA(200B)         .     54K
INA(2100B)      STA(PNY)          .     54K
STA(TNY)        .     54K
54D1 ENA(54DC)  .     54K
SAU(54DB)       .     54K
* READ NEXT BYTE. 54DR IJP2(N+4) ENI2(7) .     54K
                SKI(54D0B) SLJ(54DL+2) .     54K
                SLJ(54DE) LDA(-picbuf) .     54K
                LDQ(W)   STA(W) .     54K
                LLS(6)   STQ(W) .     54K
* BRANCH POINT. 54DB SLJ(N) .     54K
* SKIP STRING. 54DS INA(-77B) AJP1(54DR) .     54K
                SLJ(54D1) .     54K
* DECODER. 54DC AJP(54DO) INA(-20B) .     54K
                AJP3(54DR) INA(-4) .     54K
                AJP2(N+2) LRS(1) .     54K
                QJP3(54DX) SLJ(54DY) .     54K
                INA(-53B) AJP1(54DR) .     54K
                SLJ(54DR+2) .     54K
* PROCESS BEGIN STRING. 54DO LDA(TNY) STA(PNY) .     54K
                ENA(54DS) SLJ(54D1)+1 .     54K
* PROCESS X LOCATION. 54DX ENA(L+1) SLJ(54D1)+1 .     54K
                SLJ(54D1) .     54K
* PROCESS Y LOCATION. 54DY LLS(1) INA(24B) .     54K
                ALS(6) STA(TNY) .     54K
                ENA(L+1) SLJ(54D1)+1 .     54K
                RAD(TNY) ENA2(-7) .     54K
                AJP(L+6) INA(7) .     54K
                MUI(M6) SAU(L+2) .     54K
                SAU(L+2) LDA(PNYP) .     54K
                ENQ(7700B) ALS(N) .     54K
                QLS(N) SSAU(3) .     54K
                STA(3) SLJ(54D1) .     54K
                LDQ(3) SLJ(54D1) .     54K
                LDA(PNYP) LRS(6) .     54K
                STQ(3) LRS(6) .     54K

CAL PERMANENT NEW Y.
PIPY TO PICGEN CODE.
PICBUF EXHAUSTED, EXIT.
NOT END OF STRING, EXIT.
BEGIN STRING, JUMP.
LESS THAN 20, READ NEXT BYTE.
NOT A LOC, JUMP.
20 + 22 TO X, 21 + 23 TO Y.
NOT END OF FILE, EXIT.
FINAL EXIT.
TEMP BECOMES PERM YLOC.
SKIP STRING.
SKIP NEXT BYTE.
RESTORE NORMAL ROUTE.
EXTRACT THIS Y FOR USE ON NEXT LINE.

54K
<table>
<thead>
<tr>
<th>Instruction</th>
<th>Description</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>INI1(-1)</td>
<td></td>
<td>54K</td>
</tr>
<tr>
<td>LDA(PICBUF)</td>
<td></td>
<td>54K</td>
</tr>
<tr>
<td>LLS(6)</td>
<td></td>
<td>54K</td>
</tr>
<tr>
<td>INI1(1)</td>
<td></td>
<td>54K</td>
</tr>
<tr>
<td>540DE ENA(PICBUF)</td>
<td></td>
<td>54K</td>
</tr>
<tr>
<td>EXF(7721B)</td>
<td></td>
<td>54K</td>
</tr>
<tr>
<td>* RESET</td>
<td></td>
<td>55K</td>
</tr>
<tr>
<td>EXF(77203B)</td>
<td></td>
<td>55K</td>
</tr>
<tr>
<td>EXF(77211B)</td>
<td></td>
<td>55K</td>
</tr>
<tr>
<td>EXF(77211B)</td>
<td></td>
<td>55K</td>
</tr>
<tr>
<td>EXF(77241B)</td>
<td></td>
<td>55K</td>
</tr>
<tr>
<td>EXF(77305B)</td>
<td></td>
<td>55K</td>
</tr>
<tr>
<td>EXF(77431B)</td>
<td></td>
<td>55K</td>
</tr>
<tr>
<td>EXF(77403B)</td>
<td></td>
<td>55K</td>
</tr>
<tr>
<td>EXF(77411B)</td>
<td></td>
<td>55K</td>
</tr>
<tr>
<td>EXF(77411B)</td>
<td></td>
<td>55K</td>
</tr>
<tr>
<td>EXF(77411B)</td>
<td></td>
<td>55K</td>
</tr>
<tr>
<td>EXF(77411B)</td>
<td></td>
<td>55K</td>
</tr>
<tr>
<td>EXF(77411B)</td>
<td></td>
<td>55K</td>
</tr>
<tr>
<td>EXF(77411B)</td>
<td></td>
<td>55K</td>
</tr>
<tr>
<td>EXF(77411B)</td>
<td></td>
<td>55K</td>
</tr>
<tr>
<td>EXF(77411B)</td>
<td></td>
<td>55K</td>
</tr>
<tr>
<td>ENA(77B)</td>
<td></td>
<td>55K</td>
</tr>
<tr>
<td>SLJ(2EXT)</td>
<td></td>
<td>55K</td>
</tr>
<tr>
<td>* PIP UP</td>
<td></td>
<td>61K</td>
</tr>
<tr>
<td>LDA(PIPD)</td>
<td></td>
<td>61K</td>
</tr>
<tr>
<td>AJP(L+3)</td>
<td></td>
<td>61K</td>
</tr>
<tr>
<td>STA(PIPD)</td>
<td></td>
<td>61K</td>
</tr>
<tr>
<td>STA(TIM)</td>
<td></td>
<td>61K</td>
</tr>
<tr>
<td>RAO(TIM)</td>
<td></td>
<td>61K</td>
</tr>
<tr>
<td>SLJ(TIPIP)</td>
<td></td>
<td>61K</td>
</tr>
<tr>
<td>* PIP RIGHT</td>
<td></td>
<td>62K</td>
</tr>
<tr>
<td>LDA(PIPD)</td>
<td></td>
<td>62K</td>
</tr>
<tr>
<td>AJP(L+3)</td>
<td></td>
<td>62K</td>
</tr>
<tr>
<td>STA(PIPD)</td>
<td></td>
<td>62K</td>
</tr>
<tr>
<td>STA(TIM)</td>
<td></td>
<td>62K</td>
</tr>
<tr>
<td>RAO(TIM)</td>
<td></td>
<td>62K</td>
</tr>
<tr>
<td>SLJ(TIPIP)</td>
<td></td>
<td>62K</td>
</tr>
<tr>
<td>* PIP DOWN</td>
<td></td>
<td>63K</td>
</tr>
<tr>
<td>LDA(PIPD)</td>
<td></td>
<td>63K</td>
</tr>
<tr>
<td>AJP(L+3)</td>
<td></td>
<td>63K</td>
</tr>
<tr>
<td>STA(PIPD)</td>
<td></td>
<td>63K</td>
</tr>
<tr>
<td>STA(TIM)</td>
<td></td>
<td>63K</td>
</tr>
<tr>
<td>RAO(TIM)</td>
<td></td>
<td>63K</td>
</tr>
<tr>
<td>SLJ(TIPIP)</td>
<td></td>
<td>63K</td>
</tr>
<tr>
<td>* PIP LEFT</td>
<td></td>
<td>64K</td>
</tr>
<tr>
<td>LDA(PIPD)</td>
<td></td>
<td>64K</td>
</tr>
<tr>
<td>AJP(L+3)</td>
<td></td>
<td>64K</td>
</tr>
<tr>
<td>STA(PIPD)</td>
<td></td>
<td>64K</td>
</tr>
</tbody>
</table>

**53**
* INTER-KEY SUBROUTINES.

* PRINT TAB LIST.

 1PTL ENIT(0)
  LDA(TL1)  ALS(44B)
  SSU(1B1)  STA(T)
  EXF(77010B)
  ISK(17B)
  EXF(77311B)
  SLJ(2EXT)

* CAL DELTA.

 15Z SLJ(N)
  MUI(SIZE)
  SLJ(L-2)

* PIP INCREDMENTATION.

 1PIP LDA(PIPX)
  LDA(PIPX)
  LLS(47B)
  QRS(47B)
  STQ(PIPY)
  LLS(44B)
  SSU(PIP)
  EXF(77010B)
  EXF(77174B)
  ENII(1)
  EXF(77175B)
  EXF(77120B)
  SLJ(L-4)

 2PIP EXF(77010B)

* CONVERT A SIGNED 9 BIT RJ NR TO A SIGN + 2 BYTE BCD OCT NR.

 1VRT SLJ(N)
   AJP2(L+2)
   ENQ(4000B)
   STQ(TEMP)
   ENA(0)
   RAD(TEMP)
   LLS(3)
   ENA(128)
   LRS(6)
   LDA(TEMP)

   64K
   64K
   64K

   PTK
   PTK
   PTK
   PTK
   PTK
   PTK
   PTK
   PTK
   PTK
   PTK
   DLK
   DLK
   DLK
   PTK
   PTK
   PTK
   PTK
   PTK
   PTK
   PTK
   PTK
   PTK
   PTK
   PTK

   SRK
   PTK
   PTK
   PTK
   PTK
   PTK
   PTK
   PTK
   PTK
   PTK
   PTK
   PTK
   PTK
   PTK
   PTK
   PTK
   PTK
   PTK
   PTK
   PTK
   PTK
   PTK
   PTK

   VTK
   VTK
   VTK
   VTK
   VTK
   VTK
   VTK
   VTK
   VTK
   VTK
   VTK
   VTK
   VTK
   VTK
   VTK
   VTK
   VTK
<table>
<thead>
<tr>
<th>LLS(6)</th>
<th>SLJ1(VRT)</th>
<th>0</th>
<th>VFK</th>
<th>LKX</th>
</tr>
</thead>
<tbody>
<tr>
<td>1LOC</td>
<td>SLJ(N)</td>
<td></td>
<td>VFK</td>
<td>LKX</td>
</tr>
<tr>
<td></td>
<td>LDA(XLOC)</td>
<td></td>
<td>VFK</td>
<td>LKX</td>
</tr>
<tr>
<td></td>
<td>LLS(47B)</td>
<td></td>
<td>VFK</td>
<td>LKX</td>
</tr>
<tr>
<td></td>
<td>ARS(47B)</td>
<td></td>
<td>VFK</td>
<td>LKX</td>
</tr>
<tr>
<td></td>
<td>STQ(YLOC)</td>
<td></td>
<td>VFK</td>
<td>LKX</td>
</tr>
<tr>
<td></td>
<td>ENA(22B)</td>
<td></td>
<td>VFK</td>
<td>LKX</td>
</tr>
<tr>
<td></td>
<td>LAC(XLOC)</td>
<td></td>
<td>VFK</td>
<td>LKX</td>
</tr>
<tr>
<td></td>
<td>ENA(20B)</td>
<td></td>
<td>VFK</td>
<td>LKX</td>
</tr>
<tr>
<td></td>
<td>LDA(XLOC)</td>
<td></td>
<td>VFK</td>
<td>LKX</td>
</tr>
<tr>
<td></td>
<td>ARS(2)</td>
<td></td>
<td>VFK</td>
<td>LKX</td>
</tr>
<tr>
<td></td>
<td>SLJ(1LOC)</td>
<td></td>
<td>VFK</td>
<td>LKX</td>
</tr>
<tr>
<td>* PLACE XLOC AND YLOC IN PICBUF.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1PB</td>
<td>SLJ(N)</td>
<td></td>
<td>VFK</td>
<td>LKX</td>
</tr>
<tr>
<td></td>
<td>SII1(6PB)</td>
<td></td>
<td>VFK</td>
<td>LKX</td>
</tr>
<tr>
<td></td>
<td>LII2(POS)</td>
<td></td>
<td>VFK</td>
<td>LKX</td>
</tr>
<tr>
<td></td>
<td>LRS(6)</td>
<td></td>
<td>VFK</td>
<td>LKX</td>
</tr>
<tr>
<td></td>
<td>ARS2(0)</td>
<td></td>
<td>VFK</td>
<td>LKX</td>
</tr>
<tr>
<td></td>
<td>STA1(PICBUF)</td>
<td></td>
<td>VFK</td>
<td>LKX</td>
</tr>
<tr>
<td></td>
<td>ENA2(-6)</td>
<td></td>
<td>VFK</td>
<td>LKX</td>
</tr>
<tr>
<td></td>
<td>ISK1(77B)</td>
<td></td>
<td>VFK</td>
<td>LKX</td>
</tr>
<tr>
<td></td>
<td>ENI16(3)</td>
<td></td>
<td>VFK</td>
<td>LKX</td>
</tr>
<tr>
<td></td>
<td>OUT6(FUL)</td>
<td></td>
<td>VFK</td>
<td>LKX</td>
</tr>
<tr>
<td>2PB</td>
<td>ENI2(66B)</td>
<td></td>
<td>VFK</td>
<td>LKX</td>
</tr>
<tr>
<td>4PB</td>
<td>INI2(-6)</td>
<td></td>
<td>VFK</td>
<td>LKX</td>
</tr>
<tr>
<td></td>
<td>ENA(PICBUF)</td>
<td></td>
<td>VFK</td>
<td>LKX</td>
</tr>
<tr>
<td>3PB</td>
<td>SII1(POS)</td>
<td></td>
<td>VFK</td>
<td>LKX</td>
</tr>
<tr>
<td>6PB</td>
<td>ENI2(N)</td>
<td></td>
<td>VFK</td>
<td>LKX</td>
</tr>
<tr>
<td></td>
<td>ENA(0)</td>
<td></td>
<td>VFK</td>
<td>LKX</td>
</tr>
<tr>
<td>* WAIT</td>
<td>SLJ(N)</td>
<td></td>
<td>VFK</td>
<td>LKX</td>
</tr>
<tr>
<td>1WAT</td>
<td>SLJ(N)</td>
<td></td>
<td>VFK</td>
<td>LKX</td>
</tr>
<tr>
<td></td>
<td>SLJ(L-1)</td>
<td></td>
<td>VFK</td>
<td>LKX</td>
</tr>
<tr>
<td>3WAT</td>
<td>SLJ(N)</td>
<td></td>
<td>VFK</td>
<td>LKX</td>
</tr>
<tr>
<td></td>
<td>ENA(4)</td>
<td></td>
<td>VFK</td>
<td>LKX</td>
</tr>
<tr>
<td></td>
<td>ISK1(300)</td>
<td></td>
<td>VFK</td>
<td>LKX</td>
</tr>
<tr>
<td></td>
<td>ENA(-1)</td>
<td></td>
<td>VFK</td>
<td>LKX</td>
</tr>
<tr>
<td>* PRINT LEFT AND RIGHT MARGINS.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1MAR</td>
<td>SLJ(N)</td>
<td></td>
<td>VFK</td>
<td>LKX</td>
</tr>
<tr>
<td></td>
<td>INA(3)</td>
<td></td>
<td>VFK</td>
<td>LKX</td>
</tr>
<tr>
<td></td>
<td>LDQ(MX)</td>
<td></td>
<td>VFK</td>
<td>LKX</td>
</tr>
<tr>
<td></td>
<td>STA(RM1)</td>
<td></td>
<td>VFK</td>
<td>LKX</td>
</tr>
<tr>
<td></td>
<td>INA(-3)</td>
<td></td>
<td>VFK</td>
<td>LKX</td>
</tr>
<tr>
<td></td>
<td>AL4(44B)</td>
<td></td>
<td>VFK</td>
<td>LKX</td>
</tr>
<tr>
<td></td>
<td>X TO RIGHT MARGIN.</td>
<td></td>
<td>VFK</td>
<td>LKX</td>
</tr>
</tbody>
</table>

* DONT PRINT. 
* EXTEND SIGN. 
* XLOC POS. JUMP. 
* -X = NEXT BYTE. 
* +X = NEXT BYTE. 
* POSITION 2 BYTES. 
* START YLOC LIKE XLOC. 
* POSITION 2 BYTES. 
* B6 = 0= CALL PIGGEN, NOT 0= SKIP IT. 
* B1 = PICBUF WORD TO BE ADDED TO. 
* B2 = UNUSED BITS IN PICBUF(B1). 
* LLS(LS) = NEW BYTE + 77 + 76. 
* FORM NEW WORD. 
* PICBUF IS PACKED. 
* NEW WORD NOT NEEDED, JUMP. 
* PICBUF NOT FULL, JUMP. 
* PRINT PICBUF IS FULL. 
* NEW WORD, ALL 48 BITS UNUSED. 
* B6 = 0, SKIP PIGGEN. 
* DO PIGGEN. 
* SHORT WAIT. 
* .3 TO .4 SEC WAIT.
SSU(LM1) STA(LM1)  
LDA(RM2) SLJ4(IVRT)  
ALS(14B) MDQ(MKN)  
SSU(RM2) STA(RM2)  
LDA(LM2) SLJ4(IVRT)  
ALS(14B) LDQ(MKN)  
SSU(LM2) STA(LM2)  
ENI(4)  
EXF(77010B) EXF(77010B)  
OUT1(RM2) LDA(RM2)  
SUB(LM2) ENQ(0)  
AJP2(L+1) EMA(0)  
DVI(SIZE) INA(-1448)  
AJP2(L+2) INA(1446)  
ENQ(0) SLJ(L+2)  
ENQ(100B)  
STQ(T) ENI(0)  
INA(-128) AJP3(L+2)  
INIT(1) SLJ(L-1)  
INA(128) AJP(L+2)  
ENA(L+2)  
LRS(6)  
AJP(L+1) EMA(0)  
ADD(T) LLS(368)  
LDQ(MKA)  
SSU(CPL3) STA(CPL3)  
ENI(3)  
EXF(77010B) EXF(77010B)  
OUT1(CPL3) SLJ(LM1)  
* NEXT 2 HITS CONVERTED TO BINARY.  
RET SLJ(N) INA(-128)  
AJP(L+2) INA(2)  
AJP(L+1) INA(10B)  
ALS(5) STA(T)  
ENA(L+1) SLJ(01AEO)  
INA(-128) AJP(L+3)  
INA(2) AJP(L(TEXT)  
INA(10B) ALS(2)  
ADD(T) SCL[MK(1)]  
* PRINT MARKER.  
33MK SLJ(N) LDA(XLOC)  
LDQ(YLOC) QLS(148)  
LLS(1448) LDQ(MKXY)  
SSU(MARKER) STA(MARKER)  
EXF(77010B) EXF(77010B)  
OUT(MARKER) SLJ(33MK)  
BSK  
* BACKSPACE  
1BKS LDA(SIZE) BACKSPACE MARKER.
1RN8 1JP1(L+5)  
ISk2(1000B)  
SLJ(1FN)  
ZRO(0)  
2RN8 LDA2(1N)  
LDQ(-2)  
LLS(68)  
* MAIN BRANCH POINT.  
1BRA SLJ(N)  
* PROCESS STRING.  
1CST INA(-76B)  
INA(76B)  
SLJ(1RN8)  
1SPT ALS(6)  
ENA(L+1)  
ADD(W1)  
STA(W1)  
ENA(L+1)  
ADD(W1)  
ALS(33B)  
LDQ(MKXY)  
STA(W1)  
LA(UKPT)  
STA(W1)  
EXF(77010B)  
OUT(W1)  
* END STRING.  
1EST ENA4(-6)  
ENA(20B)  
SLJ(1TEST)  
* PACK  
1PAK SLJ(N)  
2PAK IJP4(L+1)  
ALS(6)  
SLJ(1PAK)  
4PAK ENA(5PAK)  
LDA(W)  
ALS(30B)  
SSU(DW)  
5PAK LRS(68)  
LLS(68)  
SSU(SW)  
ISK2(1727B)  
SLJ(1ER)  
EXF(77010B)  
ENI4(16)  
OUT(W)  
STA(W)  
* TABLE  
* READ NEXT BYTE.  
* ANY WORDS LEFT.  
* NO, EXIT.  
* LOAD NEW WORD.  
* WORD TO Q, O TO A.  
* LOAD NEXT BYTE, Q TO WORD.  
* MAIN BRANCH POINT.  
* IF BYTE IS CR OR END STRING, JUMP.  
* PACK.  
* 1ST BYTE.  
* 2ND BYTE.  
* 3RD AND LAST BYTE.  
* CTRL. ARE IN POSITION.  
* INSERT LOCATION.  
* INSERT DOT.  
* INDIW MEM.  
* SEL AND WAIT FOR DD65 MEM.  
* OUTPUT.  
* OUTPUT WORD COMPLETE, JUMP.  
* FILL WITH BLANKS.  
* WORD FULL, JUMP TO 4 OR 5 PACK  
* NOT FULL, STORE AND RETURN.  
* DW.  
* SW STRING.  
* UPDATE MEM ADDRESS.  
* AVAILABLE MEM FILLED, EXIT.  
* SEL AND WAIT FOR DD65 MEM.  
* NEW WORD.  
* TBP
| 10D | INA(00D) | SAU(L+1) | ENTER TABLE. | TBP |
| 00D | SLJ(N)   | SLJ(18G) | JUMP. | TBP |
|     | ENA(30)  | SLJ(18R) | BEGIN STRING. | TBP |
|     | ENA(3R)  | SLJ(18R) | PASS. | TBP |
|     | ENA(34)  | SLJ(18R) | | TBP |
|     | ENA(35)  | SLJ(18R) | | TBP |
|     | ENA(36)  | SLJ(18R) | | TBP |
|     | ENA(37)  | SLJ(18R) | | TBP |
|     | ENA(38)  | SLJ(18R) | | TBP |
|     | ENA(39)  | SLJ(18R) | | TBP |
| 10D | ENA(31)  | SLJ(18H) | CHARACTER STRING. | TBP |
|     | ENA(31)  | SLJ(18H) | VECTOR STRING. | TBP |
|     | ENA(31)  | SLJ(18H) | SPOT MODE. | TBP |
| 20D | ENA(32)  | SLJ(4ST) | SET X. | TBP |
|     | ENA(32)  | SLJ(4ST) | SET Y. | TBP |
|     | ENA(32)  | SLJ(4ST) | SET -X. | TBP |
|     | ENA(32)  | SLJ(4ST) | SET -Y. | TBP |
| 30D | LDA(DW)  | SLJ(1MED) | SET SMALL CHAR. | TBP |
|     | LDA(DW)  | SLJ(1LRG) | SET MEDIUM CHAR. | TBP |
|     | ENA(32)  | SLJ(1ERR) | SET LARGE CHAR. | TBP |
| 40D | LDA(DW)  | SLJ(1LFR) | USE RIGHT TUBE. | TBP |
|     | LDA(DW)  | SLJ(1LFR) | USE LEFT TUBE. | TBP |
| 50D | LDA(DW)  | SLJ(1NOR) | SET NORMAL INTENSITY. | TBP |
|     | LDA(DW)  | SLJ(1NOR) | SET BRIGHT INTENSITY. | TBP |
| 609 | ENA(2ERR) | SLJ(1ERR) | INCREMENT RIGHT. | TBP |
| 610 | LDA(DW) | SLJ(1IN) | INCREMENT DOWN. | TBP |
| 611 | LDA(DW) | SLJ(1ERR) | | TBP |
| 612 | ENA(2ERR) | SLJ(1ERR) | | TBP |
| 613 | ENA(2ERR) | SLJ(1ERR) | | TBP |
| 614 | ENA(2ERR) | SLJ(1ERR) | | TBP |
| 615 | ENA(2ERR) | SLJ(1ERR) | | TBP |
| 616 | ENA(2ERR) | SLJ(1ERR) | | TBP |
| 617 | ENA(2ERR) | SLJ(1ERR) | | TBP |
| 618 | ENA(2ERR) | SLJ(1ERR) | | TBP |
| 619 | ENA(2ERR) | SLJ(1ERR) | | TBP |
| 620 | SLJ(1FN) | SLJ(1ERR) | EXIT. NEXT DD65 ADD IN UA OF ARG. | TBP |
| 621 | SLJ(1RN8) | SLJ(1ERR) | SKIP ONE BYTE. | TBP |

* BEGIN STRING.

| 18G5 | ALS(15B) | LDQ(MKADD); | BEGIN STRING. | BSP |
| 521 | SSU(DW) | STA(DW); | | BSP |
| 522 | ENA(4PAK) | SAL(2PAK); | | BSP |
| 523 | ENI4(1) | ENA(1CST) | | BSP |

* MODE

| 1CH | SAL(2BGS) | LDA(DW); | SET CHAR. MODE. | MDP |
| 524 | SCL(MKCH) | STA(DW); | | MDP |
| 525 | LDA(SW) | SCL(MKCH); | | MDP |
| 526 | STA(SW) | SLJ(1RN8); | | MDP |
| 527 | 1VCT | SAL(2BGS) | SET VECTOR MODE. | MDP |
| 528 | SCL(MKCH) | LDA(DW) | | MDP |
| 529 | STA(SW) | SCL(MKCH) | | MDP |
| 530 | STA(SW) | SLJ(1RN8) | | MDP |

* LOCATION.

| 22AD | SCm(MKCM) | LDQ(MKX); | RESTORE ROUTE. | LCP |
| 531 | ALS(46B) | SLJ(L+4) | | LCP |
| 532 | SSU(DW) | -A = X. | | LCP |
| 533 | ENA(2B1) | A = X. | | LCP |
| 534 | SCm(MKCM) | -A = Y. | | LCP |
| 535 | ALS(2B) | A = Y. | | LCP |
| 536 | SSU(DW) | | | LCP |
| 537 | STA(DW) | | | LCP |

* SIZE.

| 15ML | SCL(MK47) | SCL(MK46); | SET FOR SMALL. | SZP |
| 538 | STA(DW) | SLJ(1RN8); | | SZP |
| 539 | 1MED | SCL(MK47) | SET FOR MEDIUM. | SZP |
| 540 | STA(DW) | SLJ(1RN8) | | SZP |
| 541 | 1LRG | SCL(MK47) | SET FOR LARGE. | SZP |
| 542 | STA(DW) | SLJ(1RN8) | | SZP |

* TUBE

| 1TRT | SST(MK11) | STA(DW) | SET RIGHT TUBE. | TBP |

60
LDA(SW)      SST(MK11)
STA(SW)      SLJ(1RN8)
* INTENSITY,
1TLF SCL(MK11) STA(DW)
LDA(SW)      SCL(MK11)
STA(SW)      SLJ(1RN8)
* SET LEFT TUBE.
1NOR SCL(MK45) STA(DW)
SLJ(1RN8)
* SET NORMAL INTENSITY.
1BRT SST(MK45) STA(DW)
SLJ(1RN8)
* SET BRIGHT INTENSITY.
1INCR SCL(MK23) STA(DW)
SLJ(1RN8)
1IND SST(MK23) STA(DW)
SLJ(1RN8)
* SET TO INCREMENT RIGHT.
* SET TO INCREMENT DOWN.
* INCRED.
1ERR INI5(4) EXF(77010B)
2ERR OUT5(ERR4) SLJ(1EN)
1ER EXF(77010B) EXF(77010B)
ENI(2)
* PRINT ERROR.
* BAD INPUT TO PICGEN.
* SEL AND WAIT FOR OD65 MEM.
* PRINT ERROR.
* END
1FN ENA(L+2) SAL(1EST)
SLJ(1EST)
ENA(3ST)
* RETURN TO HERE.
1FIN ENI1(N) SAL(1EST)
ENI2(N)
1FIN ENI5(N) ENI4(N)
2FIN ENI5(N) ENI6(N)
2FIN ENI5(N) ENI6(N)
SLJ(1ST)
* END STRING.
* RESTORE ROUTE.
* EXIT.

MACHINE CHALOC (PICBUF, POSIT, HERE, PIPX, PIPY)
* FINDS THE LOCATION IN PICBUF OF THE CHARACTER INDICATED BY THE PIPER.
  HERE = PICBUF WORD AND HERE-UA = BYTE (RT MOST=O, LF MOST=52B)
CON(M6=68)
  CON(M6=68)
  CON(M6=68)
  CON(M6=68)
  CON(M6=68)
1ST SLJ(N) SLJ(L+6) * PICBUF.
2ST ZRO(0) ZRO(0) * POSIT.
3ST ZRO(0) ZRO(0) * HERE. IF HERE IS NEG., NOT FOUND.
4ST ZRO(0) ZRO(0) * PIPX.
5ST ZRO(0) ZRO(0) * PIPY.
6ST ZRO(0) ZRO(0) * PREP.
1ST ZRO(0) LDA(3ST)
  SAU(5RN8)
  ENA(1SF)
  ENI10)
2ST ZRO(0) LDA(2ST)
  SAU(3RN8)
  ENI10)
* READ NEXT BYTE.
1RN8 IJP1(L+4) ENI1(7)
* USE OLD WORD, JUMP.
61
SRB ISK2(N)
2RB ENA1(-1)
3RB LDA2(N)
LDQ(W)
LLS(W)
* MAIN BRANCH POINT.
1RA SLJ(N)
* START FILE CHECK.
1SF INA(-1)
SLJ(3SS)
* SKIP STRING.
1SS ENA(2SS)
1BG SAU(1BRA)
2SS INA(-77B)
3SS ENA(1DC)
* DECODER.
1DC AJP(1SS)
AJP(2PX)
AJP(2PY)
AJP(2MX)
AJP(2MY)
AJP(1SM)
AJP(1ME)
AJP(1LR)
1SM ENA(2)
1ME ENA(4)
1LR ENA(10B)
STA(SIZE)
2PX ENA(1PX)
2PY ENA(1PY)
2MX ENA(1MX)
2MY ENA(1MY)
* LOCATION SEARCH.
1MX ENQ(0)
SMC(1SEV)
STA(XF)
1MY ENQ(0)
SMC(1SEV)
STA(YF)
1PY ENQ(0)
STA(YF)
AJP3(L+1)
ADD(SIZE)
* CORRECT LINE FOUND. NOW FIND CHARACTER.
1CH ENA(2CH)
SLJ(3SS)
2CH ENA(1SS)
3CH INA(-77B)
* DIGBUF EXHAUSTED, JUMP.
* NOT FOUND, EXIT.
* LOAD NEW WORD.
* BYTE IN A RJ.
* MAIN BRANCH POINT.
* NOT START FILE, JUMP.
* START FILE, GO TO DECODER.
* ROUTE STRING BELOW.
* NOT END OF STRING, JUMP.
* RESTORE ROUTE.
* START STRING, JUMP.
* SMALL.
* MEDIUM.
* LARGE.
* SIZE = 1/2 OF TRUE SIZE.
* ROUTE NEXT LOCATION BYTE.
-X.
+X.
EXIT.
-Y. NOTE-Y POSITION ALWAYS FOLLOWING X POSITION DUE TO 1LOC ROUTINE.
+Y.
IF SIZE - ABS(YF - PY) IS NEG., WRONG LINE, JUMP
REROUTE START STRING.
RESET START STRING.
ROUTE STRING BELOW.
END STRING, REROUTE.
LDA(XF)
AJP3(L+1)
ADD(SIZE)
LDA(SIZE)
RAD(XF)
1FND
SAU7(4ST)
ENA2(10)
LDA7(4ST)
1SEV
SEV7(77777B)

SUB7(5ST)
SCM(1SEV)
AJP2(1FND)
ADD(SIZE)
SLJ(1RN)
MUI(M6)
SAL7(4ST)
SLJ(1ST)
SEV7(77777B)

IF SIZE - ABS(XF - P1PX) IS
POSD, RIGHT CHAR., JUMP.
INCREMENT XF.
OUT PUT LOCATION
FOUND EXIT
ALL ONES.

MACHINE FLASH
SLJ(N), ENAI2OB, STA(P), -SLJ4(1DN),
1-SLJ4(10FF), -SLJ4(1L),
- SLJ4(10FF), -SLJ4(1I), -SLJ4(10FF),
2-SLJ4(1G), -SLJ4(10FF), -SLJ4(1Z), -SLJ4(10FF), -SLJ4(1T), -SLJ4(10FF),
3-SLJ4(10FF), -SLJ4(1T), -SLJ4(10FF), -SLJ4(1E), -SLJ4(10FF),
4-SLJ4(1I), -SLJ4(10FF), -SLJ4(1I), -SLJ4(10FF),
5ENA(20B), STA(P), ENI(6).

2ST
EXF(77202B), SLJ4(1W), EXF(77203B), SLJ4(1W), IP6(L-2), ENI(6).
EXF(77204B), SLJ4(1W), EXF(77205B), SLJ4(1W), IP6(L-2), ENI(6).
EXF(77210B), SLJ4(1W), EXF(77211B), SLJ4(1W), IP6(L-2), ENI(6).
EXF(77220B), SLJ4(1W), EXF(77221B), SLJ4(1W), IP6(L-2), ENI(6).
EXF(77240B), SLJ4(1W), EXF(77241B), SLJ4(1W), IP6(L-2), ENI(6).
EXF(77340B), SLJ4(1W), EXF(77341B), SLJ4(1W), IP6(L-2), ENI(6).
EXF(77320B), SLJ4(1W), EXF(77321B), SLJ4(1W), IP6(L-2), ENI(6).
EXF(77310B), SLJ4(1W), EXF(77311B), SLJ4(1W), IP6(L-2), ENI(6).
EXF(77302B), SLJ4(1W), EXF(77305B), SLJ4(1W), IP6(L-2), ENI(6).
EXF(77304B), SLJ4(1W), EXF(77303B), SLJ4(1W), IP6(L-2), ENI(6).
EXF(77402B), SLJ4(1W), EXF(77403B), SLJ4(1W), IP6(L-2), ENI(6).
EXF(77404B), SLJ4(1W), EXF(77405B), SLJ4(1W), IP6(L-2), ENI(6).
EXF(77410B), SLJ4(1W), EXF(77411B), SLJ4(1W), IP6(L-2), ENI(6).
EXF(77420B), SLJ4(1W), EXF(77421B), SLJ4(1W), IP6(L-2), ENI(6).
EXF(77440B), SLJ4(1W), EXF(77441B), SLJ4(1W), IP6(L-2), ENI(6).
EXF(77540B), SLJ4(1W), EXF(77541B), SLJ4(1W), IP6(L-2), ENI(6).
EXF(77510B), SLJ4(1W), EXF(77511B), SLJ4(1W), IP6(L-2), ENI(6).
EXF(77520B), SLJ4(1W), EXF(77521B), SLJ4(1W), IP6(L-2), ENI(6).
EXF(77510B), SLJ4(1W), EXF(77511B), SLJ4(1W), IP6(L-2), ENI(6).
EXF(77502B), SLJ4(1W), EXF(77503B), SLJ4(1W), IP6(L-2), ENI(6).
EXF(77602B), SLJ4(1W), EXF(77603B), SLJ4(1W), IP6(L-2), ENI(6).
EXF(77604B), SLJ4(1W), EXF(77605B), SLJ4(1W), IP6(L-2), ENI(6).
EXF(77610B), SLJ4(1W), EXF(77611B), SLJ4(1W), IP6(L-2), ENI(6).
EXF(77620B), SLJ4(1W), EXF(77621B), SLJ4(1W), IP6(L-2), ENI(6).
EXF(77640B), SLJ4(1W), EXF(77641B), SLJ4(1W), IP6(L-2), ENI(6).
EXF(77740B), SLJ4(1W), EXF(77741B), SLJ4(1W), IP6(L-2), ENI(6).

63