An alphanumeric text generator for a computer display.

William E. Mennie

University of Windsor

Follow this and additional works at: https://scholar.uwindsor.ca/etd

Recommended Citation
https://scholar.uwindsor.ca/etd/6621

This online database contains the full-text of PhD dissertations and Masters' theses of University of Windsor students from 1954 forward. These documents are made available for personal study and research purposes only, in accordance with the Canadian Copyright Act and the Creative Commons license—CC BY-NC-ND (Attribution, Non-Commercial, No Derivative Works). Under this license, works must always be attributed to the copyright holder (original author), cannot be used for any commercial purposes, and may not be altered. Any other use would require the permission of the copyright holder. Students may inquire about withdrawing their dissertation and/or thesis from this database. For additional inquiries, please contact the repository administrator via email (scholarship@uwindsor.ca) or by telephone at 519-253-3000 ext. 3208.
INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps.

ProQuest Information and Learning
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA
800-521-0600

UMI
AN ALPHANUMERIC TEXT GENERATOR
FOR A COMPUTER DISPLAY

BY
WILLIAM E. MENNIE

A Thesis
Submitted to the Faculty of Graduate Studies through the
Department of Electrical Engineering in Partial Fulfillment
of the Requirements for the Degree of
Master of Applied Science at the
University of Windsor

Windsor, Ontario
1970
ABSTRACT

This thesis describes the logical design and implementation of three variations of an alpha-numeric text generator for displaying computer information on a cathode ray tube (c.r.t.).

Each character is generated by intensifying dots which are selected from a $5 \times 7$ matrix. Selection of the dots is performed by digital logic circuits which supply unblanking pulses to the Z-axis of the c.r.t. in synchronism with a raster generator.

Three methods of raster generation are described. This permits the character generator to be used in systems of varying cost and complexity.

The most complex method uses horizontal and vertical digital to analogue (D/A) converters to generate the raster as well as to position each character. This permits several new character operations to be provided by hardware.

Three sizes of characters may be generated and two sizes may be hardware positioned as subscripts or superscripts. Back spacing enables superscripts to be positioned above subscripts or characters to be superimposed. Tabulation locates up to eight vertical columns. Carriage returns may be made to any one of nine left-hand margin positions. Line feeds may be single or double spaced. Spacing between lines and characters is automatically adjusted according to the character size used.

Control of the character generator may be accomplished through an ASR 33 Teletype and a few function keys. Thus the text generator is easier to operate than most conventional character generators.
The text generator functions equally well with a storage c.r.t. or with a refresh memory since it is capable of generating up to 53,000 characters per second.

The simplicity and versatility of the text generator makes it suitable for nearly every application which utilizes a c.r.t. display since it can readily be tailored to such user's needs as the addition of a graphics capability.
ACKNOWLEDGEMENTS

The author wishes to express his appreciation to Dr. P. A. V. Thomas for suggesting this project and for his helpful criticism. Acknowledgement must also go to the National Research Council of Canada for the financial assistance which made this project possible.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABSTRACT</td>
<td>ii</td>
</tr>
<tr>
<td>ACKNOWLEDGEMENTS</td>
<td>iv</td>
</tr>
<tr>
<td>TABLE OF CONTENTS</td>
<td>v</td>
</tr>
<tr>
<td>LIST OF ILLUSTRATIONS</td>
<td>viii</td>
</tr>
<tr>
<td>LIST OF TABLES</td>
<td>xi</td>
</tr>
<tr>
<td>I.  INTRODUCTION</td>
<td></td>
</tr>
<tr>
<td>1.1 The Desirability of Cathode Ray Tube Displays</td>
<td>1</td>
</tr>
<tr>
<td>1.2 Some Current Methods of Character Generation</td>
<td>4</td>
</tr>
<tr>
<td>1.2.1 Beam Shaping</td>
<td>4</td>
</tr>
<tr>
<td>1.2.2 Scanning</td>
<td>4</td>
</tr>
<tr>
<td>1.2.3 Stroke or Waveform</td>
<td>5</td>
</tr>
<tr>
<td>1.2.4 Dot Generation</td>
<td>5</td>
</tr>
<tr>
<td>1.2.5 Software Techniques</td>
<td>6</td>
</tr>
<tr>
<td>1.3 The Refresh Problem</td>
<td>7</td>
</tr>
<tr>
<td>1.4 Three Proposed Methods of Character Generation</td>
<td>8</td>
</tr>
<tr>
<td>1.5 Implementation</td>
<td>9</td>
</tr>
<tr>
<td>II. GENERATION OF UNIQUE CHARACTER PULSE TRAINS</td>
<td></td>
</tr>
<tr>
<td>2.1 Reasons for Selecting the 5 x 7 Dot Matrix</td>
<td>10</td>
</tr>
<tr>
<td>2.2 The Unique Pulse Train</td>
<td>12</td>
</tr>
<tr>
<td>2.3 General Description of the Character Dot Generator</td>
<td>12</td>
</tr>
<tr>
<td>2.4 The Counter</td>
<td>14</td>
</tr>
<tr>
<td>2.5 The Counter Decoder</td>
<td>17</td>
</tr>
<tr>
<td>2.6 Pulse Selection Gates</td>
<td>20</td>
</tr>
</tbody>
</table>
5.5 The Carriage Return and Tabulate ................. 69
1. Carriage Return ........................................ 69
2. Tabulate .................................................. 73

VI. CONTROL AND TESTING OF THE A/N GENERATOR

6.1 Refresh and Control of the A/N Display .......... 79
6.2 The I/O Terminal ........................................ 82
6.3 Test Results ............................................ 86

VII. CONCLUSIONS ........................................... 88

APPENDIX I CHARACTER LIST AND ASCII CODES ................. 92
APPENDIX II CHARACTER DOT PATTERNS AND PULSE NUMBERS .... 94
APPENDIX III CHARACTER SIZE AND POSITION EXAMPLES .......... 100
APPENDIX IV INSTRUCTION AND DATA WORDS ................. 101
APPENDIX V TIMING DIAGRAMS .............................. 102
APPENDIX VI LOGIC NOTATION ............................... 105
APPENDIX VII TWO OPERATING SPEEDS FOR HORIZONTAL AND VERTICAL
SWEEP CHARACTER GENERATION ............................. 106

REFERENCES .................................................. 107
VITA AUCTORIS .............................................. 108

vii

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
**LIST OF ILLUSTRATIONS**

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>Examples of Characters Generated by Different Methods</td>
<td>3</td>
</tr>
<tr>
<td>2.1</td>
<td>Use of the Unique Pulse Train</td>
<td>11</td>
</tr>
<tr>
<td></td>
<td>a. Horizontal Deflection Voltage Waveform</td>
<td></td>
</tr>
<tr>
<td></td>
<td>b. Vertical Deflection Waveform</td>
<td></td>
</tr>
<tr>
<td></td>
<td>c. Pulse Train Required to Form the Character 'A'</td>
<td></td>
</tr>
<tr>
<td></td>
<td>d. Character Which Results From Combined Waveforms</td>
<td></td>
</tr>
<tr>
<td>2.2</td>
<td>Block Diagram of Character Dot Generator</td>
<td>13</td>
</tr>
<tr>
<td>2.3</td>
<td>Counter for Horizontal Sweep Method</td>
<td>15</td>
</tr>
<tr>
<td>2.4</td>
<td>High Speed Counter for Horizontal D/A Converter Method</td>
<td>16</td>
</tr>
<tr>
<td>2.5</td>
<td>The Counter Decoder</td>
<td>18</td>
</tr>
<tr>
<td>2.6a</td>
<td>The Basic 5 x 7 Dot Matrix</td>
<td>19</td>
</tr>
<tr>
<td>2.6b</td>
<td>The Pulse Selection Gates</td>
<td>19</td>
</tr>
<tr>
<td>2.7</td>
<td>The Buffer Register</td>
<td>21</td>
</tr>
<tr>
<td>2.8</td>
<td>The Character Decoder</td>
<td>23</td>
</tr>
<tr>
<td>3.1</td>
<td>Output of Original Character Generator</td>
<td>27</td>
</tr>
<tr>
<td>3.2</td>
<td>Character Generator With Horizontal and Vertical Sweep</td>
<td>29</td>
</tr>
<tr>
<td>3.3</td>
<td>Horizontal Sweep and Vertical D/A Converter Method</td>
<td>32</td>
</tr>
<tr>
<td>3.4</td>
<td>Timing Waveforms for Horizontal Sweep and Vertical D/A Converter</td>
<td>33</td>
</tr>
<tr>
<td></td>
<td>a. Horizontal Deflection Voltage Waveform</td>
<td></td>
</tr>
<tr>
<td></td>
<td>b. Clock Pulses</td>
<td></td>
</tr>
<tr>
<td></td>
<td>c. Y-Reset Pulses to Clear the Least Significant Three Bits</td>
<td></td>
</tr>
</tbody>
</table>

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1</td>
<td>One Possible Method Using D/A Converters</td>
<td>37</td>
</tr>
<tr>
<td>4.2a</td>
<td>Normal Counter</td>
<td>40</td>
</tr>
<tr>
<td>4.2b</td>
<td>Counter Capable of Counting in Any Position</td>
<td>40</td>
</tr>
<tr>
<td>4.3</td>
<td>Size Selection Circuitry</td>
<td>43</td>
</tr>
<tr>
<td>4.4a</td>
<td>Dot Matrix</td>
<td>46</td>
</tr>
<tr>
<td>4.4b</td>
<td>Character Count Control Circuitry</td>
<td>46</td>
</tr>
<tr>
<td>4.5</td>
<td>X Count Selection Circuitry</td>
<td>50</td>
</tr>
<tr>
<td>4.6a</td>
<td>X Register (Least Significant Bits)</td>
<td>52</td>
</tr>
<tr>
<td>4.6b</td>
<td>X Register (Most Significant Bits)</td>
<td>53</td>
</tr>
<tr>
<td>4.7</td>
<td>Y Count Selection Circuitry</td>
<td>56</td>
</tr>
<tr>
<td>4.8</td>
<td>Y Reset Circuitry</td>
<td>57</td>
</tr>
<tr>
<td>4.9a</td>
<td>Y Register (Least Significant Bits)</td>
<td>59</td>
</tr>
<tr>
<td>4.9b</td>
<td>Y Register (Most Significant Bits)</td>
<td>60</td>
</tr>
<tr>
<td>4.10</td>
<td>UP-DOWN Count Selection Circuitry</td>
<td>62</td>
</tr>
<tr>
<td>5.1</td>
<td>Special Character Decoder</td>
<td>65</td>
</tr>
<tr>
<td>5.2a</td>
<td>Generation of the Line Feed Pulse</td>
<td>67</td>
</tr>
<tr>
<td>5.2b</td>
<td>Use Margin Register and Space Flip-Flop</td>
<td>67</td>
</tr>
<tr>
<td>5.3</td>
<td>X Register Clear Circuitry</td>
<td>70</td>
</tr>
<tr>
<td>5.4</td>
<td>Circuitry for Loading X Register During Carriage Return and Tabulate</td>
<td>71</td>
</tr>
<tr>
<td>5.5</td>
<td>Tabulate Algorithm</td>
<td>75</td>
</tr>
<tr>
<td>5.6</td>
<td>Circuitry Required To Generate 64's Count During Tabulation</td>
<td>76</td>
</tr>
<tr>
<td>6.1a</td>
<td>Device Selector Schematic</td>
<td>80</td>
</tr>
<tr>
<td>Figure</td>
<td>Description</td>
<td>Page</td>
</tr>
<tr>
<td>--------</td>
<td>-------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>6.1b</td>
<td>A/N Control Unit</td>
<td>80</td>
</tr>
<tr>
<td>6.2</td>
<td>Block Diagram of I/O Terminal</td>
<td>81</td>
</tr>
<tr>
<td>6.3</td>
<td>Three Raster Techniques (Photographs)</td>
<td>84</td>
</tr>
<tr>
<td>a.</td>
<td>Horizontal and Vertical Sweep</td>
<td></td>
</tr>
<tr>
<td>b.</td>
<td>Horizontal Sweep and Vertical D/A Converter</td>
<td></td>
</tr>
<tr>
<td>c.</td>
<td>Horizontal and Vertical D/A Converter</td>
<td></td>
</tr>
<tr>
<td>6.4a</td>
<td>Three Character Sizes (Photograph)</td>
<td>85</td>
</tr>
<tr>
<td>6.4b</td>
<td>Tabulate and Carriage Return (Photograph)</td>
<td>85</td>
</tr>
</tbody>
</table>

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
## LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1   X Register Counting Capabilities</td>
<td>49</td>
</tr>
<tr>
<td>4.2   Y Register Counting Capabilities</td>
<td>54</td>
</tr>
<tr>
<td>4.3   Y-Reset Positions</td>
<td>55</td>
</tr>
<tr>
<td>5.1   Tabulate and Margin Positions</td>
<td>69</td>
</tr>
</tbody>
</table>
CHAPTER I

INTRODUCTION

1.1 The Desirability of Cathode Ray Tube Displays

Digital computers are capable of extremely rapid computation and data manipulation. Since mechanical devices such as typewriters, line-printers, card readers and plotters are generally much slower than the computer, the computer must normally slow down to match the speed of the input-output (I/O) devices which are transmitting data to and receiving data from the computer. This can vary from a person typing a few characters per minute to a line printer capable of outputting perhaps 3,000 characters per second.

To avoid this problem, modern high-speed computers generally transfer the computed information into buffers which can accept data at the computer's high speed and then supply this data to the output device at its own lower speed. The reverse operation can be used for loading data into the computer. Thus the computer is able to interact with several input-output devices to ensure that the high computing capability is used to the best advantage. For the majority of scientific and business applications this type of interaction is satisfactory. The job is submitted at one time and the computed results are obtained on paper at a later time.

More recently, computers have found their way into such varied areas as data retrieval, computer-aided instruction (CAI) and computer-aided design (CAD). These uses introduce two new problems which the above
mentioned I/O devices are not readily able to cope with. First, the
user is interested in rapid results and cannot afford long delays while
waiting for printed material. Obviously, he cannot sit in front of a
line-printer all day. Secondly, the user often wishes to interact with
the computer. How is he able to obtain access to the information which
has been computed but is still in the output buffer and not yet printed
so that he may rapidly decide if the results are what he expected?

The best solution to the problem developed so far is the use of a
cathode ray tube (c.r.t.) display. Character writing speeds in excess
of 50,000 per second are easily achieved. Thus computed data can be
transmitted and displayed almost instantaneously. The addition of a
light-pen and function keys to the normal typewriter keyboard adds the
possibility of modifying or pointing to selected areas of the display
to obtain the attention of the computer for rapid two-way interaction
between the computer and the user. Adding a graphics capability en-
ables pictorial as well as printed information to be displayed.

Another important advantage of c.r.t. displays is the possibility
of having several c.r.t. screens displaying the same information sim-
ultaneously. This is advantageous, for example, in modern airports
where computers are used in flight scheduling. Displays placed in
strategic locations throughout the terminal can immediately inform the
staff and public of arrival and departure times.

The features mentioned above make c.r.t. displays an attractive
and efficient method of providing communication between the computer
and its users. In addition, they can be made esthetically pleasing to
the eye and their quiet operation permits the user to work in a peaceful
and thought-stimulating environment.
Figure 1.1 Examples of Characters Generated by Different Methods
1.2 Some Current Methods of Character Generation²,⁷

There are many different methods of generating characters for c.r.t. displays in use today. The most important requirement is that they are easily readable. A few of the currently used methods will be described.

1. **Beam Shaping**

This method utilizes a specially constructed c.r.t. with a stencil mask placed in the path of the electron beam. This stencil contains the shape of all characters which can be displayed. The unshaped electron beam generated by the accelerating gun of the tube can be deflected to a particular area of the mask by selection plates within the tube. The beam passes through the mask and is extruded in the shape of the character. The shaped beam is then deflected again to its final position on the screen. Although this method of character generation is fast, the tube is expensive and the number and shape of the characters is usually quite limited. Fig. 1.1a is an example of this type of character.

2. **Scanning**

This method utilizes a scanning raster similar to that used in television. The character data within the generator is scanned in synchronism with the raster to provide the data required to modulate the beam. This method is well suited to multiple displays such as the previously mentioned airport display since television sets can be used and interconnected by coaxial cable for a relatively low cost display. This method cannot readily be used for light pen interaction since the entire screen is being scanned by the raster and the coordinates of the beam at a particular time are not accurately known. Fig. 1.1b gives
an example of this character.

3. Stroke or Waveform

This method of generating characters probably permits the greatest flexibility of character styles. The character is stored as X deflection, Y deflection and intensity information. As the character is being generated, these three pieces of analogue data are simultaneously fed to the X, Y and Z axes of the c.r.t. tube. The character is, therefore, traced out in much the same manner as one would write characters by hand. This method is very fast but the circuitry required to store and supply the analogue information is quite complex and expensive. Fig. 1.1c illustrates the characters generated by this method.

4. Dot Generation

In this method, the characters are composed of a group of dots which are selected from a basic array or matrix. Two operations must be performed to generate a character by this method. First, the dot matrix position must be generated on the screen by either a small sweep raster or by D/A converters which can accurately position the beam within the array. Secondly, some means of determining whether to intensify (unblank) the beam at a particular position within the array must be provided for each character.

One of the most commonly used methods of generating the intensification information is to use a small core memory. If a 5 x 7 matrix of dots is used, the memory will contain 35 magnetic cores also arranged in a 5 x 7 matrix. A sense winding is threaded through the cores which correspond to the dots which are required to generate a particular character. In addition, a set and reset winding is threaded
through each core. All of the set windings of each column are connected in series as are the reset windings of each row.

To generate a character the core memory is interrogated in sequence with the raster generation. First, all of the cores of a particular column are set but the character sense windings are disabled. Then each core in the column is reset in sequence and the sense winding is activated. Only if the sense winding passes through a core which is being reset will it sense the changing magnetic field and control the intensification of the corresponding point in the CRT raster. The obvious disadvantages of this method are that characters cannot be changed without rewiring the core matrix and a complete read cycle must be performed for all 35 possible dots in the matrix making the system quite slow compared with other methods.

Other methods of selecting dots have also been used. One uses a diode matrix which is scanned to generate the x and y deflection as well as the required intensity information.

5. Software Techniques

In order to generate characters as rapidly or as inexpensively as possible, the various hardware techniques which have been mentioned have usually been employed to generate characters. Usually only the type and size of the character is supplied to the character generator. The actual generation of the character is accomplished automatically.

With the advent of extremely high speed random access memories, some manufacturers are moving away from hardware character generation. The coordinate and intensity information required to generate a particular character is stored in a memory as a subroutine which can easily be
modified to generate any shape and size of character one pleases.

A character is generated by selecting a particular subroutine. The stored information is then fed to D/A converters which control the beam positioning and intensity whenever that particular subroutine is addressed. The obvious advantages of such a method are flexibility and the use of relatively unsophisticated hardware. But until recently high speed memories have been very costly although their price is rapidly being reduced now. It may well be that in the very near future this method of character generation may overtake the conventional hardware techniques.

1.3 The Refresh Problem

It has been implied above that high speed is a desirable characteristic for a character generator. This can be desirable for two main reasons.

First, it was mentioned that one wishes to have the computer transfer its computed results as rapidly as possible. If one is utilizing a special storage c.r.t. tube which is capable of storing the image traced out by the electron beam, the computer can transfer the data to the display at a fairly high speed (say 3,000 characters/second) and the screen will retain the information. This type of display cannot be used for interactive displays which use a light pen since the display is continuous with time and does not give the computer any means of determining the position of the pen.

If one is not utilizing a storage c.r.t. tube, generating the display once is not sufficient since it fades rapidly with time. This fade-out or persistence time typically varies from between 20 to 100 milliseconds depending on the type of phosphor used in the c.r.t. This
brings us to the second and most important need for a high speed display.

To maintain the information on a non-storing c.r.t., the display must continually be regenerated or refreshed. This refreshing must be done at a rate which is high enough to prevent the eye from seeing it or the screen will appear to have annoying flicker. Thus the refresh rate could be as high as 50 or 60 frames per second depending on the phosphor of the c.r.t. and background lighting of the room. In order to display 1,000 characters on the screen, one must have a character generator capable of generating 50,000 characters per second. Hence software character generation is currently very limited due to the amounts of information required.

1.4 Three Proposed Methods of Character Generation

A 5 x 7 dot matrix was chosen to provide the character raster. The method of selecting and intensifying the correct dots for a particular character is common to all three character generators. This will be discussed in greater detail in chapter II.

The three character generators differ primarily in the method used to generate the 5 x 7 dot matrix. The first method utilizes both horizontal and vertical sweep circuits to generate the character raster. A D/A converter is also used in the vertical axis if more than one line of characters is to be displayed.

The second method utilizes a horizontal sweep circuit but only a D/A converter is used in the vertical axis to generate both the seven vertical dot positions in the matrix and the positioning of several lines of text.

The final method, uses a D/A converter for the horizontal position-
ing as well as a vertical D/A converter similar to that used in the second method.

The three methods of generating the character rasters have been constructed and tested. They will be discussed in detail in chapters III and IV along with their advantages and disadvantages.

1.5 Implementation

The circuitry which will be described in the remainder of this thesis was constructed from Digital Equipment Corporation (DEC) "FLIP CHIP" modules. This is mentioned at this time, since some of the circuitry which is to be discussed is very dependent on DEC hardware. However, when this is the case, enough theory is given so that one could easily implement the circuit using other circuitry.

The majority of the logic was implemented using DEC 'R' series logic modules which can operate up to a frequency of 2 MHz. These modules use negative "NAND" logic in which 0 volts represents a "zero" or "false" state and -3 volts represents a "one" or "true" state.

The D/A converters were DEC 'A' series modules. A few 'B' series delay modules were used since they provide for delays as short as 50 nsec. 'W' series clamped loads were used whenever additional signal clamping was required.

At this point, it is suggested that the reader familiarize himself with the symbols and logic notation described in Appendix VI.
CHAPTER II

GENERATION OF UNIQUE CHARACTER PULSE TRAINS

2.1 Reasons for Selecting the 5 x 7 Dot Matrix

At an early stage in the development of the character generator, it was decided that a Teletype Model 33 ASR (Automatic Send-Receive) would be used to generate the ASCII character codes which would then be fed to the display via the computer or computer-refresh system. Thus only upper case (capital) letters would be generated. It is possible to generate all of the printing characters found on the keyboard with a 5 x 7 dot matrix.

If one had wished to generate lower case characters, at least a 7 x 9 matrix would be necessary to provide sufficient dot resolution for legible and distinct characters. But the addition of only two dots to the horizontal and vertical axes of the matrix means that a total of 63 dot positions would be required instead of 35. This means that approximately 75% more hardware would be required just to provide the basic dot matrix generation not to mention additional hardware required to select the greater number of dots which constitute each character.

In addition to increased hardware, speed becomes an important factor when a larger dot matrix is used. Since each dot is generated sequentially in time, it would require a 75% increase in the character generating time for a 7 x 9 matrix if the same dot frequency were used. Hence if a refresh system were to be used, the number of characters which could
Figure 2.1 Use of the Unique Pulse Train
be displayed would be nearly cut in half!

If one were to generate a 7 x 9 dot matrix at a higher speed, the cost of the logic circuitry would be increased. In addition to this, modulation of the Z-axis (intensity) of the c.r.t. becomes difficult at high frequencies. Since one wishes to generate dots and not smeared lines, intensification times must be much less than 100 nsec ($10^{-7}$ seconds) for the desired character generating speed. This factor would needlessly add to the cost of the display c.r.t.

2.2 The Unique Pulse Train

As the 35 positions of the 5 x 7 matrix are being generated sequentially with time, information must be fed to the c.r.t. intensification circuits to determine whether a particular dot is to be intensified or not. Fig. 2.1 illustrates this for the simple case of a horizontal and vertical sweep raster. As the raster is being generated sequentially in time, the unique pulse train for the letter A is being supplied to the Z axis. The resulting character is shown in Fig. 2.1d. Thus some method of generating a unique pulse train is required for all printing characters being generated by the display. This will be discussed in the remainder of this chapter.

2.3 General Description of the Character Dot Generator

The actual generation of the unique pulse trains for each character is common to all three character generators which are discussed in the remainder of this thesis. Only the control unit and the raster generator vary in the three systems and one slight modification must be made to a counter.

The character dot generator itself consists of a clock, a six bit
Figure 2.2 Block Diagram of Character Dot Generator
counter, a counter decoder, gating circuitry to select unique pulse trains for each character and a buffer to hold the ASCII code of the character whose pulse train is to be gated to the Z-axis. In addition, an inhibit gate is used to prevent the pulse train from reaching the Z-axis in special cases. A block diagram of the character dot generator is given in Fig. 2.2.

2.4 The Counter

A six bit binary up counter which is capable of counting up to \(2^6 = 64\) is used. Thirty-Five counts correspond to the 35 dots in the 5 x 7 matrix. However, space must be inserted between characters.

If a sweep circuit is used to provide the horizontal deflection, the beam is moving at a constant rate. To provide two horizontal raster units (dot positions) of spacing between characters, enough time must pass to provide for an additional 14 dots (i.e. two columns of 7 dots each). Hence 49 clock pulses must be counted. This type of counter is used by the raster generators which will be discussed in Chapter III.

If the horizontal beam position is produced by a binary counter and a D/A converter, only one additional pulse after the 35 matrix counts will be required to provide the space. This will be discussed in detail in Chapter IV.

A NAND gate detects a counter value of 35 for the D/A converter method or a count of 48 for the horizontal sweep method. The output of this gate enables the DCD gate of a pulse amplifier. Thus the 36th or the 49th clock pulse passes through the pulse amplifier. This pulse is used to clear the counter, the buffer and the "GENERATE" flip-flop which is a device busy flag in the control unit. When this flip-flop is set,
Figure 2.3 Counter for Horizontal Sweep Method
Figure 2.4 High Speed Counter for Horizontal D/A Converter Method
it enables the counter; clearing it disables the counter. The 36th clock
pulse also generates the character space in the case of the D/A converter
method. Fig. 2.3 illustrates the counter which would be used with a
horizontal sweep circuit. It is capable of operating at a clock frequency
of 1.08 MHz.

One serious problem which arose from using DEC 2 mo modules was
that each flip-flop has a nominal carry propagation of 70 nsec. This
means that one must wait for up to 420 nsec for the counter to reach a
steady state. But at 2 MHz, the clock pulses are 500 nsec apart. Hence
only 80 nsec is available during which intensification with its associ-
ated gating propagation delays can take place. This is insufficient.

To eliminate carry time a DC carry chain was used. This permits
the construction of a counter of any size with all flip-flops switching
simultaneously. It consists of six interconnected diode gates with two,
three,...,seven inputs each of which has an output only when the input
to that gate and to all the gates of lesser significance are in the one
state (i.e. a carry will occur on the next count). In this way the in-
put DCD gates of the flip-flops through which a carry would propagate
are enabled. Since the clock is fed to all flip-flops in parallel, those
with enabled DCD gates will be complimented simultaneously. Hence no
propagation delay will occur for a carry. Fig. 2.4 illustrates a 2 MHz
counter which would be used for the D/A converter method.

2.5 The Counter Decoder

A sequence of 35 pulses is required to intensify each dot of the
5 x 7 matrix. A series of 35, 7-input NAND gates is used in the counter
decoder. Six of the inputs are connected to the outputs of the six
Figure 2.5 The Counter Decoder

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
(a) The Basic 5 x 7 Dot Matrix

**Figure 2.6b** The Pulse Selection Gates
flip-flops of the counter in such a way that the first gate detects a count of one, the second a count of two, up to the thirty-fifth which detects a count of 35.

The seventh input to each of the gates is a delayed clock pulse. Refer to Fig. 2.5. The clock must be delayed because the clock pulses are first used to change the counter. After the counter is changed by the clock pulse and the 35 gates have reached a steady state condition, the delayed clock pulse reaches the gates. This prevents the ambiguity which could occur if the 35 decoder gates were sampled at the same time the counter was changing.

Thus each NAND gate will have an output when its particular count is present in the counter. Since only one gate can be enabled at any one time, the gates provide 35 distinct pulses which are sequential in time.

2.6 Pulse Selection Gates

Each character which is to be generated is composed of unique dots selected from the basic 5 x 7 matrix which is shown in Fig. 2.6a. As the various dot positions are generated by the raster generator (sweep or D/A converter), a decision as to whether to intensify a dot or not must be made for each particular character. This is done by the pulse selection NAND gates.

The inputs to these gates are the outputs of the counter decoder NAND gates. If a character requires dot #1 of the matrix, the output of the first counter decoder NAND gate is used. Hence the pulse selection gate for a particular character will have as many inputs as there are dots in the completed character. Since the inputs were outputs of
Figure 2.7 The Buffer Register
the preceding counter decoder NAND gates the following Boolean expression will be true if pulses 8, 9, 15, 16 were to be used:

\[
PULSE\ \text{TRAIN} = PULSE\ #8 \land PULSE\ #9 \land PULSE\ #15 \land PULSE\ #16
\]

\[
= PULSE\ #8 \lor PULSE\ #9 \lor PULSE\ #15 \lor PULSE\ #16
\]

Thus we see that the pulse selector NAND gates act as OR gates. The output of these gates will be a unique pulse train for each character to be generated.

At present a sixty character set of "printing" characters is used but this could be expanded to any size by the addition of more pulse selector NAND gates. A list of all the pulses required to form each character and a model of each character can be found in Appendix II. Samples of the gates are shown in Fig. 2.6b.

2.7 The Buffer Register

This register holds the ASCII code for the character which is currently being generated. Although the ASCII code requires 8 bits to define the full character set, not all bits need be utilized. If one only wishes to generate the printing characters of the ASR 33 Teletype, the six least significant bits of the code would be sufficient since \(2^6 = 64\) characters could be selected.

In our case, seven bits were used so that non-printing operations such as line feed, carriage return, tabulate and back space could be derived directly from the teletype keyboard. This also provides for expansion of up to \(2^7 = 128\) characters. A complete list of the ASCII codes used will be found in Appendix I.

The buffer which holds the ASCII code consists of seven flip-flops which can be loaded in parallel. The control of the loading operation
Figure 2.8 The Character Decoder
would be by computer or by a special control unit. This will be dis-
cussed further in the following chapters. The display buffer register
(DBR) could be the accumulator of the computer or the memory buffer
register of a refresh memory. Fig. 2.7 shows the buffer inputs as be-
ing the seven least significant bits of the refresh buffer which will
be described briefly in chapter VI.

2.8 The Character Decoder

The reader has probably noted that the unique pulse train for ev-
every character was generated simultaneously by the Pulse Selection Gates.
Some means of selecting only the character pulse train whose ASCII code
is contained in the buffer must be provided. This selection is per-
formed by the character decoder.

At present, sixty printing characters are generated and they only
require six bits of the buffer to define them uniquely. Thus the sev-
enth bit is not needed at this point and will be discussed further in
the chapter on Special Characters.

The character decoder is comprised of a set of sixty 7-input NAND
gates with a gate corresponding to each character. The fact that only
six bits define the sixty characters provides a substantial savings in
hardware since one less input is needed for each gate (an 8-input NAND
gate would have been required if all seven buffer bits were used). Six
of the inputs are fed by the output of the six least significant buffer
flip-flops. Each decodes a different ASCII code. The seventh input is
fed by the output of the Pulse Selection gate which generates the pulse
train for the character specified by the ASCII code on the other six
inputs. The character decoder is illustrated in Fig. 2.8. The follow-
ing Boolean expression applies:

\[
\text{DECODER OUTPUT} = \text{PULSE TRAIN} \land \text{ASCII CODE}
\]

or \[
\text{DECODER OUTPUT} = \text{PULSE TRAIN} \land \text{ASCII CODE}
\]

Hence the character decoder acts as an AND gate.

The outputs of the sixty character decoder gates must be 'OR'ed together and then fed to the intensification circuitry. This would require a sixty input 'OR' gate which obviously exceeds the fan-in capabilities of DEC 2 MHz circuits. Notice, however, that only one character decoder gate can be active at one time (i.e. their outputs are exclusive). By simply wiring the outputs of the sixty decoder gates together with a single clamped load we are able to perform the 'OR' operation and not use any logical hardware. In fact, we have freed 59 clamped loads which could be used elsewhere. Fig. 2.8 gives the logic diagram of the character decoder.
CHAPTER III

TWO METHODS OF BEAM POSITIONING USING AN ANALOGUE HORIZONTAL SWEEP

3.1 Introduction

In 1968, a project was begun by this department which has now brought us close to the completion of a complete graphics terminal. This will be discussed further in chapter VI.

Originally it was decided that a simple, low-cost character generator should be constructed which used a raster sweep method. At that time, the method of unique pulse generation which was described in chapter II was developed. A test model was constructed to demonstrate the feasibility of the method. It was then decided to incorporate the character generator into a larger, more sophisticated system.

During the subsequent development, three methods of raster generation have been tested. All utilize the same method of selecting the character pulse trains described in the preceding chapter. Each method arose as the project proceeded toward a goal of increased speed and flexibility.

The two methods of beam positioning which use a horizontal sweep circuit will be described in the remainder of this chapter.

3.2 Horizontal and Vertical Sweep Method

In 1968-69, another Master's candidate designed and constructed a limited test model of an A/N character generator which successfully generated the letters "A" and "C".

26
Figure 3.1 Output of Original Character Generator
His method of generating the vertical sweep was to utilize the saw-tooth output of a TEKTRONIX type 545 oscilloscope triggered by pulses 1, 8, 15, 22 and 29 which were derived from the counter decoder. This produced five linear sweeps which were fed to the Y-axis of a TEKTRONIX type 564 storage oscilloscope.

To provide a horizontal sweep, he used an integrate and hold circuit which charged a capacitor at a uniform rate during the actual generation of the character. The final voltage was held until the next character was generated and the capacitor would again be charged to a higher voltage. The output of this integrator was fed to the X-axis of the storage oscilloscope.

When the output of the character decoder 'OR' gate was fed to the Z-axis of the storage oscilloscope, characters could be generated as shown in Fig. 3.1.

While this method worked, it was severely limited in speed by the integrate and hold circuit. Due to the large capacitor which was used, more than 100 msec were required to discharge it. This meant that, while a line could be written at a fairly high character speed, a full tenth of a second was required before the next line could be written.

For a low cost character generator utilizing a storage c.r.t. the above raster generating technique would be sufficient. If a normal telephone channel capable of transmitting 1,200 bits/sec. were used to provide the ASCII codes to the character generator, this method would be ideal.

Of course, much faster sweep circuits can be constructed as can integrate and hold circuits. Since the vertical sweep circuit must only sweep out a single character height, it could be relatively non-linear.
Figure 3.2 Character Generator With Horizontal and Vertical Sweep

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
without resulting in noticeable character degradation. Hence a low cost circuit could be used for the vertical sweep circuit.

Instead of using the integrate and hold circuit to provide the horizontal beam positioning, a continuously operating horizontal sweep could be used if the ASCII character codes are supplied in a rapid sequence such as would occur in a refresh system. This sweep circuit must be quite linear since it must sweep over the entire width of the display screen without noticeable variations in character width.

During the testing of the pulse selecting circuitry for the full sixty character set described in chapter II, a temporary test circuit was constructed to provide the vertical and horizontal sweeps. This resulted in the display system illustrated in Fig. 3.2.

The sweep circuits were identical except for the RC time constants which controlled sweep time. In addition, they could be reset by a trigger pulse. This is especially desirable in the horizontal axis to provide a carriage return function (i.e. resetting the beam to the left-hand margin). The horizontal sweep could be reset in one character time so that when the ASCII code for a carriage return was decoded, no delay was required before the next character could be generated (recall the 1/10 second reset time of the integrate and hold circuit).

Appendix VII describes two different operating modes that were tested. The low speed circuits were set to generate 1,020 characters per second with a dot intensification of 10 μsec. This would be ideal for a storage system such as the TEKTRONIX type 601 storage display unit.

The higher speed circuits were used in a refresh mode and could supply up to 153 characters flicker free. This figure was chosen since the PDP-8/S computer was used as a refresh memory and it was not able
to supply data to the buffer at a faster rate. However, the vertical sweep circuit could operate effectively at 125 KHz. Thus a character rate of over 17,500 per second could have been attained which would permit 440 flicker-free characters to be generated.

Since it is not the purpose of this project to design the actual analogue sweep circuits, those which were used are not presented here. However, many sweep circuits are available which could fulfill the requirements mentioned above. A sweep circuit with a period of 3.5μsec. would permit a maximum clock frequency of 2 MHz which is the theoretical upper limit of the DEB logic circuitry used in the decoders. This would permit a character writing speed of 41,000 characters per second.

The generator was able to display several lines of text by the use of a 3 bit down counter which fed a D/A converter. Whenever the ASCII code for a line feed was decoded, the counter was decremented. The output of the D/A converter was summed with the vertical sweep circuit and the resulting analogue voltage was fed to the Y-axis of the oscilloscope. Obviously the number of lines could easily be varied by changing the number of bits in the counter and D/A converter.

The sweep method of horizontal and vertical raster generation provides a relatively low-cost display. It is rather limited in flexibility since one is unable to place a character at a particular position on the screen. For instance, if one wished to generate a character at the right-hand margin one must wait until the sweep circuit had reached that position on the screen. In addition, this method is not well suited to being included in a complete graphics terminal since very little of the character generator circuitry could be shared by the graphic generating circuits and because of the impossibility of random location of charac-
Figure 3.3 Horizontal Sweep and Vertical D/A Converter Method

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
Figure 3.4 Timing Waveforms for Horizontal Sweep and Vertical D/A Converter
ters. Also only one size of character is possible since the sweep circuits have constant sweep times.

3.3 Character Generator with Horizontal Sweep and Vertical D/A Converter

This method utilizes the same horizontal sweep circuit which was used above. However, the vertical raster generation is carried out solely through a D/A converter. If an 8 bit D/A converter is used a possibility of 256 vertical positions is possible.

A general block diagram of the system is shown in Fig. 3.3. An 8 bit up-down counter register which may be loaded in parallel provides the input levels to the D/A converter. When the counter is cleared to zero, the lowest line on the screen is accessed.

During the generation of a character, the least significant 3 bits of the counter are cleared by clock pulses 1, 8, 15, 22, 29 (note: these are not the delayed pulses which come from the counter decoder). A complete description of how this works will be found in Section 4.4. While the counter is being cleared, it is disabled to prevent accidental carries. Hence the vertical beam position after clearing will always occur at a position xxxxxxx000, where the x's can represent any one of $2^5 = 32$ character line positions.

The character is then generated by counting up six more positions from these lower points to provide for the seven vertical dot positions of the 5 x 7 matrix. Since this is repeated 5 times while the horizontal sweep is functioning, a raster may be formed as shown in Fig. 3.4e. By connecting the output of the character decoder to the Z-axis intensification circuits as before, selected dots may be intensified to produce a character.
A carriage return is generated by resetting the horizontal sweep circuit. A line feed may be produced by counting down in the 4th or 5th least significant bits to produce either a single or double space. This will be fully described in the chapter on Special Characters.

Although the use of a vertical D/A converter is more expensive than a vertical sweep circuit, it permits the character generator to be incorporated into a modest graphics terminal. Since the counter register may be loaded to any value by a parallel data transfer, random vertical dot positioning is also possible. Hence functions can be plotted with the horizontal sweep representing the independent variable.

Thus the use of a vertical D/A converter provides additional flexibility. Obviously more bits can be added to the vertical D/A converter to provide greater vertical resolution but this requires more circuitry and would therefore add to the cost of the display unit.
CHAPTER IV

BEAM POSITIONING BY USING D/A CONVERTERS

4.1 Introduction

Although the use of a vertical D/A converter with a horizontal sweep circuit provided increased flexibility, it is not well suited to a general graphics terminal. The goal of building a complete graphics terminal imposed several additional design constraints which had to be included in the final character generator.

First, a refresh system was to be used. This implies high character writing speeds. Recall that the horizontal sweep circuits required an additional $\frac{1}{4}$ dot times to generate the spacing between characters. The additional 40% which must be added to the character generating time was actually wasted in terms of the useful information which was being displayed. Obviously it would be better to provide the spacing in one or two dot times.

Second, since a graphics capability was to be added, it was desirable to be able to share some of the dot positioning circuitry of the character raster generator. This means that a completely random dot positioning capability must be provided in both the horizontal and vertical direction.

Third, three character sizes would be an asset to a graphics terminal. If sweep circuits were used for character generation, their time constants would have to be altered under digital control. The
Figure 4.1 One Possible Method Using D/A Converters
associated circuitry would become very complex and the casual user of
the system would probably have difficulty keeping track of the current
beam position if the sweep time varied from character to character.
Also subscripting and superscripting should be provided by hardware
rather than software.

Finally, we wished to make the operation of the character generator
trivial to the user who is not familiar with a computer system and still
permit him to use the full capability of the character generator through
the keyboard and a few additional function keys.

If two up-down-counter registers are used so that one controls a
vertical D/A converter as before, and the other supplies input levels
to a horizontal D/A converter, completely random positioning of the beam
anywhere on the screen is possible. Thus all of the above design con­
straints can be readily satisfied.

4.2 One Possible Method

Fig. 4.1 very briefly illustrates a possible method of beam position­
ing which utilizes horizontal and vertical D/A converters. The character
raster would be generated by two 3-bit up counters each controlling a
D/A converter. The vertical counter would initially be cleared. It
would then count up six times, be reset, count up six times, etc. Re­
peating this 5 times would produce 5 columns of 7 dots. If the horizon­
tal D/A converter were initially cleared and then incremented whenever
the vertical counter was reset, the 5 x 7 dot raster would be generated.

The output of the two D/A converters could be amplified to provide
different character sizes. Then the resulting voltage would be summed
with the appropriate X and Y position voltages to yield the final beam
positioning voltages. Character positioning would be completely random.

Spacing must be provided by changing the value of the horizontal position register before the next character is to be generated. Similarly, line feeds as well as positioning for subscripting and superscripting would be provided by changing the value of the vertical position register.

This method requires four D/A converters, two scaling amplifiers, two summing amplifiers and complex circuitry to coordinate the two registers which are associated with each axis. If one were to use a c.r.t. with both electrostatic and electromagnetic deflection, this method might be advantageous. The beam could be positioned electromagnetically and the character raster generated at this position electrostatically. This would eliminate the summing amplifiers since the deflection systems would provide the summing function.

This method of beam positioning has the advantage of completely random dot and character positioning. However, keeping track of the current beam position to provide for automatic subscripting, superscripting and variable spacing between different sized characters could be excessively complicated.

4.3 The General Method Used

From the outset, we had decided to use electrostatic beam deflection alone. The above method would, therefore, be very wasteful since 13 bits of D/A converter yield only 10 bits of actual beam positioning due to summation of the output voltages. In addition, scaling and summation require expensive amplifiers.

One other factor must also be considered. Is it really necessary
Figure 4.2 (a) Normal Counter

Figure 4.2 (b) Counter Capable of Counting in Any Position

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
to be able to provide completely random character positioning? Perhaps it would be sufficient to provide for one of 64 possible lines for vertical character positioning. It would certainly simplify subscripting and superscripting. This was the approach which was chosen.

The problem now became one of using the position register to also provide the character raster and scaling. Admittedly, the control circuitry would be complex but it would be less costly than the operational amplifier and D/A converter required to implement the other method.

The normal method of operating a counter is to always count in the least significant bit position and let the output carry over to higher significant bits for successive counts. Refer to Fig. 4.2a.

The method which was used for the X & Y register permits pulses to be gated by control circuitry to the appropriate counter bit. Hence, instead of counting two pulses in the unity position, a single count in the two's position yields the same final value in the counter. Thus one can cause large changes in the value of the counter with a single pulse (i.e. a single pulse in the 64's position is the same as 64 pulses in the 1's position). Fig. 4.2b illustrates this. Notice that each bit can be triggered to count by either a carry from a lesser significant bit or by the count pulse being gated to its trigger. The pulse amplifier (PA) is necessary to convert the carry levels to pulses which are used to trigger the next flip-flop. If a true level were permitted to remain on the OR gate no further pulses would be able to pass through it.

The actual implementation of the X and Y registers and their associated control circuitry is quite complicated and there is substantial
overlapping between normal characters, special characters and graphics. For this reason, the description of these circuits will be continued into the chapter on Special Characters. It will also be included in the general block diagram of the graphics terminal which will be discussed briefly in Chapter VI.

4.4 Character Size Selection

Before continuing with the description of the registers, two important considerations must be introduced. First, the terminal can recognize two types of 12 bit information words; an instruction word and a data word. The instruction word provides information to set the display into a particular operating mode. For example, the mode could be alphabetic-numeric or any one of several graphics modes. Once a mode has been selected, all data words which follow are interpreted as data in the context of that particular mode. As many data words may follow as the user of the system wishes and each will be interpreted the same way until another instruction word is encountered. For instance, the data word could contain the 7-bit ASCII code plus other information about the character. A description of the A/N data and instruction words will be found in Appendix IV.

The second consideration is that of character size. The X and Y registers are each 10 bits in length. Thus \(2^{10} = 1,024\) distinct positions can be selected along each axis. This means that on a 10 inch square screen, the dot spacing (raster unit) would be \(\frac{10}{1,024} \approx \frac{1}{100}\) inch. Since a character is seven dots high (six spaces between them), if every dot were to be used, a character would only be about \(\frac{6}{100} = \frac{1}{16}\) of an inch high. Obviously this is too small.
Figure 4.3 Size Selection Circuitry
Three character sizes can be generated by the character generator. The smallest uses every second dot position, the normal character uses every fourth dot position and the largest uses every eighth position. Hence the characters are about $1/8$, $1/4$ and $1/2$ inch high. Appendix III illustrates the exact relationship between character size and position on the screen.

Now one can see the advantage of having the position register capable of counting in the 2's, 4's or 8's position. All characters may still be generated by the same 35 clock pulses but they will be gated to the appropriate counter bit according to the desired character size.

Fig. 4.3 illustrates the circuitry required to select the proper character sizes. The middle sized character is the normal character. The smallest character may be positioned with respect to the normal character as either a subscript or a superscript by hardware. The largest character could be used as a capital letter for the normal characters.

This is suitable for a 10" display screen but what about a smaller screen? If the screen were 5" high, the smallest character would be $1/16$" high. Thus, when selecting the A/N mode, the user has the option of specifying a double mode. In this mode, the smallest character is no longer available. The double sized (Capital) character becomes the normal character, and the middle sized character may be hardware positioned as a subscript or a superscript.

Referring to Fig. 4.3, one can see three flip-flops. The double flip-flop can be set during a change to the A/N mode. It is always cleared first and then if bit 8 of the display buffer is in the '1' state it will be set.

The other two flip-flops are set by the data word. They cannot be
cleared before they are loaded as was the case for the "DOUBLE" flip-flop. This is because the 36th clock pulse (which is used to clear the data register), is also used to provide the proper spacing between characters. Since character spacing varies with character size, we cannot destroy the status of these two flip-flops. Thus the inverters (labelled 16 and 17) permit these two flip-flops to be loaded in a "jam-transfer" mode.

The flip-flop labelled "SUPER" is ignored by the gating circuitry unless the "SCRIPT" flip-flop is set. Thus if the "SCRIPT" flip-flop is cleared, a normal character will be generated. Recall, however, that we wish to have three character sizes including a double size character (capital). If the "DOUBLE" flip-flop was not set during instruction time we must accomplish this in another manner. This is done by NAND gate #1 as follows:--

\[ \text{CAPITAL} = \text{SCRIPT} \land \text{SUPER} \]

The output of this gate plus the "0" output of the "DOUBLE" flip-flop are 'OR'ed in the NAND gate #2:--

\[ \text{DOUBLE} = \text{DOUBLE} \land \text{CAPITAL} \]

or \[ \text{DOUBLE} = \text{DOUBLE} \lor \text{CAPITAL} \]

Thus the double size character may be chosen in either the double or the normal mode.

The outputs of the final character size selection gates are as follows:--

\[ \text{DOUBLE} \land \text{SCRIPT} = \text{DOUBLE} \]

\[ \text{NORMAL} \land \text{DOUBLE} \land \text{SCRIPT} = \text{NORMAL} \lor (\text{DOUBLE} \land \text{SCRIPT}) \]

\[ \text{NORMAL} \land (\text{SUB} \land \text{SCRIPT}) \]

\[ \text{NORMAL} \land (\text{SUPER} \land \text{SCRIPT}) \]
Figure 4.4a Dot Matrix

Figure 4.4b Character Count Control Circuitry

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
Thus we have five decoded outputs which indicate:

(a) a double sized character
(b) a normal character or a double scripted character
   (which are actually the same character)
(c) a normal subscript
(d) a normal superscript
(e) a double subscript
(f) a double superscript

These outputs provide levels which are used by the X and Y count selection circuitry.

4.5 Character Count Control Circuitry

To generate the 5 x 7 dot raster, the 35 clock pulses must be gated to the X and Y registers at the appropriate times. This is done by the "character count control circuitry". Recall that the intensification signals which were derived from the character decoder were actually delayed with respect to the clock to prevent ambiguity while the counter was changing. Actually, we must also ensure that the X and Y D/A converters have reached a steady state voltage (i.e. the beam has settled in position) before we intensify the dot. Thus we must use the actual clock pulses to count in the X and Y position register. Fig. 4.4b shows the required circuit.

The beam of the c.r.t. must first be initialized to one of the possible vertical positions for character generation (i.e. position #1). Then six clock pulses are supplied to the Y register at the appropriate count position to generate the vertical column of dots 2 to 7. Refer
to Fig. 4.4a. The beam must then be vertically reset to the initial
position and spaced over horizontally to the right (i.e. position #8).
This is repeated four times to yield the 35 dots of the matrix.

The load data pulse triggers the delay #1 which is actually a pulse
stretcher providing a level for the length of the delay time. The level
which is produced is inverted and supplied to NAND gates #3 and #5. This
inhibits the counting in the Y direction and enables the beam to be re-
set vertically. The X count is also disabled at this time.

NAND gate #2 detects delayed clock pulses from the "counter decoder".
Pulses #7, #14, #21 and #28 are effectively 'OR'ed as follows:

$$\#7 \land \#14 \land \#21 \land \#28 = \#7 \lor \#14 \lor \#21 \lor \#28$$
The output is inverted and used to trigger delay #2. The output of this
delay is inverted and supplied to NAND gates #1 and #3. It is also fed
without inverting to gate #4. The delay is enabled by the "X count en-
able" level derived from the special character decoder. It is only dis-
abled if a NULL (zero ASCII) code is decoded.

NAND gate #1 'OR's the inverted outputs of the two delays and enables
gate #5 which has the clock at its other input. Thus gate #5 allows
the first clock pulse which arrives after LD A/N DATA or #7 or #14 or
#21 or #28 to pass through (i.e. 1, 8, 15, 22, 29). The output pulse
will be referred to as "Y CHAR RESET" and is used to initially position
the beam at the bottom of the five columns in the 5 x 7 dot matrix.

The NAND gate labelled #4 is enabled by the output of delay #2 and
gates the first clock pulse to arrive after the enable level to increment
the X-counter (i.e. 8, 15, 22, 29). The output pulses are referred to
as "X CHAR COUNT".

GATE #3 is enabled whenever either delay is not active. The delays
are adjusted so that they are active only long enough to permit one clock pulse to reset the Y counter and increment the X counter. Thus all clock pulses except 1, 8, 15, 22 and 29 are allowed to pass through gate #3. The output pulses are referred to as "Y CHAR COUNT" and when gated to the Y counter, provide the pulses which increment the counter.

The output of gate #1 is also inverted and called the "Y RESET LEVEL". This is required to prevent the Y counter from performing a carry while it is being reset.

Timing diagrams are provided in Appendix V.

4.6 X Register Count Selection

Table 4.1 lists the requirements of the X register.

<table>
<thead>
<tr>
<th>COUNT POSITION</th>
<th>UP</th>
<th>DOWN</th>
<th>USE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>✓</td>
<td>✓</td>
<td>GRAPHIC 1's</td>
</tr>
<tr>
<td>2</td>
<td>✓</td>
<td>✓</td>
<td>1's CARRY</td>
</tr>
<tr>
<td>4</td>
<td>✓</td>
<td>✓</td>
<td>GRAPHIC 2's small char</td>
</tr>
<tr>
<td>8</td>
<td>✓</td>
<td>✓</td>
<td>2's CARRY</td>
</tr>
<tr>
<td>16</td>
<td>✓</td>
<td>✓</td>
<td>GRAPHIC 4's medium char small space</td>
</tr>
<tr>
<td>64</td>
<td>✓</td>
<td>✓</td>
<td>4's CARRY</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>8's CARRY</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>32's CARRY</td>
</tr>
</tbody>
</table>

X Register Counting Capabilities

TABLE 4.1

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
Figure 4.5 X Count Selection Circuitry
The graphic inputs (i.e. GR 2's) have been provided for use by the graphics portion of the terminal and will not be discussed here. The "TAB COUNT 64" input will be discussed in the chapter on Special Characters. Fig. 4.5 illustrates the required circuitry to select the appropriate counting position. The 36th clock pulse which clears the buffer and counter is also used to provide the "CHAR SPACE" pulse.

The carry inputs to the NAND gates are not directly connected to the output of the preceding flip-flop. Since the carry outputs from flip-flops are levels, they would prevent any other counting pulses from passing through the gate if they were permitted to remain positive. Thus the output of the flip-flop is used to trigger a pulse amplifier which only generates a 100 nsec pulse and then returns to a negative level. It is this pulse which is connected to the input of the NAND gate (i.e. l's CARRY UP). Note that the X register is an up-down counter. The up carry pulse amplifier is triggered when the '1' output of the flip-flop goes to ground. The down carry pulse amplifier is triggered when the '0' output goes to ground.

The "X COUNT" outputs are the inputs to the various counting positions of the X register.

4.7 The X Register

The X register is a 10 bit up-down counter which is also capable of being loaded in parallel. The circuit is shown in Fig. 4.6.

The flip-flop outputs which are used to trigger carry pulse amplifiers (i.e. l's CARRY UP) are shown leaving the flip-flop but the pulse amplifiers are not shown.

The inputs labelled "DOWN COUNT X" and "UP COUNT X" are derived from
Figure 4.6a X Register (Least Significant Bits)
Figure 4.6b X Register (Most Significant Bits)
control circuitry which will be discussed later. The input labelled "LOAD X" is used for graphics as are the 10 level inputs "DBR #2, ..., DBR #11". The "Display Buffer Register" (DBR) is a 12 bit memory buffer register which is used to accept data and instruction words being read from the refresh memory.

The "CLEAR X" input is also a graphics operation which always precedes the loading of the register. The "CLEAR X" pulse is also used by the carriage return which will be discussed as a Special Character in the next chapter.

The "1" outputs of the ten flip-flops are connected to the digital inputs of the 10 bit D/A converter. This yields the final analogue voltage which is used to drive the X deflection circuitry of the c.r.t.

4.8 Y Register Count Selection

This circuitry functions in exactly the same manner as the X-register count selection circuitry but provides different counting capabilities. Table 4.2 lists the counting requirements of the Y register.

<table>
<thead>
<tr>
<th>COUNT</th>
<th>POSITION</th>
<th>UP</th>
<th>DOWN</th>
<th>USE</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>GRAPHIC 1's</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>1's CARRY</td>
<td>GRAPHIC 2's</td>
</tr>
<tr>
<td>4</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>2's CARRY</td>
<td>small char</td>
</tr>
<tr>
<td>8</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>4's CARRY</td>
<td>medium char</td>
</tr>
<tr>
<td>16</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>8's CARRY</td>
<td>double char</td>
</tr>
<tr>
<td>32</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>16's CARRY</td>
<td>line feed</td>
</tr>
<tr>
<td>64</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>32's CARRY</td>
<td>line feed</td>
</tr>
<tr>
<td>128</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>64's CARRY</td>
<td>line feed</td>
</tr>
</tbody>
</table>

Table 4.2 Y Register Counting Capabilities
<table>
<thead>
<tr>
<th>Character Size</th>
<th>Character Type</th>
<th>Initial Y Position</th>
<th># of Possible Positions</th>
<th># of lines of text for each character size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Largest</td>
<td>DOUBLE</td>
<td>xxxxx000000</td>
<td>16</td>
<td>16</td>
</tr>
<tr>
<td>Medium</td>
<td>DOUBLE SUBSCRIPT OR NORMAL</td>
<td>xxxxx000000</td>
<td>16</td>
<td>32</td>
</tr>
<tr>
<td></td>
<td>DOUBLE SUPERSCRIPT</td>
<td>xxxxx100000</td>
<td>16</td>
<td></td>
</tr>
<tr>
<td>Smallest</td>
<td>NORMAL SUBSCRIPT</td>
<td>xxxxx000000</td>
<td>32</td>
<td>64</td>
</tr>
<tr>
<td></td>
<td>NORMAL SUPERSCRIPT</td>
<td>xxxxx100000</td>
<td>32</td>
<td></td>
</tr>
</tbody>
</table>

Y-Reset Position

TABLE 4.3

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
Figure 4.7 Y Count Selection Circuitry
Figure 4.8 Y-RESET Circuitry
As one can see, four sizes of line feeds are possible. This occurs since a single spaced or double spaced line feed may be user specified and the size of spacing depends on the character size specified. The "DOUBLE SPACE", "SINGLE SPACE" and "LINE" control signals are derived from the Special Character Circuitry. Fig. 4.7 illustrates the Y Count Selection circuitry.

4.9 Y-Reset Circuitry

It was previously mentioned that as the character matrix is being generated the Y register must be reset five times to some initial value depending on the vertical positioning and size of the character. In addition to this, hardware generated subscripts and superscripts must be properly positioned with respect to the main line of text.

The Y-RESET circuitry is used to gate the Y-RESET pulse (which is generated by the Character Count Control circuitry) to the Y register in such a way as to correctly initialize each of the 5 vertical columns of dots which comprise the dot matrix. Appendix III illustrates all possible character positions and Table 4.3 lists the initial values which must be loaded into the Y register for different characters.

Fig. 4.8 illustrates the circuitry which is required to reset the Y counter. The outputs of the 4 reset gates are connected to the outputs of the appropriate flip-flops. For example, the DOUBLE-SUPER-SCRIPT level and the Y reset pulse are gated by a NAND gate to the 32 collector. This means that the output of the gate is connected to the "ZERO" output of the 32's flip-flop. When this is forced to the "ZERO" state by the NAND gate the other side of the flip-flop is set to the "one" state.

The "CLEAR Y" pulse is a graphics pulse which is used to clear the
Figure 4.9a Y Register (Least Significant Bits)
Figure 4.9b Y Register (Most Significant Bits)
Y register. Notice that it clears the 512, 256, 128, 64, 32, 16 flip-flops of the Y register directly but is NANDed with the Y-RESET pulse to clear the 8, 4, 2, 1 flip-flops. This NAND gate acts as an OR gate so that the least significant 4 flip-flops may be cleared during character generation without changing the values of the other 6 flip-flops.

4.10 The Y Register

The Y register is a 10 bit up-down counter which is capable of being loaded in parallel. Fig. 4.9a and b illustrates the circuit.

The "UP Count Y" and "DOWN Count Y" both are derived from the UP-DOWN count control circuitry. The "LOAD Y" pulse is a graphics pulse which is used to strobe the contents of DER bits 2-11 into the Y register in parallel.

The "1" outputs of the flip-flops drive the 10 bit Y D/A converter. This D/A converter yields the final analogue voltage which is fed to the Y-axis driving circuits of the c.r.t.

4.11 UP-DOWN Count Selection Circuitry

This circuitry is required to control the X and Y registers. If one is generating a normal character, both the X and Y registers must count up. But in special cases either or both may be required to count down.

Fig. 4.10 illustrates the circuitry. The 4 DCD gate inputs to the flip-flops are used for the graphics "DOT" mode and will not be discussed here.

When the A/N buffer is being loaded, the two flip-flops are also set to the "one" state in anticipation of a normal character. If the character is to be a "back space", pulse #1 from the counter decoder
Figure 4.10 UP-DOWN Count Selection Circuitry
clears the X count flip-flop so that it will count down instead of up. Similarly a "line feed" (LF) or a "carriage return" (CR) or a NULL (all zero ASCII code) will enable pulse #1 to set the down count of the Y count flip-flop. The "tabulate" (TAB) will permit pulse #14 to also clear the Y flip-flop for a down count. The actual generation of these four special characters will be discussed in detail in the next chapter. Notice, however, that either the "Y reset level" or the "Inhibit Normal Character" level will disable the Y up count. During special character generation, the Y register must not count in the UP direction. Also, while the Y register is being reset by the "Y reset pulse", the Y register must be prevented from counting since clearing a flip-flop could cause a carry to propagate through the counter.

In addition to the above cases, the UP output of the X count flip-flop is also gated. If the Y register is set to a down count and the "A/N MODE" level is present, the gating inhibits the X count up, since if any input to the NAND gate is a zero the output will be a one (negative) which cannot be used to enable the counter.

Notice that the flip-flops are always set during buffer loading but may be modified later. The flip-flops are not cleared at the end of generating a character by pulse #36 as the buffer is. This is necessary to keep the X count up enabled during the generation of the character space.
CHAPTER V

SPECIAL CHARACTERS

5.1 Introduction

In addition to the sixty printing characters which can be generated four additional characters can be recognized. These are the carriage return which resets the beam to the left hand margin, the line feed which moves the beam down to a new line, the back space, and the horizontal tabulate which permits eight vertical columns to be automatically left justified.

These characters have been implemented to permit the beam to be rapidly positioned at the proper point on the screen solely through the keyboard of the Teletype. This makes it easy for the user to operate the text generator. He is not consciously required to load the X and Y register with actual numeric values since the hardware does this for him.

The left-hand margin can also be selected by the user. Thus if he wishes to generate a column at the right-hand side of the screen, he need not generate a carriage return-line feed to the left margin and then have to space across to the right again. This reduces the number of characters which must be refreshed so that more useful information may be displayed.

5.2 Special Character Decoder

Recall that only six bits of the seven bit buffer were used by the
Figure 5.1 Special Character Decoder

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
character decoder to determine which character was to be generated. All seven bits are required for the special character gates. These gates are also seven input NAND gates since they are not required to gate a unique pulse train (i.e. they are non-printing).

Fig. 5.1 illustrates the special character decoder. In addition to the carriage return, line feed, back space and tabulate, the null or zero ASCII code is detected. This permits the display to remain in the A/N mode for a fixed refresh time by cycling through an area of memory of constant length. If there is not enough information to fill this block of memory, the vacant location can be filled with zeros. When they are transferred to the A/N generator no character will be generated but the clock will run for the same time required by any other character. This is done by permitting pulse #1 to set the Y COUNT selection flip-flop of Fig. 4.10 to the DOWN state. This will then prevent any counting in the X register so that no displacement will be possible in the X direction during a NULL.

The outputs of the 5 decoding gates are "OR"ed by the 5 input NAND gate. The output of this is inverted yielding a positive output. This is sent to the "INHIBIT NORMAL CHARACTER" input of Fig. 4.10 preventing the Y register from counting up. In addition, this level inhibits the pulse train from the character decoder from reaching the Z-axis. This is necessary since the six bit code detected by the character decoder will not uniquely decode the seven bit code being used. For example, the carriage return ASCII code is 215 while the letter M is 315. The least significant six binary bits are 001 101 in both cases. Thus the character decoder will try to generate a unique pulse train for the letter M in both cases. However, if bit seven is a "0" as is the case
Figure 5.2a Generation of the Line Feed Pulse

Figure 5.2b Use Margin Register and Space Flip-Flop
for the carriage return, the pulse train is inhibited and does not intensify the Z-axis while the carriage return is being executed.

5.3 The Back Space

This is the simplest of the special characters. Referring again to Fig. 4.10, one can see that the BACK SPACE output of the special character decoder enables pulse #1 to set the X register to count down. Thus as the character is generated the five vertical columns will move from right to left followed by the appropriate spacing also to the left. Notice that since we are counting down instead of up and no other changes are made, we can still specify a character size for the back space. Since no intensification takes place, this character is not visible while it is being generated. The back space permits the user to place superscripts directly over subscripts or to produce new characters by superimposing several characters.

5.4 The Line Feed

When the Y count control was discussed, the gating for the line feed was given (See Fig. 4.7c). Depending on the character size and whether a single or double spaced line feed was required, the Y register was made to count down in the proper position. Pulse #1 was gated to select the Y down count (See Fig. 4.10) and the circuitry of Fig. 5.2 generates the appropriate control signals.

When the line feed is generated, it is normally double spaced. However, if the user desires he sets bit #4 of the data word which in turn sets the SINGLE SPACE flip-flop during the loading of the A/N buffer. Thus a single spaced line feed may be generated.
5.5 The Carriage Return and Tabulate

1. Carriage Return

These two characters share the same circuitry and, therefore, must be discussed together. It was decided that the tabulate should move the beam to every tenth position of the smallest size character regardless of the current beam position or size of character being generated. Since a maximum of eighty-five small characters can be placed on a line, eight possible positions 10, 20, ..., 70, 80 may be used. The same positions are possible left margin settings so that a total of 9 carriage return positions including the extreme left are possible. Table 5.1 lists the actual X register values for these positions.

Since each small character takes 12 horizontal raster positions, ten characters are 120 raster positions.

<table>
<thead>
<tr>
<th>DECIMAL VALUE</th>
<th>OCTAL VALUE</th>
<th>DECIMAL VALUE</th>
<th>OCTAL VALUE</th>
<th>BINARY VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>120</td>
<td>170</td>
<td>0</td>
<td>000</td>
<td>0 0 0 1 1 1 0 0 0 0 0</td>
</tr>
<tr>
<td>240</td>
<td>360</td>
<td>0</td>
<td>000</td>
<td>0 0 1 1 1 1 0 0 0 0 0</td>
</tr>
<tr>
<td>360</td>
<td>550</td>
<td>0</td>
<td>000</td>
<td>0 1 0 1 1 0 1 0 0 0 0</td>
</tr>
<tr>
<td>480</td>
<td>740</td>
<td>0</td>
<td>000</td>
<td>0 1 1 1 1 0 0 0 0 0 0</td>
</tr>
<tr>
<td>600</td>
<td>1130</td>
<td>1</td>
<td>000</td>
<td>1 0 0 1 0 1 1 0 0 0 0</td>
</tr>
<tr>
<td>720</td>
<td>1320</td>
<td>1</td>
<td>000</td>
<td>1 0 1 1 0 1 0 0 0 0 0</td>
</tr>
<tr>
<td>840</td>
<td>1510</td>
<td>1</td>
<td>000</td>
<td>1 1 0 1 0 0 1 0 0 0 0</td>
</tr>
<tr>
<td>960</td>
<td>1700</td>
<td>1</td>
<td>000</td>
<td>1 1 1 1 0 0 0 0 0 0 0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>MARGIN REGISTER CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>N/A</td>
</tr>
</tbody>
</table>

TABLE 5.1 Tabulate and Margin Positions
Figure 5.3 X Register Clear Circuitry
Figure 5.4  Circuitry for Loading X Register During Carriage Return and Tabulate
One can immediately observe 3 characteristics of the binary values:

(a) The 64 flip-flop always contains a "1"
(b) Bits 4, 2, 1 are always zero
(c) $512 = \overline{32}$ (i.e. 512 is the compliment of 32)

Further observe that $8 = 2^3$. Hence 3 flip-flops can be used to uniquely select each of the 8 positions.

Let us deal first with a normal carriage return to the left margin. Pulse #1 sets the Y count down (See Fig. 4.10). This in turn prevents the X counter from counting up even though the flip-flop is set to count up. Obviously, then, the X counter cannot count at all. The carriage return can be accomplished simply by clearing the X register. Fig. 5.3 illustrates the required circuitry. The CR level from the special character decoder enables pulse #7 to directly clear 512, . . . , 64 of the X register and by further gating to clear 32, . . . , 1 of the X register.

A carriage return to some other position requires further circuitry. Fig. 5.4 shows circuitry required to load the X register during a carriage return to a preset margin or a tabulation.

The Margin Register consists of three flip-flops labelled 1, 2, 3 with 3 being the most significant bit. If, when the A/N instruction is decoded, bit five of the instruction word is a "one", the LOAD MARGIN REGISTER is set. Setting the flip-flop triggers a pulse amplifier which clears the Margin Register. The next data word following the instruction is then loaded into the A/N buffer. In this case the Margin Register is
an extension of the buffer. A NULL ASCII code is loaded into the regular buffer and inhibits generating a character. The LOAD MARGIN REGISTER flip-flop enables a NAND gate and permits the LOAD A/N DATA pulse to strobe the contents of bits 2, 3 and 4 of the DER into the Margin Register. Pulse #7 is used to clear the LOAD MARGIN REGISTER flip-flop to prevent any further changes in the Margin Register until another A/N instruction is decoded. Hence subsequent data words contain standard A/N data. Pulse #7 also terminates the data cycle by clearing the counter, the buffer and the GENERATE flip-flop since the remainder of the cycle is idle.

Now if one wishes to generate a carriage return and use the Margin Register, bit #4 of the data word, previously set to "1", sets the same flip-flop which was used for the SINGLE SPACED line feed. The output of this is 'NAND'ed with the CR level of the special character decoder.

Referring to Fig. 5.3, one can see that pulse #7 will still be gated by the CR level to clear the X register. Then pulse #28 will be gated by NAND gates 1, 2,...,6, 7 (depending on the status of the margin register flip-flops) to the "0" collectors of the appropriate X register flip-flops. Recall that this acts as a direct set for the particular slip-flop. Notice that gates 1, 2, 3 strobe the contents of the Margin Register into the 512, 256 and 128 bits of the X register. Gate 4 always sets the 64 bit. Gates 5, 6 and 7 strobe the compliment of the Margin Register into the 32, 16 and 8 bits of the X register and bits 4, 2, 1 remain cleared. Thus the beam has been positioned according to Table 5.1.

2. Tabulate

The tabulate works in much the same way as a carriage return
which uses the Margin Register. However, it is further complicated by the fact that no matter where the beam is currently located, it must move to the next higher position given in Table 5.1. For example, if the beam is currently at octal position 234 it must be tabulated to position 360. If at position 344, we must still tabulate to position 360. Since the tabulate is to be performed by hardware, we cannot read the contents of the X register into the computer, execute an algorithm to determine the next tabulate position and then reload the X register. In addition, the graphics terminal has no arithmetic unit so that some other hardware algorithm which does not use addition or subtraction must be found.

Indeed, one very simple algorithm was found but this fails to work in every case. Consequently, four special gates act as subprograms to the main algorithm to yield the correct value. The strategy is to treat the most significant 3 bits of the X register as the margin register which was used by the carriage return. If one can successfully place the correct value in those three bits, they can uniquely define the new tabulate position.

Recall that the smallest character is 12 raster units wide. Thus we need not consider bits 4, 2 and 1 of the X register in our algorithm since any combination of these three bits is less than one character width.

If we ignore bit 8 of the X register, it could in some cases yield a value in combination with bits 4, 2, 1 which would be greater than the width of a character and lead to erroneous results. Several algorithms which consider only the seven most significant bits of the X register were considered. No general algorithm was found which worked
START

512 = 32, 256 = 16, 128 = 8, 64 = a one?

Yes

Increment 64's position

A

No

B

512 = 16 = a one?

Yes

256 = 32 = a one?

Yes

128 = 64 = 32 = 16 = a one?

Yes

512 = 256 = 64 = 32 = 16 = a one?

Yes

FINISHED

64 = zero?

No

Increment 64's position

Clear 32, ..., 1 of X Register

Load 512 into 32
Load 256 into 16
Load 128 into 8

Fig. 5.5 Tabulate Algorithm
Figure 5.6 Circuitry Required to Generate 64's Count During Tabulation
in all cases. This is due in part to the fact that the 512 bit is a "zero" for half the algorithm and "one" for the remainder, but the midpoint of the screen is not one of the tabulate positions.

Extensive computer simulation developed an algorithm which failed in only three character positions. Two gates are used to detect these cases and make corrections. Fig. 5.5 gives the algorithm used for tabulation. The circuitry required to place the correct values in 512, 256, 128 and 64 is shown in Fig. 5.6.

The 3 exclusive-OR circuits are used to compare bit pairs of the X register. Only if both inputs are the same will an exclusive-OR yield an output. The 3 circuits are interconnected so that their output is true only when all three pairs match. Gate a yields a positive output only when

\[(512 = 32) \land (256 = 16) \land (128 = 8) \land (64 \text{ is a one})\]

This actually detects when the beam is already within one character width of the tabulate position. When this is true we wish to go to the next position. Thus the gate enables pulse #7 to count one in the 64's position. This of course makes the 64 bit a zero (i.e. 64 is a one). Thus pulse #14 is able to cause a further count in the 64's position. These two counts have guaranteed that the most significant 3 bits of the X register will have been incremented to the next tabulate position. Thus if any of the gating determines that a count in the 64's position is necessary, a new value will always be set into the most significant four bits. Notice that the 64 bit always is set to the "one" state.

Once we have the new value in bits 512, 256 and 128 of the X register we have uniquely defined the new tabulate position and we are ready to load the least significant bits of the X register. Referring
to Fig. 5.3, we can see that pulse #21 will clear bits 32, 16, ..., 2, 1 of the X register.

Gates 8, 9 and 10 of Fig. 5.4 enable pulse #28 to load $\overline{512}$ into bit 32, $\overline{256}$ into bit 16 and $\overline{128}$ into bit 8 of the X register.

Hence, the algorithm has been completed. A new value was placed in 512, 256 and 128 by counting in the 64's position. The 64 bit was left in the "one" state. The compliments of 512, 256 and 128 were placed in 32, 16 and 8 respectively and bits 4, 2 and 1 were left in the "zero" state. Reference to table 5.1 will convince the reader that the c.r.t. beam has indeed been positioned at the next tabulate position.
CHAPTER VI

CONTROL AND TESTING OF THE A/N GENERATOR

6.1 Refresh and Control of the A/N Display

The A/N generator was designed specifically to operate with the PDP-8/S digital computer. Control of the generator is accomplished by 3 pulses:

(a) SET A/N MODE
(b) CLEAR INSTRUCTION
(c) LOAD DATA

In the I/O terminal, these pulses are supplied by the control unit associated with the refresh memory. However, the PDP-8 family of computers has the capability of generating 3 input-output pulses (IOP) for a given device code. Hence, if one wishes, the computer could be used as the refresh memory by having it loop through a portion of memory containing the display information. The computer must execute a minimum of 5 instructions per data word. This requires 220 \( \mu \text{sec} \) for the PDP-8/S and limits the display to 4,500 characters/sec. Thus only 150 characters may be displayed if a 30 Hz refresh rate is used. It is for this reason that we require a separate refresh system which can supply data at a much higher rate.

However, if one were to use a PDP-8/L or PDP-8/I computer with a negative logic bus, the computer alone could be used for a refresh memory. The same five instructions require only 16 to 20 \( \mu \text{sec} \) on these.
Figure 6.1a Device Selector Schematic

Figure 6.1b Control of A/N Generator

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
Figure 6.2 Block Diagram of I/O Terminal
computers. If a 2 MHz clock were used in the A/N generator, the minimum character generating time would be 18.5μsec. However, DEC has informed us, and it has been observed, that their flip-flops will not count reliably at 2 MHz. Hence the clock must be run at a frequency slightly less than this speed. Since the character generating speed is longer than the 5 instruction cycle time of these computers, they are certainly capable of refreshing the A/N display.

Fig. 6.1 (a) illustrates the device selector. When the appropriate device code is used, the IOP pulse sent by the computer passes through the pulse amplifier to produce the appropriate control pulse.

Fig. 6.1 (b) illustrates the control unit for the A/N generator. The clear instruction pulse clears the mode flip-flops for all modes of the display unit including graphics (i.e. A/N mode, DOT mode, Vector mode). Then the SET A/N MODE pulse sets the A/N mode flip-flop and other information such as the LOAD MARGIN REGISTER flip-flop referred to previously. The A/N MODE flip-flop enables the NAND gate so that the LOAD DATA pulse may pass through it and become the LOAD A/N DATA pulse. This is required since, if the display were in the VEC TOR MODE and the LOAD DATA pulse arrived, the A/N buffer must not be loaded or a character would be generated at the same time as a vector. This would result in chaos.

6.2 The I/O Terminal

Fig. 6.2 illustrates the complete I/O terminal in block diagram form. Two other Master's candidates are working on the system as well. One is responsible for the control unit, computer interface and refresh core memory; the other is constructing the graphics circuitry.
The actual intensification of the beam during character generation will be provided by the graphics hardware to eliminate redundant circuitry. The intensity normally varies with character size since the dots are more closely spaced for smaller characters making them appear brighter. Thus the character size is sent to the intensification circuits as well as the unique pulse train. Once the intensity is standardized for all three sizes, it may be uniformly set to three different brightness levels.

A light pen is currently being designed as part of the control unit and should be added to the system in the immediate future. There is a provision in the A/N instruction word to make a character light-pen sensitive. Thus it will be possible to permit the light pen to "see" certain characters while making other characters "invisible" to the pen. This will help prevent one from accidentally pointing to the wrong information. The light pen feature would be very useful in editing text since one could point to the character to be changed and the computer would then know the memory location containing the character. Hence the new character could simply be written into the same location resulting in the corrected display. Also if a linked list data file were used, complete lines of text could be inserted or deleted.

Provision has also been made in the data words for the addition of character rotate and blink functions. These also will be performed by graphics circuitry. The character rotate will permit characters to be rotated 90° counter-clockwise for labelling axes of graphs, etc. The blink feature will permit certain characters or lines to flicker or blink. This can be useful for attracting the attention of the user to specific areas of the display.
Figure 6.3 Three Raster Techniques (Photographs)
Figure 6.4a Three Character Sizes (Photograph)

Figure 6.4b Tabulate and Carriage Return (Photograph)
Capability for future expansion has been one of the key features of the design strategy of the I/O terminal. Each major block (i.e. A/N generator, core) has been made modular. The blocks are interconnected by cables which easily plug into them. Thus the blocks can be rearranged as more space is required for expansion.

The total system including computer, high speed paper tape reader, teletype interface and display system is housed in two 19" racks which are mounted together on wheels. When completed, a c.r.t., Teletype and function keys will be mounted on a desk which will be interconnected to the computer-display system by cables.

6.3 Test Results

The A/N text generator has been completed and tested both with the computer and with the refresh core providing refresh. The three photographs of Fig. 6.3 illustrate the same character generated by the different raster techniques described in this thesis. The photographs are deliberately over-intensified to show the raster as well.

At this time, the graphics generator has not been completed. Hence the intensification circuitry is not available. A temporary pulse amplifier has been used to modulate the Z-axis of the c.r.t. Due to the slow switching speed of the pulse amplifier, the minimum character generating time without noticeable smearing of the dots is 70 $\mu$sec (about 4 times the desired character generation time). However, with the clock set as high as 1.9 MHz, the 5 x 7 dot matrix can be generated with acceptable resolution. Thus if the final intensification circuitry can be made to respond rapidly enough, the character generator will be able to generate about 53,000 characters/sec.
Fig. 6.4a illustrates the three character sizes. The double A/N mode would primarily be used on a small c.r.t. to permit subscripting and superscripting. In the normal mode, all three character sizes may be generated. The two equations on each side of the vertical dotted line were generated from identical data words. Only the instruction word was different. Notice that a superscript may be placed directly above a subscript by backspacing.

Fig. 6.4b demonstrates the tabulate and carriage return. A normal carriage return returns the beam to the extreme left of the screen as shown by the letter N typed at this position. The digits 0,...,7 were located by tabulating. Notice that they occur at every 10th position of the smallest character size shown at the top of the display (85 characters being shown in a complete line).

The digits 0,...,7 also represent the value which is contained in the margin register to provide a carriage return to the margin setting. For example, 0 was loaded into the margin register and a carriage return was performed using the margin register. Then the letters ABCDE were typed. A carriage return-line feed then set the beam to position F. The four columns of characters were generated by using 4 different margin settings. The sixty characters at the bottom of Fig. 6.4b illustrate all of the possible printing characters.
CHAPTER VII

CONCLUSIONS

The principle of using a unique pulse train which is derived from a binary counter solely through simple gating, provides a relatively inexpensive and easily maintained character generator. In its current form, each character is associated with certain gates. If a character or any portion of a character fails, one simply need replace those gates until the problem is corrected. If a particular dot is not present in all characters, one can quickly replace the counter decoder gate which decodes that pulse. Hence the character generator can be easily user maintained.

Since the character generator was experimental, certain inefficiencies in hardware resulted. For example, DEC provides two input NAND gates which can be expanded by the addition of diode inputs. However, the minimum diode expander module contains seven diodes. Hence a 7-input NAND gate which is frequently used in this design requires only seven of the nine available diodes.

The use of NAND logic requires that inverters be used throughout the count control circuitry. If AND-OR logic had been employed, these inverters could have been eliminated.

The combinations of different size characters as well as single and double spaced line feeds permits one to place characters at a large variety of vertical positions. However, the reset circuitry does prevent
completely random vertical positioning. If one felt that this were
necessary, another 10 bit register could store a vertical position.
Then the Y-reset pulse would strobe the contents of this register into
the Y register instead of collector triggering the Y register to cer-
tain fixed values. Similarly a 10 bit margin register could be used to
permit the left margin to be positioned anywhere on the screen. Of
course these 10 bit registers will add to the cost of the display and
little will be gained by them.

At present the switch register of the computer is being 'OR'ed
into the data words to provide for subscripting, superscripting double
and single spaced line feeds, and carriage returns using the margin re-
gister. One simply sets the bit of the data word he wishes and types
the appropriate character. The ASCII code from the Teletype then has
these extra bits of information added to form the 12 bit data word which
is then transferred to the refresh core. It is proposed that when com-
pleted, the terminal will have function keys which will make it easy
for the user to select the instruction mode as well as character size,
etc.

Many character generators use core memories to provide the inten-
sification information and many use weighted resistor networks to pro-
vide the positioning information. Neither of these methods can be im-
plemented in integrated circuits. Since our method of selecting unique
pulse trains relies solely on the use of logic gating circuitry which is
identical except for the interconnections, integrated circuits on printed
circuit boards could reduce the size of the character generator from
about two hundred and fifteen modules to only a few.

With the advent of large scale integration (LSI), it is feasible
that the complete pulse selection circuitry could be placed on a single chip with only 9 signal inputs and one output. If a complete system excluding the D/A converters were implemented further savings could probably be made. The use of this technology would be justified if these circuits were to be used in sufficient quantities to offset the cost of developing the masks and minimizing interconnections. However, a truly portable A/N display weighing perhaps only a few pounds and consuming little power could be the results of such an effort. With the expected advances in display media perhaps a fully solid state display screen using light emitting diodes could replace the c.r.t. and eliminate the need for D/A converters. Then the entire terminal could probably be carried about in a small briefcase. Note, however, that it is the fact that digital rather than analogue character generation is used, which makes this idea a possibility.

Use of such hardware features as tabulation, margin setting and subscripting and superscripting has, to the best of my knowledge, never been implemented before. This results in much more displayed information with fewer instruction and data words. If, for example, a tabulate position were to be calculated in the computer by software, the contents of the X register would have to be read into memory, extensive bit manipulation or table look-up would be required to determine the new beam position and the new beam position would then have to be transferred back to the refresh memory. This could result in a delay of several milliseconds the first time the TAB character was typed. However, the hardware accomplishes the same result within one character generation time.

The three methods of raster generation makes the method of selecting
unique pulse trains adaptable to three different display systems of varying cost and complexity without any modification. Hence a much wider market could be served by the one design.
## APPENDIX I

### Character 7 Bit Binary Code

<table>
<thead>
<tr>
<th>Character</th>
<th>7 Bit Binary Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>1 000 001</td>
</tr>
<tr>
<td>B</td>
<td>1 000 010</td>
</tr>
<tr>
<td>C</td>
<td>1 000 011</td>
</tr>
<tr>
<td>D</td>
<td>1 000 100</td>
</tr>
<tr>
<td>E</td>
<td>1 000 101</td>
</tr>
<tr>
<td>F</td>
<td>1 000 110</td>
</tr>
<tr>
<td>G</td>
<td>1 000 111</td>
</tr>
<tr>
<td>H</td>
<td>1 001 000</td>
</tr>
<tr>
<td>I</td>
<td>1 001 001</td>
</tr>
<tr>
<td>J</td>
<td>1 001 010</td>
</tr>
<tr>
<td>K</td>
<td>1 001 011</td>
</tr>
<tr>
<td>L</td>
<td>1 001 100</td>
</tr>
<tr>
<td>M</td>
<td>1 001 101</td>
</tr>
<tr>
<td>N</td>
<td>1 001 110</td>
</tr>
<tr>
<td>O</td>
<td>1 001 111</td>
</tr>
<tr>
<td>P</td>
<td>1 010 000</td>
</tr>
<tr>
<td>Q</td>
<td>1 010 001</td>
</tr>
<tr>
<td>R</td>
<td>1 010 010</td>
</tr>
<tr>
<td>S</td>
<td>1 010 011</td>
</tr>
<tr>
<td>T</td>
<td>1 010 100</td>
</tr>
<tr>
<td>U</td>
<td>1 010 101</td>
</tr>
<tr>
<td>V</td>
<td>1 010 110</td>
</tr>
<tr>
<td>W</td>
<td>1 010 111</td>
</tr>
<tr>
<td>X</td>
<td>1 011 000</td>
</tr>
<tr>
<td>Y</td>
<td>1 011 001</td>
</tr>
<tr>
<td>Z</td>
<td>1 011 010</td>
</tr>
<tr>
<td>0</td>
<td>0 110 000</td>
</tr>
<tr>
<td>1</td>
<td>0 110 001</td>
</tr>
<tr>
<td>2</td>
<td>0 110 010</td>
</tr>
<tr>
<td>3</td>
<td>0 110 011</td>
</tr>
<tr>
<td>4</td>
<td>0 110 100</td>
</tr>
<tr>
<td>5</td>
<td>0 110 101</td>
</tr>
<tr>
<td>6</td>
<td>0 110 110</td>
</tr>
<tr>
<td>7</td>
<td>0 110 111</td>
</tr>
<tr>
<td>8</td>
<td>0 111 000</td>
</tr>
<tr>
<td>9</td>
<td>0 111 001</td>
</tr>
<tr>
<td>!</td>
<td>0 111 010</td>
</tr>
<tr>
<td>$</td>
<td>0 111 011</td>
</tr>
<tr>
<td>&amp;</td>
<td>0 111 100</td>
</tr>
<tr>
<td>'</td>
<td>0 111 101</td>
</tr>
<tr>
<td>(</td>
<td>0 111 110</td>
</tr>
<tr>
<td>)</td>
<td>0 111 111</td>
</tr>
<tr>
<td></td>
<td>0 100 000</td>
</tr>
<tr>
<td></td>
<td>0 100 001</td>
</tr>
<tr>
<td></td>
<td>0 100 010</td>
</tr>
<tr>
<td></td>
<td>0 100 011</td>
</tr>
<tr>
<td></td>
<td>0 101 000</td>
</tr>
<tr>
<td></td>
<td>0 101 001</td>
</tr>
<tr>
<td>Character</td>
<td>7 Bit Binary Code</td>
</tr>
<tr>
<td>-----------</td>
<td>------------------</td>
</tr>
<tr>
<td>*</td>
<td>0 101 010</td>
</tr>
<tr>
<td>+</td>
<td>0 101 011</td>
</tr>
<tr>
<td>,</td>
<td>0 101 100</td>
</tr>
<tr>
<td>-</td>
<td>0 101 101</td>
</tr>
<tr>
<td>.</td>
<td>0 101 110</td>
</tr>
<tr>
<td>/</td>
<td>0 101 111</td>
</tr>
<tr>
<td>;</td>
<td>0 111 010</td>
</tr>
<tr>
<td>;</td>
<td>0 111 011</td>
</tr>
<tr>
<td>&lt;</td>
<td>0 111 100</td>
</tr>
<tr>
<td>&gt;=</td>
<td>0 111 101</td>
</tr>
<tr>
<td>?</td>
<td>0 111 110</td>
</tr>
<tr>
<td>[</td>
<td>1 011 011</td>
</tr>
<tr>
<td>\</td>
<td>1 011 100</td>
</tr>
<tr>
<td>]</td>
<td>1 011 101</td>
</tr>
<tr>
<td></td>
<td>1 011 110</td>
</tr>
<tr>
<td>↓</td>
<td>1 011 111</td>
</tr>
<tr>
<td>TAB</td>
<td>0 001 001</td>
</tr>
<tr>
<td>LF</td>
<td>0 001 010</td>
</tr>
<tr>
<td>CR</td>
<td>0 001 101</td>
</tr>
<tr>
<td>BS</td>
<td>0 011 000</td>
</tr>
<tr>
<td>NULL</td>
<td>0 000 000</td>
</tr>
</tbody>
</table>

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
APPENDIX II

CHARACTER DOT PATTERNS AND PULSE NUMBERS

Dot Patterns

A B C D E
F G H I J
K L M N O
P Q R S T
U V W X Y
Z
0 1 2 3 4
5 6 7 8 9
!"#$%'
( )*+,
- . / :;
< = > ? [\]^ _
### Pulse Numbers Required to Generate Characters

<table>
<thead>
<tr>
<th>Character</th>
<th>Pulse numbers</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>1, 2, 3, 4, 5, 10, 13, 17, 21, 24, 27, 29, 30, 31, 32, 33</td>
</tr>
<tr>
<td>B</td>
<td>1, 2, 3, 4, 5, 6, 7, 8, 11, 14, 15, 18, 21, 22, 25, 28, 30, 31, 33, 34</td>
</tr>
<tr>
<td>C</td>
<td>2, 3, 4, 5, 6, 8, 14, 15, 21, 22, 28, 30, 34</td>
</tr>
<tr>
<td>D</td>
<td>1, 2, 3, 4, 5, 6, 7, 8, 11, 14, 15, 18, 21, 22, 28, 30, 31, 32, 33, 34</td>
</tr>
<tr>
<td>E</td>
<td>1, 2, 3, 4, 5, 6, 7, 8, 11, 14, 15, 18, 21, 22, 28, 29, 35</td>
</tr>
<tr>
<td>F</td>
<td>1, 2, 3, 4, 5, 6, 11, 14, 18, 21, 28, 35</td>
</tr>
<tr>
<td>G</td>
<td>2, 3, 4, 5, 6, 8, 14, 15, 17, 21, 22, 24, 28, 30, 31, 34</td>
</tr>
<tr>
<td>H</td>
<td>1, 2, 3, 4, 5, 6, 7, 11, 18, 25, 29, 30, 31, 32, 33, 34, 35</td>
</tr>
<tr>
<td>I</td>
<td>15, 16, 17, 18, 19, 20, 21</td>
</tr>
<tr>
<td>J</td>
<td>2, 8, 15, 22, 30, 31, 32, 33, 34, 35</td>
</tr>
<tr>
<td>K</td>
<td>1, 2, 3, 4, 5, 6, 7, 11, 17, 19, 23, 27, 29, 35</td>
</tr>
<tr>
<td>L</td>
<td>1, 2, 3, 4, 5, 6, 7, 8, 15, 22, 29</td>
</tr>
<tr>
<td>M</td>
<td>1, 2, 3, 4, 5, 6, 7, 13, 19, 27, 29, 30, 31, 32, 33, 34, 35</td>
</tr>
<tr>
<td>N</td>
<td>1, 2, 3, 4, 5, 6, 7, 13, 19, 25, 29, 30, 31, 32, 33, 34, 35</td>
</tr>
</tbody>
</table>

96

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
<table>
<thead>
<tr>
<th>Character</th>
<th>Pulse numbers</th>
</tr>
</thead>
<tbody>
<tr>
<td>O</td>
<td>2, 3, 4, 5, 6, 8, 14, 15, 21, 22, 28, 30, 31, 32, 33, 34</td>
</tr>
<tr>
<td>P</td>
<td>1, 2, 3, 4, 5, 6, 7, 11, 14, 18, 21, 25, 28, 33, 34</td>
</tr>
<tr>
<td>Q</td>
<td>2, 3, 4, 5, 6, 8, 14, 15, 17, 21, 23, 28, 29, 31, 32, 33, 34</td>
</tr>
<tr>
<td>R</td>
<td>1, 2, 3, 4, 5, 6, 7, 11, 14, 17, 18, 21, 23, 25, 28, 29, 33, 34</td>
</tr>
<tr>
<td>S</td>
<td>2, 5, 6, 8, 11, 14, 15, 18, 21, 22, 25, 28, 30, 31, 34</td>
</tr>
<tr>
<td>T</td>
<td>7, 14, 15, 16, 17, 18, 19, 20, 21, 28, 35</td>
</tr>
<tr>
<td>U</td>
<td>2, 3, 4, 5, 6, 7, 8, 15, 22, 30, 31, 32, 33, 34, 35</td>
</tr>
<tr>
<td>V</td>
<td>3, 4, 5, 6, 7, 9, 15, 23, 31, 32, 33, 34, 35</td>
</tr>
<tr>
<td>W</td>
<td>1, 2, 3, 4, 5, 6, 7, 9, 17, 23, 29, 30, 31, 32, 33, 34, 35</td>
</tr>
<tr>
<td>X</td>
<td>1, 2, 6, 7, 10, 12, 18, 24, 26, 29, 30, 34, 35</td>
</tr>
<tr>
<td>Y</td>
<td>6, 7, 12, 15, 16, 17, 18, 26, 34, 35</td>
</tr>
<tr>
<td>Z</td>
<td>1, 2, 7, 8, 10, 14, 15, 18, 21, 22, 26, 28, 29, 34, 35</td>
</tr>
<tr>
<td>Ø</td>
<td>2, 3, 4, 5, 6, 8, 10, 14, 15, 18, 21, 22, 26, 28, 30, 31, 32, 33, 34</td>
</tr>
<tr>
<td>1</td>
<td>8, 13, 15, 16, 17, 18, 19, 20, 21, 22</td>
</tr>
<tr>
<td>2</td>
<td>1, 2, 3, 6, 8, 11, 14, 15, 18, 21, 22, 25, 28, 29, 33, 34</td>
</tr>
<tr>
<td>Character</td>
<td>Pulse numbers</td>
</tr>
<tr>
<td>-----------</td>
<td>---------------------------------------------------</td>
</tr>
<tr>
<td>3</td>
<td>2, 6, 8, 14, 15, 18, 21, 22, 25, 28, 30, 31, 33, 34</td>
</tr>
<tr>
<td>4</td>
<td>3, 4, 10, 12, 17, 20, 22, 23, 24, 25, 26, 27, 28, 31</td>
</tr>
<tr>
<td>5</td>
<td>2, 4, 5, 6, 7, 8, 11, 14, 15, 18, 21, 22, 25, 28, 30, 31, 35</td>
</tr>
<tr>
<td>6</td>
<td>2, 3, 4, 5, 6, 8, 11, 14, 15, 18, 21, 22, 25, 28, 30, 31, 34</td>
</tr>
<tr>
<td>7</td>
<td>1, 7, 9, 14, 17, 21, 25, 28, 33, 34, 35</td>
</tr>
<tr>
<td>8</td>
<td>2, 3, 5, 6, 8, 11, 14, 15, 18, 21, 22, 25, 28, 30, 31, 33, 34</td>
</tr>
<tr>
<td>9</td>
<td>2, 5, 6, 8, 11, 14, 15, 18, 21, 22, 25, 28, 30, 31, 32, 33, 34</td>
</tr>
<tr>
<td>!</td>
<td>15, 17, 18, 19, 20, 21</td>
</tr>
<tr>
<td>&quot;</td>
<td>12, 13, 14, 26, 27, 28</td>
</tr>
<tr>
<td>$</td>
<td>2, 5, 9, 11, 13, 15, 16, 18, 20, 21, 23, 25, 27, 31, 34</td>
</tr>
<tr>
<td>&amp;</td>
<td>2, 3, 5, 6, 8, 11, 14, 15, 17, 18, 21, 23, 26, 27, 29, 31</td>
</tr>
<tr>
<td>.</td>
<td>19, 20, 21</td>
</tr>
<tr>
<td>(</td>
<td>2, 3, 4, 5, 6, 8, 14</td>
</tr>
<tr>
<td>)</td>
<td>22, 28, 30, 31, 32, 33, 34</td>
</tr>
<tr>
<td>*</td>
<td>3, 5, 7, 11, 12, 13, 19, 25, 26, 27, 31, 33, 35</td>
</tr>
<tr>
<td>+</td>
<td>3, 10, 15, 16, 17, 18, 19, 24, 31</td>
</tr>
</tbody>
</table>

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
<table>
<thead>
<tr>
<th>Character</th>
<th>8,10,11,16,17,18</th>
</tr>
</thead>
<tbody>
<tr>
<td>-</td>
<td>3,10,17,24,31</td>
</tr>
<tr>
<td>.</td>
<td>8,9,15,16</td>
</tr>
<tr>
<td>/</td>
<td>2,10,18,26,34</td>
</tr>
<tr>
<td>:</td>
<td>8,9,11,12,15,16,18,19</td>
</tr>
<tr>
<td>;</td>
<td>8,10,11,13,14,16,17,18,20,21</td>
</tr>
<tr>
<td>&lt;</td>
<td>4,10,12,16,20,22,28</td>
</tr>
<tr>
<td>=</td>
<td>3,5,10,12,17,19,24,26,31,33</td>
</tr>
<tr>
<td>&gt;</td>
<td>8,14,16,20,24,26,32</td>
</tr>
<tr>
<td>?</td>
<td>6,14,15,17,18,21,26,28,34</td>
</tr>
<tr>
<td>[</td>
<td>1,2,3,4,5,6,7,8,14,15,21</td>
</tr>
<tr>
<td>\</td>
<td>6,12,18,24,30</td>
</tr>
<tr>
<td>]</td>
<td>15,21,22,28,29,30,31,32,33,34,35</td>
</tr>
<tr>
<td>↑</td>
<td>5,13,15,16,17,18,19,20,21,27,33</td>
</tr>
<tr>
<td>←</td>
<td>4,10,11,12,16,18,20,25,32</td>
</tr>
</tbody>
</table>

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
APPENDIX IV

INSTRUCTION AND DATA WORDS

Instruction Word

<table>
<thead>
<tr>
<th>1</th>
<th>0</th>
<th>0</th>
<th>1</th>
<th>0</th>
<th>Ma</th>
<th>R</th>
<th>S</th>
<th>L</th>
<th>Int</th>
</tr>
</thead>
</table>

Data Word

<table>
<thead>
<tr>
<th>0</th>
<th>B</th>
<th>SS</th>
<th>M</th>
<th>7 Bit ASCII Code</th>
</tr>
</thead>
</table>

Symbols Used

Ma: 0 - No Action
1 - Load bits 2, 3, 4 of next data word into Margin Register

R: 0 - Normal Character
1 - Rotate 90° Counter-clockwise

S: 0 - Normal A/N Mode
1 - Double A/N Mode

L: 0 - Light Pen Insensitive
1 - Light Pen Sensitive

Int: 00 - Normal Intensity
01 - Dim
10 - Bright
11 - Blanked

B: 0 - No Action
1 - Blink

SS: 00 - Normal Position
01 - Superscript
10 - Subscript
11 - Capital

M: 0 - Normal CR or Double Spaced LF
1 - CR to position in Margin Register or Single Spaced LF
APPENDIX V

TIMING DIAGRAMS

LOAD A/N DATA

Clock

Delayed Clock

GENERATE

X Count Enable

Y RESET LEVEL

Y Count Enable

Pulse #36
Character complete, clear buffer, clear counter, clear GENERATE

Normal Character Waveforms

102
The following waveforms are for Special Characters. Only those waveforms which differ from the normal character are illustrated for these characters.

INHIBIT NORMAL CHARACTER

Set X COUNT DOWN

Back Space (BS)

INHIBIT NORMAL CHARACTER

Set Y COUNT DOWN

Generate Line-Feed

#7

Line Feed (LF)

Set Y COUNT DOWN

Null
INHIBIT NORMAL CHARACTER

Set Y COUNT DOWN

CLEAR X Register

Load X Register from Margin Register if necessary.

Carriage Return (CR)

INHIBIT NORMAL CHARACTER

Count in 64's Position if necessary

Set Y COUNT DOWN

CLEAR 512,...,64 of X Register and Prepare to Load New Value into X Register

Load New Value into X Register

Tabulate (TAB)
APPENDIX VI

LOGIC NOTATION

**Direct**
- Clear — £
- Set

**Flip-Flop Delay**
- Pulse
- Input
- Level must be present at least 400 nsec before gate is pulsed.

**DCD Gate**

This gate acts as an AND gate. It provides a logical delay which is essential for sampling flip-flops at the time they are changing.

- Positive $ightarrow$ Preferred Pulse Input
- Negative

**Input**

**Interconnections**

**P.A.**
- Pulse Amplifier

**Delay**

**Inverter**

**NAND Gate**

**OR Gate**
## APPENDIX VII

**TWO OPERATING SPEEDS FOR HORIZONTAL AND VERTICAL SWEEP CHARACTER GENERATION**

<table>
<thead>
<tr>
<th>Function</th>
<th>Low Speed</th>
<th>Medium Speed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clock Frequency</td>
<td>50 KHz</td>
<td>300 KHz</td>
</tr>
<tr>
<td>Character Rate</td>
<td>1,020 Char/Sec</td>
<td>6,120 Char/Sec</td>
</tr>
<tr>
<td>Vertical Sweep</td>
<td>7.14 KHz</td>
<td>42.86 KHz</td>
</tr>
<tr>
<td>Horizontal Sweep</td>
<td>25.5 Hz</td>
<td>153 Hz</td>
</tr>
<tr>
<td>Characters/Line</td>
<td>40</td>
<td>40</td>
</tr>
<tr>
<td>Number of Lines</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>Dot Intensification Time</td>
<td>10 μsec</td>
<td>1 μsec</td>
</tr>
<tr>
<td>Frame Rate</td>
<td>N/A</td>
<td>40 Frames/Sec</td>
</tr>
<tr>
<td>Flicker Free Characters</td>
<td>N/A</td>
<td>153</td>
</tr>
</tbody>
</table>
REFERENCES


1945 Born on June 28, in Windsor, Ontario, Canada.

1959 Completed elementary education at the Alice Street Public School, Essex, Ontario, Canada.

1964 Completed Grade XIII at Essex District High School, Essex, Ontario, Canada.

1967 Graduated from the University of Windsor, Windsor, Ontario, Canada, with the degree of B.Sc. in Physics.

1970 Candidate for the degree of M.A.Sc. in Electrical Engineering at the University of Windsor, Windsor, Ontario, Canada.