A DISPLAY SYSTEM FOR COMPUTER GRAPHICS

by

FRANK ROBERT HAMILTON

B.A. Sc., University of British Columbia, 1968

A THESIS SUBMITTED IN PARTIAL FULFILMENT
OF THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF APPLIED SCIENCE

in the Department

of

Electrical Engineering

We accept this thesis as conforming to the
required standard

THE UNIVERSITY OF BRITISH COLUMBIA

September, 1974
In presenting this thesis in partial fulfilment of the requirements for an advanced degree at the University of British Columbia, I agree that the Library shall make it freely available for reference and study.

I further agree that permission for extensive copying of this thesis for scholarly purposes may be granted by the Head of my Department or by his representatives. It is understood that copying or publication of this thesis for financial gain shall not be allowed without my written permission.

Department of **ELECTRICAL ENGINEERING**

The University of British Columbia
Vancouver 8, Canada

Date **SEPT 26/74**
ABSTRACT

A display system for computer graphics is described in this thesis. The requirements of the display system are defined, with the emphasis on use in computer-aided network design. Several basic design alternatives are examined, and a basic approach is chosen toward development of a prototype system.

A detailed description of the graphics display system architecture and operation is provided. Details relating to the implementation of the prototype system are provided, and the performance of the system is examined.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>LIST OF ILLUSTRATIONS</td>
<td>v</td>
</tr>
<tr>
<td>ACKNOWLEDGEMENT</td>
<td>vii</td>
</tr>
<tr>
<td>1. INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td>1.1 Statement of the Problem</td>
<td>1</td>
</tr>
<tr>
<td>1.2 Scope of the Thesis</td>
<td>1</td>
</tr>
<tr>
<td>2. DESIGN OF THE GRAPHICS DISPLAY SYSTEM</td>
<td>2</td>
</tr>
<tr>
<td>2.1 Graphics Display System Requirements</td>
<td>2</td>
</tr>
<tr>
<td>2.2 Design Considerations</td>
<td>3</td>
</tr>
<tr>
<td>2.2.1 Basic Display System Architecture</td>
<td>3</td>
</tr>
<tr>
<td>2.2.2 Analog Versus Digital Techniques</td>
<td>5</td>
</tr>
<tr>
<td>2.2.3 Display-Element Generation</td>
<td>5</td>
</tr>
<tr>
<td>2.2.3.1 Display Program Subroutines</td>
<td>5</td>
</tr>
<tr>
<td>2.2.3.2 Character Generation</td>
<td>6</td>
</tr>
<tr>
<td>2.2.3.3 Vector Generation</td>
<td>6</td>
</tr>
<tr>
<td>3. FUNCTIONAL ORGANIZATION OF THE GRAPHICS DISPLAY SYSTEM</td>
<td>8</td>
</tr>
<tr>
<td>3.1 Display System Overview</td>
<td>8</td>
</tr>
<tr>
<td>3.2 Display Instructions</td>
<td>11</td>
</tr>
<tr>
<td>3.2.1 Control Instructions</td>
<td>12</td>
</tr>
<tr>
<td>3.2.2 Transfer Instructions</td>
<td>12</td>
</tr>
<tr>
<td>3.2.3 Point Instructions</td>
<td>13</td>
</tr>
<tr>
<td>3.2.4 Graph-Point Instructions</td>
<td>14</td>
</tr>
<tr>
<td>3.2.5 Character Instructions</td>
<td>14</td>
</tr>
<tr>
<td>3.2.6 Short-Vector Instructions</td>
<td>16</td>
</tr>
<tr>
<td>3.2.7 Long-Vector Instructions</td>
<td>16</td>
</tr>
</tbody>
</table>

iii
<table>
<thead>
<tr>
<th>Chapter</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.3</td>
<td>Host Computer Interface</td>
<td>18</td>
</tr>
<tr>
<td>3.4</td>
<td>The Display Processor</td>
<td>24</td>
</tr>
<tr>
<td>3.4.1</td>
<td>The Immediate Instruction Sequence</td>
<td>28</td>
</tr>
<tr>
<td>3.4.2</td>
<td>The Single-Word Instruction Sequence</td>
<td>29</td>
</tr>
<tr>
<td>3.4.3</td>
<td>The Half-Word Instruction Sequence</td>
<td>29</td>
</tr>
<tr>
<td>3.4.4</td>
<td>The Double-Word Instruction Sequence</td>
<td>32</td>
</tr>
<tr>
<td>3.5</td>
<td>The Display Oscilloscope Interface</td>
<td>32</td>
</tr>
<tr>
<td>3.6</td>
<td>The System Clock</td>
<td>34</td>
</tr>
<tr>
<td>3.7</td>
<td>Display-Element Generators</td>
<td>36</td>
</tr>
<tr>
<td>3.7.1</td>
<td>The Point and Graph-Point Generator</td>
<td>36</td>
</tr>
<tr>
<td>3.7.2</td>
<td>The Character Generator</td>
<td>39</td>
</tr>
<tr>
<td>3.7.3</td>
<td>The Short-Vector Generator</td>
<td>44</td>
</tr>
<tr>
<td>3.7.4</td>
<td>The Long-Vector Generator</td>
<td>47</td>
</tr>
<tr>
<td>4.</td>
<td>IMPLEMENTATION OF THE GRAPHICS DISPLAY SYSTEM</td>
<td>51</td>
</tr>
<tr>
<td>4.1</td>
<td>A Modular Approach</td>
<td>51</td>
</tr>
<tr>
<td>4.2</td>
<td>Purchased Components</td>
<td>51</td>
</tr>
<tr>
<td>4.3</td>
<td>The Logic Family</td>
<td>51</td>
</tr>
<tr>
<td>4.4</td>
<td>Construction Techniques</td>
<td>52</td>
</tr>
<tr>
<td>5.</td>
<td>PERFORMANCE OF THE GRAPHICS DISPLAY SYSTEM</td>
<td>53</td>
</tr>
<tr>
<td>6.</td>
<td>CONCLUSIONS</td>
<td>56</td>
</tr>
<tr>
<td>APPENDIX 1</td>
<td>THE VECTOR GENERATION ALGORITHM</td>
<td>58</td>
</tr>
<tr>
<td>REFERENCES</td>
<td></td>
<td>60</td>
</tr>
<tr>
<td>Figure</td>
<td>Illustration Description</td>
<td>Page</td>
</tr>
<tr>
<td>--------</td>
<td>-----------------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>3.1</td>
<td>System Block Diagram</td>
<td>9</td>
</tr>
<tr>
<td>3.2</td>
<td>Display Instruction Formats</td>
<td>10</td>
</tr>
<tr>
<td>3.3</td>
<td>The ASCII Character Set</td>
<td>15</td>
</tr>
<tr>
<td>3.4</td>
<td>Typical Short Vectors</td>
<td>15</td>
</tr>
<tr>
<td>3.5</td>
<td>A Sample Long Vector</td>
<td>17</td>
</tr>
<tr>
<td>3.6</td>
<td>CPU Interface</td>
<td>19</td>
</tr>
<tr>
<td>3.7</td>
<td>CPU Interface Timing</td>
<td>23</td>
</tr>
<tr>
<td>3.8</td>
<td>Display Processor</td>
<td>25</td>
</tr>
<tr>
<td>3.9</td>
<td>The Immediate Sequence</td>
<td>27</td>
</tr>
<tr>
<td>3.10</td>
<td>The Single-Word Sequence</td>
<td>27</td>
</tr>
<tr>
<td>3.11</td>
<td>The Half-Word Sequence</td>
<td>30</td>
</tr>
<tr>
<td>3.12</td>
<td>The Double-Word Sequence</td>
<td>31</td>
</tr>
<tr>
<td>3.13</td>
<td>The Display Oscilloscope Interface</td>
<td>33</td>
</tr>
<tr>
<td>3.14</td>
<td>The System Clock</td>
<td>35</td>
</tr>
<tr>
<td>3.15</td>
<td>Clock Timing</td>
<td>35</td>
</tr>
<tr>
<td>3.16</td>
<td>The Point, and Graph-Point Generator</td>
<td>38</td>
</tr>
<tr>
<td>3.17</td>
<td>Point, and Graph-Point Timing</td>
<td>38</td>
</tr>
<tr>
<td>3.18</td>
<td>Beam Positioning During Character Generation</td>
<td>40</td>
</tr>
<tr>
<td>3.19</td>
<td>The Character Generator</td>
<td>41</td>
</tr>
<tr>
<td>3.20</td>
<td>Character Timing</td>
<td>42</td>
</tr>
<tr>
<td>3.21</td>
<td>The Short-Vector Generator</td>
<td>45</td>
</tr>
<tr>
<td>3.22</td>
<td>Short-Vector Timing</td>
<td>46</td>
</tr>
<tr>
<td>3.23</td>
<td>The Long-Vector Generator</td>
<td>48</td>
</tr>
<tr>
<td>3.24</td>
<td>Long-Vector Timing</td>
<td>50</td>
</tr>
</tbody>
</table>
### LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>Input-Output Transfer Instructions</td>
<td>21</td>
</tr>
<tr>
<td>5.1</td>
<td>Display System Performance</td>
<td>54</td>
</tr>
<tr>
<td>5.2</td>
<td>Graph Generation Performance</td>
<td>54</td>
</tr>
</tbody>
</table>
I am grateful to the National Research Council of Canada for support received during the term of this project.

I would like to thank Dr. A.D. Moore for his invaluable guidance while supervising this project and Dr. M.R. Ito for reading this thesis and providing helpful suggestions.

Special thanks go to Dr. J.S. MacDonald, Messrs. D. Heywood, S. Semrau, G. Austin and N. Thompson for the many useful ideas resulting from discussions.

I wish to acknowledge the assistance provided by Mr. B. Nuttal who constructed the display system, and by my wife who typed this thesis.
1. INTRODUCTION

1.1 Statement of the Problem

Computer graphics has long been recognized as a desirable (perhaps essential) approach to information display in certain applications (1,2,3,4). In particular, the use of graphics display techniques in interactive, computer-aided design has proven to be invaluable. (5)

Computer graphics techniques have been utilized on the PDP-9 installation at the University of British Columbia Department of Electrical Engineering for several years. Limitations on the existing graphics facility have been noted, the most important of which are the restrictive central processing unit (CPU) overheads during the maintenance of a graphics display, and the excessive display generation time (primarily due to a low bandwidth display oscilloscope), resulting in "flickering" displays.

1.2 Scope of the Thesis

The thesis will describe the steps leading to the implementation of a graphics display system for use at the University of British Columbia Department of Electrical Engineering. Chapter 2 examines the requirements of such a system, and presents the basic design considerations pertinent to these requirements. Chapter 3 provides the detailed functional organization of the display system, and Chapter 4 describes the implementation details. System performance is described in Chapter 5, and conclusions presented in Chapter 6.
2. DESIGN OF THE GRAPHICS DISPLAY SYSTEM

2.1 Graphics Display System Requirements

Prior to the design of a graphics display system, some general requirements must be established. To establish these requirements, a graphics facility for use in computer-aided network design and analysis is considered.

Display system requirements for computer-aided network design and analysis include the capability of generating and altering schematic circuit diagrams in a highly interactive fashion. The capability of displaying the response of a network to various input signals is also required. Ideally, it is desirable to be able to generate a continuously varying response (perhaps illustrating the effect of a continuous variation in a network parameter).

Clearly, the computer supervising the display system must be capable of more than the support of a static display. Computer overheads associated with the generation of the display must be minimized.

The oscilloscope utilized in the graphics system must permit highly dynamic displays, eliminating the possibility of a storage oscilloscope. The oscilloscope should have a resolution of at least 100 points to the inch, with a usable display area of 10 inches by 10 inches. At least 8 distinct intensity levels should be provided.

The system must be capable of generating a complex display with a refresh rate of at least 20 times each second.

The graphics system should be simple to use. Generation of common display elements, including points, graphs, vectors, and characters should be simple from the user point of view.
2.2 Design Considerations

2.2.1 Basic Display System Architecture

Consideration of possible forms of computer/display system interaction suggests several possibilities for the basic display system architecture. Choice of a basic architecture is made on the basis of complexity and the effect on host computer overheads.

One possible form of computer/display system interaction is exemplified by the existing PDP-9 facility. In this case, a PDP-9 program must initiate the display of each point via input-output transfer (IOT) instructions. The main advantage of this approach is its simplicity; the primary disadvantage is the extensive CPU involvement in display generation and maintenance. Computer involvement could be lessened considerably by addition of hardware to generate common display elements, such as characters and vectors. Certain types of displays still require considerable host computer involvement; for example, maintenance of an irregular graph containing 1000 points (such as a network response to a step input) would fully load most small computers if all data is passed via IOT instructions.

A second form of computer/display system interaction is via the direct memory access (DMA) facility available on most small computers (6,7). In this case, data representing an entire display can be constructed by the host computer in its main memory. In the simplest case, the memory resident display data (the "display program") could consist of sequential pairs of X and Y co-ordinate values, and a beam unblanking indication. To generate a display, the host computer provides the display system with the data table address and a word count via IOT instructions. On receipt of a "GO" indication, the display
system accesses sequential data words via the DMA facility and constructs the desired display. Following generation of the last point, the display system interrupts the host computer, and awaits further instructions. The entire process is repeated as often as required to ensure a flicker-free display.

The decrease in host computer overhead associated with this type of display system can be illustrated using the 1000 point graph example. Typically, each DMA access "steals" 1 microsecond from the available host CPU execution time; 1000 X-Y co-ordinate pairs would reduce available CPU time by 2 milliseconds. Since at least 30 repetitions per second are required to eliminate flicker, generation of the graph reduces host computer capabilities by approximately 6%.

The DMA approach provides a significant decrease in computer overheads compared with the programmed I/O approach, but has the disadvantage of increasing the display system complexity.

A third basic system architecture would utilize a large storage system dedicated to the display system (6). With this approach, the host CPU would transfer a display program to the display system memory via either IOT instructions or DMA. Subsequent display generation would be performed using the dedicated memory in a fashion similar to the DMA approach described above.

With this approach, host CPU involvement is limited to loading the display system memory. In the case of a static display, host CPU overheads are negligible. In the case of a highly dynamic display, overheads are not significantly decreased from those associated with the previously described architectures.

A disadvantage of the third approach is a considerable increase in complexity through addition of the large dedicated-memory
system.

The display system was implemented using the DMA approach, since this basic architecture provides a reasonable compromise between complexity and host CPU overheads.

2.2.2 Analog Versus Digital Techniques

Analog display generation techniques (8) provide certain advantages, particularly in the generation of vectors and characters; however, they have some major disadvantages compared to digital techniques. These include:

a) analog circuits tend to drift, necessitating frequent adjustment,

b) analog implementation is generally more difficult, and time consuming.

On the basis of these considerations, implementation using digital techniques was adopted.

2.2.3 Display-Element Generation

The DMA approach has severe limitations if all display elements are generated as described for the 1000 point graph example. Both core requirements and CPU overheads are restrictive in the case of complex displays.

The addition of hardware display-element generators eliminates these restrictions. A vector generator, for example, requires only 2 words of core storage and 2 DMA accesses if the vector is represented by a delta-X and delta-Y co-ordinate pair.

2.2.3.1 Display Program Subroutines

A subroutine capability provides a means of decreasing core requirements for most graphic displays.
A single-level subroutine capability requires storing a single display program address within either a display system register, or a dedicated host CPU memory location. A multiple-level (nested) subroutine capability requires unique storage for a number of display program addresses, hence a considerable increase in hardware complexity over the single-level case.

To minimize complexity a single-level subroutine capability was included in the display system. Since many complex display elements are generated by hardware (as described in subsequent sections) the nested subroutine capability is seldom desired for most graphic displays.

2.2.3.2 Character Generation

A simple approach to character generation is to generate each character via a corresponding subroutine. This approach decreases core requirements, but requires several DMA accesses during the generation of most characters.

Recent introduction of pre-programmed large capacity read-only-memories (ROM's) provides a more attractive alternative. The ROM's provide a simple method of converting a 7-bit ASCII (American Standard Code for Information Interchange) code into 35 binary values used to generate the character in a 5 by 7 dot-matrix format. A detailed description of the ROM-driven character generator is provided in Chapter 2.

2.2.3.3 Vector Generation

Two common techniques are often used in a digital approach to vector generation. Both techniques produce only an approximation to a true vector.

The first technique utilizes digital differential analyzers
(DDA's) to produce a very accurate approximation. The technique requires quite complex circuits.

A second method utilizes a binary rate multiplier (BRM) algorithm. The algorithm produces an adequate, but poorer approximation (in some cases) than the DDA techniques, but implementation is considerably less complex.

The BRM approach (described in detail in Chapter 3) was chosen on the basis of simplicity.
3. **FUNCTIONAL ORGANIZATION OF THE GRAPHICS DISPLAY SYSTEM**

3.1 **Display System Overview**

A simplified block diagram of the graphics display system and the associated host computer is provided in Figure 3.1.

The host computer is a Data General Supernova, a fast third-generation mini-computer.

The major components of the graphics display system include a set of internal storage registers, three D/A converters, a set of display-element generators, and the display processor.

General registers provide temporary storage for data used in the generation of a particular display element. Examples of their use include the storage of an ASCII code during character generation, or the storage of data used during vector generation.

Additional registers provide storage for X, Y and Z-axis co-ordinates. Associated D/A converters directly drive the display oscilloscope, an Optimation CDO-6100. X and Y co-ordinate registers are parallel-loading up/down counters, rather than simple latch registers.

Generation of specific display elements, either character, long vector, short vector, point or graph-point is controlled by a corresponding display-element generator. Each display-element generator converts the data contained in the general registers to the control signals (±X, ±Y, UNBLANK) required to generate a specific display element.

A system clock provides basic timing signals for display element generation.

The heart of the graphics display system is the display
FIGURE 3.1  SYSTEM BLOCK DIAGRAM
<table>
<thead>
<tr>
<th>MODE NUMBER</th>
<th>MODE NAME</th>
<th>MODE</th>
<th>ADDRESS FORMAT</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>CONTROL</td>
<td>0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15</td>
<td>0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15</td>
</tr>
<tr>
<td>1</td>
<td>TRANSFER (JMP)</td>
<td>0, 0, E</td>
<td>DISPLAY PROGRAM ADDRESS</td>
</tr>
<tr>
<td>1</td>
<td>(JSR)</td>
<td>0, 1, E</td>
<td>DISPLAY PROGRAM ADDRESS</td>
</tr>
<tr>
<td>1</td>
<td>(RSR)</td>
<td>1, 1, E</td>
<td>PI</td>
</tr>
<tr>
<td>2</td>
<td>POINT GRAPH-POINT</td>
<td>NXMD, B, AX, D</td>
<td>CORD</td>
</tr>
<tr>
<td>3</td>
<td>CHARACTER</td>
<td>C2, E, C1, E</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>SHORT VECTOR</td>
<td>B, D2, N2, E, B, D1, N1, E</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>LONG VECTOR</td>
<td>1, 1, 0, B, S, DY</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>NXMD, X, DX</td>
<td></td>
</tr>
</tbody>
</table>

**FIGURE 3.2** DISPLAY INSTRUCTION FORMATS
processor. The display processor accesses host CPU memory locations, directs the transfer of data to internal storage registers, and supervises operation of the display-element generators.

Initiation of the display process is performed by the host CPU. Thereafter, the host CPU is essentially free to perform any related or totally unrelated function.

The display processor accesses instructions and data from the host CPU memory. Each instruction is interpreted by the display processor and any associated data is routed to one or more internal registers. Following register loading, a "start function" signal is sent to the appropriate display-element generator.

The display-element generator executes the required function and indicates completion by asserting a "function complete" signal. Occurrence of the "function complete" signal initiates the next instruction execution cycle. An entire oscilloscope display is generated by accessing and executing a set of display instructions.

3.2 Display Instructions

A display program consists of a set of display instructions defining a complete graphic display. Each display instruction defines one or two elements of the total graphic display. Figure 3.2 provides a summary of the display instruction set.

Display instructions are interpreted by the display processor according to the current display processor mode. Display processor modes include control, transfer, point, graph-point, character, short vector and long vector. A field in each display instruction defines the display processor mode for interpretation of the next instruction.
In one form of display system operation, the display processor accesses sequential display instructions directly from the host CPU memory. The first instruction accessed is always interpreted in control mode.

3.2.1 Control Instructions

Control mode instructions are used primarily to set up various display system registers. The registers, including intensity (INT), scale (SC) and "user", tend to remain constant over large segments of any display program.

The intensity register provides for eight levels of display oscilloscope intensity. The scale register defines a scale factor, 1, 2, 4 or 8, applied in graph-point, character and vector element generation. A "user" register is maintained by the display system for the programmer's convenience. The register can be read under program control, and can be used, for example, to indicate the portion of a display program being executed at the time of a light pen interrupt.

An interrupt field, PI, Optionally generates a host CPU interrupt, and a light pen control bit, LP, is provided to enable or disable light pen interrupts.

The next mode field, NXMD, specifies the display processor mode for interpretation of the next sequential instruction.

3.2.2 Transfer Instructions

Transfer instructions allow modification of the display program counter without host CPU intervention. Three types of transfer instructions are differentiated by the code in instruction bits 0 and 1.

A jump (JMP) instruction, code 0, sets the display program
counter to the address specified by the display program address field.

Code 1 identifies a subroutine jump instruction, JSR. On execution, the display program counter is incremented, saved in a subroutine register, and a new value for the display program counter is obtained from the display program address field.

A return from subroutine (RSR) instruction, code 3, transfers the subroutine register contents to the display program counter. An interrupt field, PI, permits the generation of a host CPU interrupt.

The "E" bit in each transfer instruction specifies the mode for execution of the next instruction. If the "E" bit is set, the display processor will enter control mode. When the "E" bit is clear, a mode change will not occur.

3.2.3 Point Instructions

Point instructions are used to load the X and Y co-ordinate registers with absolute values.

Each point instruction modifies a single co-ordinate register, as specified by the "AX" field. When AX contains zero, the X register is modified; otherwise, the Y register is modified.

On point instruction execution, the absolute value contained in the "CORD" field is transferred to the specified co-ordinate register. A time delay is generated to permit the oscilloscope beam to settle in the new position. The length of the time delay is indicated by the "D" field content. When D contains zero, a time delay appropriate to a one-inch beam deflection is generated. A one in the D field results in a time delay permitting a worst case beam deflection (15 inches).

On expiry of the time delay, the beam is unblanked if the "B" field contains one. The display processor is placed in the mode
specified by the "NXMD" field, and the next sequential instruction is executed.

3.2.4 Graph-Point Instructions

Graph-point instructions provide a simple mechanism for generating graphs on the display oscilloscope.

Execution of a graph-point instruction is almost identical to the execution of a point instruction. The absolute value contained in the "CORD" field is transferred to the co-ordinate register indicated by the "AX" field. At the same time, however, the other co-ordinate register is incremented by 1, 2, 4 or 8, as specified by the current content of the scale register.

Interpretation of all other graph-point instruction fields is identical to that of the point instruction.

3.2.5 Character Instructions

Character instructions are used to display alpha-numeric text on the display oscilloscope. Each instruction provides for the generation of up to two characters.

Sixty-four distinct characters can be generated, as illustrated in Figure 3.3. A specific character is indicated by its corresponding ASCII code.

The low-order eight bits of a character instruction specify the first character to be generated. The character indicated by the ASCII code contained in the Cl field is generated in a 5 x 7 dot-matrix format. The size of the dot-matrix is determined by the content of the scale register. When the "E" field contains a zero, the character specified by the high-order eight bits is generated.

Sequential characters are generated from left to right on
FIGURE 3.3 THE ASCII CHARACTER SET

FIGURE 3.4 TYPICAL SHORT VECTORS
the display oscilloscope. The character display process continues until a non-zero "E" field is encountered, resulting in a transition to control mode.

### 3.2.6 Short-Vector Instructions

Vectors limited in both size and direction are generated using short-vector instructions. Each instruction defines up to two short vectors.

A short vector is defined by one of 8 directions as specified in the "D" field, and its length, (1 to 8 points) as indicated by the "N" field content. Figure 3.4 illustrates some of the possible short vectors.

The low-order 8 bits of a short-vector instruction specifies the first vector to be generated. A vector is generated by deflecting the oscilloscope beam by equal increments in the specified direction. After each increment, the beam is unblanked if the "B" field is non-zero. The size of each increment is specified by the content of the scale register, and the number of increments as indicated by the "N" field content.

Successive half-words are processed in short-vector mode until a non-zero "E" field is encountered, causing a display processor transition to control mode.

### 3.2.7 Long-Vector Instructions

Vectors less constrained in length and direction than short vectors are generated using long-vector instructions. A long-vector instruction consists of two 16-bit words.

The DX and Dy fields specify corresponding signed delta-X and delta-Y co-ordinate values. Each delta co-ordinate value is in sign-magnitude format, where a non-zero value of "S" indicates a
FIGURE 3.5 - A SAMPLE LONG VECTOR
negative value.

Long vectors are generated using a binary rate multiplier algorithm. Implementation of the algorithm (described in Section 3.7.4) results in a vector consisting of a series of points at approximately equal intervals. Since a digital approach restricts points to a grid, most vectors are only approximations to a straight line.

A sample vector is illustrated in Figure 3.5.

DX and DY specify the number of points in the vector. Point spacing is determined by the content of the scale register. The oscilloscope beam is unblanked at each vector point position if the "B" field is non-zero.

The contents of the "NXMD" field specify the display processor mode for execution of the next instruction.

3.3 Host Computer Interface

The interface to the host computer, illustrated in block diagram form in Figure 3.6, provides the interface between the display processor and the host CPU.

The CPU interface consists of bus drivers and receivers, a buffer register for output data, a four-channel input multiplexer and an interface control unit.

All data and control signals between the host CPU and the display system are provided by the I/O BUS. Each I/O BUS line is terminated by a simple resistive network matching the characteristic impedance of the line. Signals from the host CPU are buffered by line receivers, each a simple inverter presenting a single unit load to the I/O BUS. Signals to the host CPU are presented to the I/O BUS via open-collector bus drivers. Bidirectional I/O BUS data lines
FIGURE 3.6 CPU INTERFACE
are split into two unidirectional buses, the IN BUS and the OUT BUS, by pairs of bus drivers and receivers.

Data received from the host CPU is temporarily stored in the output buffer register. Outputs from the sixteen simple latches form the buffered data out bus, DATBUS.

The input multiplexer permits the switching of four 16-bit channels of data to the INBUS.

Supervision of all input/output transfers is provided by the interface control unit. The interface control unit performs three major functions: the decoding and processing of display system IOT instructions issued by the host CPU, the supervision of direct memory access fetches from host CPU memory and the assertion of a host CPU interrupt.

Table 3.1 provides a summary of the IOT instructions used for display system operation. Two device codes with mnemonics DCO and DC1 are used to specify the display system.

Four data-input IOT instructions are used to read the four input multiplexer channels. The interface control unit decodes the input IOT request and switches the appropriate multiplexer channel to the INBUS.

The interface control unit provides for two modes of operation during display generation. The first mode, DMA execution mode, provides for display generation with minimal host CPU program control. IOT instructions are used to initiate the display process, and display instructions are fetched using the direct memory access facility. A second mode, IOT execution mode, provides for display generation under host CPU control. That is, display instructions are presented to the
<table>
<thead>
<tr>
<th>IOT INSTRUCTION MODIFIERS</th>
<th>FUNCTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>START (S)</td>
<td>STANDARD MANUF. CONVENTION</td>
</tr>
<tr>
<td>CLEAR (C)</td>
<td>STANDARD MANUF. CONVENTION</td>
</tr>
<tr>
<td>IOTLC (P)</td>
<td>INIT. DMA EXECUTION MODE</td>
</tr>
</tbody>
</table>

**TABLE 3.1** INPUT-OUTPUT TRANSFER INSTRUCTIONS
display system via program IOT instructions.

Prior to display generation in DMA execution mode, the address of the display program must be output to the display processor. Execution of "DOA AC, DCO" by the host CPU results in the display program address on the I/O BUS. The interface control unit strobes the address into the output buffer register and sends a signal, LOAD DPC, to the display processor.

Execution of an IOT instruction with the START modifier enables display system interrupts and direct memory accesses.

On decoding an IOT instruction with the IOPLS modifier, the interface control unit asserts the INIT line, placing the display processor in control mode. The signal WDRQ, requesting a display instruction, is also generated and initiates the display generation process.

The interface control unit supervises the DMA fetch of a display instruction. On receipt of the signal WDRQ, a DMA fetch is performed and the resulting data is strobed into the output buffer register. The signal LDWD is sent to the display processor to indicate that a display instruction word is available for processing.

Figure 3.7 illustrates the interaction between the host CPU and the interface control unit during direct memory accesses.

On receipt of the first WDRQ, a DMA request is sent to the host CPU. When the host CPU is able to honour the DMA request, the signal ADDRESS IN is sent to the interface control unit; the control unit responds by switching multiplexer channel 0, the display program counter, onto the I/O BUS. The host CPU accesses the indicated memory address and outputs the contents on the I/O BUS, accompanied by the
FIGURE 3.7. CPU INTERFACE TIMING
signal DATA OUT. The data is strobed into the output buffer register, and the signal LDWD is generated, indicating completion of the DMA request.

To minimize the average time between WDRQ and LDWD, the interface control unit initiates another DMA cycle immediately after LDWD is generated. In cases where display instruction processing is a lengthy operation, the next display instruction is available when WDRQ is next asserted; hence LDWD is immediately returned. In cases of short display instruction execution time, the DMA cycle is in progress when WDRQ is asserted, and LDWD is generated on completion of the cycle.

The DMA scheme necessitates double-buffering of the data. The display processor must ensure that the display program counter is updated and the output buffer register is free for use shortly after LDWD is generated.

IOT execution mode is initiated by the execution of a "DOC AC, DCO" instruction. Decoding by the interface control unit results in the assertion of INIT, placing the display processor in control mode. Execution of "DOBS AC, DCO" places a display instruction on the I/O BUS. The display instruction is strobed into the output buffer register, display system interrupts are enabled, and the signal LDWD is generated. The display processor, on receipt of LDWD, processes the display instruction and asserts WDRQ on completion. On receipt of WDRQ, the interface control unit requests a host CPU interrupt and awaits program intervention.

3.4 The Display Processor

The display processor, illustrated in block diagram form in Figure 3.8, supervises the execution of all display instructions.
FIGURE 3.8 DISPLAY PROCESSOR
Fundamental components of the display processor include: the display program counter and multiplexer, the subroutine register, the mode register and the timing generator. Two of the internal storage registers, IR1 and IR2, are also shown in Figure 3.8.

The display program counter, DPC, is a parallel loading, 12-bit up-counter. The associated DPC multiplexer permits parallel loading of the DPC from either the output buffer register (via the DAT BUS), or the subroutine register. The DPC contains the current display program address during DMA execution mode.

The subroutine register is a parallel loading register consisting of 12 RS flip-flops. The subroutine register can retain a display program address during instruction execution.

Internal register 2, IR2, is a 16-bit, parallel loading register constructed of RS flip-flops. IR2 is provided for the storage of data used in display-element generation.

Another internal register, IR1, is identical to IR2. A multiplexer associated with IR1 allows the switching of either the low-order or high-order 8 bits of IR1 to the IRM BUS. IR1 is normally used for storage of the display instruction during display-element generation.

The mode register is a 3-bit flip-flop register used to store the current mode code during instruction execution.

The timing generator consists of mode decode circuitry, monostable multivibrators, and associated logic used to generate the fundamental system timing signals.

Four distinct timing sequences are generated during display instruction execution. An "immediate" sequence applies during
FIGURE 3.9 THE IMMEDIATE SEQUENCE

FIGURE 3.10 THE SINGLE-WORD SEQUENCE
the execution of control and transfer instructions. The "single-word" sequence is utilized during the execution of point and graph-point instructions. A third sequence, the "half-word" sequence, is used during the execution of character and short-vector instructions. The final sequence, the "double-word" sequence, is applicable during long-vector instruction execution.

3.4.1 The Immediate Instruction Sequence

The immediate instruction sequence, illustrated in Figure 3.9, is initiated on assertion of WDRQ, a request for the next display instruction. Completion of the memory access is denoted by LDWD. The leading edge of LDWD is used to increment the DPC, and the trailing edge triggers a monostable multivibrator to produce timing pulse 1, TP1. TP1 is used to transfer the display instruction from the output buffer register (DAT BUS) to IR1. The trailing edge of TP1 triggers a second monostable multivibrator, producing timing pulse 2, TP2. TP2 is used to generate WDRQ, initiating the next timing sequence.

On execution of a control instruction, the scale, intensity and user registers are loaded from the DAT BUS using TP1. If specified, a program interrupt is also initiated by TP1. TP2 is used to transfer the mode (for execution of the next instruction) from IR1 to the mode register.

LDWD is used to transfer the DPC to the subroutine register during execution of a JSR transfer instruction. TP1 is used to load the DPC with the new program address (via the DAT BUS) on execution of both a JSR and a JMP transfer instruction. On execution of an RSR transfer instruction, TP1 is used to load the DPC from the subroutine register. If the "E" field of the transfer instructions is non-zero,
TP2 is used to clear the mode register, restoring the display processor to control mode.

3.4.2 The Single-Word Instruction Sequence

A single-word sequence is illustrated in Figure 3.10. During a single-word sequence, LDWD, TP1 and TP2 are generated in the same manner as in an immediate sequence. LDWD is used to increment the DPC, and TP1 is used to strobe the display instruction into IR1. The appropriate "start function" line is asserted by TP2, and the display processor enters an idle state while a display-element generator performs its function.

On completion of the appropriate function, the display element-generator asserts "function complete", and WDRQ is generated. The mode register is modified, if required, on the leading edge of "function complete".

3.4.3 The Half-Word Instruction Sequence

A half-word sequence, used for the generation of characters and short vectors, is similar to a single-word sequence. "Start function", resulting from the initial WDRQ, initiates the generation of the element defined by the low-order 8 bits of the display instruction. Subsequent timing is dependent upon the content of the display instruction "E" field.

On completion of the function associated with the low-order 8 bits of the display instruction, "function complete" is asserted. If the "E" field (of the low-order half-word) contains zero, generation of another character or short vector is indicated, and "start function" is immediately reasserted, as illustrated in Figure 3.11a. Following element generation associated with the high-order 8 bits of the display instruction, "function complete" is asserted. If the "E" field contains zero, generation of another character or short vector is indicated, and "start function" is immediately reasserted, as illustrated in Figure 3.11b.
a) CASE I - PROCESSING 2 CONSECUTIVE HALF WORDS

b) CASE II - PROCESSING LOW ORDER HALF WORD ONLY

FIGURE 3.11 THE HALF WORD-SEQUENCE
FIGURE 3.12 THE DOUBLE-WORD SEQUENCE
instruction, "function complete" is again generated, resulting in WDRQ, a request for a new display instruction.

If the "E" field (associated with either half-word) is non-zero, the mode register is cleared in preparation for a control instruction, and WDRQ is asserted. Figure 3.11b illustrates the case where the "E" bit of the low-order half-word of the display instruction is non-zero.

During a half-word sequence, the appropriate 8 bits of the display instruction are switched onto the IRM bus for use by the display generator.

3.4.4 The Double-Word Instruction Sequence

The first half of a double-word sequence (Figure 3.12) is identical to an immediate sequence except that TP1 is used to load the first word of a long-vector instruction into IR2. The second half of a double-word sequence is identical to a single-word sequence.

3.5 The Display Oscilloscope Interface

The interface to the display oscilloscope is illustrated in block diagram form in Figure 3.13. Major axis components include: X and Y up/down counters, and D/A converters. A scale register, an intensity register and D/A converter, and beam unblanking circuitry complete the interface.

The X up/down counter is a 10-bit, synchronous counter constructed from J-K flip-flops. The counter can be parallel loaded from the DAT BUS. A count direction input indicates count up or count down, and a toggle line initiates the counter state change. The counter can be incremented or decremented by 1, 2, 4 or 8 units for each toggle pulse, as specified by the scale register content.
FIGURE 3.13 THE DISPLAY OSCILLOSCOPE INTERFACE
The content of the X up/down counter is transferred to the 10-bit, buffered, X D/A converter which directly drives the X-axis of the display oscilloscope.

The Y-axis up/down counter and associated D/A converter are identical to the X-axis circuitry.

The intensity register is a 3-bit register consisting of simple latches. Signals from the intensity register are input to a 3-bit D/A converter which directly drives the Z-axis of the display oscilloscope.

A simple two-stage transistor amplifier is used to drive the low impedance, beam unblanking input of the display oscilloscope.

When "UNBLANK" is asserted, a point with co-ordinates specified by the X and Y axis D/A buffers and intensity specified by the intensity register is displayed.

### 3.6 The System Clock

Fundamental timing signals used in the generation of all display elements are derived from the system clock. The clock frequency is programmable, permitting the optimization of display element generation times.

Figure 3.14 is a simple block diagram of the system clock, which consists of two monostable multivibrators, M1 and M2, and a resistor switching network. The monostable multivibrators are triggered by a rising edge applied to the "T" terminal, and can sustain a duty cycle near 100 percent. Duration of the monostable output is determined by an RC network. The resistor switching network permits the selection of 16 distinct pulse durations from M1, as specified by the TIMING SELECTION inputs.
FIGURE 3.14 THE SYSTEM CLOCK

(a) - SINGLE CLOCK CYCLE

(b) - MULTIPLE CLOCK CYCLES

FIGURE 3.15 CLOCK TIMING
Figure 3.15 illustrates typical waveforms obtained from the system clock. A single clock cycle will result from a brief assertion of "ENABLE CLOCK", as illustrated in Figure 3.15a. The rising edge of "ENABLE CLOCK" triggers M1, producing the first clock phase, Cl. M2 is triggered on the falling edge of Cl, generating C2, the second clock phase.

If "ENABLE CLOCK" remains asserted, multiple clock cycles are generated, as shown in Figure 3.15b. C2 is fed back to re-trigger M1, providing a free-running clock as long as "ENABLE CLOCK" is asserted.

The resistor switching network consists of 4 simple transistor switches providing 16 different RC time constants for M1. The duration of Cl ranges from 400 nanoseconds to 10 microseconds. The duration of C2 is fixed at 400 nanoseconds.

The two phases of the system clock are used in a standard fashion by most display-element generators. Normally, a single point within a display element is generated during one complete clock cycle. During the first clock phase, a display-element generator modifies the X and Y display co-ordinates as required for a specific display point. The optimum time duration for Cl can be selected by each element generator to provide the appropriate beam excursion and settling time. C2 is normally used to unblank the CRT beam, displaying the single point.

3.7 Display-Element Generators

3.7.1 The Point and Graph-Point Generator

Points and graph-point functions are performed by a single display-element generator since most of the processing requirements
are common to both. In each case, one display axis is set to an absolute value, and in the case of graph-point instructions, the other display axis is incremented.

Figure 3.16 provides a simple block diagram of the point and graph-point generator, indicating the signals used in the generation process. Figure 3.17 illustrates the fundamental timing waveforms during element generation.

Element generation is initiated when the display processor asserts either "START POINT" or "START GRAPH-POINT". The start signal is used to generate either "LOAD X" or "LOAD Y" (according to the content of the "AX" instruction field), transferring the absolute coordinate to the X or Y up/down counter via the DAT BUS. In the case of the graph-point instruction, the start signal is used to generate a toggle signal, (TOGGLE Y or TOGGLE X) incrementing the appropriate co-ordinate axis counter. The start signal is also used to initiate one full cycle of the two-phase clock by asserting "ENABLE CLOCK".

Phase 1 of the two-phase clock, C1, is used to transfer the contents of the X and Y up/down counters to their associated D/A buffer registers. The duration of C1 is determined by the content of the "D" instruction field, providing for either a typical (1-inch), or worst case (15-inch) beam deflection.

When C1 expires, the second clock phase, C2, is generated. C2 is optionally used to unblank the display oscilloscope beam, according to the content of the "B" instruction field. C2 is used to assert "FUNCTION COMPLETE", terminating the element generation process.
FIGURE 3.16 THE POINT AND GRAPH-POINT GENERATOR

FIGURE 3.17 POINT AND GRAPH-POINT TIMING
3.7.2 The Character Generator

Alphanumeric characters are constructed in a 5 by 7 dot-matrix format by the character generator.

Characters are generated using the X-axis and Y-minor axis D/A converters. Figure 3.18 illustrates the beam positioning during character generation. A character is generated in 5 vertical segments, with each segment produced by incrementing the Y-minor axis 6 times. The Y-minor axis co-ordinate is then cleared and the X-axis counter incremented to position the beam for the next vertical segment. When the 5 vertical segments are complete, an additional X-axis increment positions the beam in readiness for the next character.

Figure 3.19 provides a simplified block diagram of the character generator. The diagram illustrates the primary components of the character generator; these include the horizontal and vertical counters, the character read-only-memory (ROM), Y-minor axis D/A converter, a shift register and the control unit.

The horizontal counter is a 3-bit, modulo-5 counter used to sequence through the 5 vertical segments. The vertical counter is a 3-bit, modulo-7 counter which is incremented through all states during the generation of each vertical segment.

The character read-only-memory is a National Semiconductor Corporation static MOS 3072-bit ROM used to convert a 6-bit ASCII character code to the required dot-matrix pattern. The result of each ROM access determines which of the seven beam positions are to be unblanked during the generation of a vertical segment. The low-order bits of the ROM address are the 6-bit ASCII character code, and the high-order bits are the vertical segment number, defined by the
FIGURE 3.18  BEAM POSITIONING DURING CHARACTER GENERATION
FIGURE 3.19 THE CHARACTER GENERATOR
FIGURE 3.20 CHARACTER TIMING
horizontal counter.

A shift register is used to hold ROM data during vertical segment generation.

Y-minor axis components include a 3-bit Y-minor register and the associated Y-minor D/A converter.

The character generator control unit provides the control signals required for character generation. Figure 3.20 illustrates a portion of the timing waveforms during the generation of a single character.

"START CHAR" is asserted by the display processor to initiate character generation. The low-order 8 bits of IR1 are switched onto the IRM BUS; the horizontal and vertical counters are cleared; and the two-phase clock is enabled. The 7-bit ROM word specified by the horizontal counter and the ASCII code is parallel-loaded into the shift register by the first clock phase, Cl. Cl is also used to strobe the vertical counter into the Y-minor axis buffer register and the X-axis up/down counter to the X-axis buffer register. The rising edge of C2 is used to increment the vertical counter, and the display oscilloscope beam is unblanked during the second clock phase. Shift register data is shifted on the falling edge of C2.

The remaining 6 points in the first vertical segment are generated in a similar fashion. Following the seventh point, the vertical counter overflows, resetting it to zero and incrementing the horizontal counter. A new ROM word is transferred to the shift register, the X-axis up/down counter is incremented, and the generation of the next vertical segment is initiated.

The process is repeated until 5 vertical segments have been generated, completing the character. An extra clock cycle is allowed,
during which an additional X-axis increment is generated, the Y-minor
register is cleared and "CHARACTER COMPLETE" is asserted.

3.7.3 The Short-Vector Generator

Short vectors, as described in 3.2.6, are constructed by
the short-vector generator. Short vectors are generated using the
X and Y major axis D/A converters.

Figure 3.21 provides a simplified block diagram of the
short-vector generator. Major components include: the length counter,
the vector-direction decoder, and the control unit.

The length counter is a 3-bit modulo-7 counter used to
determine the length of a short vector.

Vector direction is established by decoding the "D" field of
the short-vector instruction. The vector-direction decoder translates
the "D" field into the required X and Y-axis up/down counter direction
signals, and routes the master toggle signal (generated by the control
unit) to the appropriate axis-counter.

Control signals used during short-vector generation are
derived from the control unit. Figure 3.22 illustrates the fundamental
waveforms associated with the generation of a single short vector.

Short-vector generation is initiated when the display pro-
cessor asserts "START SHORT VECTOR". The low-order 8 bits of IR1 are
switched onto the IRM BUS, and the two-phase clock is enabled. The
first occurrence of C1 is used to load the length counter from the IRM
BUS.

The rising edge of C2 is used to toggle the appropriate X and
Y-axis up/down counters. The length counter is incremented by the
rising edge of C1, and the axis up/down counters are transferred to the
FIGURE 3.21  THE SHORT-VECTOR GENERATOR
FIGURE 3.22 SHORT-VECTOR TIMING
appropriate D/A buffer registers. If the "B" bit of the short-vector instruction is set, the display oscilloscope beam is unblanked during the second occurrence of C2, displaying the first point in the vector. The second occurrence of C2 is also used to toggle the axis up/down counters, initiating the generation of the second vector point.

Additional short-vector points are generated in this fashion until the length counter is incremented to the zero state. The last vector point is displayed, completing a single short vector. Assertion of "SHORT VECTOR COMPLETE" then restores control to the display processor.

3.7.4 The Long-Vector Generator

Long vectors, capable of spanning the entire display oscilloscope screen, are constructed by the long-vector generator. Vector length and direction are specified by delta-X (DX) and delta-Y (DY) values in sign-magnitude form. The long-vector generator provides an approximation to the vector by displaying points at roughly equal intervals along the path of the true vector. Complete details of the vector generation algorithm are provided as Appendix 1.

Figure 3.23 provides an overview of the long-vector generator. The long-vector generator consists of the vector counter, an X and Y change detector and the control unit.

The vector counter is a 10-bit special-purpose up-counter used to establish the length of a long vector. In order to satisfy the vector generation algorithm, the size of the vector counter need only be N bits, where DX and DY are each less than $2^N$. To minimize the time required to produce a specific vector, some vector-counter
Figure 3.23 The Long-Vector Generator
bits are automatically disabled (according to the highest power of 2 contained in the larger of DX or DY).

The X and Y change detectors compare the vector-counter content to the values of DX and DY, respectively, and modify the major axis up/down counters as required to produce the desired vector.

Control signals required for vector generation are provided by the control unit. Figure 3.24 illustrates the timing signals used in the generation of a specific vector. In this case, the vector corresponds to DX=7 and DY=4, as shown in Figure 3.5.

Vector generation is initiated by the signal "START LONG-VECTOR", which results in the enabling of the two-phase clock. The rising edge of C1 toggles the vector counter, in preparation for display of the first vector point.

C2 is used to modify the X and Y-axis up/down counters through the signals TOGGLE X and TOGGLE Y (C2 is blocked by the X and Y change detectors when an axis modification is not required). The contents of the X and Y-axis up/down counters are transferred to the corresponding D/A converters on the next occurrence of C1. The display oscilloscope beam is unblanked during the second occurrence of C2 and completes the display of the first vector point.

The remaining vector points are generated in an identical fashion until the vector counter overflows to zero. On completion of the last vector point, "LONG VECTOR COMPLETE" is asserted, restoring control to the display processor.
FIGURE 3.24 LONG-VECTOR TIMING
4. IMPLEMENTATION OF THE GRAPHICS DISPLAY SYSTEM

4.1 A Modular Approach

The graphics display system was implemented in a highly modular form for the following reasons:

a) to simplify trouble-shooting (modularity permits the removal of some circuit boards without affecting the operation of others, providing a means for quickly isolating many faults)

b) to permit the addition of special-purpose display-element generators.

Circuitry associated with each display-element generator is independent from all others, so addition of a new element generator is relatively simple. The main restriction is that any display-element generator must conform to one of the standard instruction sequences.

4.2 Purchased Components

Where possible, standard "off-the-shelf" components were purchased for use in the graphics display system. The major purchased components include:

a) the display oscilloscope,

b) the major axis - D/A converters,

c) power-supplies and

d) circuit board mounting panels.

4.3 The Logic Family

The graphics display system was implemented using 7400 and 3000 series TTL integrated circuits. These logic families provide several advantages, including the existence of a large number of functions in the product lines, low propagation delays, compatibility with
4.4 Construction Techniques

The graphics display system was constructed using Digital Equipment Corporation H-series wire-wrap mounting panels. The H-series panels accommodate up to 32 circuit boards, with 72 conductor positions for each board.

Circuit boards with up to 50 wire-wrap sockets for 14- or 16-pin integrated circuits provided the basis for construction of the display system. Approximately 15 circuit boards were required for implementation of the special-purpose display system circuitry.

Wire-wrap techniques were generally utilized for circuit interconnection. The primary advantage offered by this technique is the ability to quickly make changes to circuits.
5. PERFORMANCE OF THE GRAPHICS DISPLAY SYSTEM

Table 5.1 provides a summary of observed element generation time and calculated host processor overhead. CPU overheads are based on DMA fetches from the Supernova processor. All times are approximate.

Element generation times are based primarily on the settling and point display requirements of the display oscilloscope. Character, short-vector and long-vector elements require at least 1.6 microseconds per point (1.2 microseconds to allow oscilloscope settling, and 400 nanoseconds for point display).

The total "flicker-free" capacity of the display system is largely dependent on the scale factor used during long and short vector generation; as the scale factor is increased, the capacity is also increased. The subjective appearance of display elements (particularly long vectors) is adversely affected by an increase in scale factor. The minimum scale factor provides for the generation of a long vector on a 0.01-inch grid. Since the oscilloscope spot diameter is 0.02 inches, a vector generated using the minimum scale factor appears continuous and fairly "smooth" to the human eye. A vector generated using the maximum scale factor is constructed on a .08-inch grid, and the eye is very aware of the discrete point make-up of the vector.

Flicker-free capacity of the display system for graph generation is dependent on both the scale factor and the programmable delay. Capacity and CPU overheads are summarized in Table 5.2.

Small characters are generated on a 0.02-inch grid using scale factor 2. At this scale factor, characters are approximately
<table>
<thead>
<tr>
<th>ELEMENT</th>
<th>GENERATION TIME (µsec)</th>
<th>CPU OVERHEAD (µsec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>POINT</td>
<td>SHORT DELAY 5</td>
<td>1</td>
</tr>
<tr>
<td>GRAPH-POINT</td>
<td>LONG DELAY 15</td>
<td>1</td>
</tr>
<tr>
<td>CHARACTER</td>
<td>60</td>
<td>0.5</td>
</tr>
<tr>
<td>SHORT VECTOR</td>
<td>$2 + 2N$ (1)</td>
<td>0.5</td>
</tr>
<tr>
<td>LONG VECTOR</td>
<td>$4 + 2N$ (1)</td>
<td>2</td>
</tr>
</tbody>
</table>

(1) $N =$ NUMBER OF POINTS IN ELEMENT

**TABLE 5.1 DISPLAY SYSTEM PERFORMANCE**

<table>
<thead>
<tr>
<th>PROGRAMMABLE SCALE DELAY</th>
<th>SCALE FACTOR</th>
<th>APPROXIMATE CAPACITY (1)</th>
<th>APPROXIMATE CPU LOADING</th>
</tr>
</thead>
<tbody>
<tr>
<td>SHORT</td>
<td>MINIMUM</td>
<td>10</td>
<td>20%</td>
</tr>
<tr>
<td>SHORT</td>
<td>MAXIMUM</td>
<td>80</td>
<td>20%</td>
</tr>
<tr>
<td>LONG</td>
<td>MINIMUM</td>
<td>3</td>
<td>6.5%</td>
</tr>
<tr>
<td>LONG</td>
<td>MAXIMUM</td>
<td>26</td>
<td>6.5%</td>
</tr>
</tbody>
</table>

(1) CAPACITY - The number of full-screen (10") graph plots which can be accommodated at a repetition rate of 20 frames per second.

**TABLE 5.2 GRAPH GENERATION PERFORMANCE**
0.14-inch high by 0.1-inch wide, and 70 characters can be accommodated across the full width of the screen. At scale factor 2, the discrete point make-up of each character is only detectable on close examination. 20 full lines of characters can be displayed without objectionable flicker (20 repetitions per second), at the expense of less than 2% of host CPU capacity.

Up to 5000 short vectors (containing an average of 4 points each) can be displayed without annoying flicker. Capacity is independent of the scale factor employed. Host CPU loading at full capacity is approximately 5%.

Long vectors can be generated at various rates ranging from 5 inches per millisecond (minimum scale factor) to 40 inches per millisecond (maximum scale factor). At minimum scale factor, at least 25 ten-inch vectors can be generated without annoying flicker, at the expense of approximately 0.1% of host CPU capacity. The capacity and CPU loading are increased by a factor of 8 when the maximum scale factor is employed.

Typical displays will consist of a mixture of the various element types. Total host CPU loading is a function of the specific display, but will normally not exceed 10% of the Supernova capacity.
6. CONCLUSIONS

A computer-based display system was constructed to augment the existing PDP-9 facility. The limitations of the PDP-9 system were overcome through the use of a high bandwidth display oscilloscope, and a more sophisticated display processor. By accessing and executing display instructions stored in host CPU memory, the display processor relieves much of the host CPU overhead associated with the maintenance of a graphics display. The ability to generate complex display elements (for example, characters and vectors) from single-display instructions reduces host CPU display software requirements considerably.

During the four years between construction of the display system and the writing of this thesis, digital technology has advanced considerably. Examination of alternative solutions to display system implementation in light of the current technology is appropriate. An obvious alternative solution to the problem is to utilize the design of the graphics display system (essentially as outlined in this thesis), but implement the design with state-of-the-art components. Extensive use of medium and large-scale integrated circuits would significantly reduce the size and complexity of the display system.

Another alternative would take advantage of programmable read-only memories (PROM's) now available. This approach would require a new design, the heart of which would consist of a high-speed PROM-driven processor similar to those now used in commercially-available "micro-programmable" mini-computers. This approach could be adopted for the display processor only, or it could be extended to encompass
the functions performed by the display-element generators. The use of PROM rather than standard read-only-memory would facilitate alterations during system implementation as well as later additions or alterations to the display system.
This appendix provides a description of the binary rate multiplier algorithm used in long-vector generation.

Consider an arbitrary vector between the origin and a point whose X and Y co-ordinates are DX and DY respectively. If the vector is generated in time T, then the equation for the X and Y components of the display oscilloscope beam as a function of time are simply

\[ X = DX(t/T) \]
\[ Y = DY(t/T) \]

To produce an approximation to the desired vector, an algorithm is required which will provide, for example, DX equal increments of the X-axis co-ordinate value at approximately equal intervals over the time T.

Consider restricted values of DX and DY which can be represented as:

\[ DX = X_22^2 + X_12^1 + X_02^0, \]
\[ DY = Y_22^2 + Y_12^1 + Y_02^0 \]

where \( X_i \) and \( Y_i \) are binary coefficients. An algorithm for producing DX increments along the X axis, and DY increments along the Y axis over \( 2^3-1 \) time intervals is illustrated in the following table.

<table>
<thead>
<tr>
<th>Time</th>
<th>Action</th>
<th>X-Axis</th>
<th>Y-Axis</th>
</tr>
</thead>
<tbody>
<tr>
<td>( 2^2 )</td>
<td>(INCREMENT CO-ORDINATE IF SPECIFIED COEFFICIENTS ARE NON-ZERO)</td>
<td>( \emptyset \emptyset \emptyset )</td>
<td>( \emptyset \emptyset \emptyset )</td>
</tr>
<tr>
<td>( 2^1 )</td>
<td>Don't increment</td>
<td>( X_2 )</td>
<td>( Y_2 )</td>
</tr>
<tr>
<td>( 2^0 )</td>
<td></td>
<td>( X_1 )</td>
<td>( Y_1 )</td>
</tr>
<tr>
<td></td>
<td>( \emptyset \emptyset )</td>
<td>( X_2 )</td>
<td>( Y_2 )</td>
</tr>
<tr>
<td></td>
<td>( \emptyset 1 )</td>
<td>( X_0 )</td>
<td>( Y_0 )</td>
</tr>
<tr>
<td></td>
<td>( 1 \emptyset )</td>
<td>( X_2 )</td>
<td>( Y_2 )</td>
</tr>
<tr>
<td></td>
<td>( 1 \emptyset )</td>
<td>( X_1 )</td>
<td>( Y_1 )</td>
</tr>
<tr>
<td></td>
<td>( 1 \emptyset )</td>
<td>( X_2 )</td>
<td>( Y_2 )</td>
</tr>
</tbody>
</table>
In general terms, for values of DX and DY up to $2^N-1$, the algorithm can be stated as:

a) locate the least significant non-zero power of two, $P$, in the current time,

b) increment the X co-ordinate if the coefficient $X_{N-P}$ is non-zero,

c) increment the Y co-ordinate if the coefficient $Y_{N-P}$ is non-zero.
REFERENCES


