CRYPTOLOG

January - February 1983

Rcvd 13 OCT 83

SPECIAL ISSUE

CISI ESSAY CONTEST [U]

CRYPTOLOG

To submit articles or letters
via PLATFORM mail, send to

cryptolg at barlc∅5
(bar-one-c-zero-five)
(note: no 'O' in 'log')

Editorial

"Something different" is what we promised
for this issue and this is, for CRYPTOLOG,
something different—an entire issue devoted
to one general field.

Opinions will differ about whether a publi-
cation, even a relatively informal one such as
this is, should always have "something for
everyone." On one level, this question some-
times crops up in the form of "should we have
lots of short articles on a variety of sub-
jects, or should we run fewer but longer arti-
cles?"

Pragmatically, our approach to the question
involves looking at just what articles and
items come in for publication. We don't do a
lot of recruiting of material for the maga-
zine. The board of editors does some, but
they are all busy people and working for the
magazine is a sideline for them. Then, we try
to keep up with what people are doing and what
their current interests are. This means,
among other things, that more articles about
computers are showing up in these pages at the
same time that a general resurgence of
interest in computers is evident, both inside
and outside the agency. That doesn't mean we
aren't still interested in other topics, but
it probably means that we aren't getting many
submissions about other topics. Art Salemme
used to respond to, "Why aren't there more
articles about my field?" with "Why don't you
write one?" That's a good thought.

Finally, and perhaps most important to us,
there is a clear need to develop the exposi-
tory skills around this place. Those who are
particularly good at certain skills have to
find a way to tell others about what they know
and do. This takes practice, and practice
requires a place to publish. CRYPTOLOG is one
such place.

D: 4011963

1982 CISI Essay Awards

In 1982, CISI began an annual CISI Essay Awards competition. The purpose of this competition is to encourage NSA employees to share their expertise and experience in computer and information sciences with the NSA community. The papers submitted were judged in the following three categories:

- Hardware and Software Systems
- Applications, Information Systems, Computing Milieux
- Methodologies and Theory

Each category was refereed and judged separately and monetary prizes of $100 for first place and $75 for second place were awarded in each category. The general criteria for judging were:

- relevance to the fields of computer and information science
- writing style
- organization
- significance to NSA

P.L. 86-36

Two first prizes were awarded, one to [   ] of L091 for her paper "The Future Brightens for Flat-Panel Displays" and one to [   ] of E53 for her paper "Menu Selection as a Tool for Human/Machine Interaction."   P.L. 86-36

Three second prizes were awarded. They went to [   ] of R532 for her paper "Improving Raster Graphics Images by Anti-Aliasing," to [   ] of T333 for his paper "Managing Our Systems for Performance," and to [   ] of B62 for her paper "A Tutorial on Color Theory and Human Color Perception the Color Graphics Programmer."

CISI would like to thank the following judges for their time and effort in judging the fifteen papers that were submitted:

[   ]   [   ]

[   ]
CISI Member-at-Large

P.L. 86-36

Special Issue: CISI Essay Contest

TABLE OF CONTENTS

P.L. 86-36

DOCID: 4011963

The Future Brightens for

FLAT-PANEL

Displays (U)

P.L. 86-36

by [ ]

T Intern

INTRODUCTION

In the United States, more than two million cathode-ray tubes (CRTs) are installed and, despite the recession, market demands continue to grow.[2] Surveys conducted by Booz Allen and others suggest that by 1990, one-quarter of the white-collar clerical and professional employees in the US will have display terminals on their desks.[3] What is important to note is that many of these terminals will not be the traditional CRTs of the past.

The major challenge to the cathode-ray tube is coming from flat-panel displays. Although the CRT has been relatively well accepted for many years, numerous limitations have made it a prey for competing technologies. This has encouraged a strong and rapidly expanding interest in the development and production of flat-panel displays, one of the most viable alternatives in overcoming the many disadvantages inherent to the CRT.

This paper will pursue a thorough examination of flat-panel display technology. The first area of investigation includes a detailed explanation of CRT and flat-panel display technology. We will then proceed to take a close look at four major flat-panel displays currently being either researched or developed, to include a discussion of their physical design, operating characteristics, strengths, and weaknesses. The third area will cite the major advantages and disadvantages of CRTs and flat-panel displays. Fourthly, we will briefly summarize some of

> "There's one feature that a CRT cannot have, and the lack of that feature makes it awkward in certain applications...a CRT cannot be flat."[1]   Larry Tannas

the current research in flat-panel technology. The final area will examine the present and future markets for flat-panel displays.

CREATING THE DISPLAY IMAGE

Displays may be classified according to a number of attributes, a significant one being their screen addressing technique. This section of the paper will discuss the methods that both the CRT and flat-panel devices employ to create a display image.

The CRT uses beam addressing, a process which utilizes a deflected focused beam of electrons to trace the display image on the screen. In CRTs, this can be accomplished in generally two ways (Figure I). First, in both calligraphic and direct-view storage displays, the beam is deflected in a random motion across the phosphor-coated tube to create the display image, somewhat analogous to "painting" the image on the screen.[4] In this manner, the electron beam may be deflected in any sequence given by the computer. Raster displays employ a second method, applying a fixed deflection system which forces the electron beam to scan the CRT in a definite sequence, usually from left to right and from top to bottom. As the electron beam is

FIGURE I
Display Image Creation
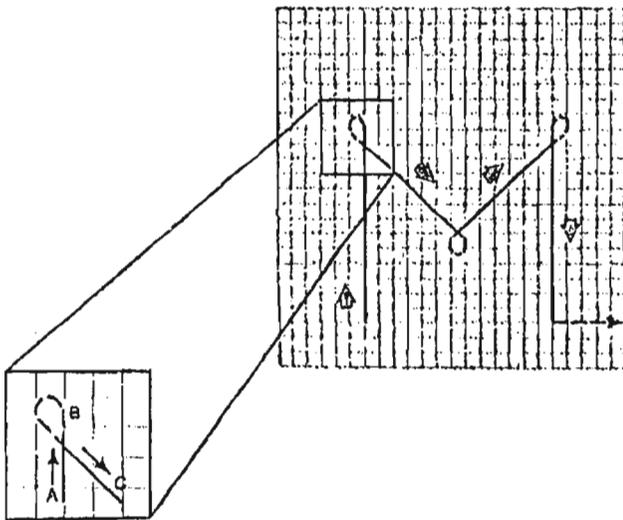
In calligraphic and storage tube displays, vectors are drawn with "strokes" of the beam on the display surface.
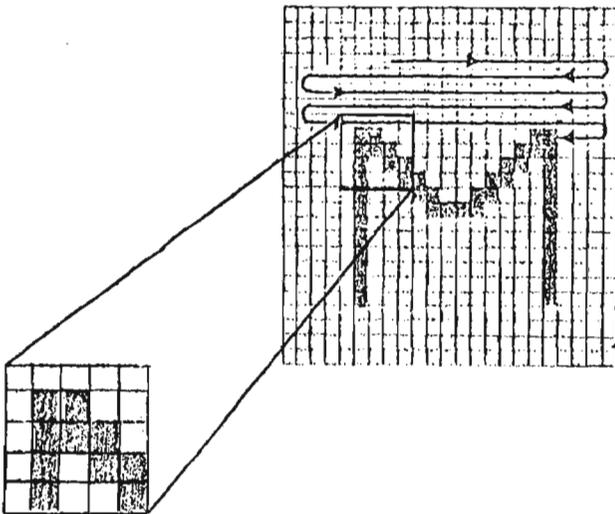
On raster systems, vectors are displayed by turning pixels on and off as the beam moves back and forth down the display surface.

deflected line by line, the beam current may be adjusted to cause variations in brightness along the lines. In television applications, this raster-scanning technique usually applies a method called interlace, which produces a

more random scan sequence.[5] The effect of interlace is to create the display image as two separate fields, the first field containing the even-numbered scan lines and the second field containing the odd-numbered scan

FIGURE II
Interlaced Scan Sequence

——————— first field (even scan lines)
—·—·—·— second field (odd scan lines)
················ horizontal retrace
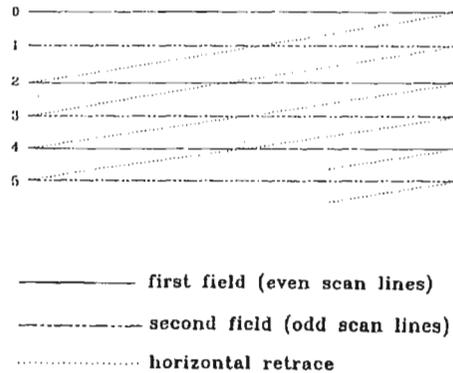
lines. Figure II illustrates the interlace process. This works well with television images, since the pictures they generate do not differ markedly from line to line; this is not always the case in computer graphics; therefore interlace may not always be implemented.[6]

Unlike CRTs, flat-panel displays make use of matrix addressing to create a screen image. This technique employs a matrix of horizontal and vertical microelectrodes, the intersections of which are discrete picture elements, known as pixels.[7] To illuminate a point on the screen, the row and column microelectrodes intersecting at that specific location are activated. In practice, as each row is energized, all of the column lines intersecting that row at locations which require illumination are simultaneously activated. That horizontal row is subsequently turned off and the cycle is repeated with all the remaining rows. In this manner, flat-panel display images are created a line at a time, whereas CRT images are traced a dot at a time.[8]

Each addressing scheme places a different requirement on the display element. Beam addressing requires the display element to be capable of responding to extremely short drive pulses, due to the manner in which the electron beam rapidly sweeps across the CRT screen, point by point, to create an image.[9] In contrast, matrix addressing requires each display element to incorporate a threshold function so that the element will not respond to any excitation it may receive when adjacent

elements are activated.[10] This threshold effect is achieved through the application of a non-linear voltage-versus-light characteristic.[11]

Flat-panel displays all contain some substance which will either glow (emissive) or change the way light is reflected (non-emissive) when a voltage is applied. If such displays employed a linear voltage-versus-light characteristic, as any pixel is selectively excited, all of the pixels along the associated column and row electrodes would glow dimly, activated by a partial voltage.[12] This is inadequate for a satisfactory matrix-addressed display, therefore a non-linear voltage-versus-light characteristic is applied. In this way, as voltage is increased between the horizontal and vertical electrodes, insignificant light emission or reflectance occurs at the selected pixel until a threshold is passed, at which time the light output rises steeply.[13] A pictorial representation of both the linear and non-linear voltage-versus-light characteristic may be found in Figure III.

FIGURE III
Threshold Effect
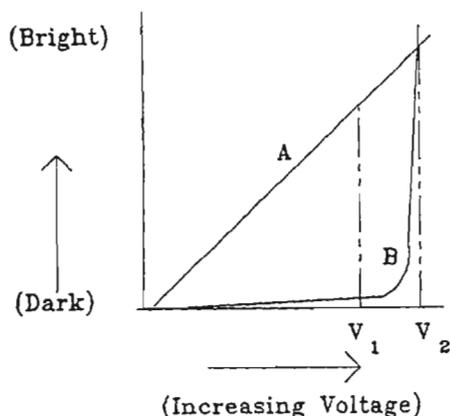
Threshold effect is needed so a single circuit can drive all pixels along one line. Otherwise, for a linear curve (A), all the other pixels along the selected line would glow when any voltage is applied. But with nonlinear curve (B), they would not glow even when a large voltage V_1 is applied to the row. Combined with a small additional voltage to the column electrodes, the threshold V_2 is passed, creating a bright dot at that crosspoint.

As a result of this non-linear characteristic, a fairly high voltage can be applied to all the elements of a selected row with a single driver. While this voltage is maintained below the threshold level, all the pixels in the selected row will remain dark. When small voltages are applied to selected column drivers, the potential at the crosspoints of these row and column electrodes will rise above the threshold level, causing selected pixels along the activated row to be illuminated. The duration or strength of the voltage can be varied to control the brightness or gray levels of the selected pixels. By repeating this process for each row of the display device, an entire image can be created, line by line, on the viewing surface.[14]

In selecting a display device, an important factor to consider is its particular screen-addressing technique. If the application demands high-resolution moving images with full color and good contrast, a CRT may be the best selection. On the other hand, if full color is not essential, but high precision and good resolution are, a flat-panel display may be the better choice. Whatever the application, understanding the special advantages each screen-addressing technique has will provide assistance in making a selection.

EXPLORING FLAT-PANEL TECHNOLOGY

Among the many flat-panel displays currently being researched and marketed, several esoteric technologies may be identified as being potentially the most viable alternatives to the CRT. In this section of the paper, four flat-panel display implementations will be examined in terms of their design, operating characteristics, strengths, and limitations. These four technologies include gas plasma, electroluminescent, electrophoretic, and liquid-crystal displays.

Gas plasma displays

Of the major flat-panel displays currently available, gas plasma panels appear to be the most prolific. This may not be a function of their technical superiority, but rather a result of being marketed longer.

The notion that light could be produced by an electric discharge through a gas has been known for over 200 years. Scientific interest in this phenomenon was stimulated as early as the mid-1800s, when Geissler fabricated glass tubes with electrodes sealed at opposite ends and filled them with gases such as oxygen, carbon dioxide, and hydrogen under low

pressure.[15] While these tubes had a short operating life, they were of technical interest for the study of radiation spectra of different gases.[16] Following the discovery of neon in the early 1900s, elongated tubes containing this gas came into use for advertising signs.[17] Soon after World War I small neon-filled bulbs became available commercially because of their usefulness as on/off indicators for line voltage.[18] Today, the demand for information displays in digital computer systems and the persistent desire for flat-panel displays has spurred a steady growth in the development of gas plasma displays.

The essential picture element (pixel) in gas plasma displays is an electrical gas discharge that develops between two electrodes when the applied voltage exceeds a threshold level. In a simple display, such as an on/off indicator, there is only a single discharge site. In recently developed high-resolution matrix displays, however, there are over one million discharge sites.[19] If such a high resolution matrix required two million electrodes, the development of such a display would be completely impractical. In matrix displays, electrodes are usually shared by a row or column of discharge sites. One million discharge sites in this configuration would require only 2,000 electrodes, a thousand for each axis. This makes development of such high-resolution matrix displays extremely practical and highly desirable.

Gas plasma displays can be classified according to a number of attributes. They are usually identified as being either AC or DC displays.

▶ In AC versions, dielectric surfaces separate the electrodes from the gas.

▶ In DC versions, the electrodes are immersed in the gas.[20]

Another classification relates to their mode of operation as storage or nonstorage.

▶ In storage mode, the memory that holds the image information is inherent to the display device.[21]

▶ In nonstorage or cyclic mode, the memory is external to the display. Unless the image can be represented by a single discharge, the image information is transferred to the device and displayed sequentially, usually one row or column at a time. To avoid flicker, this entire image must be continually refreshed.[22]

FIGURE IV
DC Plasma Panel
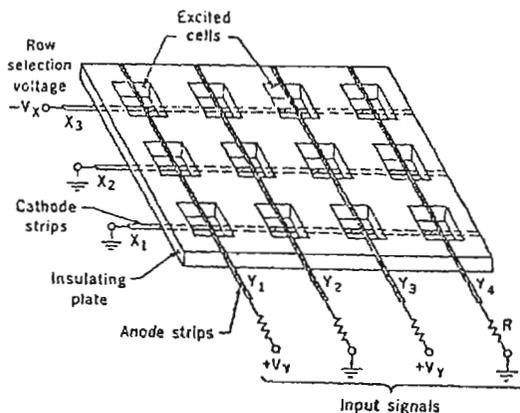
The basic structure for a matrix-addressed DC plasma display is presented in Figure IV. The individual cells in this display type are defined by holes in the insulating plate. In line with the holes is a set of cathode strips on the lower surface of the plate. Running perpendicular to these is a set of anode strips on the opposite surface. While not depicted in this diagram, it must be assumed that this entire structure is contained between a pair of glass plates that are vacuum-sealed at the edges and filled with gas at a low pressure.[23] To operate, a pulse voltage, $-V_x$, is applied to one of the cathode strips while input signals in the form of pulse voltages, $+V_y$, are applied simultaneously to selected anode strips. Since the voltage sum $V_x + V_y$ is assumed to be greater than a threshold voltage, V_t, for initiating a gas discharge, light emission is produced at the excited cells, wherever the two voltages intersect. Since the individual voltages V_x and V_y are assumed to be less than the threshold voltage V_t, no other cells of the matrix will be excited. Similarly, other rows of cells can be repetitively addressed in rapid sequence to produce a flicker-free image.[24]

An alternative gas plasma device in commercial use today is the AC plasma panel, depicted in Figure V. In this display, the X and Y conductors are fabricated on the inner surfaces of two glass plates. These electrodes are then covered with a thin layer of glass so there is no direct contact between the electrodes and the gas contained between the two plates.[25]

FIGURE V
AC Plasma Panel
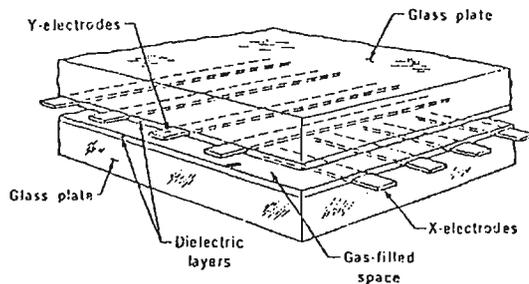
During operation of this plasma display, an AC sustaining voltage is maintained between the sets of X and Y conductors. This voltage alone is insufficient to excite any cells in the off state. If voltage pulses of suitable magnitude are applied to a selected pair of X-Y conductors, a discharge can be initiated. This discharge will quench itself within approximately one millisecond, due to the buildup of charges on the insulating walls resulting from the current flow. Following this, the cell will continue to fire on successive half-cycles, since the voltage built up on the walls during each half-cycle will add to the applied voltage of the next half-cycle. Subsequently, triggering selected cells in sequence, a complete image can be stored.[26]

The plasma panel has several advantages and disadvantages. Its ruggedness makes it more appropriate for some military and industrial applications where CRTs would be unacceptable.[27] Its small size makes it suitable in aircraft and certain office environments where space may be a critical consideration. With a diffusing surface on the back or within the panel, optical images can be superimposed on the digitally-generated images.[28] Finally, the plasma panel produces steady and flicker-free images and text that is easy to read and interpret.[29]

The disadvantages of plasma display are its poor resolution and complex addressing and wiring requirements.[30] Its inherent memory may be useful in certain applications, although it is not as flexible as the frame-buffer memory available in CRTs.[31] While gas discharge technology has stimulated efforts to

develop color displays, this area remains primarily in the research stage due to the technical problems and costs associated with this enhancement.[32]

Electroluminescent displays

The flat-panel display technology most likely to challenge the characteristics of the CRT is the electroluminescent display--in particular, the thin-film electroluminescent layer display. As early as 1937, G. Destrian showed that light could be obtained from specially prepared zinc sulfide powder layers when an AC electric field was applied.[33] Unfortunately, development of this technology was delayed until 1950, due to a lack of satisfactory transparent electrodes for viewing the phosphor.[34] In 1950, however, workers at Sylvania, using a similar phosphor powder, were successful in producing cells which employed recently-developed transparent coatings of tin oxide on glass.[35] From that point, research has expanded to include improving electroluminescent phosphors and developing display devices based upon these materials.

FIGURE VI
Electroluminescent-layer Display
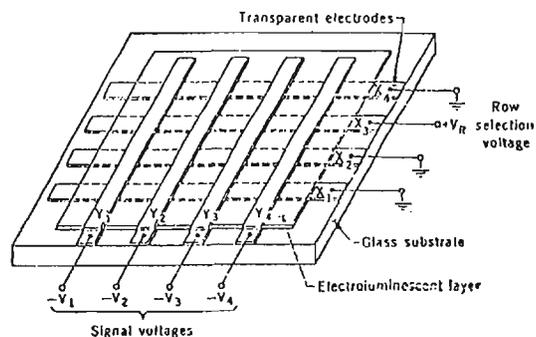
In the early 1970s an electroluminescent layer was developed that could be excited with a DC (or pulsed DC) voltage and that exhibited highly desirable characteristics for display applications. These devices are of particular interest for displays containing a large number of pixels, due to their proven high light output obtainable through short DC pulses and sharp threshold voltages.[36] As with gas plasma displays, electroluminescent layer displays employ a matrix addressing scheme to reduce the number of electrodes necessary for the display. The operation of

this device is highly similar to that of the gas plasma display, and tests have shown that a flicker-free image can be produced by applying the pulse voltage, $+V_r$ to successive rows and cyclically repeating this addressing process.[37] (Figure VI) Panels of this type have been developed for experimental purposes to display both alphanumeric information and television images.[38] Electroluminescent panels, 20 by 27 centimeters in size, containing 224 x 224 elements, have been developed and successfully exhibited images with good gray scale and highlight brightness of 10fl (foot-lamberts), approximately one-tenth the brightness exhibited on commercial television receivers.[39]

Current efforts have been directed more toward developing thin-film electroluminescent displays, as an alternative to powder layer displays. Initially, these attempts resulted in films with poor light output and limited duration.[40] In the period between 1964 and 1970, workers at Sigmatron were successful in developing much improved films. They achieved this by using manganese-activated zinc sulfide powder, applying additional insulating films on one or both sides of the panel, and by coating the rear surface of the phosphor with an additional light-absorbing layer of arsenic selenide, enabling the transparent phosphor to appear black while in its off state.[41] This led to the development of display panels that could be viewed with good contrast even in high ambient light.

Much like the DC powder layers, thin-film phosphors react to a very sharp threshold voltage, also enabling them to be designed in an X-Y matrix-addressing scheme. Thin-film panels of this type have been developed and used to display alphanumeric information and television images.

Thin-film electroluminescent panels may be scanned at video rates, due to the rapid response of their phosphor.[42] When a given pixel is excited, its light output diminishes to approximately half its original brightness in about a millisecond. This persistence level, comparable to that of CRT phosphors designed for video displays, contributes to the panel's average brightness.[43]

Thin-film panels have been developed to display television images with good gray scales, an important attribute for displaying high-quality pictures. In CRTs, displaying video images with gray scale is an analog function of the electron beam. Thin-film electroluminescent panels typically feed the video input through an analog-to-digital converter, which codes it into 16 levels of gray and converts it to a 4-bit digital number. When the signals are transferred to the column drivers, the 4-bit code associated with each column is used to set the column drivers to one of 16 levels of voltage, thereby varying the brightness levels.[44]

Research indicates that arrays of thin-film panels may be successfully used as memory devices if the manganese content of the zinc sulfide is increased (from approximately 1 percent by weight to approximately 5 percent).[45] While this storage mechanism is still in the experimental stage, panels of this type exhibit the potential for displaying stationary images with a large number of pixels, since slow writing speeds may be used to create images totally free of flicker.

While thin-film electroluminescent panels show promise for a wide range of display applications, several disadvantages are inherent to their design and the elements used to create their displayed images. The high-voltage addressing circuits required to drive these displays are currently a disadvantage of this type of flat-panel device. This problem may be resolved in the near future as suitable low-cost integrated circuits become available.[46] Another problem is the relatively low efficiency of electroluminescent layers resulting in considerable power dissipation in the panel and drive circuits. As a result, electroluminescent layers exhibit only a small percent of the efficiency of the finest electron-bombarded phosphors.[47] Limitations of the materials presently used in constructing X-Y-addressed thin-film panels leads to a maximum obtainable brightness for full television resolution of approximately one-tenth that available from cathode-ray tubes.[48] Researchers have pursued developing phosphors with colors other than the yellow-orange characteristic of the standard thin-film panels, but they tend to exhibit a significant decrease in efficiency.[49]

Electrophoretic displays

Another flat-panel device, the electrophoretic display, is a nonemissive device based upon the principle of electrophoresis--the migration of charged particles in an electric field.[50] This display makes use of a thin layer of dyed fluid in which pigment particles of a highly contrasting color or reflectivity are suspended.

FIGURE VII
Electrophoretic Display

TRANSPARENT ELECTRODE

o +ELECTRODE SEGMENT

SUBSTRATE

COLOR OF DYE

OBSERVER

CHARGED PIGMENT PARTICLES (−)

DYED SUSPENDING MEDIUM

COLOR OF PIGMENT

o −ELECTRODE SEGMENT

SPACER/SEAL

GLASS PLATE
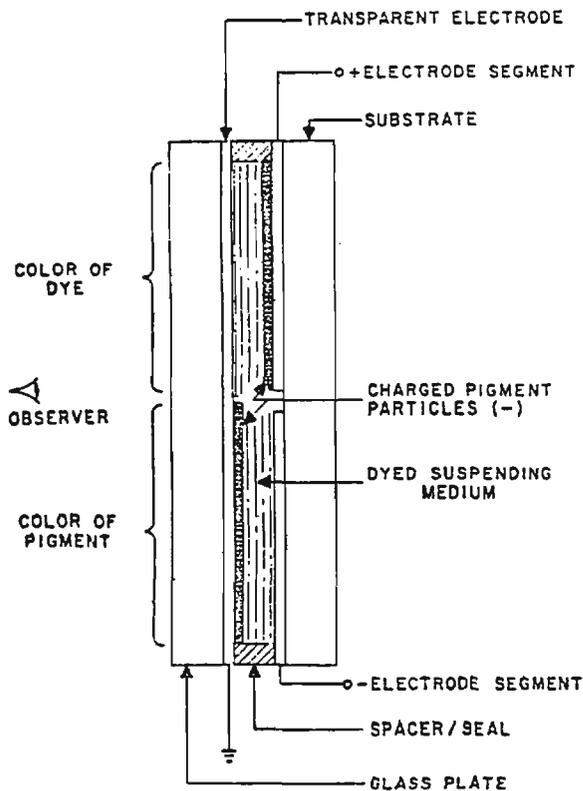
Electrophoretic display images are formed by light-colored pigment particles being deposited on a viewing electrode against a dark background of dielectric fluid. By electrostatically driving the pigment to the viewing surface of the display, the pigment color becomes observable at that specific picture element. Conversely, by electrostatically driving the pigment toward the back electrode and thus, away from the viewing surface, the pigment particle is masked from sight by the dark colored fluid.[51] (Figure VII)

The advantages of electrophoretic displays, including high contrast, wide viewing angles, inherent memory, and low power consumption, have encouraged researchers to apply this medium to large-information-content flat-panel devices.[52] Two methods investigated to accomplish this goal include the direct matrix and active matrix approaches.

The direct matrix, containing only the display medium held between the row and column electrodes on their respective cell walls, is

relatively easy to build and simple to operate. Not only does this provide for an attractively low overall system cost, but a further advantage is its inherent memory which eliminates flicker, avoids the necessity of refreshing, and allows for high levels of multiplexing.[53]

While the early electrophoretic fluids often lacked the well-defined voltage threshold required for direct matrix addressing, several techniques have been researched which successfully ameliorate this problem. One technique introduces an external threshold by fabricating three sets of electrodes and operating them in a triode mode. This successfully permits matrix addressing and also reduces addressing times.[54] A second technique has proven successful by building an inherent threshold into the electrophoretic suspension. This is accomplished by adjusting the particle-to-particle and particle-to-electron interaction. The voltage threshold is thereby created by either altering the composition of the suspension or the nature of the surfaces.[55]

The active matrix approach, with matrix drivers and local storage capacitors integrated into the display, appears to be the most viable display technique for a high-speed, page-size, interactive electrophoretic display.[56] An experimental 32 x 32 varistor-capacitor array-driven electrophoretic device has been developed which creates images a line at a time without crosstalk, flicker, or refreshing.[57] Future varistor-capacitor array-driven electrophoretic devices should have higher resolution and larger viewing areas for better utility.

Liquid-crystal displays

The final flat-panel device to be discussed is the liquid-crystal display. Like the electrophoretic display, this device is a nonemissive display, since it controls the transmission or reflection of external light.[58] This characteristic makes it especially useful for displays that must be viewed in high ambient light.

Liquid-crystal materials belong to a class of organic compounds which, under certain temperature constraints, exhibit several optical and electrical properties characteristic of crystalline solids.[59] Interest in liquid-crystal materials for display devices first began in the 1960s when it was demonstrated that an electric field applied across a thin

layer could produce significant changes in light transmission with very low power consumption.[60] Since that time, following extensive research, liquid-crystal displays have proliferated in the market in watches, pocket calculators, and various other electronic instruments.

Molecules of all liquid-crystal materials are generally elongated in shape, causing variations in the index of refraction depending upon the orientation of the molecules. Most liquid-crystal displays currently available use twisted nematic molecules.[61] The alignment of these nematic liquid crystals is essentially parallel. In addition, if the alignment direction is established at the liquid's boundary, such as at the surface of the glass plates restraining the liquid material, all the molecules tend to assume the same alignment.[62]

FIGURE VIII
Liquid-crystal Display

A twisted nematic display is composed of a layer of nematic material contained between the transparent conductive surfaces of two glass plates. This structure is then placed between two polarizers. Before constructing the panel, the glass plates are specially treated to cause the molecules at the surface to align in a direction nearly parallel to the surface of the plate. Figure VIII(a) shows this configuration, which also represents the appearance of a twisted nematic liquid-crystal cell when no voltage is being applied. In this configuration, light is guided by the helical molecules so that the panel is bright with the voltage off.[63]

When an AC voltage is applied across the cell, nearly all the molecules will align themselves perpendicular to the glass surfaces, as shown in Figure VIII(b). With the molecules aligned vertically between the two plates, light is blocked by the polarizers.[64]

Liquid-crystal displays consume less than 1 microwatt per square centimeter, considerably less power than that required for luminescent displays.[65] This makes them highly practical for portable, battery-operated devices. An important disadvantage of liquid-crystal displays is their relatively slow response, their turn-on and turn-off times both requiring approximately 0.1 seconds at room temperature.[66]

Considerable effort has been directed toward developing liquid-crystal displays that do not require polarizers. Success has been achieved by dissolving dichroic dyes in the liquid-crystal material. This improves brightness and provides for a wider viewing angle, two significant limitations of the twisted nematic displays. In panels of this type, positive-contrast images (colored images on a colorless background) or the reverse can be produced in a variety of colors depending on the dyes used.[67]

X-Y-addressed arrays with good contrast over an acceptable viewing angle (more than 10 rows) have not yet been developed for liquid-crystal displays. This is largely due to the lack of a sharp threshold function.[68] Attempts to overcome this limitation involve either incorporating a field effect transistor at each pixel to prevent voltage from appearing along unselected elements or utilizing collimated light to exploit other optical effects in liquid crystals whose threshold is much sharper. This second technique should allow display images to be created with several hundred rows.[69]

ADVANTAGES AND DISADVANTAGES OF THE CRT

Despite the competition from flat-panel and other display technologies, the cathode-ray tube remains preeminent in today's market. Its numerous advantages make it a stiff competitor for less researched and developed technologies. The CRT is a high-resolution display, adaptable for both individual and small group viewing.[70] Its method of screen addressing is through the use of a deflected focused beam of electrons. This is especially useful for applications where the information to be displayed is supplied serially.[71] The CRT is the basic component for the direct-view storage tube, calligraphic displays, and raster displays, adding to its continued success.[72] The wide range of phosphors and careful filtering have made it possible to achieve good contrast in extremely bright ambients.[73] Phosphors exist which also exhibit good and predictable lifetime characteristics.[74] CRTs are inexpensive and relatively easy to manufacture.[75] They can produce high-resolution color displays having wide viewing angles without degradation of brightness or contrast.[76] Their phosphors emit light instantaneously and decay quickly, thereby permitting real-time video displays.[77]

While the CRT has proven adaptable to the demands of many applications, numerous disadvantages render it inappropriate or useless in others. CRTs are bulky devices, occupying considerable desk space. This becomes a serious concern in applications where space considerations are paramount. They lack ruggedness, with their thin glass envelopes susceptible to breakage.[78] Their screen surfaces reflect light, a characteristic which can both distract and annoy the user.[79] Limited success has been achieved in developing self-luminous projection CRTs. It appears difficult to produce and collect sufficient light from a self-luminous CRT to provide bright, high-resolution, large-screen displays with adequate cost and lifetime.[80] The CRT is an analog device, making it inadequate for interactive graphic applications where true digital displays are necessary.[81] CRTs nearly always exhibit some distortion because the electron beam must travel different distances to strike all areas of the screen surface, striking it at different angles and with varying spot size and shape.[82] Finally, the CRT is a high-voltage device, a definite disadvantage under low-pressure conditions, such as in aircraft.[83]

BENEFITS AND LIMITATIONS OF FLAT-PANEL DISPLAYS

Functionally, flat-panel displays offer a viable alternative to CRTs, especially in applications where space considerations play an important role in the design of a system. Some of the numerous advantages of flat panels, which make them especially attractive for military and certain industrial applications, include their ruggedness, portability, shock resistance, ability to withstand wide temperature variances, long-life characteristics, and low power consumption requirements.[84] Flat panels are inherently free of geometric distortion and have such "built-in" features as orthogonality, linearity, and resolution.[85]

Economically, flat-panel displays are still too expensive to replace CRTs: a 10-to-1 price ratio currently exists between flat-panels and CRTs.[86] Production costs of AC and DC plasma panels, liquid-crystal displays, and electroluminescent panels are significantly greater than those of CRTs with relatively the same resolution.[87]

Two major factors contributing to the high cost of flat-panel display terminals are their expensive drive electronics and low production volumes. Flat-panel display terminals continue to maintain a high price tag because they are still produced in small quantities, compared to CRTs.[88] Even if production increased dramatically, the cost of drive electronics would have to drop significantly before the market would open for flat-panel displays.[89]

Progress is currently being made in that direction. Texas Instruments (TI) has developed a 32-channel integrated circuit that significantly reduces the display-drive electronics. "Where it once took two diodes and a resistor to drive each line of a 512 x 512 display, it now takes only 32 TI integrated circuits."[90]

Despite the high costs, flat-panel displays should not be dismissed when searching for a display technology that best suits the needs of a specific application. In many situations, flat-panel terminals can be justified due to their size, durability, and low power consumption.

FLAT-PANEL DISPLAY APPLICATIONS

As long as complicated drive circuitry and low-volume production continue to make flat-panel displays expensive, applications for these devices may remain limited to military and highly specialized commercial users.

IBM's flat-panel displays are available with their retail and banking systems and are primarily targeted for high-volume-usage applications.[91] The Q-1 Corp., Hauppauge, N.Y., has incorporated flat-panel terminals into systems for its small business and banking customers, who appreciate the durability and space savings of the displays.[92] Companies such as General Digital Corp., East Hartford, Conn., and Science Applications, Inc., La Jolla, Calif., sell flat-panel display terminals to industrial customers for use in factories and on oil-drilling platforms, applications demanding a rugged display device.[93] A DC plasma display terminal marketed by General Digital Corp., is being used for applications such as an instructor's controller in advanced aircraft simulators, a metal-rolling-mill controller, an automated electronic testing device, an executive desk-top information retrieval terminal, and a control and monitoring device in manufacturing and food processing plants. Named the VuePoint, this flat-panel device incorporates a Burroughs display, the Self Scan II, and uses a touch-sensitive screen overlay that eliminates the need for a keyboard.[94]

Until the high cost of flat-panel technology is considerably reduced, many suppliers will have to depend heavily on military contracts to remain in the market. The military is generally able to base purchasing decisions on utility rather than cost, which has enabled them to investigate and purchase flat-panel terminals applicable to their specific needs.[95] Elliot Schlam, Chief of the display division of the US Army's Electronics Research and Development Command (ERADCOM), Fort Monmouth, N.J., says "the Army decided 12 years ago that it had to use automated data-processing equipment on the battlefield rather than rely on antiquated means of communication, such as teletypewriters, walkie-talkies, and hard-copy message deliveries. Flat-panel displays, rather than CRTs, are playing a major role in the military's communication scheme.... A battle unit needs a rugged, portable terminal that can withstand shock, vibration, and a wide range of temperatures, one that will have a long life, and one that will not consume much power. Considering those criteria, CRTs do not match up to flat-panel displays."[96]

Among the many flat-panel display devices being used in military applications is a portable unit that fits into a briefcase, built by Hycom, Irvine, Calif., a subsidiary of Sharp.[97] This particular system is intended for a wide range of functions, to include command and control, intelligence-data relay, radar display, air-traffic control, and ballistic calculations. Other flat-panel devices being built for the military include large-screen displays for use in command centers. One such display is being manufactured by Photonics, Luckey, Ohio, an AC plasma display supplier. It is a large-screen AC plasma device, assembled by Magnavox, measuring 1 meter diagonally and containing 2 million light-emitting pixels.[98] In addition, Photonics and other flat-panel suppliers are producing small displays that can be incorporated into hand-held computers, transparent map-overlay displays, and instrumentation displays for aircraft and submarines.[99]

While plasma displays have primarily been designed into military terminals over the past 10 years, Elliot Schlam says most future contracts under his control will be for electroluminescent displays and states, "In a point-by-point comparison of plasma and electroluminescent displays, electroluminescent models weigh less, consume less power, and are priced lower than plasma displays."[100] In his opinion, electroluminescent technology is clearly the one to pursue.

Hycom has made its first large-scale commercial sale for a portable terminal that will be used for such applications as field service, diagnostics, and electronic mail. Riley Holly, an assistant vice president for the company says, "It will be like an Apple [computer] in a suitcase. The price should be comparable to what a small Apple would cost, but the user will be able to carry the system around with him."[101] He also states that the unit will be battery operated and may use CMOS nonvolatile memory.

CURRENT RESEARCH IN FLAT-PANEL TECHNOLOGY

The CRT has maintained its preeminent position in the field of television and information displays largely as a result of its many outstanding characteristics, including high resolution, wide range of colors, high brightness, good contrast, low cost, and firmly established manufacturing techniques. Its numerous disadvantages, however, have encouraged researchers to actively investigate alternative display technologies. This section of the paper will briefly describe some

research currently being undertaken in the area of flat-panel technology as attempts to solve these limitations.

The Defense Advanced Research Projects Agency (DARPA) has partially supported the development of a 512-character electrophoretic display by Philips Laboratories, Briarcliff Manor, New York. This matrix-addressed electrophoretic image display, using a control electrode principle, is a non-emissive flat-panel display having 228 columns and 148 rows with pixels 0.25 mm square. Commercially available high-voltage shift registers and microprocessor control have been used in driving this display device.[102]

An important characteristic of this electrophoretic image display is the application of the control electrode principle. This entails creating an electrostatic threshold for switching by introducing a third electrode between the two electrodes of the basic device. This third electrode is in the form of a mesh and is detailed in Figure IX. The significant advantage of this matrix addressing approach is its extremely fast addressing time.[103]
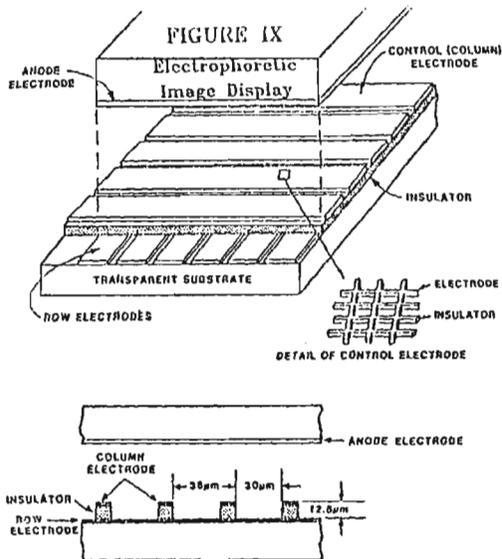
Research indicates that the control electrode principle provides a practical and dependable method for the introduction of an electrostatic threshold into an electrophoretic image display. The basic structure is simple and the threshold is not materials dependent, but is controlled by the geometry of the device. It does not require special suspensions and it appears that the size of the display and the number of pixels can vary to include large screen displays with few serious limitations. The developers of this electrophoretic image display believe it will make the use of non-emissive displays in large information content applications highly practical in the near future.[104]

Individuals at IBM's Watson Research Center, Yorktown Heights, New York, have investigated the notion of a storage CRT display using a thin-film electroluminescent faceplate.[105] After summarizing their findings in the areas of selective erasure, attainable levels of brightness, contrast, resolution, and the constraints and concerns associated with the development of such a large thin-film electroluminescent faceplate, they support the concept of developing this storage CRT. It show promise for exhibiting such outstanding display features as high luminance, good contrast, no flicker, high information capacity, and reasonably high rates of interactivity.[106] Technical problems in fabricating a practical device of this type remain, yet the developers do not perceive them to be insurmountable. Solving these problems will most likely yield a highly useful and marketable display device.

James L. Ferguson of American Liquid Xtal Chemical Corp., Kent, Ohio, has investigated the use of birefringence in liquid-crystal displays and has developed what he terms "the fastest mode of drive for a nematic liquid crystal yet available."[107] While many of the characteristics of this device remain to be investigated, Ferguson asserts that a high-performance matrix display can be produced which has the capability of presenting a moving image without tailing or sticking, and attainable multiplexing levels in excess of 30 to 1.[108]

Jacques Robert, partially sponsored by the Centre National d'Etudes des Telecommunications of France, has developed a complex liquid-crystal matrix display to be used as a television screen.[109] The three major components of his experimental video display, developed mainly for videophone applications, include a liquid-crystal cell, driving electronics, and projection optics. It has

FIGURE IX
Electrophoretic
Image Display

ANODE ELECTRODE

CONTROL (COLUMN) ELECTRODE

INSULATOR

TRANSPARENT SUBSTRATE

ROW ELECTRODES

ELECTRODE

INSULATOR

DETAIL OF CONTROL ELECTRODE

ANODE ELECTRODE

COLUMN ELECTRODE

INSULATOR

ROW ELECTRODE

128 x 128 display elements, multiplexing capabilities allowing a display of eight shades of gray and an image change rate greater than five times per second. Intermediate results have demonstrated that fairly complex pictures of faces can be produced with the stated system characteristics. Robert suggests this is simply the first stage in the development of a liquid-crystal video display. He asserts that performance can be enhanced by such improvements as increasing the number of display elements to 256 x 256, increasing the levels of gray scale to 16 or 32, and improving the image rate.[110]

A low-threshold-voltage thin-film electroluminescent device has been developed by members of the Faculty of Engineering Science, Osaka University, Osaka, Japan.[111] Although high brightness, long life, and limited color have been achieved for thin-film electroluminescent displays, the problem of high voltage requirements has remained. This has severely hampered the use of commercially available integrated circuits, leading to highly complicated and expensive driving circuits. This voltage requirement has also prevented improvement of the device's reliability. The low-threshold-voltage thin-film electroluminescent has been successfully implemented at 60 volts. This is nearly one-fourth the voltage of most conventional thin-film electroluminescent devices.[112] In addition, it has successfully exhibited brightness levels of 300fl at 60 volts, with a maximum attainable of more than 800fl.[113]

Exxon has been particularly active in flat-panel display research. Kylex, an Exxon subsidiary in Mountain View, Calif., has recently developed a liquid-crystal flat-panel device capable of displaying eight lines of characters. While this appears quite restrictive, there is speculation that Kylex displays will soon be expanding to CRT capacities.[114]

Innovative research is being conducted by a wide variety of groups. Hopefully, their continued efforts will assist in promoting greater interest in the development of all types of flat-panel displays, leading to an enhanced display technology without the identifiable disadvantages characteristic of the CRT.

THE PRICE DIFFERENTIAL

Ken Bosomworth, President of International Resource Development, Inc., Norwalk, Conn., suggests that there is a 10-to-1 price ratio between flat panels and CRTs.[115] A cost comparison of the two technologies easily substantiates this figure. Purchase price estimates for a basic CRT range from $100, for a single-unit purchase, to as low as $60 for volume purchases.[116] In contrast, Electro-Plasma of Millbury, Ohio, manufactures a 256 x 512-pixel AC plasma display which sells for $2,450 in single-unit orders and $1,500 for volume purchases greater than 1,000.[117] In a similar price range, Sharp manufactures a 240 x 320-pixel electroluminescent display that sells for $2,800 in single-unit orders, with no quantity discounts.[118]

Despite these high costs, many terminal manufacturers remain in the market and appear enthusiastic about the future of flat-panel display technology. An example of one such manufacturer is Televideo, Inc., San Jose, Calif. Mel Snyder, Vice President for Marketing and Sales of Televideo, refers to flat-panel terminals as the "products of the future" and adds that his company is particularly interested in the development of electroluminescent technology.[119]

As such optimism continues to grow and pervade the manufacturing community, the research and production of flat-panel technology will undoubtedly surge. Ultimately, this should lead to a significant drop in flat-panel display prices, closing the price gap between CRTs and flat panels, and enabling flat-panel technology to become more economically competitive with the CRT.

CURRENT MANUFACTURERS OF FLAT-PANEL TECHNOLOGY

Despite the limited market and high costs presently associated with the development of flat-panel displays, many companies remain actively involved in the research and production of this technology. Companies as large as IBM, Nippon Electric Company, Burroughs, and Sharp, and as small as Electro-Plasma and Photonics, are highly supportive of advancing the development of flat-panel display products.[120] Manufacturers in the process of developing prototype TV displays using plasma or liquid-crystal devices include Fujitsu, Hitachi, Matsushita, and Seiko Denki Company.[121] Military data terminals utilizing gas plasma panels are currently in production by Magnavox, SAI, Interstate, Singer Librascope, and Norden.[122] Aerojet Electro Systems, Azusa, Calif., and Rockwell Electronics Research, Thousand Oaks, Calif., are developing new thin-film electroluminescent panels with self-contained integrated circuit drivers

for upcoming enhancements in Army data terminals.[123]

Hycom, a US-based subsidiary of Sharp, is one of the leading manufacturers of flat-panel display technology. Hycom has produced a thin-film electroluminescent panel for use in the Army's Digital-Message Device, a portable battlefield terminal. While manufacturing costs of a comparable CRT would be considerably less, this panel requires much less power and has a longer life. Hycom speculates that high-volume production will bring manufacturing costs down to CRT levels within the next two years.[124] Hycom is nearing delivery of a Tactical Video Display, which has demonstrated the compatibility of a thin-film electroluminescent display with a standard video signal. Company officials claim commercial versions of the Tactical Video Display will sell for less than $500 each for production rates of one thousand or more per month. A special feature of this device is a black layer on the rear surface of the display. This will provide for greater contrast in high ambient light, such as will be found in field operations or in helicopters.[125] Hycom is currently marketing electroluminescent panels with high resolution and picture element density applicable for both graphic and alphanumeric displays.[126] Hycom and GTE's Lamp Division, Salem, Mass. are in the process of developing fully transparent thin-film electroluminescent displays for Army Tactical Data terminals. These displays will be used as overlays on top of field maps.[127] In addition, Hycom is currently developing flat-panel displays for the "electronic desk" of the future, possibly even for the "electronic briefcase."[128]

Rockwell Electronics is nearing completion of a miniature thin-film electroluminescent device that could possibly provide full video capabilities. With a screen size of 1 x 1.4 inches, the device produces a high-quality video display of 512 x 683 lines, aptly demonstrating the high-resolution capabilities of electroluminescent technology. The display's total power consumption is less than five watts, making it highly suitable for such military applications as thermal weapon sights, head-mounted cockpit displays, mortar-locating radar displays, and terminal displays.[129]

CONCLUSIONS

The goal of this paper has been to pursue a thorough examination of flat-panel technology. To achieve this result, the paper has explored contrasting methods of creating display images on both CRT and flat-panel devices, examined in depth four major flat-panel displays currently being researched or developed as viable CRT alternatives, cited major advantages and disadvantages of CRTs and flat-panel displays, and summarized some current research in flat-panel technology.

Despite the distinctive characteristics of each of the flat panels explored within this paper and the different immediate and specific goals of these technologies, it appears they all share a common goal; namely, the achievement of a level of performance comparable or superior to that of the cathode-ray tube. While the present limitations of these technologies may hinder their achieving this goal within the next few years, it is clearly just a matter of time before flat-panel display technology will successfully displace the cathode-ray tube in both military and commercial markets.

FIGURE X

Estimated flat-panel display market*
(US shipments - $million)
1980-1990

Word- and data-processing application	Military	Consumer
1980	15	-
1982	80	-
1985	130	-
1990	320	130

* Displays with more than 200 characters

Figure X presents the International Resource Development's forecast for flat-panel display shipments, based on current technology and pricing trends. The table indicates that by 1985, flat panels will be offering CRTs a serious challenge in military applications and by 1990, they will have made great strides into the consumer market.[130]

Saul Kuchinsky, president of Quantum Systems, a consulting firm that specializes in flat-panel displays, says there are "literally thousands" of applications for flat-panel displays in the commercial market.[131] He comments, however, that the real problem lies not so much with the technology as with conservative management policies. Kuchinsky estimates that within five to 10 years, flat-panel display terminals will account for 25 percent

of the display market.[132] He predicts, "One company, probably a Japanese company, will have success with flat-panel terminals, and all of a sudden everyone will get involved. Cautious companies need proof of someone else's success."[133]

Although low-volume production and high costs currently suppress market demands for flat-panel terminals, innovative research, in both the commercial and private sectors, is expanding at a phenomenal rate, creating a bright future for flat-panel technology.

Footnotes

1. Catalano, Frank. "Exploring Flat-Panel Technology." Mini-Micro Systems, Dec. 1981, p. 126.
2. DeJackmo, Margaret. "Major Challenge to CRTs from Flat-Panel Displays." Mini-Micro Systems, Nov. 1980, p.113.
3. Ibid.
4. Tutorial: Computer Graphics. Proceedings of the Spring Compcon79 on Computer Graphics. 26 Feb.-1 Mar. 1979. Institute of Electrical and Electronics Engineers, Inc.: New York, N.Y., 1979, p. 23.
5. Newman, William M., and Robert F. Sproull. Principles of Interactive Computer Graphics. New York: McGraw-Hill, Inc., 1979, pp. 35-36.
6. Ibid., p. 217.
7. Meyer, Franklin. "Picture Brightens for Flat-Panel Displays." High Technology, March/April 1982, p. 36.
8. Ibid.
9. Anderson, L.K. "The Cathode Ray Tube Display: Why Use Anything Else?" Symposium on Display Materials and Devices, Murray Hill, New Jersey. 3 May 1973, p. 4.
10. Ibid.
11. Meyer, p. 36.
12. Ibid.
13. Ibid.
14. Ibid.
15. Kazan, b. "Material Aspects of Display Devices." Science, Vol. 208, 23 May 1980, p. 931.
16. Ibid.
17. Ibid.
18. Ibid.
19. Tutorial: Computer Graphics, p. 89.
20. Ibid.
21. Ibid.
22. Ibid.
23. Kazan, p. 932.
24. Ibid.
25. Ibid.
26. Ibid.
27. Catalano, p. 125.
28. Tutorial: Computer Graphics, p. 99.
29. Ibid.
30. Kazan, p. 932.
31. Ibid.
32. Ibid., p. 933.
33. Ibid., p. 928.
34. Ibid.
35. Ibid.
36. Ibid., p. 929.
37. Ibid.
38. Ibid.
39. Ibid.
40. Ibid.
41. Ibid.
42. Ibid., p. 930.
43. Meyer, p. 37.
44. Ibid.
45. Kazan, p. 930.
46. Ibid.
47. Ibid.
48. Ibid.
49. Ibid.
50. 1980 Biennial Display Research Conference. Proceedings of a Conference on the Current Research in Information Display. 21-23 October, 1980. Institute of Electrical and Electronics Engineers, Inc.: New York, N.Y., 1980, p. 10.
51. Meyer, p. 38.
52. 1980 Biennial Display Research Conference, p. 10.
53. Ibid.
54. Ibid.
55. Ibid.
56. Ibid.
57. Ibid.
58. Kazan, p. 933.
59. Ibid.
60. Ibid.
61. Ibid.
62. Ibid.
63. Ibid.
64. Ibid., p. 934.
65. Ibid.
66. Ibid.
67. Meyer, p. 38.
68. Kazan, p. 935.
69. Ibid.
70. Anderson, p.3.
71. Ibid.
72. Newman, pp. 33-50.
73. Anderson, p. 10.
74. Ibid.
75. Meyer, p. 36.
76. Ibid.
77. Ibid.
78. DeJackmo, p. 113.
79. Ibid.
80. Anderson, p. 20.
81. Ibid., p. 21.
82. Meyer, p. 33.
83. Anderson, p. 22.
84. Catalano, p. 126.

DOCID: 4011963

85. Meyer, p. 36.
86. Catalano, p. 125.
87. DeJackmo, p. 114.
88. Catalano, p. 130.
89. Ibid.
90. Ibid.
91. Ibid., p. 125
92. Ibid.
93. Ibid.
94. Ibid.
95. Ibid., p. 126.
96. Ibid.
97. Ibid.
98. Ibid.
99. Ibid.
100. Ibid., p. 128.
101. Ibid., p. 132.
102. 1980 Biennial Display Research Conference, p. 26.
103. Ibid.
104. Ibid.
105. Ibid., p. 154.
106. Ibid.
107. Ibid., p. 178.
108. Ibid.
109. 1978 Biennial Display Research Conference. Proceedings of a Conference on the Current Research in Information Display. 24-26 October, 1978. Institute of Electrical and Electronics Engineers, Inc.: New York, N.Y., 1978, p. 52.
110. Ibid
111. 1980 Biennial Display Research Conference, p. 143.
112. Ibid
113. Ibid
114. DeJackmo, p. 114.
115. Catalano, p. 125.
116. Ibid
117. Ibid
118. Ibid
119. Ibid
120. Ibid
121. DeJackmo, p. 114.
122. Meyer, p. 39.
123. Ibid.
124. DeJackmo, p. 114.
125. Ibid.
126. Ibid.
127. Meyer, p. 39.
128. DeJackmo, p. 114.
129. Meyer, p. 40.
130. Catalano, p, 130.
131. Ibid., p. 132.
132. Ibid.
133. Ibid.

GLOSSARY

AC: Alternating current: an electric current that reverses direction at regular intervals, having a magnitude that varies continuously in a sinusoidal manner.

ambient: completely surrounding; encompassing.

anode: the positively charged electrode, plate, strip, or terminal of a cell, battery, vacuum tube, etc.

birefringence: the separation of a ray of light into two unequally refracted, polarized rays, occurring in crystals in which the velocity of light rays is not the same in all directions.

cathode: the electrode that emits electrons or gives off negative ions and toward which positive ions move or collect; the negative pole of a battery or other source of electric current.

cathode-ray tube: a vacuum tube in which cathode rays, usually in the form of a slender beam, are projected onto a phosphor-coated screen to produce a luminous spot.

CMOS: capacitive metal oxide semiconductor.

collimated light: an optical system that transmits parallel rays of light.

CRT: cathode ray tube.

DARPA: Defense Advanced Research Projects Agency.

DC: direct current: an electric current of constant direction, having a magnitude that does not vary or varies only slightly.

dichroic: exhibiting of essentially different colors by certain solutions in different degrees of dilution or concentration.

dielectric: a nonconducting substance; insulator.

diode: a device, as a two-element electron tube or semiconductor, through which current can pass only in one direction.

electrode: a conductor used to establish electrical contact with a nonmetallic portion of a circuit (as in an electrolytic cell, an electron tube, etc.)

electrophoresis: the migration of charged particles in an electric field.

ERADCOM: Electronics Research and Development Command.

fc: foot-candle; the quantity of light reaching a surface or the illumination as measured in foot-candles (fc = $lumens/foot^2$).

fl: foot-lamberts; the quantity of light radiated by a surface per unit area in a given direction, or its luminance as measured in foot-lamberts. The foot-lambert is defined in such a way that a perfect uniformly diffuse reflector (white bond paper is a reasonable approximation of a lambertian surface) illuminated with 1 fc will have a brightness of 1 fl.

flicker: In relation to graphics, flicker may be thought of as an unsteady image resulting from a display device with an inadequate refresh rate (number of times per second a picture is redrawn). The main determinant of the refresh rate is the phosphor's persistence, the time from the removal of excitation to the moment when phosphorescence has decayed to 10% of the initial light output. The longer the persistence, the lower the required refresh rate to produce a flicker-free picture. A picture appears constant or flicker-free to the viewer even though in reality any given point is "off" much longer than it is "on."

frame buffer memory: a large digital memory used to drive raster displays. Modern frame buffer memories generally use random-access integrated memory circuits. Given 8 bits or more of intensity precision, the frame buffer is capable of producing color and monochrome images whose complexity and quality are limited only by the quality of the TV monitor on which they are displayed. The frame buffer offers the only satisfactory display device for applications requiring shading, solid areas of color, high-quality text, or any type of image processing. Disadvantages of the frame buffer include the large amount of memory it requires to represent the display image, thus making it expensive, and the time it takes to fill this memory or change it, thus slowing down interactive response time.

index of refraction: the ratio of the velocity of light or other radiation in the first of two media to its velocity in the second as it passes from one into the other, the first medium usually being taken to be a vacuum.

integrated circuit: an interconnected group of circuit elements, as of resistors and transistors, in a single tiny wafer of semiconductor material.

linearity: the faithfulness with which an output signal of an electronic reproducing system reproduces an input signal; more specifically, the faithfulness with which the shape and arrangement of the elements in a television picture reproduce the shape and arrangement of the original televised image.

multiplexing: In general terms, the word "multiplexing" refers to the use of a single facility to handle simultaneously several similar but separate operations. Most computers, for example, have high-speed multiplexing input/output channels to handle many peripheral devices such as line printers or card readers, all of which may operate simultaneously. The main use of multiplexing, however, is in the field of data communication, where it is used for the transmission of several lower-speed data streams over a single higher-speed line. The primary motivation behind multiplexing is the reduction of costs, although in many cases an increase in reliability is an additional benefit.

nonvolatile memory: a memory that retains its contents even if the power supply is removed.

orthogonality: a system of surfaces consisting of two families whose components are mutually perpendicular where they intersect.

pixel: a picture element.

precision: with respect to graphic display devices, the measure of the programmer's ability to address a point on the screen.

refresh: the repeated passing of an image to the display device.

resistor: a device, the primary purpose of which is to introduce resistance into an electric circuit.

resolution: with respect to graphic display devices, the measure of the user's ability to distinguish points on the screen from one another.

semiconductor: a substance whose electric conductivity at normal temperatures is intermediate between that of a metal and an insulator.

suspension: the state in which the particles of a substance are mixed with a fluid but are undissolved.

transistor: an electronic device made of semiconducting material and equipped with three or more electrodes. It performs functions similar to those of a vacuum tube without requiring current to heat a cathode.

Improving Raster Graphics Images
by Anti-Aliasing

or

How to Get Rid of
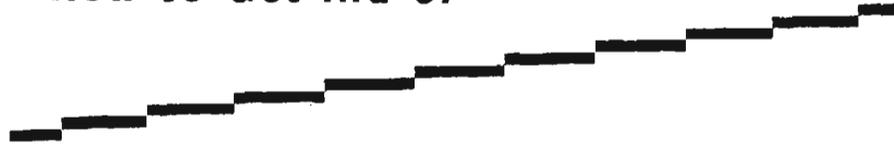
the Jaggies(U)

by [ ] R53

I. Introduction
A. Graphics Devices

There are two main classes of graphics devices: continuous and discrete. A continuous or random-scan device draws with a beam on the display surface much as a person draws with a pencil on paper, with smooth, continuous strokes. A discrete or raster-scan device divides the display surface into a matrix or raster of picture points (or pixels). The picture is created by lighting up the pixels that it would cover if laid on top of the display surfaces.

Continuous devices, such as the calligraphic and storage tube displays, are excellent for representing line drawings because of their smooth, precisely positioned lines. Calligraphic display devices have the advantages of moving images and selective erasure, since the images must be redrawn 30 times a second, but the complexity of the image is limited. Storage tubes can represent very complex images, but the whole image has to be erased and redrawn every time one part of it has to be erased. In either device, filled-in areas are difficult to represent and there is no broad range of colors available.

Discrete devices can represent any amount of information possible without affecting performance. Areas are filled easily. A raster device can provide a full range of colors or intensities, which makes it possible to represent realistic images. Selected areas can be erased without affecting the rest of the image. However, animation is not very practical, since a lot of computation would be needed for each image and the precision to which objects can be placed in the image is usually not as high as in a continuous device.

B. The Raster Display

Since this paper is concerned with the raster device in particular, a little more information about it is needed. The raster device is very much like a television set controlled by a computer. The image is drawn, line by line, 30 times a second. These lines are called scan lines and the order in which they are drawn is called an interlace pattern. The information about whether or not to turn a pixel on is located in a frame buffer. This frame buffer is read 30 times a second and the values in it are converted to intensity or color information for the beam. Each pixel may have a number of bits of information about it if its intensity can vary. On a color display, the red, green, and blue components of a pixel's color have to be specified separately since three beams are used. Many raster displays use what is known as a color lookup table, which contains a list of the components for a number of specified colors. The number in the frame buffer corresponding to a pixel points into the color lookup table and specifies the color with which to draw the pixel. In this way, a few bits of information per pixel can specify a color from a wider range than they would directly.

One way to draw a picture on a raster display device is to send commands to the device which tell it to choose a color, draw a

line, fill an area, etc. The device uses line-generating algorithms to figure out which pixels are affected and changes the corresponding locations in the frame buffer to the appropriate value. The next time the frame buffer is read, this new information is displayed on the screen.

There is another way to draw a picture that will be considered in this paper; it is called scan conversion. A geometrical representation of the scene (i.e., a list of lines, polygons, etc.) is examined by testing the image at each pixel point to see if there is anything there. If there is, its color is determined and the value is put into the frame buffer directly. This is called sampling the image.

Because of decreasing memory and processor costs and the proven inexpensive television technology used, raster displays are becoming more popular. Other attractions are the realistic images, color, shading, and the ability to display a substantial amount of information. But the discrete quality of the device and the sampling process, along with low resolution, lead to poor image quality due to the attempt to represent continuous scenes with discrete points. It is important to solve this problem so that complex images, free from any inaccuracies can be presented on a raster device.

This paper will discuss the kinds of image defects that are produced by the discrete quality of a raster device and briefly list the proposed solutions. It will be shown that one of these, area sampling, is the most effective and logical solution. The paper will then explain the theory involved in this solution and present some algorithms used to implement the solution. Finally, there will be a discussion of the advantages, disadvantages, and possible future use of this technique.

II. Manifestations of the Sampling Problem

The defects due to the discrete quality of a raster display occur in a few distinct places: along edges on the silhouette of an object, a crease on a surface, or along a colored patch on a surface; in very small or thin objects; and in areas of complicated detail.[3] There are a few basic types of defects caused that are very apparents to a viewer.

The most obvious problem is that of jagged edges (or "the jaggies"). Since a raster device can only display dots in given positions, lines made of these dots have to follow a regular grid defined by the raster scan. This stair-step effect is most noticeable on edges

that are nearly horizontal or nearly vertical (see Figure 1). Also, lines cannot be arbitrarily positioned but must start and end at dot positions.[4, 7]

Test pattern synthesized at a resolution of 256 samples by using techniques similar to those of conventional hidden-surface algorithms.
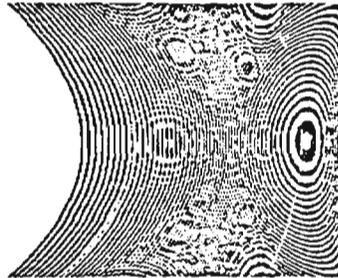

Figure 2: Moire Patterns (from reference [3])

Another general category of defects is the misrepresentation of details that are smaller than the size of a pixel. Small objects can "fall between the dots." This occurs because each dot on the image represents an infinitesimally small point in the scene being depicted. If the object is small enough, it is possible that none of it will fall on a sample point. A small object may disappear entirely; a long, thin object may appear in some places and not in others, looking like a barber pole; a very complicated object may lose some of its detail.[3]

Moiré patterns are produced in periodic images with closely spaced lines. The spaces in thin, broken lines and the jagged places in thicker lines are repeated from line to line with slight variations, causing patterns.[2, 3, 7] Figure 2 shows these patterns appearing in the raster image of closely spaced parabolic arcs.

In animated sequences of images, these problems become much more obvious and some additional problems appear. As slopes change, "armies of ants" appear to be running along the edges of objects as the edge detail changes. Small moving objects appear and disappear in successive frames. When they do appear, they are suddenly represented by a whole pixel, causing a distracting flashing, or scintillation. Slow-moving objects appear to change shape, due to the fact that the edges can only change positions in pixel increments and the edges change at different times. Since the object does not move smoothly, this is called crawling and it occurs only in animated sequences. A horizontal edge which looks fine in a still image will appear to jump from scan line to scan line as it moves vertically.[3, 7]

The next section of this paper will discuss proposed solutions to this problem, which are called anti-aliasing techniques for reasons that will be explained later.

III. Area Sampling as the Solution to Aliasing

A. Proposed Solutions

There are three basic techniques for improving the quality of raster display. The one that first comes to mind is to increase the resolution of the CRT. By increasing the number of scan lines and pixels per scan line, finer details can be represented and the "jaggies" are diminished. It can be shown that a "typical CRT" (17-inch diagonal screen screen viewed from 25 inches) would require 3,577 scan lines in order for the human eye to be unable to resolve the dots. Even at this resolution, some defects (such as scintillation) will still be visible unless the input is limited.[4] Today the highest resolution commonly available on a raster display is 1,024 lines, so it will probably take some time to develop a device with at least 3,577 lines. Since the computation time for most images is proportional to the square of the resolution, the cost of producing an image would be high and the picture generation time too long, especially in an interactive environment.[3,7].

The simplest and cheapest technique to get rid of the "jaggies" is to blur the image by actually defocusing the device or applying contour smoothing to the image after it has been computed. Although this does lessen the impact of the jagged edges, it does nothing to restore lost detail. As a matter of fact, the detail and sharpness that other techniques preserve is lost and the amount of information displayable is limited.[3,7]

A technique which is a combination of the first two is the wobbled raster. This is implemented by an addition to the deflection circuitry of the regular raster device. The wobbled raster actually doubles the horizontal and vertical resolution by quadrupling the number of dots per line and wobbling the beam one quarter of a scan-line width above and below the line twice as it scans across what was one pixel. Each scan line produces two consecutive lines simultaneously, offset by half a pixel width. Like the half-toning process used to produce continuous images in printing, the new pixels are aligned at angles, which tends to reduce the visibility of the basic dot structure. If all four new pixels are given the same color, the effect is the same as blurring the image.[9] While the idea of having all pixels the same distance

from their neighbors is a good idea, using the wobbled raster as a solution to jagged lines seems contrived and has no advantage over the other two techniques.

The third and best technique is to make each sample point represent a finite area in the scene rather than an infinitesimal point. For example, a very small object would occupy a fraction of the small area that would correspond to a pixel on the output image. The color of that pixel would be computed as the sum of the colors of the object and background, weighted by their respective areas. In the simple black-and-white case, the intensity would just be the fraction of area the object occupies in the pixel.

This approach corresponds closely to what actually takes place in television picture reproduction. The television camera receives light reflected from a finite area to form the color for every dot. While the other techniques mentioned offer somewhat ad hoc solutions to the problem, this technique is based on sound principles and makes sense intuitively.[3] The remainder of this paper will go into more detail about this technique and will refer to it, in general, as area sampling.

B. The Theory of Area Sampling

1. General Introduction

The difference between sampling an image at points and area sampling is qualitative. Area sampling will always produce the same intensities at any translation of the image. In point sampling, the sum of all the intensities will vary as an image is translated. For example, a small object may appear in one scene and not in the next, depending on how it falls on the pixels.[2] A properly adjusted digital raster display consists of regularly spaced dots (the pixels) which overlap by around one half. Ideally, the intensity displayed at each pixel should represent the intensity and size of whatever in the scene is covered by the area of the dot on the screen. Most points in the image would contribute to the intensities of three or four pixels.[4]

A model of the display which is easier to understand and manipulate is what will be referred to as the simple filter. Represent each pixel as a square centered on the pixel center with the side equal to the distance between the centers of two adjacent pixels. This divides the scene into a rectangular grid. The intensity of the pixel is proportional to that area of the pixel which is covered by an object. That is, the intensity

of the the pixel is the average visible intensity of the scene over the square area. If a color display is used, the pixel color is the sum of the colors of all the objects in the square weighted by their areas.[2]

Using this method, one can represent smooth and arbitrarily positioned lines and edges and small objects of any size and in any position. Varying intensity levels create the appearance of details lying between pixel positions (sub-pixel details). To see this, consider a small object the size of a pixel. This spot can be made to appear to be moving smoothly across the screen from pixel to pixel by dimming one pixel while brightening the one next to it. Adjacent pixels at half intensity will give the appearance that the object is lying between them.

Similarly, varying intensity levels allow nearly horizontal or nearly vertical lines to appear smooth instead of jagged. On a raster display, a line of pixel width which is more horizontal than vertical (i.e., the slope is less than 1) will be represented by one pixel per column at full intensity by a regular digital vector generator. Using the simple filter, one can see that the line will actually cross two or three pixels per column. If these pixels are intensified proportionally to the area the line occupies over each of them, the line appears smooth. Figure 3 illustrates the application of the simple filter to a line. While the lines represented this way are thicker and not as sharply defined as their jagged counterparts, their apparent positions are actually more accurate since they are derived implicitly from their environment rather than being set explicitly by the display device.[4]

2. Aliasing and Rastering

Having the value of each pixel represent a finite area instead of a point has the effect of applying a convolutional filter to the scene before it is sampled.[3] For the present, filtering can be thought of as blurring the scene before it is point-sampled.[5] By producing "blurred" images, we have shown that greater precision is possible. This is true only because the sharp images produced by displays without using varying intensities, or gray scale, contain inherent inaccuracies.[4]

In the field of signals processing, it is well known that a signal cannot be faithfully reproduced from digital samples if the sampling frequency is less than twice the highest frequency in the signal. In terms of reproducing a scene on the graphics device, the signal is the mathematical model of the scene to be displayed. A faithful reproduction of that scene on a graphics device is one which contains just as much detail and accuracy as the original scene, without any added information. For us to make a faithful reproduction of a scene, the scene cannot change more frequently than once per sample point. If this is not the case, an image with less detail than the original scene will be produced. This is called an alias of the scene, and aliasing refers to the defects produced. For example, highly periodic scenes, such as a picket fence, may appear as a few broad stripes instead of as many thin lines if these lines occur more than once every two sample points. Convolving a scene with a two-dimensional filter is done to ensure that the high frequencies in the scene do not exceed one-half the sampling rate. Techniques which attempt to reduce the effects of aliasing are methods of anti-aliasing.[3]

Aliasing is the consequence of improper filtering of a scene before it is sampled. Another term, rastering, is also used to refer to the defects described above, but actually it is a result of improper filtering of the image during reconstruction on the device, rather than at construction. Rastering appears as "ghosts" of the original image, since the pixels are not being displayed as intended. If the beam is not properly focused, the results are due to rastering as are the dark lines that may appear between scan lines. If aliasing can be thought of as getting less information than intended, rastering can be thought of as getting extraneous information that did not exist in the original scene.[3, 7]

3. Filtering

It has been said that aliasing is caused by neglecting to filter a scene before sampling it and that the cure, making each sample represent a finite area, has the effect of applying a filter to the scene. Now a more detailed definition of filtering is needed.

Filtering is an averaging process; the intensity of a pixel is determined by the scene within a small distance of the pixel center, not at a single point. This averaging is what eliminates the "high frequencies" that cause aliasing. Filtering is controlled by a filtering function, which supplies a weighting function for the averaging process. Convolution by a function just refers to this weighting process. The filter function describes the distribution of light emitted by a pixel on the display. Typically, a pixel is brightest at the center and decreases in intensity rapidly in all directions away from the center. Filter functions do not have to match

CID: 4011963

a. The infinite resolution line.

b. The image of the same
line on a raster display.

c. The simple filter applied
to the line. Note that the line
lies on up to 3 pixels per column.

d. The resulting filtered line
on a raster display. The intensity
of each pixel is proportional to the
part of it that is covered by the line.
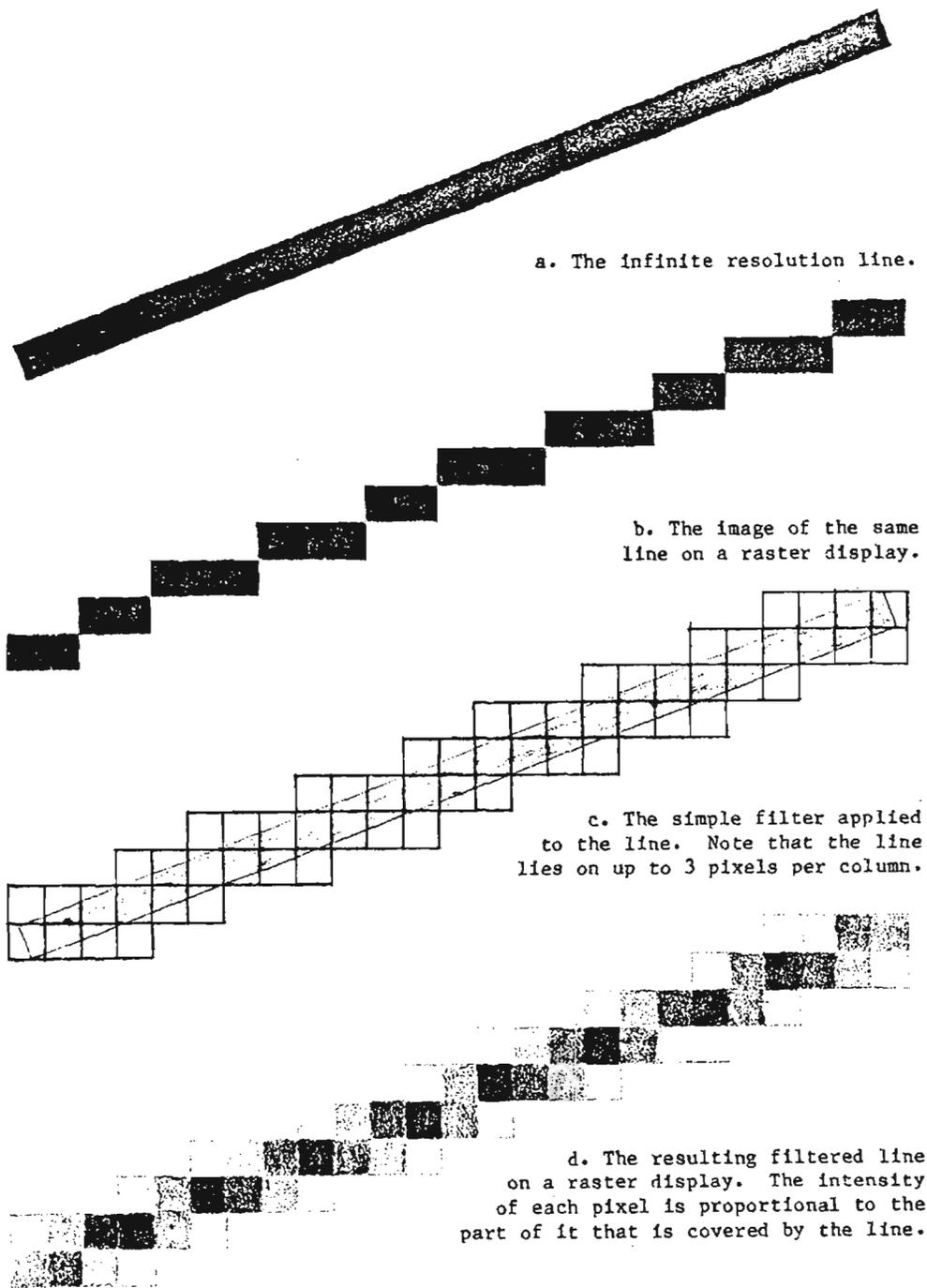
Figure 3: Applying the Simple Filter to a Line

the light emission properties of a display device exactly. Usually functions are chosen that are easy to manipulate, and their parameters are varied until the image looks good.[6]

While the difference between area sampling and point sampling is qualitative, the difference between the simple filter and better, more complex ones is quantitative. Most good filters will integrate the intensities in a small area around the pixel center; as resolution increases, all filters produce the same image.[2] The filter function is usually only non-zero in a small area around the pixel. The width of the filter corresponds to the diameter of the area to be sampled. Wider filters are lower-pass filters, which sample a larger area and tend to blur the image. Narrow filters sample a small area and tend to undersample. Point sampling can be thought of as an infinitesimally narrow filter.

The shape of a filter determines the relative weights assigned to various points in the small area. Flat filters, such as the simple filter described above, weight all points in the area equally and are lower-pass filters. Most good filters are circularly symmetrical, weighted more heavily at the center and falling to zero at the perimeter. These filters approximate the pixel's intensity distribution. Filters that are weighted very heavily in the center and fall off rapidly will cancel the blurring effect of a wide filter. As long as filters have the same width and the same general shape, the results are not noticeably different.[10]

The simple filter is equivalent to convolving the scene with a "box" function; it is a flat filter with a width equal to the distance between pixels but is not circularly symmetrical.[2] The most popular filters are the simple filter (used for its simple representatin of pixels) and a roughly conical filter about one-and-a-half to two times as wide as the distance between pixels (used to more accurately approximate the pixel's properties and to take advantage of circular symmetry).

4. An Added Benefit: Increased Resolution

There is an interesting result of the anti-aliasing technnique of using gray scale levels to represent a scene. Not only does it achieve its goal of eliminating the effects of aliasing, it also results in a noticeable gain in resolution. This also occurs in the half-toning process, which is just the opposite of the anti-aliasing method described above. While anti-aliasing converts size into various intensities, half-toning converts intensities into different sizes of dots. In newspapers these dots are all printed in black, but the

image produced appears to be continuous and to possess gray scale. This is because the dots are small enough to be point sources.

When looking at a point source, the human eye does not distinguish between the dot's size and its intensity, even though one could resolve the dots by looking more closely. This means that the resolution at which dots become point sources is less (i.e., fewer dots per inch) than the resolving power of the eye. For good-quality half-toning, that resolution is between 85 and 133 dots per inch. A picture with 100 dots per inch held at distance of one foot corresponds to 480 scan lines on our "typical CRT." Experiments have shown that for this screen size at average brightness, 440 lines is the minimum resolution for which size and intensity of dots are interchangeable. This figure agrees well with one derived from the half-toning analogy.

Since in a point, source intensity and size are equivalent, subpixel details can be represented using varying intensities. Therefore, the "effective" resolution of a display using anti-aliasing will be greater than the pixel resolution. For example, consider a large object with its left edge overlapping a column of pixels by one half. The pixels in that column will be calculated at one-half full intensity, while the ones to the left of the column will be at zero intensity and those to the right at full intensity. Studies have shown that the brain reacts to this image the same as it does to an infinite-resolution picture of the same edge. Therefore subpixel positioning of the edge can be represented by using varying intensities. Furthermore, changing the intensity of the intermediate column of pixels is interpreted as a corresponding movement of the edge.

The effective resolution of a display is determined not only by the number of pixels but also by the number of the differentiable intensity levels the device is capable of producing. Factors such as room lighting, phosphor characteristics, spot size, and screen reflectivity all determine the eye's ability to distinguish one intensity from another. The device itself may be limited in the number and range of intensities it can display. The ratio of the brightest to the dimmest intensities displayable on our "typical CRT" is about 25 to 1. Using this figure and the fact that on the same CRT the eye can distinguish a 4% difference in the intensities for single spots, the number of levels needed is 83. These levels are on an exponential scale, each intensity a 4% increase over the previous one.[7] When looking at large areas of varying intensity, though, the eye is able to distinguish a 2% difference, which implies 162 levels, also on an exponential scale. But in

order to properly represent spot positions, for example, two spots at half intensity should have the same brightness as one at full intensity. In this case a linear scale, in which each intensity differs from the previous one by a fixed amount, would be more useful. Since this concern is more important than distinguishable levels, the linear scale is the most common kind.[4] The 83 exponential levels would translate into 628 linear levels. Studies have shown, however, that 256 levels are sufficient with only a little roughness in variation noticeable in the low intensities.[7]

One way to look at these 256 levels is as follows: Divide each pixel into a 16-by-16 grid, forming 256 "subpixels." Then, using the 256 intensities to represent lighting from none to all of these subpixels results in a factor of 16 increase in the "effective" resolution of the device. A 512-line CRT would have an effective resolution of 8,192 lines, but since only around 3,600 can be resolved, less than half of that would be "usable." Of course this is not real resolution, but it is perceptually equivalent. Objects as small as one-256th the size of a pixel can be represented, and objects can be positioned to one-sixteenth pixel precision. It should be noted that the 256 intensity levels require eight bits of memory per pixel, and for a full-color display one would need three times this much.[7] Some successful anti-aliasing schemes use only 16 intensity levels, but even with this amount, 12 bits are needed per pixel. All combinations of the colors available are needed to accurately render any pixel, so it appears that using anti-aliasing on a device which uses a color lookup table would be difficult. Most of the colors needed for the table would be implicitly determined by a few explicitly chosen colors. (See Section IV.C).

We have shown that the anti-aliasing technique of area sampling an image eliminates the effects of aliasing and also allows, to some extent, the representation of detail finer than the resolution of the display and a corresponding increase imprecision. The next section of this paper will show how this theory is applied to a general scene and then how problems in specific areas can be eliminated with simpler, but perhaps not as accurate, techniques.

IV. Applications of Anti-Aliasing

A. The General Case

The general case that anti-aliasing will be applied to is that of a fully described, or modeled, three-dimensional scene of solid objects. Each object is modeled as a polyhedron, each face of which is a polygon. By increasing the number of faces, a polyhedron can be constructed to approximate any solid object. Since a polyhedron is described by a relatively small number of vertices, it is easy to represent and manipulate mathematically. In translation, rotation and scaling, the primary parts of a polyhedron (points, lines, and planes) all retain their properties. Any scene to be depicted is thus a list of these geometric objects with information about their positions and colors.[8]

Techniques for achieving realism in three-dimensional raster graphics are mostly concerned with how to restore the information about the third dimension to the images. These include projections, shading, and hidden surface removal (removal of hidden parts from images of solid objects). In recent years, these techniques, especially hidden surface removal, have improved dramatically. Recently developed hardware processors for hidden surface removal can create images at 30 frames per second, fast enough for real-time applications. . Unfortunately, anti-aliasing techniques have not progressed as rapidly. Most techniques are "ad hoc" ones to remove the most obvious effects of aliasing, such as
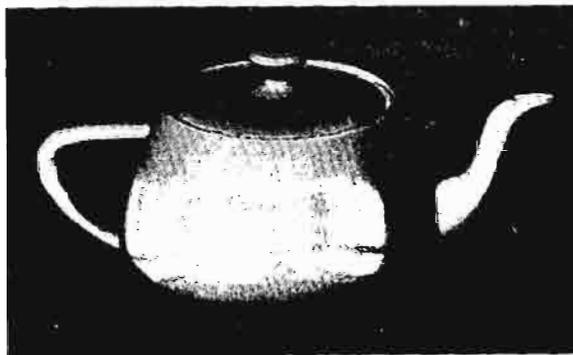
A computer-generated image of a teapot with simulated specular reflections of light entering through a window. *Courtesy University of Utah.*

Figure 4: Shaded Image with Jagged Edges (from reference [8])

jagged lines.[2, 8] How realistic is an image of a shaded object that has a smoothly curved and shaded surface, if the edges appear jagged? For an example of this contrast, see Figure 4 (which originally appeared in reference [8]).

1. Hidden Surface Removal

To properly solve the aliasing problem in the general case, one must find solutions to both the hidden-surface problem and the filtering problem. That is, for each pixel one must implement a hidden-surface algorithm to find out what is visible in that pixel and

then use the filter function to determine the resulting intensity. A "simple-minded" anti-aliasing algorithm would not properly take into account what is visible. For example, if it just summed the intensities of all the objects falling on a pixel, a completely hidden color might contribute significantly to a pixel. Errors like these are quite visible, even though they may occur in an area one-millionth of the screen area.[2]

When implementing hidden-surface algorithms with anti-aliasing, it saves time to find the places where aliasing is likely to occur, such as polygon edges, silhouettes, and creases and to restrict the time-consuming filtering process to these places. When approximating a curved surface with a polyhedron, the shading techniques used to give the polyhedron the appearance of a smooth surface make anti-aliasing unnecessary at the polygon boundaries on these surfaces.[3]

Most hidden-surface algorithms are applied after a scene is transformed into a two-dimensional image. This image is a list of overlapping polygons with a certain depth or priority associated with them.[8] The algorithms are separated into two basic types: depth algorithms and scanning algorithms. Depth algorithms process each polygon separately and determine the color of a pixel by the color of the closest polygon falling on that pixel. Scanning algorithms generate the image scan line by scan line by keeping a list of all polygons falling on that line. Scanning algorithms are the most usable ones with anti-aliasing, since all the information needed is available; i.e., all visible surfaces in the neighborhood of a sample point. The depth algorithms do not recognize or keep track of any relationships between polygons. Using this kind of algorithm would require pointers to neighboring polygons and keeping track of the edges and their amount of contribution to a sample point.[3]

2. Algorithms

This section gives two different examples of algorithms for hidden-surface removal with anti-aliasing. In both cases, the input to the algorithm is a list of two-dimensional polygons with their associated priorities and colors.

### a. Example 1: Scanning Hidden-Surface Algorithm

The first algorithm is described by Catmull.[2] It uses the simple filter and a scanning hidden-surface algorithm. Basically, it performs a hidden-surface algorithm at every pixel and then integrates the intensities. Everything needed for anti-aliasing is

provided to much more precision than is available to the display. This precision is only limited by the computer running the algorithm. Finding which pieces of polygons are visible is like the original hidden-surface algorithm, except that there are two simplifications. First, we are interested only in the sum of the intensities of each piece weighted by its area, and not in the exact coordinates of the vertices. Second, much of the work in sorting the polygons has already been done in the higher-level hidden-surface algorithm.

The basic algorithm follows, and a simple illustration is given in Figure 5. A more detailed description can be found in reference [2]. There is an active polygon list which is a list of all polygons in the current scan line. Polygons are added to and deleted from this list as necessary as each scan line is processed. For each scan line, set the pixels to background color. Each pixel has a "bucket," which is a list of all polygon pieces which fall on it. Clip each polygon in the active polygon list to the part which falls on the scan line. What is left is a list of very narrow polygons. For efficiency, clip these polygons into three sections: a piece in the center, which is all solid pixels, and the two irregularly shaped pieces on either side of it. Sort these pieces into the buckets by the X coordinate of the leftmost pixel in the piece. For every pixel in the scan line, sort the pieces in the bucket by priority, putting a solid piece of background color last. If the first piece is a solid piece, put its color into the pixel. If it is an irregular piece in front of a solid piece, find its area and use it to weight the two colors. In any other case, the first two pieces are irregular and a special hidden-surface algorithm, called the pixel integrator, is used.

The pixel integrator is like the polygon subdivision algorithm, which will be discussed in the next example. This algorithm is entered with a list of polygons in sorted order. An edge of the first polygon is selected as a dividing edge, and every polygon in the list is clipped against that edge. Two lists are formed, one for the polygon pieces lying to one side of the edge and another for the other side. If the algorithm is recursively applied to both of the resulting lists, then very shortly the first polygon in each list will cover all the ones behind it since everything else has been clipped away and is in another list. The area of this polygon can be found, the color of the polygon is weighted by it, and the result is returned. The sum of these weighted intensities from all the lists (one for each visible polygon) gives the final average intensity for the pixel.
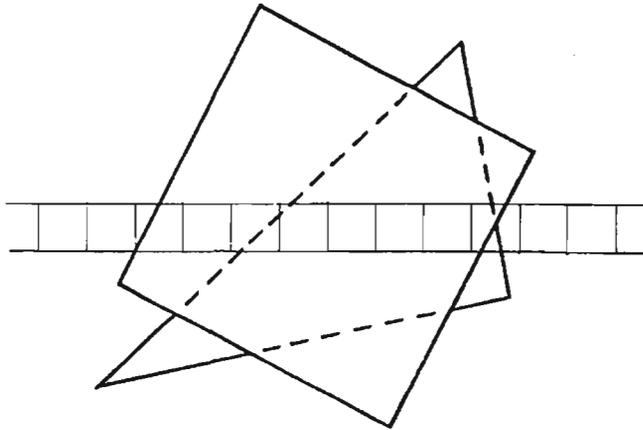
The scene is a red square on top of a green triangle.
The current scan line is superimposed over the scene.

Red Polygon
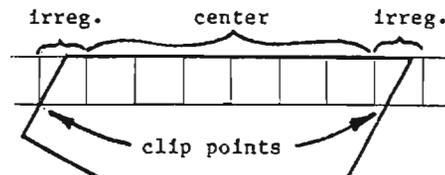
Green Polygon
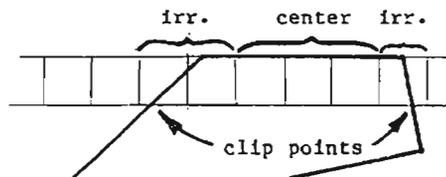
Clipping the polygons to the scan line
and dividing them into irregular and center pieces

For the first pixel, an irregular piece is in front of a solid pixel, so the
    pixel's area is used to weight red with the background color.
For the middle pixels, the first piece is a solid piece, so the pixels' colors
    are all red.
For the last pixel, the first two pixels are irregular pieces, so call the
    pixel integrator to return the pixel's color, which will be a weighted sum
    of red, green, and the background color.

Figure 5: Scanning Hidden-Surface Algorithm with Anti-Aliasing

b. Example 2: Polygon Area Sorting Algorithm

The second algorithm is described by Feibush, Levoy, and Cook.[5] It uses a newly developed hidden-surface algorithm called the polygon area sorting algorithm, which is based on the polygon subdivision algorithm. The polygon subdivision algorithm clips all polygons against the ones in front of them and discards the covered parts, so that the final output of the algorithm is a list of polygons which do not overlap, somewhat like a jigsaw puzzle representation of the scene.[8] The pixel integrator described above is similar to this, but is just concerned with the polygons' areas and not with their vertices. Figure 6 illustrates an example of the algorithm in the particular case of the pixel integrator.

The basic hidden-surface algorithm is as follows: Polygons are a list of vertices, with the edges between them marked as "clipped" or "unclipped." Find the first unclipped edge in the closest polygon in the list. If there is only one polygon in the list, or if there is no unclipped edge, return the polygon as is. Otherwise, clip all polygons in the list against that edge and put them into two lists, one for parts on each side of the edge. Set the clip flag for the clipped edge to "clipped." Reenter the algorithm for each of these two lists, combine the two resultant lists of polygons, and return this list. The final result will be a list of all visible parts of polygons.[2]

After this list is constructed, it is a relatively straightforward, but involved, matter to do the anti-aliasing. This algorithm uses a conical filter and keeps a lookup table to store the volumes above selected right triangles in the sample area, one vertex of which is the pixel center. A complicated computation finally results in the volume above any of the polygon pieces in a sample area, which is the weight for the color. The results from each polygon in a sample area are added together to get the final result. An advantage of using lookup tables is that the filter function can be changed easily by just changing the values in the table.

The same paper gives a method for anti-aliased texturing of the interiors of polygons. Texture is defined as a rectangular array of points with varying intensities. These points give the appearance of roughness or patterning to a polygon. For any visible polygon to be textured, all pixels that it falls on are mapped onto the texture definition. All texture points in the pixels are translated back to the image, and the value of each pixel is determined by weighting the points with the same filter used in the rest of the algorithm.

## B. Algorithms for Specific Cases

The algorithms above are used to solve the aliasing problem in the general case, which is any scene (no matter how complicated) that has been completely described before the algorithm is applied. These algorithms just send the image to the device as a list of colors, one for each pixel. This is very useful for real-time applications, such as flight simulation, where the entire scene is known beforehand. But this is not always the case. The graphics device is commonly used as an interactive device, with the user adding, changing, and deleting objects to create the final image. He is really drawing on the screen, and each new piece being drawn is considered to be independent from what is already on the screen.

In these cases, there probably is neither the need nor the money available for a system that keeps track of everything drawn and which essentially recomputes the image every time the screeen is redrawn. (This technique is called real-time conversion and is similar to what happens in calligraphic devices.) What is needed is a set of anti-aliasing algorithms that are specific to the elementary pieces that make up a scene, such as lines, polygons, and text. While these algorithms will be simpler and faster than the general-case ones, they will also be less accurate, especially when two or more pieces interact, since the relationships between pieces is not known. They have been described as "ad hoc" techniques, but sometimes they are the most feasible ones and their results are acceptable for most real-life applications. These applications are probably not concerned with shading, texturing, shadows, and reflections. They deal with simple objects and need fast responses.

This section will describe a few algorithms that are designed specifically to treat aliasing problems in these elementary pieces. The effectiveness of these algorithms can be seen in the illustrations in the articles mentioned. A few of these illustrations have been included in this paper, but because of the various reproduction processes used, their quality is not representative of the results seen directly on the displays.

1. Lines, Curves, and Polygons

Lines, curves, and polygon edges will all be treated as the same case. In most graphics applications, curves are approximated by a set of points connected by short line segments. Polygon edges will be seen as an extension of the case for lines. A faster algorithm specifically for drawing anti-aliased lines

red   green   background
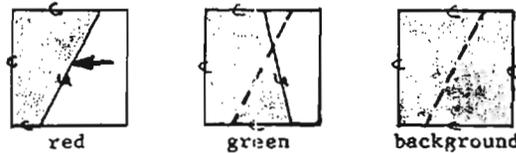
The list of polygons, the topmost on the left, which is the input to the first level of the algorithm. The edges are marked "u" or "c" for unclipped or clipped. The arrow points to the chosen unclipped edge.

red   green   back.

Polygons lying to the left of the clipping edge. Input to a second level of the algorithm.

green   background

Polygons lying to the right of the clipping edge. Input to a second level of the algorithm.

red
or
1/2 red

There are no unclipped edges in the first polygon, so it or its color value is returned to the main level of the algorithm.

green   back.   background
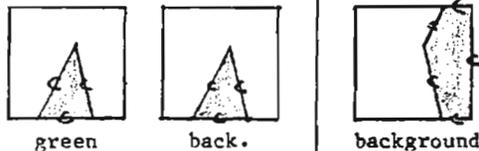
Polygons lying to the left of the edge. Input to third level.

Same for the right side.

green
or
1/6 green

background
or
1/3 background
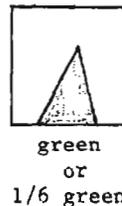
There are no unclipped edges in either list, so these polygons or their color values are returned to the second level, which combines them and returns the result to the main level of the algorithm.

The main level of the algorithm combines the results from the two second level algorithms and returns the final list of non-overlapping polygons, or a color value which is the sum of the colors of the polygons weighted by area.

Figure 6: Pixel Integrator (or Polygon Subdivision Algorithm)
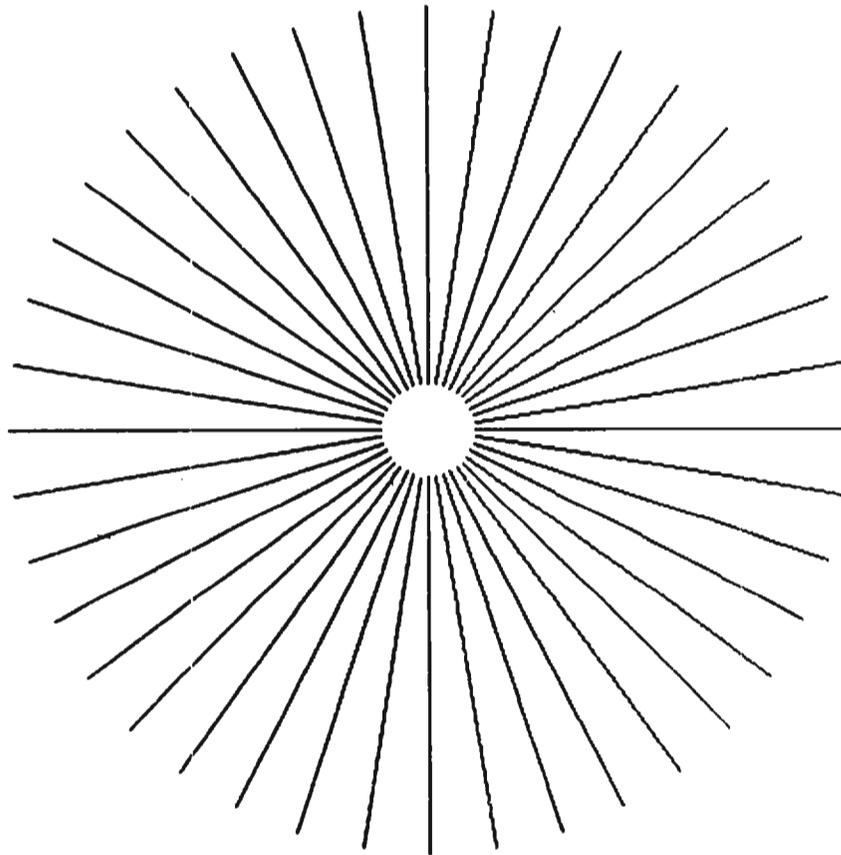
The lines on the right are normal.
The lines on the left are anti-aliased.

Figure 7: Comparison of Jagged and Anti-Aliased Lines
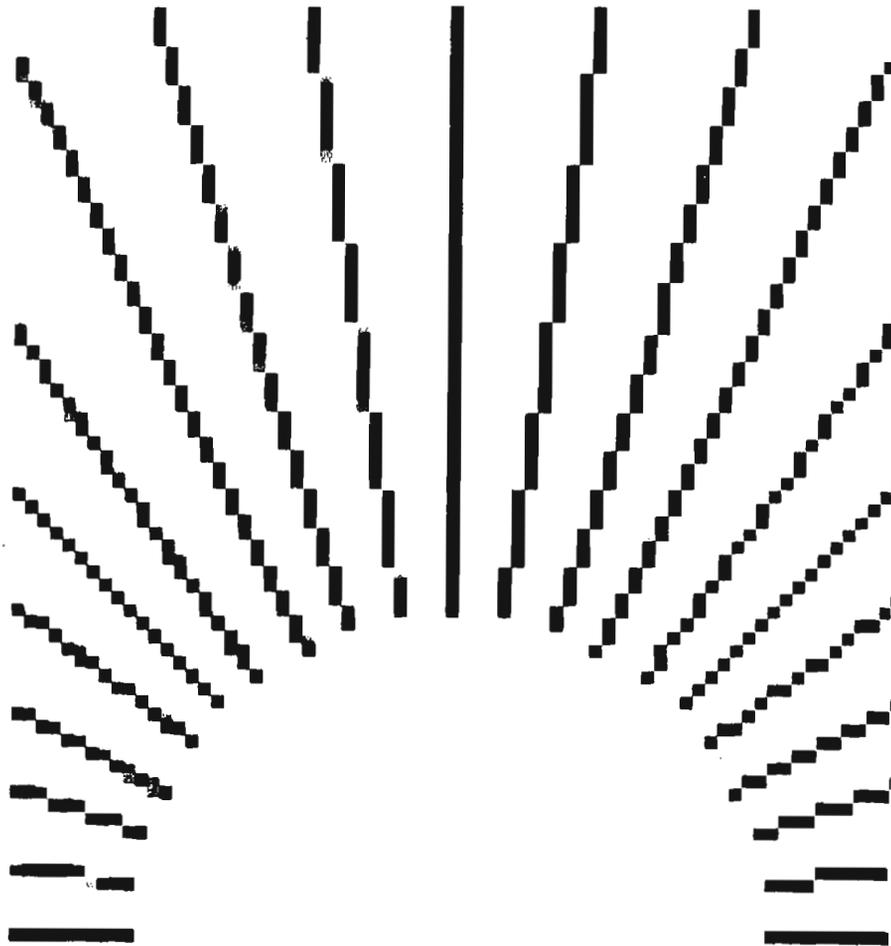
Figure 8: Enlargement of Figure 7 to Show Pixel Intensities

takes advantage of the "spatial coherence" of lines. That is, given a small part of a line, the remainder of it can be easily extrapolated. If the representation of a line on one scan line is known, changing it incrementally will provide its representation on the next scan line.[4]

a. Example 1: Bresenham's Algorithm with Anti-Aliasing

The following algorithms for drawing anti-aliased lines can be used to draw them directly onto the screen (or actually into the frame buffer) and is an extension of Bresenham's algorithm, a commonly used technique for drawing lines on raster devices. It is described in full detail by Gupta and Sproull.[6] This algorithm uses table lookup to reduce the computation of intensity levels, and variations can be used to draw lines of varying thicknesses and to smooth the edges of polygons. Figure 7 compares the lines drawn with an implementation of this algorithm to jagged lines and Figure 8 is an enlargement of the same image showing the different intensities.

The basic idea of the algorithm follows: The filter used is a conical function which has its maximum value at the center of the pixel and decreases linearly to zero at a distance of one, in units of pixel-to-pixel distance, from the pixel center. The function is such that the volume of the cone is also 1. When a line passes through a pixel, the pixel's intensity should be proprotional to the volume of the cone intersected by the line. Because the cone is circularly symmetrical, this depends only on the line's width and the perpendicular distance from the pixel to the line. Lookup tables, which have these volumes listed for certain distances from point to line, can be constructed for any line width. The precision of these tables depends on how many intensity levels are desired.

The discussion of the algorithm is restricted to lines of unit thickness in the first octant (i.e., the slope is positive but no greater than 1). These lines intersect two or three pixels in each column of pixels. The algorithm keeps track of the location of the pixel that the center of the line passes through, the center pixel, and the perpendicular distance from the center of that pixel to the line. At each column the center pixel and the pixels above and below it are shaded according to their perpendicular distances from the line, which are easily calculated from the that of the center pixel. These distances point to intensity values in a lookup table. Then the center pixel for the next column is found and the new perpendicular distance is calculated.

Endpoints have to be treated differently. A separate lookup table has intensities for the six pixels that are affected by an endpoint, the three in the endpoint's column and the three adjacent to them. Since these values vary with the line's slope, the slope is the index into this table. Extending this algorithm to lines of all slopes is just a matter of switching the roles of the X and Y coordinates for slopes greater than 1, and of replacing the Y coordinate with its negative for negative slopes.

The variations on this algorithm make it a very useful one. Lines of different thicknesses can be produced by calculating separate lookup tables for various thicknesses and choosing the appropriate one. The algorithm may have to be modified so that it intensifies more than three pixels per column. Similarly, the endpoint table for each thickness will have to have more than six pixels per endpoint. Different endpoint shapes, such as rounded or beveled instead of squared-off, can be accommodated with different endpoint tables. Polygon edges can be produced with a table which contains intensities based on how much of a pixel is covered by an edge.

Different filters, needed for different output devices, can be accommodated by lookup tables. As in the case of thicker lines, the algorithm may have to intensify more than three pixels per column. If these filters are not circularly symmetrical, though, a second parameter, slope, is needed to select the correct lookup table.

Any background shade and any line shade can be used by mixing the intensities. The same applies to colors; the red, green, and blue components are mixed independently.

Endpoints need not be at pixel centers. Precise endpoints are needed to avoid problems in repetitive patterns where the endpoints should appear to be aligned.[4] They are also needed to allow smooth motion in moving lines. Subpixel precision endpoints are a problem because their accurate rendering requires either many lookup tables or a lot of computation.

b. Example 2: Filtering Tiler

Crow [3] describes a tiler with anti-aliasing. A tiler is a procedure which generates the individual pixels which form a solid polygon from a list of the polygon's vertices. This tiler has no hidden-surface removal; it just draws a convex polygon with anti-aliased edges. This kind of tiler is needed because if the polygon's border were drawn first as anti-aliased lines, a hardware fill of the area would not work. Hardly any

of the pixels making up the border would be the same color as the polygon interior so the tiler wouldn't know when to stop, unless it were "smart" enough to recognize different intensities of the same color.

A filtering tiler differs from a conventional one in that it must keep track of the edges that fall on a given scan line, and edges that are very short or nearly vertical cannot be ignored. The algorithm is straightforward and uses the simple filter. Lists are kept, one of the edges on the left side of the polygon that intersect the current scan line and another for those on the right side. Starting at the top vertex, the polygon is created, scan line by scan line, until the bottom vertex is reached. At each scan line the edge lists are updated. Starting at the left side, the intensities of the pixels that hold irregular pieces are calculated by going through the list of the left edges and adding the weighted area on the right of that edge to the corresponding pixels it intersects. It does the same for the right side, but subtracts the weighted area to the right, to correctly render objects thinner than a pixel. It then fills in the middle of the scan line. Crow notes that this algorithm takes from two to five times longer than an ordinary tiler, depending on the number of edges.

c. Intersections

In a typical drawing, lines and polygons are not isolated. They meet, intersect, and overlap. When two objects affect the same pixel, a rule is needed to determine the ruling intensity. The simplest method is to just overwrite the old intensity. However, this will cause gaps in earlier objects where they are overwritten by dimmer pixels from later objects. Ideally, the intensity should be based on the areas occupied by the two objects. But this is not practical, since it would require information about the relationship between the objects, such as whether or not they overlap. A compromise is to sum the intensities, making sure that the sum does not exceed the maximum possible intensity. This can lead to the problems of colors showing through solid objects and of two dim lines intersecting in a bright spot, which happens on a calligraphic display. If the frame buffer does not have have a readback capability, these methods are not even possible. That is, if the previous intensity of the pixel cannot be found out, no sum or comparison can be made.[4]

2. Characters

Another important part of graphics image is text. Dot matrix characters look fine rotated in increments of 90°, and scaled or translated at pixel increments, but otherwise the results are terrible.[4] Other techniques for representing characters, such as high-resolution bitmaps (similar to dot matrices), or as curved outlines which are treated as filled polygons, or as a set of "strokes" (lines), work fine on high-resolution devices. But at low resolutions the characters are bad representations because they are undersampled. Usually the low-resolution matrices have to be constructed by hand.[10]

Warnock has addressed this problem.[10] The motivation for developing his technique was a need to represent high-quality text for a graphics application used to design page layouts. A technical article may have as many as 30 different fonts. The page layout must accurately represent the styles and shapes of the fonts that will appear on the final copy. The spacing, layout, and appearance are all very important. The requirements for character sets for use with a low-resolution raster display are:

[]   they must be faithful to the masters, even at low resolution;
[]   they must be free of aliasing, that is, no holes, dark spots, or blurring;
[]   they should look properly spaced regardless of the display's resolution.

The method is to make a high-resolution black-and-white representation, or bitmap, of the character and to sample areas of it for each pixel. The area sampled depends on the size of the character to be drawn and the width of the filter. The bits that are turned on in the sampling area are weighted by a filter matrix, which has values in it corresponding to the values of the filter function at these points. The sum of these is the intensity for that pixel. Like the line-drawing algorithm, the filter can be changed by changing the matrix values. When two characters affect the same pixel, the intensities are added, since the characters are known to overlap.

Using only 16 intensity levels and filter similar to the one in the line-drawing algorithm, the results are very impressive. Text of only five pixels high was readable by content, six pitch was sufficient to recognize letters, and at seven pixels the text was perfectly readable. The characters produced are more faithful to their masters than directly sampled ones, there are many fonts available, and the characters can be rotated and placed to subpixel precision. Very small fonts can be used for thumbnail layouts to get the feel of the page, even though the text may not be readable. Figure 9 contains some of the illustrations from the paper, showing a comparison of non-filtered and filtered

characters, rotated characters in various fonts, and an enlargement of a character.

C. Hardware with Anti-Aliasing Features

Several algorithms have been devised to eliminate the effects of aliasing in specific areas. The ones mentioned above are just a few examples of them. They have been written to coincide with the hardware implementations of their ordinary counterparts. This section will look at a commercially produced display device, the AED767, which has some anti-aliasing features. The only information available about this device is a product review from August, 1981 [1] so this discussion is not very complete or up-to-date, but it will give something of an idea of what the situation is currently. The review says that this device is the first raster graphics device produced that has "anti-aliased vector generation in the terminal hardware/firmware."

The AED is an enlargement of the AED512, an earlier color raster device. It has a resolution of 575 lines, with 768 pixels per line. There are a maximum of eight bits of information per pixel, which point into a color lookup table which holds 256 entries. The anti-aliasing feature can be turned on or off. When it is turned on, vectors can be drawn in 16 programmable base colors, each of which has 16 intensities. The algorithm used is probabbly much like the one described in this paper. The 16 intensities correspond to a quadrupling in effective resolution to 2,300 lines, which may be enough for effective anti-aliasing, since the screen size is smaller than our "typical CRTs."

The review states that "intersecting vectors [are] accurately rendered by a proprietary technique." The device has a readback capability from the color lookup table, which implies that the proprietary technique probably consists of choosing the maximum of the two values. In order to represent every possible mixing of any two base colors, many more colors than the 256 available (29,056 to be exact) would be required. It is doubtful that they are using a technique which somehow represents each pixel by a list of color lookup table entries which it sums upon scanning to compute the beam intensities.

The review does not mention filled polygon edges, so it is assumed that they will still have jagged edges. (See Section IV.B.1.b above about the problems with hardware fills to anti-aliased edges.) The contrast between jagged polygon edges and smooth lines may be distracting. The limitation the color lookup table poses on the accurate rendering of intersections is unfortunate. The lines can

still only strart and end at pixel precision. All of these facts lead one to question the benefit derived from using a device that only does the job half-way.

V. The Costs and Future Use of Anti-Aliasing

It has been shown that using area sampling to generate gray-scale images is an effective technique for eliminating the effects of aliasing on raster display devices. It also has the added benefit of increasing the effective resolution of the display device. To balance these advantages, there are some problems with the technique which will be described below, along with the solutions to them.

One problem is that a loss of acuity is apparent. A field of small objects appears as a solid gray mass instead of individual objects. Actually, in this case the eye would not be able to resolve the objects either, and that is how it would appear to the eye also.[7] The only problem would arise if the viewer got closer to the screen and expected to see more detail.

Another problem is non-linearity. Two pixels at half intensity should have the same total brightness as one pixel at full intensity. Non-linearity in the phosphor, digital-to-analaog converters, or any other transformation the calculated intensities go through before they are displayed may contribute to the problem, which makes lines look "barber-poled." Techniques which involve getting new intensities from compensation tables have been developed to compensate for the non-linearities.[7, 10]

Rastering is still a problem. Unless the display is properly adjusted, the dark lines between scan lines will be easily visible. Hardware techniques, such as the wobbled raster or more complicated interlacing schemes are used to reduce the line structure. In color CRTs the effect is not as bad since each pixel is represented by three spots instead of one.[7]

The most serious problem with anti-aliasing techniques is that they are time-consuming. The hidden-surface algorithm with anti-aliasing runs three times slower than a regular hidden-surface algorithm.[2] Even in the case of the line-drawing algorithm, three times as many pixels per column are intensified, and the computation for each pixel is more complicated. There is also the higher cost of the hardware needed to implement varying intensities and the cost of the memory to store several bits of information per pixel.[4]

CID: 4011963

An example of TIMESROMAN
1234567890 @#$%&()< >
An example of TIMESROMAN
1234567890 @#$%&()< >

Comparison of directly sampled black and white text and
filtered gray scale text. The font is 8 pixels high.

Examples of rotated text in various fonts, 8 pixels high.
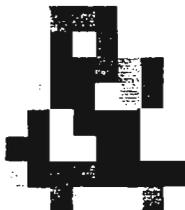
An enlarged "&" showing grayscale values, and a row of
the same character as it appears on the screen.

Figure 9: Anti-Aliased Characters (from reference [10])

One way to reduce the time factor is to put the anti-aliasing capability under user control. A rough sketch of the image can be constructed without using anti-aliasing, since the preliminary sketching is probably the longest part of the job. Once the final image is constructed, the anti-aliasing can be turned on and a high-quality image can be produced. The algorithms which use lookup tables avoid the complex filtering computations and reduce time considerably. Also, the techniques for specific cases are less time-consuming because information about the structure of the object is used.

As more display terminals contain their own micro-processors instead of hardwired logic, it seems reasonable that the complexity of the operations performed in the terminal itself will increase. An implementation of these specific algorithms in a processor would not be very expensive. With the advances being made in the speed and computing power of these processors, there is every reason to think that anti-aliasing will soon be a "universally available" feature in raster display terminals.[4]

The VLSI (Very Large Scale Integration) technology has already been used in a computer graphics design and the result is an increase in speed of image generation by a factor of 1,000. This is because many operations are performed at the same time, instead of sequentially. Currently raster graphics systems which produce full-color shaded images in real time have a smaller capacity for manipulating images with many edges than a calligraphic system and cost much more. A system has already been devised, using the VLSI technology, for producing real-time anti-aliased movement in two dimensions. Soon it may be possible to produce completely anti-aliased real-time raster images with as much detail as a calligraphic system, with the added benefits of color and shading.[11]

VI. Conclusions

In view of the technology that is commonly available today, the main factors governing the decision of whether or not to use anti-aliasing are the higher cost of hardware and the slower image generation time versus the low quality of raster images due to misrepresentation of detail. But anti-aliasing need not be as time-consuming as it is thought to be. A good realistic setup for a raster graphics system would include a set of specific anti-aliasing techniques, which would be available at user discretion. For good-quality hardcopy from a low- to medium-resolution display device, the results are definitely worth the time involved. The method is much cheaper than getting a device

with the comparable actual resolution. A more costly and time-consuming system for producing high-quality final images would include a scan-convedrting algorithm with anti-aliasing.

Aliasing effects will occur at any resolution, so this technique will have to be used if accurate images are desired. It need not be a time-consuming process, and the results will be worth the effort.

References

[1] -----, Product Review for AED767, Advanced Electronics Design, Inc., August, 1981 (distributed at SIGGRAPH 1981).

[2] Catmull, Edwin, "A Hidden-Surface Algorithm with Anti-Aliasing," Computer Graphics, Vol. 12, No. 3, pp. 6-11, August, 1978 (Proceedings of SIGGRAPH 1978).

[3] Crow, Franklin C., "The Aliasing Problem in Computer-Generated Shaded Images," Communications of the Association for Computing Machinery, Vol. 20, No. 11. pp. 799-805, November 1977.

[4] Crow, Franklin C., "The Use of Grayscale for Improved Raster Display of Vectors and Characters," Computer Graphics, Vol. 12, No. 3, pp. 1-5, August, 1978 (Proceedings of SIGGRAPH 1978).

[5] Feibush, Eliot A., Marc Levoy, and Robert L. Cook, "Synthetic Texturing Using Digital Filters," Computer Graphics, Vol. 14, No. 3, pp.294-301, July 1980 (Proceedings of SIGGRAPH 1980).

[6] Gupta, Satish, and Robert F. Sproull, "Filtering Edges for Gray-Scale Display," Computer Graphics, Vol. 15, No. 3, pp. 1-5, August, 1981 (Proceedings of SIGGRAPH 1981).

[7] Leler, William J., "Human Vision, Anti-Aliasing, and the Cheap 4000 Line Display," Computer Graphics, Vol. 14, No. 3, pp. 308-313, July 1980 (Proceedings of SIGGRAPH 1980).

[8] Newman, William M., and Robert F. Sproull, Principles of Interactive Computer Graphics, Second Edition, McGraw-Hill, New York, 1979.

[9] Ward, John E., "Wobbled-Raster High-Resolution Display," Electronic Systems Laboratory, MAC-MEMO-431, March, 1974.

[10] Warnock, John E., The Display of Characters Using Gray Level Sample Arrays," Computer Graphics, Vol. 14, No. 3, pp. 302-307, July, 1980 (Proceedings of SIGGRAPH 1980).

[11] Weiman, Carl F. R., "Continuous Anti-Aliased Rotation and Zoom of Raster Images," Computer Graphics, Vol. 14, No. 3, pp. 286-293, July, 1980 (Proceedings of SIGGRAPH 1980).

CID: 4011963

1982 Local Area Network Status (U)

by [   ] T44

P.L. 86-36
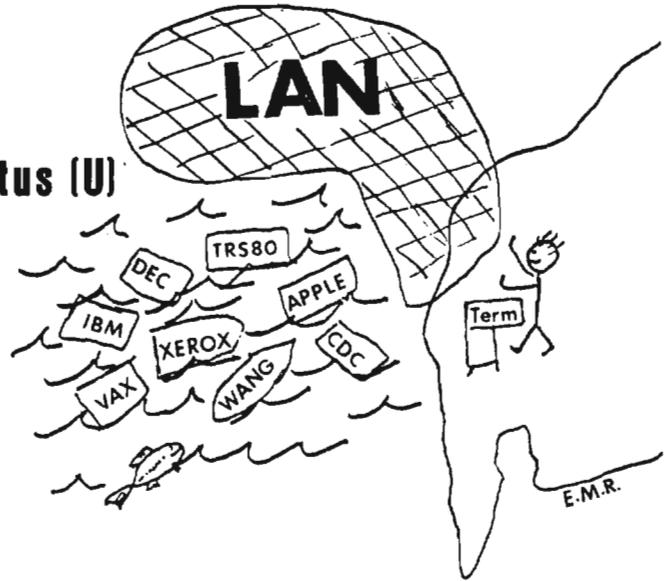
OVERVIEW

The purpose of this paper is to describe the current status of Local Area Networks (LANs). LAN technology is one of the fastest growing areas of computer communications. Early systems have been operational since 1977. In the past two years there has been an increased interest by vendors and the business community to utilize new technology and reduce computer processing costs. In this paper it is assumed that the reader has a basic knowledge of networking concepts and that there is interest in the basic components of LAN architectures. This paper is not intended to cover extensive long-range developments or to get into technical details below what is necessary to explain current LAN architecture. Future LAN objectives will be mentioned with the purpose of expanding the knowledge of NSA/DOD standards, requirements, and applications.

The demand of the business community to consolidate data storage and computer processing while reducing operating costs is one of the motivating factors of LAN development. Technical advances in computers and communications have brought both entities closer together. Many computers are beginning to rely on LANs to perform communication functions external to them. For example, LANs are utilized to move data from a computer to a central storage area for future access. Magnetic tape and printer functions can be centralized by using LAN technology thus minimizing computer workloads and processing time. Data can be moved on the LAN at speeds ranging from 9.6 Kbps up to 50 Mbps. The data can be moved error-free, to the users' eyes and be structured in a form to meet vendors' computer requirements.

Basic LAN Structure

Local Area Networks function as community interconnection media within broader network architectures to allow rapid communications among members of user groups within limited physical areas at relatively low costs. LANs are frequently used to provide for connectivity, office automation, data transfers within distributed processing systems, and for terminal-to-host computer connections. The basic components of a LAN are its host systems, communication medium (twisted pair, CATV cables, microwave, fiber optics), hardware that interfaces hosts to the communication medium (generally called bus interface units or BIUs), and protocols (generally implemented in software). A particular LAN implementation depends on the user's requirements, the available technology, physical constraints, and relevant network standards. Anticipated developments within communications and computer technology dictate that LANs be flexible in order to accommodate different types of processing equipment and to provide for BIU upgrade without requiring protocol changes.

Basic Network Structure

LAN Communication Media

Communication technology advancements are minimizing the costs of LANs while providing for rapid delivery of data over short distances. The most commonly used transmission technologies are twisted pair, microwave, fiber optics, and coaxial cable (CATV).

Twisted pair have a high bandwidth which makes them reliable for high-speed data transmissions. By twisting the wires, requirements for shielding are reduced, but they are still very susceptible to external interference. The twisted pair are easy to install for a point-to-point connection. Some vendors, such as IBM and the telephone company, use twisted pair in their network structures.

Microwave transmissions are the most expensive LAN systems to install, mainly because they require special transmitters and receivers to move the data through the air. This is used where a line-of-sight transmission is possible and a physical connection is too expensive or not possible. An example of such a situation is where LAN hosts are in two different buildings with a main road between them. In this case it may be less expensive to install microwave equipment than attempt a physical connection.

Fiber optic lines transmit data at speeds in the Gigahertz range with a very low bit error rate. It transmits data in one direction only which means two cables would be required for a LAN installation. Current fiber optic technology can only be used in point-to-point connections because cable splicing has not been perfected. Fiber optic lines are relativity free from line interference and are the most secure for a DOD mode of operation. Fiber optics will be the transmission medium of the future when the technical problems are solved and the costs come down.

Coaxial cable (CATV) is currently the most widely used transmission medium. It combines a very low data loss with high bandwidth transmission. The CATV (75-ohm) has been used in television and antenna connections for many years and is very reliable. This means it is a commercially produced product with connection methods and installation techniques being readily available at a low cost (approximately 31 cents per foot). The two most mentioned CATV transmission media are baseband and broadband. Baseband is simple to use, but it uses the entire transmission bandwidth to move the data from one host to another. You may have to use two lines if the volume of transmissions is large. The baseband system is used by XEROX ETHERnet and Ungermann & Bass LANs, to name a few. Baseband on a bus topol-

ogy is a very reliable and fast communication medium. Broadband, which is the other type of bus transmission, employs up to 41 separate channels (frequency ranges) on one cable. Each cable functions independently and is monitored by the BIU. The BIU must be tuned to the frequency or frequencies required. A channel on the broadband system can transmit analog, digital, voice, and video data on different channels at the same time. A host connected to the BIU can be a dumb terminal, intelligent terminal, large computer, telephone, or a TV camera. Broadband is fast and has to be developed more but it has great potential, especially for Agency field site and in-house use. Currently, baseband and broadband cannot communicate with each other. This is where gateway technology, which will be mentioned later, fits into the LAN picture.

Access Methods

An access method on the LAN communication lines is but one of the items necessary for rapid data movement. Some of the basic access methods are circuit switching, token passing, slotted ring, and bus contention transmissions (baseband and broadband). The type of access method is usually determined by the vendor and LAN topology selected to meet the requirement.

Private Automatic Branch eXchange (PABX) is a circuit-switched system with dedicated line transmission that has been in service since 1977. The ROLM CBX corporation has produced over 6000 systems which are in use today. Early PABX systems used analog switching but the newer systems are digitized. Voice data is first digitized and then sent over the network. The architecture of today's telephone system uses the PABX technology and has been in use for several years. INTERCOM IBX, a subsidiary of EXXON Corporation, has one large PABX system installed with approximately 20 more systems on order. The EXXON system is too new for comments.

Token passing is an access method used on ring and loop LAN topologies. It is very fast (1-10 Mbps) with high reliability. A special token character configuration is sent around the ring or loop when there is no data to be transmitted. When a host wants to send data on the network the token must be identified before the host can transmit. This eliminates the possibility of data collisions on the network.

Slotted ring is also an access method used on ring and loop LAN topologies. It is very fast (1-10 Mbps) with high reliability. The slotted ring consists of empty slots moving around the ring of a fixed slot size. Each slot size must be predefined to match the block size of data to be transferred. When

the host wants to transmit data, it finds the first empty slot and moves a block of data.

Bus contention transmission is a random procedure with no preestablished time slots or order of transmission. Usually Carrier Sense Multiple Access with Collision Detection (CSMA/CD) is the mode of transmission. A host will listen for activity on the network before it attempts to transmit the data. There is a possibility that two hosts could sense a clear network and begin transmitting at the same time. Most well-defined vendor networks have algorithms that sense the collision and retransmit the data again without the user's knowledge.

LAN Topology

LANs have architectures that follow basic topologies like ring, loop, star, tree, and bus. To aid in understanding how these LAN topologies function each will be described below. LAN topologies are usually designed to meet a specific requirement. Some requirements dictate that two or more topologies be combined to serve the users needs within the LAN.

LAN TOPOLOGIES

Loop

Loop topology is basically the same as the ring except there is a controlling node in the ring. This node can monitor and control the data as it passes around the ring.

Ring

A ring topology is a single closed path between any two users. Communication is usually over a one-way path and very fast (about 10 Mbps) As the data passes by a connecting node on the communication line the node tests the data address to see if the data belongs to it. If the data belongs to that node it will be extracted from the ring and sent to that host. If the data does not belong to the node, it just passes the data to the next node in the ring.

Star

Star topology is a master/slave type LAN architecture with the master host being the hub of the network and all network functions passing through it. The slave hosts are attached directly to the master host. In this topology the master node has full control of the network. The star topology has the capability to control such things as priority type transmissions and large data base files for the slave hosts. A disadvantage of this architecture is that if the master host ever goes down the entire LAN will not function.

Tree

A tree topology provides a single non-closed path between users. It supports two-way communications after the connections are made. PABX circuit switching is the access media on most of the tree architectures. For the connection to be made, all nodes switch to make the path on which the data will flow. Data must pass through all nodes in the circuit to reach its destination. Telephone company network systems utilize the tree architecture for voice communications.

Bus
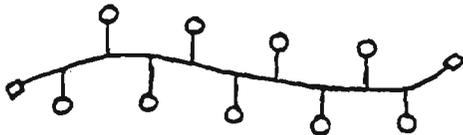
The bus topology is a single non-closed path between users. Bus networks are broadcast-type communications with mostly CSMA/CD as the transporting protocol. The bus topology is the most talked-about LAN architecture today. One of the more controversial subjects of bus architectures is the type of communication media (CATV or fiber optics and baseband or broadband). Each has advantages and the LAN architecture based on the requirements would dictate what is used.

Commercial Status

There has been little cooperation among vendors to establish a unified approach to developing hardware and software for LANs. Rapid advances in communication technology have inspired many old and new vendors to develop their own LANs independently. Most

suppliers are still developing their own components and few have working hardware in production. Some vendors have developed solutions to specific technical problems but have not produced the algorithms or appropriate hardware to implement them. For example, Ungermann & Bass has proposed a solution to provide communication between baseband and broadband network systems and has just started production. Another example is SYTEK Corporation which advertises gateways to handle inter-network communications. This is not yet in production either. SYTEK is over a year late delivering an operable System 40 (host-to-host) BIU line of hardware. Still another example is IBM, which has started support of X.25 but has not released any product specifications. These are only a few examples of products which are advertised but are not yet in production. Most vendors have directed their technology towards capturing a specific part of the LAN market. Some developers (such as SYTEK and MITRE) are addressing general LAN applications that exist now, as well as those expected in the future.

Government agencies and private industries which require LANs now are forced to buy the ones which come closest to meeting their requirements, realizing they may have to change their LANs in the future to meet requirements for inter-network communications. The cost of such an upgrade would be diminished if a set of host interface standards were adopted.

Protocols

In addition to the diversity of hardware, there are presently no accepted standards for LAN protocols. Protocol development is a costly part of building a LAN. In general the BIU hardware and vendor architecture dictate which protocols are implemented in the BIU or which are required to be written by the host's software group. Vendors such as SYTEK propose to supply a large quantity of protocol software within the bus interface unit (BIU) itself. The cost of procuring this type of BIU will be higher than for a less robust BIU, but protocol development in the host will be reduced.

Every vendor develops its own version of a protocol to meet its own product requirements, both within the network itself and for host access to the network. Protocol standards for LANs are currently being considered by the Institute of Electrical and Electronics Engineers (IEEE), National Bureau of Standards (NBS), and other standards organizations. Once such protocol standards become a reality they are more prone to be implemented in firmware. This would allow vendors to concentrate on simpler uniform host interfaces.

However, it must be realized that user requirements may dictate exceptions to the protocol standards. At least there will be protocol standards from which to begin.

LAN Security

The security aspects of protecting data from compromise has been addressed by very few LAN vendors. None has yet produced a totally secure system. There are two types of LAN security of concern for our mode of operation, namely TEMPEST and transmission security. The TEMPEST problem involves electronic radiation from the BIU, communication media, and line connectors. Vendors have not addressed this problem at all. A commercial LAN would need to be TEMPESTed by the purchaser unless the vendor agrees to do it at additional cost.

Transmission security is the encryption of data being sent over the communication lines. Several vendors are addressing transmission security by putting an encryption algorithm in the BIUs. There are no such off-the-shelf units available to date. Network Systems Corporation and SYTEK corporation are two vendors that are addressing data encryption. The only other approach to transmission security is one in which the host provides the software to encrypt and decrypt the data itself.

LAN In-House Activities

NSA has had working packet switching and local area networks in operation for several years. The experience and knowledge gained through these efforts will prove invaluable in the development and integration of LAN technology to meet agency needs.

T443 will be evaluating LAN technology for office automation through the use of the XEROX ETHERnet. This allows office clericals, professionals, and managers using workstations to interact with other users on the same LAN. Office memoranda, inter office mail, and data files can be shared with every connected LAN user. A prototype system will be used to evaluate LAN data flows between users. Basic XEROX hardware is being tested to attain maximum network operational efficiency. Functional procedures are being generated to create an operational test package. Upon completion of the prototype, test evaluations will be coordinated in T443 for the planned implementation of the full operational office automation network.

Since 1976, T41 has been developing a local network using Network System Corporation's HYPERCHANNEL adapters. The HYPERCHANNEL hardware has proved very reliable with a 6000-hour Mean Time Between Failures (MTBF). These adapters are employed in three major

processing centers where extremely high data transmission rates are required. The cost for this type of network is 40 to 50thousand dollars for each host connection permitting an advertised data rate of 50 Megabits per second.

R81 has been actively involved in network technology for several years. They have studied different vendors' hardware and software and are currently contributing to the development of MITREBUS technology for general applications. They intend to model and test baseband and broadband technologies on LSI-11/Z8000 and Motorola 68000 hardware. T44 has been directing many of its LAN inquires to R81.

R63 is developing a fully TEMPESTed LAN for field site use, also based on MITREBUS technology. Their efforts in LAN technology are of interest to T44 since it appears that the requirements driving the development of field site LANs are similar to those we may have in-house. T44 is currently evaluating the R63 requirement in order to assess the level of compatibility with both LAN architectures.

Gateways will be required for inter-network communications with LANs and other networks. R63 and T44 are currently studying one such gateway problem. This gateway will permit a field site LAN to be connected to PLATFORM.

Future LAN Plans

With the rapid development of communications and computer technology the cost of LANs is expected to come down within a few years. Technical advances will no doubt stabilize the technology to allow mass production of more reliable LANs. This should solve the majority of user requirements for communicating locally. The next significant development will come when the LANs need to be connected to other networks. Worldwide communication on digital packet switched networks is now becoming a reality. LAN technology and the development of gateways will make it possible for the local user to achieve local and global communications. Current LAN architectures must consider the global communication requirements of the future, whenever possible, in order to meet the ever expanding Agency commitments.

Conclusion

This brief overview of LAN topologies and access methods depicts that current commercial status and Agency development is progressing. Within two years LAN concepts will be stabilized and a user will be able to select a well defined LAN to meet the requirements of his application.

LOGIC DESIGN EXCEEDING BOOLEAN CAPABILITIES

1v1=1

by [   ] **R53**

BACKGROUND

Contemporary digital computer programming makes use of "languages" which have progressed from the entirely machine-dependent, tedious, "assembly" stage to a nearly machine-independent, expedient, "high-level" stage. High-level language efforts continue to push towards even higher levels (e.g., Ada). The equivalent of a few statements in a latest high-level language could require hundreds of assembly language statements.

A hardware counterpart, digital logic design, makes use of methods which have "progressed" from 2-valued Boolean algebra, truth tables, and Karnaugh maps, to those in combination with hardware design languages, system development hardware, etc. These additional means for logic design include art forms and various algorithms developed since Karnaugh maps.

There is at least one difference between the progress made in programming and that in digital logic design. Programmers, having advanced from the assembly language environment, now create almost entirely with expedient, yet efficient, high-level languages. Logic design practitioners (in contrast to theorists or researchers) still rely heavily on a tedious 2-valued Boolean algebra when not engaged in design art. An expedient, acceptable "high-level" version of 2-valued Boolean algebra has not evolved.

PROBLEMS

A problem with today's use of 2-valued Boolean algebra, the "assembly language" of digital logic design, is that it forces the algebra into service where it really does not apply. Two-valued Boolean algebra applied quite well (transitions between values aside) back in the days when SSI (small scale integration) circuits dominated. In fact, SSI is an attempt to realize 2-valued Boolean algebra.

Besides using 2-valued Boolean algebra, logic design practitioners now "fiddle around" (an art form) with bits, forcing them into devices which, for the most part, do not really operate on bits. Those devices mostly operate on buses of bits--inputs having more than two values--in other words, multivalued inputs!

THE STATUS QUO

One might argue that the logic design methods used today, however described, must be working. After all, technology is nearly at the point where microprocessors could be sold by the pound from a barrel like dried beans.

Logic circuits today are designed using a combination of art and science, and they may always be. A problem with that is balance. Given that logic design is much more an art, then if person A is born with artistic talent, only A may be able to produce "good" designs. Artistic methods cannot be reproduced without

interpretation. Scientific methods can, but can also be tedious and even encumber innovation (an art) when the basis of such methods has been overwhelmed by progress. Such appears to be true for today's logic design methods when they present the designer with 2-valued Boolean algebra as a principal tool.

But who cares? One must admit, microprocessors by the pound is by no means a small achievement.

"Microprocessors by the pound" was fueled with money. Business and government invested and still plan to invest "megabucks" (millions of dollars) in complex systems and commit for long term amortizations in order to produce today's technology. A problem with the megabucks approach applied to processes which include outdated components is that the massiveness of the approach tends to perpetuate the included obsolescence.

CHANGE

At some point a new decision regarding continuation of the present megabucks method is needed. Technology is near the limits of speed and "real estate" (room) on IC (integrated circuit) substrates. Gross parallelism and concurrency, both candidates for megabucks support, are brute force solutions to the limits problem. But is that where the money really should be spent?

One wonders, is the technology at hand being used anywhere near its capacity? Fearfully, the suspicion is that it is not. How about the designs placed on substrates—are they known to be minimized? No. Can some justification for the status quo be made on the basis of expediency? In the megabucks environment, yes, but technological limits are eroding the validity of this argument.

One component of a "smarter" approach to future logic design is to provide a better logic design mathematics. Such a "math" would allow logic design to become less of an art. Even if incapable of producing minimized results under all criteria for minimization, a new design math equipped to be compatible with current and future technologies, yet of itself costing nothing in hardware and software monies, might yield optimal mixes of expediency and efficiency.

This idea is far from new. Algebras and other calculi that could be used in logic design proliferate. Multivalued logics have been around for quite some time, and attempts have been made to convince industry to use them. But two problems with multivalued logics are their complexity and diversity. Add to that their implied required technology

changes, and rejection results. The required technology changes are fearsome. Recall all the megabucks of industrial investments and long term amortizations. Add to that another fear on the part of industry's customers that all their products could quickly be made obsolete.

How can the stalemate be broken? Altruistically, perhaps the federal government could help by attraction: introduce and use a design math that eliminates at least the scary problems discussed. But remaining even with that idea is human resistance to change. Logic design practitioners may fear that the algebra which they have known all their lives will be taken away. Given the math discussed in the next section, nothing is taken away. More is added.

A FIRST RESULT FROM RESEARCH

A "mix-valued" algebra denoted by "Mx" (which we pronounce, "mix") was first reported on [1] while it was in the throes of pre-result research. Mx has since grown to a first level of usable maturity. Now, as a first result of research from R53 [2,3], Mx may be a step in the direction of a math suitable for future logic design.

As proposed, instead of having only AND, OR, and NOT gates, Mx has seven others. Possibly, the additional primitives will enhance designer creativity. (However, one may use Mx as a 2-valued Boolean algebra only.) Each of Mx's ten gates is capable of accepting a whole bus at each of its inputs and can deliver a whole bus as a single output (this includes bus width=1). These bus-handling attributes may help to reduce logic design complexities. Instead of being constrained to the value universe {0,1}, Mx allows one to redefine it for every use of any gate. Such a freedom may enable whole design outlooks to change to the point of inspiration. Each of Mx's ten gates can also operate on combinations of buses of any radix and multivalued signals, without concern for technology. This independence may help Mx to be resistant to obsolescence. Possibilities for compounds of the proposed Mx-gates, analogous to the NAND and NOR gates developed from 2-valued Boolean algebra's three operators, can be imagined. Clearly, there should also be use for the fact that the proposed Mx-gates can be arranged to form sequential and memory circuits.

Although not as mature as 2-valued Boolean algebra, meaning it remains fertile in many ways, Mx has been used successfully in modest, practical logic design experiments. A review of 2-input instances of Mx's proposed gates, some of which were used in those experiments, are pictured below. (Yes, Mx's NOT gate can

have more than one input.)
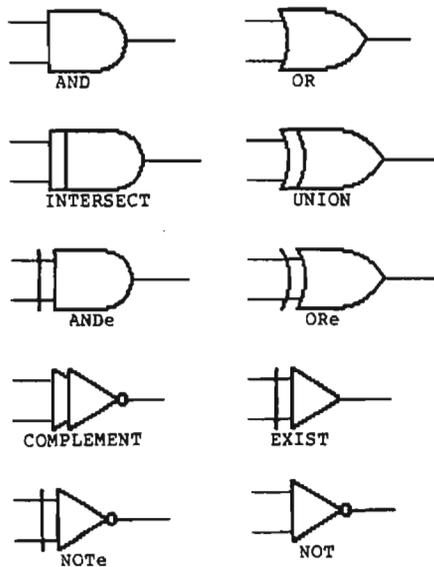
ANDe, ORe, and NOTe are pronounced, respectively, "existential AND," "existential OR," and "existential NOT."

Each Mx-gate has an associated connective symbol. The following table shows them used with output z and inputs $x_1, x_2, \ldots, x_n$.

gate	usage
AND	$z = x_1 x_2 \ldots x_n$
OR	$z = x_1 + x_2 + \ldots + x_n$
NOT	$z = x_1\verb|^| x_2\verb|^| \ldots \verb|^| x_n$
INTERSECT	$z = x_1 \wedge x_2 \wedge \ldots \wedge x_n$
UNION	$z = x_1 \vee x_2 \vee \ldots \vee x_n$
COMPLEMENT	$z = x_1 " x_2 " \ldots " x_n$
ANDe	$z = x_1 \& x_2 \& \ldots \& x_n$
ORe	$z = x_1 \Leftphi x_2 \Leftphi \ldots \Leftphi x_n$
NOTe	$z = x_1 \phi x_2 \phi \ldots \phi x_n$
EXIST	$z = x_1 \exists x_2 \exists \ldots \exists x_n$

Concatenation in the expression for AND may be replaced with dots, resulting in

$$x_1 x_2 \ldots x_n = x_1 \cdot x_2 \cdot \ldots \cdot x_n.$$

"$\verb|^|$" may be replaced with "$'$" in expressions where the NOT gate is to operate entirely within 2-valued Boolean constraints, that is, when the gate's value universe is $\{0,1\}$ and $n=1$. $x_1' x_2' \ldots' x_n$ is undefined for $n>1$ because 2-valued Boolean NOT is unary. Notice that $x' \neq x\verb|^|$.

Each Mx-gate instance g operates with respect to its own value-universe, called a "reference set," denoted by the variable, "r_g." A reference set instance may be ordered or not, as needed. Since r_g is a variable, a gate's reference set may be changed as often as desired. There are no limitations on r_g contents for any Mx-gate. However, the "reference set" for every 2-valued Boolean gate instance is the constant, $\{0,1\}$, and is ordered, $0<1$. (This is why designers can use Mx yet not do anything different than when using 2-valued Boolean algebra--2-valued Boolean algebra is a subalgebra of Mx!) When a Mx-gate's inputs contain an element not in r_g, the gate will produce the null, "ϕ" [2,3]. ϕ can also be the valid result of a Mx-gate's operation.

ϕ can have one of two effects at Mx-gate inputs:

a. The ϕ-bearing input effectively vanishes, needing no consideration in the gate output determination.

b. The ϕ-bearing input appears to have no value.

With (b), a gate can be made to wait for a non-ϕ on all, none, or exactly one of the inputs before generating an output.

ϕ-generation results in a situation which may be unfamiliar to many: the usual algebraic properties of association and distributivity involving any gate receiving a ϕ are disallowed. As an example regarding associativity, let two AND Mx-gates be connected in series, the first having inputs a, b, and c, and the other having the additional inputs d, e, and f. The expression for their output z cannot be correctly written as, z = abcdef. Instead, the correct expression is z = (abc)def, where (abc) must be evaluated first.

Definitions of the operations of the proposed Mx-gates first requires definition of the "atomizer function." Specifically, the atomizer function A on a set X of n inputs, i.e., A(X), where $X = \{x_1, x_2, \ldots, x_n\}$, decomposes the sets ranged over by all its member input variables x_i, $i \in \{1,2,\ldots,n\}$, so that no member of the result set is itself a set. For example,

if $X = \{x_1, x_2\}$, $x_1 \in \{3,4,\{7,13\},6\}$,

and $x_2 \in \{2,4,\{6,13\},25\}$,

then $A(X) = \{2,3,4,6,7,13,25\}$.

Definitions of the operations of the proposed Mx-gates follow and are different from those originally reported [1]. Except where stated otherwise, ordering of r_g members is not needed in those definitions. Notice that for $r_g = \{0,1\}$ and $x_i \in \{0,1\}$, the AND, OR, and NOT Mx-operations produce results identical to those same-named operations in 2-valued Boolean algebra.

$$AND(X) \equiv \begin{cases} min(A(X)) & \text{if } \forall x_i \in X, \; x_i \subseteq r_g \\ \emptyset & \text{otherwise} \end{cases}$$

$$OR(X) \equiv \begin{cases} max(A(X)) & \text{if } \forall x_i \in X, \; x_i \subseteq r_g \\ \emptyset & \text{otherwise} \end{cases}$$

$$ORe(X) \equiv \begin{cases} x_1 & \text{if } x_1 \neq \emptyset, \; x_2 = x_3 = x_4 = \ldots = \emptyset \text{ and } x_1 \subseteq_g \\ x_2 & \text{if } x_2 \neq \emptyset, \; x_1 = x_3 = x_4 = \ldots = \emptyset \text{ and } x_2 \subseteq r_g, \\ \quad \vdots \\ x_n & \text{if } x_n \neq \emptyset, \; x_{n-1} = x_{n-2} = \ldots \\ \qquad = x_{n+1} = x_{n+2} = \ldots = \emptyset, \text{ and } x_n \subseteq r_g \\ \emptyset & \text{otherwise} \end{cases}$$

Total ordering of r_g members is needed for the following NOT(X) definition. Also, the least valued member or its r_g must behave like a zero, the next higher must behave like a one, the next higher like a two, etc.

$$NOT(X) \equiv \begin{cases} max(r_g) - A(X) & \text{if } \forall x_i \in X, \; x_i \subseteq r_g \\ \emptyset & \text{otherwise} \end{cases}$$

$$EXIST(X) \equiv \begin{cases} max(r_g) & \text{if } \forall x_i \in X, \; x_i \subseteq r_g \text{ and } x_i \neq \emptyset \\ \emptyset & \text{otherwise} \end{cases}$$

$$UNION(X) \equiv \begin{cases} union(\{A(x_1)\},\{A(x_2)\},\ldots,\{A(x_n)\}) & \text{if } \forall x_i \in X, \; x_i \subseteq r_g \\ \emptyset & \text{otherwise} \end{cases}$$

$$INTERSECT(X) \equiv \begin{cases} intersection(\{A(x_1)\},\{A(x_2)\},\ldots,\{A(x_n)\}) \\ \qquad \text{if } \forall x_i \in X, \; x_i \subseteq r_g \\ \emptyset & \text{otherwise} \end{cases}$$

"-" is set subtraction in the following COMPLEMENT definition.

$$COMPLEMENT(X) \equiv \begin{cases} \{r_g - union(\{A(x_1)\},\{A(x_2)\},\ldots,\{A(x_n)\})\} \\ \qquad \text{if } \forall x_i \in X, \; x_i \subseteq r_g \\ \emptyset & \text{otherwise} \end{cases}$$

$$ANDe(X) \equiv \begin{cases} X & \text{if } \forall x_i \in X, \; x_i \subseteq r_g, \text{ and } x_i \neq \emptyset \\ \emptyset & \text{otherwise} \end{cases}$$

$$NOTe(X) \equiv \begin{cases} r_g & \text{if } x_1 = x_2 = \ldots x_n = \emptyset \\ \emptyset & \text{otherwise} \end{cases}$$

Did somebody say "**MX**"?

No, he said "**Mx**" (pronounced "Mix")!

USING THE FIRST RESEARCH RESULT

Step-by-step Mx-designs of five combina-
torial memoryless circuits, ranging from very
simple to not-so-simple, have been reported
[2], making use of a "First-Order Logic
Design" procedure. [2] An improved statement
of the Procedure has also been reported [3].

A listing of specifications and results of
the five Mx-designs referred to in the preced-
ing paragraph, follows. When compared to
bused I/O (input or output or both) versions
designed using 2-valued Boolean algebra, those
Mx-design results are obviously significantly
simpler. Such simplicity will have more value
once direct realization ICs for the Mx-gates
exist. (A direct realization effort is under
way, and a patent has been applied for.)

DESIGN #1

Specification: Transmit to an output z any
value from the set {a,b,c} common to input
buses p, q, and r.

Result: $z = p \wedge q \wedge r$, with the reference set shown.
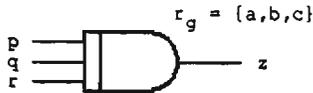
DESIGN #2

Specification: Same as for Design #1 except
instead of transmitting to the output the
identical value from the set {a,b,c}, send
only one signal to indicate that all three
inputs are one of a, b, or c.

Result: $z = \exists(p \wedge q \wedge r)$, with the reference sets shown.
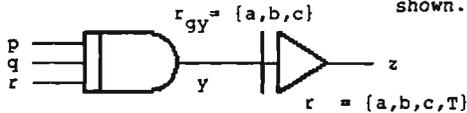
DESIGN #3

Specification: Design a multiplexer having
bus output variable z, data bus input vari-
ables p, q, and r, and a "select" bus input
variable s, so that via s, value "a" selects
p, value "b" selects q, and value "c" selects
r.

Result: z = ps + qs + rs, with the refer-
ence sets shown and where a>p, b>q, and
c>r. (Notice that associativity cannot be
applied to the result expression.)
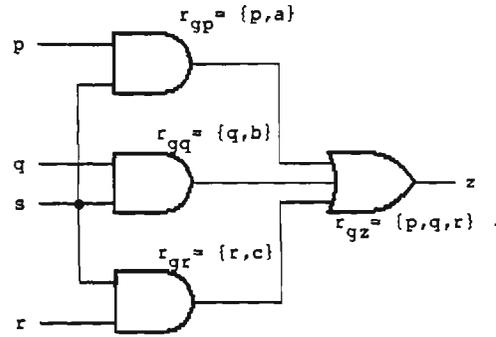
DESIGN #4

Specification: Convert a single binary bus
input x of hexadecimal code to octal, deliver-
ing the results to binary bus outputs y and z.

Result: $y = (\exists x) \oplus (\exists(\&x))$ and $z = (\&x) \oplus (\&x^{\wedge})$,

with the reference sets shown.
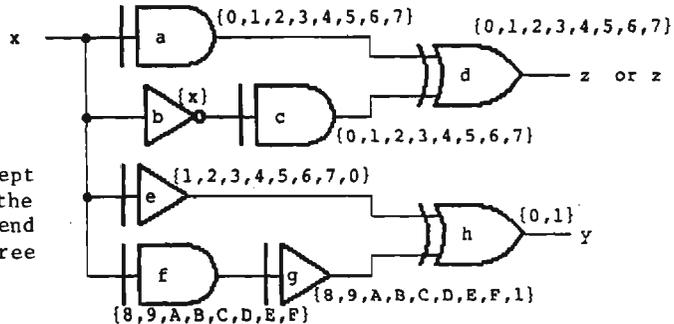
DESIGN #5

Specification: Design a comparator which
will indicate at one of its outputs g, e, or 1
(lower case L), that at its inputs x and y,
x>y, x=y, or x<y, respectively.

Result: $g = \exists(\exists(x \wedge y) \wedge (xy) \wedge y))$,

$e = \exists(x \wedge y)$, and $1 = \exists(\exists(x \wedge y) \wedge (xy) \wedge x))$,

with the reference sets shown.
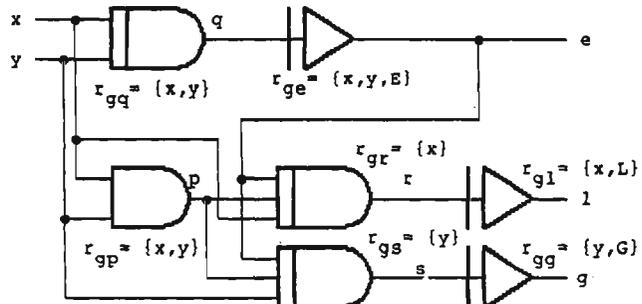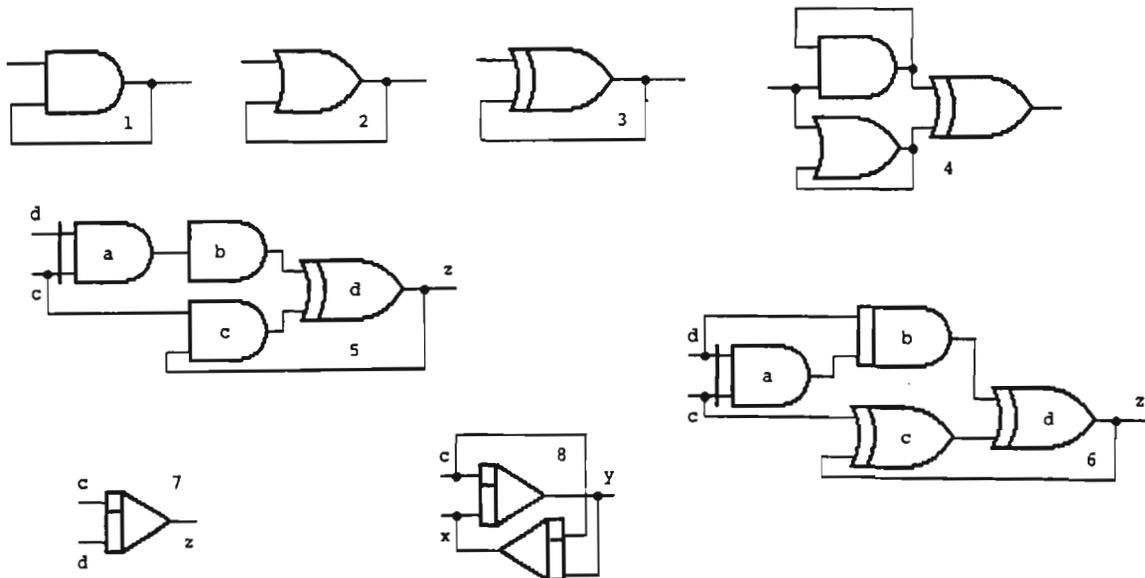
ITEMS STILL IN RESEARCH

The above figures (an assortment of heretofore unpublished memory circuits for the reader to ponder) use only the proposed Mx-operators.

1. A "min" latch. The output will hold the minimum of all values it has received since being activated. Resets to ϕ upon introduction of a value not in the reference set.

2. A "max" latch. The output will hold the maximum of all values it has received since being activated. Resets to ϕ upon introduction of a value not in reference set.

3. An accumulating data latch. The output will hold all inputs. Resets to ϕ upon introduction of a value not in the reference set.

4. A min-max latch (could be used as a "truest and most false" value finder). The output holds the least and greatest values received since being activated. The min or max value is reset to ϕ upon introduction of a value not in the AND or OR gate reference set, respectively. Both the min and max values are reset to ϕ upon introduction of a value not in either reference set.

5. An arbitrary-value, one-level memory. $c \in \{\phi,C\}$, $C>d$, $d \equiv$ data, $r_{ga} = r_{gb} = \{C,d\}$, $r_{gc} = r_{gd} = \{d\}$. A new value of d is loaded in memory whenever $c=C$. (When c

is a clock signal, C is a high potential, and ϕ is a low potential, this circuit also describes the single-direction behavior of the nMOS pass transistor used in IC chip design.)

6. An arbitrary-signal, one-level memory. $c \in \{\phi,C\}$, $d \equiv$ data, $r_{ga} = r_{gb} = \{C,d\}$, $r_{gc} = r_{gd} = \{d\}$. A new instance of d is loaded in memory whenever $c=C$. This is a generalized version of circuit 5; d may be other than a value (e.g., a set). Therefore, under the same assignments of potential values for C and ϕ as described for circuit 5, this circuit also describes the single-direction behavior of the nMOS pass transistor.

7. A polarized, single symbol used to represent circuit 6, i.e., a gated single-level memory.

8. A connection of two circuits (7). This configuration accounts for the bidirectional behavior of the nMOS pass transistor. When allowed to represent a connection of two circuits (6), this configuration more specifically accounts for the bidirectional behavior of the nMOS pass transistor. Input c still represents a clock signal, but the remaining terminals are now each I/O terminals. Hence, the remaining terminals are relabeled impartially, "x" and "y." This configuration additionally requires an additive "Logic Current Law," analogous to Kirchoff's Current Law, for complete operation description. The Logic Current Law is beyond the scope of this paper.

CONCLUSIONS AND SUGGESTIONS

Although at a usable level, Mx is still immature. It needs enhancements born of feedback from users and researchers (who may even find fatal problems with it). But potential users may find little immediate incentive to try Mx. Having a variety and abundance of "building block" ICs, those users may not want to learn a new design algebra, no matter how good it may be. Logic design today is largely a matter of "putting together" (an art form) large, IC-realized, functional modules. Having to work a little harder every now and then, designing-in some SSI "glue chips" by means of a "tedious" 2-valued Boolean algebra, may be acceptable. And, having even greater capability yet with the same size building block ICs, possibly resulting from use of an updated design math, may have little importance to them.

Yet among them may be those who would like to see logic design done more expediently and methodically, with the possibility of mathematical verification, and at the same time would like to try Mx. For them, beginning suggestions follow: First, throw away nothing currently used in design, i.e., methods, ICs, systems, etc. Next, while designing in the manner accustomed to, try to identify logic design tasks which seem tedious. Try some of the Mx design methods known [2,3] and possibly add some of your own. Note how those methods may or may not have applied; then with that experience, look for other areas in your design that could use improvement. (These steps should help to increase one's ability to think in terms of Mx-gates and properties.)

When sufficiently brave, try a small design from scratch using Mx. If there are parts of the design for which no Mx method is known, patch in what has worked in the past. When done, one can either put the resultant logic design away until the day when direct-realization chips for Mx exist or try to create realizations.

Realizations can be created by:

a. building them from non-programmable "catalog" parts,

b. building them from programmable "catalog" parts, and

c. making custom ICs.

Method (a), although expedient, may not result in the best solution in terms of space, speed, and power consumption. Method (b), using PLAs (programmable logic arrays) or variations on ROMs (read-only memories), may be slightly less expedient and possibly result in slower operation than (a). But for modest-size reference set and input cardinality combinations, (b) probably uses much less space and power than (a). Method (c), if one can monetarily afford it, may be the most efficient from the standpoint of circuit performance but may also have the longest period from design to realized circuit (bad for "breadboarding" expedience). Using a table-lookup approach (suggested by Dr. Richard K. Kunze, R53, 24 August 1982, and also used in method (b)), PLA or ROM versions of Mx circuits have regularity, readily lending themselves to "very large scale" IC implementation. Physical circuit compaction beyond that possible by using Mx alone may be available through electrical and physics "tricks" and sharing of on-chip resources. (Such an extra-method of compaction is analogous to the greater compactions in assembly language achievable when human intelligence is applied to the object code produce by a software compiler.)

Has the time come to move towards a logic design mathematics compatible with today's and future technologies?

References   P.L. 86-36
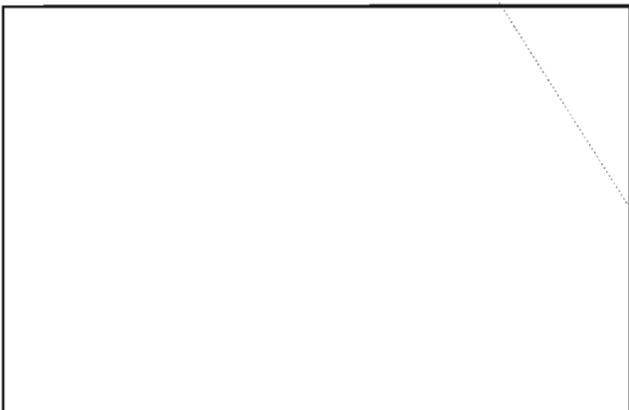
I don't care what you say about it. I still like Boolean algebra, by George!

Boole

It's OK, George — Mx includes your stuff.

Author

CID: 4011963

Ada:
Conquering the
TOWER OF BABEL(U)

Ada Augusta, Countess of Lovelace, the daughter of Lord Byron and a gifted mathematician who worked with Charles Babbage, a computer pioneer. She is the person after whom the Ada language is named.

by

INTRODUCTION

In an effort to implement standards, to stay steadily increasing software costs, and to create a universal language for embedded computer systems, a long-term research effort began at the Department of Defense (DoD). The result of this effort is Ada, a new programming language.

This paper will explore the development and capabilities of Ada. In the process, it will illustrate to the reader that several factors have to be considered when examining the performance of a programming language. It is not enough to look at only the characteristics of the language. The language must be studied within the context of its intended purpose.

The following sections will examine the Ada language. Ada's history will be presented as will some features which are not common to higher-level languages. Factors which will be important to the long-term success of Ada will be presented in the "Conclusion." It will be apparent that Ada is still in an evolutionary process and its success is not assured.

GENERAL OVERVIEW

The influx into the marketplace of a variety of programming languages and resultant software began with the commercial acceptance of computers. The user, who was originally a participant in the creation process, was replaced by specialists who provided languages and software which did not necessarily meet the user's specific needs.[1] Changing technology and demands for increased capabilities aggravated this problem. Many approaches were initiated to alleviate the resultant software crunch. This paper will focus on Ada, one Department of Defense (DoD) solution to software problems. Here, the pendulum has swung back again and the user, in conjuction with the specialists, is a participant in the process.

A programming language cannot be created in a sterile environment. It must be designed with the same care an architect would take in designing a house. The architect must be aware of the customer's needs and objectives before initiating design plans. He must also be aware of other factors which will interact with the house. These will include, but are not limited to, public utilities, the neighborhood, and the customer's lifestyle. Once the architect has developed a clear understanding of his objective, he can begin to draft a design fitting the customer's desires and relating them to the physical environment. The architect's interaction with the customer, the zoning commissioners, and others should not end here if the plans are to be acceptable to all involved. To ensure that the final product meets the requirements, the design must be reviewed at intermediate stages, preferably with those involved, to ensure that the original objectives have not been lost.[2]

The creation of a programming language should proceed in the same manner. The creator must determine what the objectives of the language are to be, and with what it will interact. Upon determination of what the programming language should contain, design plans should allow for adequate reviews throughout the process. The final product should be a programming language which meets the needs of the user and can interact well with the user's environment.

All languages which have been created to date have been created to perform certain functions. FORTRAN (FORmula TRANslation) is a mathematically based language intended for use by both scientists and engineers. It is well-suited for the handling of complex mathematical problems but is not well-suited for handling large amounts of input and output (I/O). COBOL (COmmon Business Oriented Language) was designed for business-oriented problems which may involve a large amount of file processing and I/O, but only involve simple mathematical functions. Because of its business orientation, the language was designed to closely resemble English, thus being made easy to code and read as well as self-documenting. Unlike FORTRAN, it has

Also see page 155.

* Ada is a trademark of
DOD Ada Joint Program Office

special provisions to make it easier to manipulate and process alphanumeric data. PL/1 (Programming Language 1) combines the advantages of COBOL and FORTRAN, i.e., the file processing, I/O, and mathematical capabilities. It is a multi-purpose language which can efficiently handle either scientific or commercial problems as well as combinations of the two. These examples illustrate the diversity of programming languages. It should be evident that to date there is no language which can perform all functions optimally. Tradeoffs have to be made when deciding which programming language to use.

The programmer thus has to consider several factors when choosing a programming language. Among them are:

[] Language availability,
[] Language familiarity of programmers,
[] Ease of program maintenance,
[] Cost of programming,
[] Time needed to write the program,
[] Time needed to execute the program, and
[] Characteristics of the problem.[3]

Whether a language is or can be supported by his system must also be considered. These are factors which help to determine which language will be chosen for a particular application. They are also factors which will be included in the determination of whether Ada will become the standard Department of Defense language.

EVOLUTION OF ADA

The creation of a new programming language was not originally considered as a solution when it was discovered through several studies in the early 1970s that a major problem at the Department of Defense (DoD) was language proliferation. Instead, the DoD high-order-language standardization program was initiated. This program considered standardizing seven already established programming languages in order to alleviate the problem. Feedback from the users, however, indicated that seven languages would still be too many languages to simplify the problem notably. More studies were initiated to determine what the requirements of the language would be. It was discovered that there were no significant differences in the tri-service (Army, Navy, Air Force) requirements. Furthermore, $100 million per year would be saved by converting to a simple common language. The decision was made to consider a single programming language.[4]

The first definition of requirements for a single common language was presented in 1975 by the High-Order Language Working Group (HOLWG) which had been created to identify and recommend solutions. Existing programming languages were reviewed to see if any could fulfill all the requirements. None could satisfy more than 75% of them. Factors which led to the decision to create a new programming language included the inability of current languages to handle easily and efficiently such functions as parallel processing, real-time input and output (I/O), and exception handling. It was also noted that in many projects at DoD it was necessary to modify existing languages in order to provide for needed enhanced capabilities. Once the decision was made that a language needed to be created, competitive bids for a language design that met the requirements were requested. The design was to use one of three languages for its base: ALGOL 68, PASCAL, or PL/1. Of the three, PASCAL was the most popular.[5] In 1977 the field of competitors was cut from four to two, and in early 1978, CII HONEYWELL BULL's language design was accepted as the preliminary definition of Ada.[6]

It is important to note that there was a great deal of input from many sources about the requirements of the language. As was stated in the Foreword of the Reference Manual for the Ada Programming Language:

"The reviews and comments, the numerous evaluation reports received at the end of the first and second phases, the more than 900 language issue reports, comments, and test and evaluation reports received from 15 different countries during the third phase of the project, and the on-going work of the IFIP Working Group 2.4 on system implementation languages and that of the LTPL-E or Purdue Europe all had substantial influence on the final definition of Ada."[7]

The requirements documents, all of which were circulated for comments, went through five revisions. The culmination of this effort was the STEELMAN Report published in June 1978, which set forth the final requirements of the language.[8] Because the importance of the support environment is realized, the same approach used to develop the STEELMAN requirements was used to develop the requirements for a support environment.[9]

Both the Army and Air Force have awarded contracts to build the Ada compiler and develop a program development environment.[10] In their contract, the Army specified that the compiler be capable of running on four systems: the VAX-11/780, the PDP-11/70, the AN/GYK-12, and the Litton L3050.[11] The initial compiler will be designed to run on the VAX-11/780 and will generate code for various target machine environments. The completion of this compiler is slated for early 1983.[12]

DOCID: 4011963

The Air Force contract initially specifies a compiler to run on the IBM 370 series with future systems including the Perkin-Elmer Corp. model 8/32, the Dec system 10, and the CDC 6600. The completion date is slated for mid-1983.[13] Because some universities and hardware manufacturers have also begun compiler development projects, it is considered likely that there will be a workable full Ada compiler by the end of 1983.[14]

The Ada programming language was completed in July 1980. By December 1980, it was designated Military Standard 1815 [15] with the intent of using the language to create software which will "implement such applications as command, control and communications, fire systems, storage and retrieval, and tactical systems."[16] Ada was submitted by the Department of Defense in April 1981 for approval by the American National Standards Institute (ANSI) and will be submitted by ANSI to the International Standards Organization (ISO) for approval as an international standard.[17] While Ada has been designed to act as common language for embedded computer systems, indications are that it might also act as a standard language for general application computer systems.

AN ANALYSIS OF FEATURES OF ADA

The structure of the Ada language is not unique in itself as Peter Wegner illustrates in the following overview of the levels of Ada's program structure:

[] Characters, which are the lowest-level atomic constituents of a program;
[] Lexical units, which are the atomic units of meaning (semantic units);
[] Expressions, which specify a computation that computes a "value";
[] Assignment statements, which assign the value computed by an expression to a variable:
[] Control structures, which can control the sequence in which assignment statements and other statements of the program are executed;
[] Declarations, which define the attributes of identifiers used in the statements of a program;
[] Program Units, which associate declarations defining the attributes of identifiers with statements which use them;
[] Compilation Units, which are the units of structure for program development and separate compilation.[18]

The majority of these features are common to all higher-level programming languages. What makes Ada unique is the versatility of the compilation units. These include procedures and functions, as well as two newer concepts,

packages and tasks. Each unit is separately compiled and subsequently placed in a program library. The compiler can then check for syntax errors and error of type compatibility of calls throughout its compilation process. Types will be discussed later.

The Ada Package is considered one of Ada's most significant features. It is defined in the Reference Manual as a "unit specifying a collection of related entities such as constants, variables, types, and subprograms."[19] By grouping these together, it facilitates a logical view of the unit as well as allowing the modularity sought in top-down design. It is a flexible construct which can be used for a variety of functions. One is to create a package which allows the sharing of information in a common area apart from any one program. This is similar to the FORTRAN COMMON block. The difference lies in the fact that data types may also be provided via the package. Another use, which deals with the information hiding feature to be discussed later, is grouping related subroutines. This allows the grouped subroutines to share the same variables while inhibiting access by modules outside the package.[20]

It is the structure of the package which allows information to be hidden from the user of the package. The package is partitioned into two sections. These are the package specification and the package body. The package specification is basically a sequence of declarations. It is divided into a visible part and a private part. Within the visible part are those entities which may be used by units outside the package. The private part contains that which is necessary for the compiler but not for the user of the program. In the package body is contained the code necessary to implement those resources specified in the visible part of the program specification. Neither the declarations nor the code in this section is accessible to the user. An example of a package and its call follows.

```
package RATIONAL_NUMBERS is
  type RATIONAL is private; --hides
                           representation
                           from users
  function EQ (X,Y:RATIONAL)return BOOLEAN;
  function "+"(X,Y:RATIONAL)return RATIONAL;
  function "*"(X,Y:RATIONAL)return RATIONAL;
  function "/"(M,N:RATIONAL)return RATIONAL;
private
  type RATIONAL is
    record
      NUMERATOR:INTEGER;
      DENOMINATOR:INTEGER
              range 1..INTEGER'LAST
    end record
end RATIONAL_NUMBERS; --end of package
                      specification
```

```
package body RATIONAL_NUMBERS is
  procedure SAME_DENOMINATOR
            (X,Y:in our RATIONAL)is
  begin
    --reduces X and Y to the same denominator
  end;
  function EQ(S,Y;RATIONAL)return BOOLEAN is
    U,V:RATIONAL          --body of EQ
  begin
    U:=X;
    V:=Y;
    SAME_DENOMINATOR(U,V);
    return(U,NUMERATOR=V,NUMERATOR)
  end EQ;
  function "+"(X,Y:RATIONAL)return RATIONAL
                    is ...end"+";
  function "*"(X,Y:RATIONAL)return RATIONAL
                    is ...end"*";
  function "/"(M<N:RATIONAL)return RATIONAL
                    is ...end"/";
end RATIONAL_NUMBERS:


    To use the Rational Numbers Package:

with RATIONAL_NUMBERS; --make compilation
                        unit visible
procedure USE_RATIONAL is
  use RATIONAL_NUMBERS; --allow unqualified
                      use of +,*,EQ,/,RATIONAL
  X,Y,Z:RATIONAL;          --declare three
                          RATIONAL objects
begin
  X:=3/4;              --rational number creation
                        and assignment
  Y:=6/8;
  if EQ(X,Y)then  --rational number equality
                    testing
    Z:=X*Y;       --rational no multiplication
                    and assignment
  else
    Z:=X+Y;       --rational number addition
                    and assignment
  end if;
end [21]
```

The Ada Task is defined by the Reference Manual as a routine that may operate in parallel with other routines.[22] In essence, what occurs is a simulation of a multi-processing system. Through the use of keywords, execution of task routines can be interleaved. While this is common to assembly level languages, it is not common to higher level languages. Benefits include increased efficiency and "aid in conceptualizing certain applications."[23]

The construct of the task is the same as the package. Consequently, the same information hiding feature is present. The difference is that the body part of a task contains a routine that can be run in parallel with other tasks.

Another prominent feature of Ada is strong typing. The data type determines which set of values and corresponding operations are applicable for a given identifier that has been declared. Strong typing restricts the value and operations of the declared variable to only those that are applicable to the declared type. The type of every variable and expression can be determined at compile time, thereby reducing run-time errors. The attention the programmer needs to give to typing of variables also reduces errors during the writing of the program. The user is given some flexibility in that there are four classes of types. These are scalar types, composite types, access types, and private types. Furthermore, Ada has powerful type-definition capabilities which allow for the defining of new types by the user.

The first class of types to be addressed is scalar, which is subdivided into discrete and numeric subtypes. Discrete, in turn, includes enumeration and integer types. Scalar types can be used for indexing, loop iteration, and choices in case statements and record variants. The enumeration type explicitly declares its values in the type definition. It is useful because it can be used to define finite sets of objects such as colors, weekdays, or directions. The predefined types 'character' and 'boolean' are enumeration types. Numeric types provide the means for performing numerical computations. It can be viewed as being subdivided into integer and real types. The integer type is predefined and consists of a set of consecutive integers. While it has an implicit set of values, a range of values can be explicitly set by establishing a range constraint in the type definition. Approximate computations can be accomplished through real types which are subdivided into floating point types and fixed point types.

Composite types describe arrays and records. An array is an aggregate of identically typed elements which are identified by indices. A record can be viewed as a structured object consisting of named elements of possibly different types. The elements are selected through their identifiers. Composite typing allows a good deal of flexibility to the programmer. The elements in both arrays and records can be manipulated individually, treated as aggregates and/or be directly assigned to compatible structures. The bounds of an array need not be specified until object declaration time. Thus it is possible to define an array with unspecified bounds and allow other arrays to be defined within a specified range. For example:

```
type INDEX is
  range 1..1000;   --type INDEX is
                       used to define
type VECTOR is
  array(INDEX) of INTEGER;   --array type
                               VECTOR with
                               unspecified bounds
U,V:VECTOR(1..20):   --20 elem vector objects
W:VECTOR(1..10);     --10 elem vector objects
[27]
```

Also permitted by Ada, are records having elements of varying size or type. They may also be defined dynamically, however, the size must be specified at record allocation time.

The last two types to be reviewed are access and private types. The access type is required in order to define a group of dynamic variables. The variables are generated internally and assigned internal names. Because these are not static variables, their existence begins at execution time and ends at the termination of the program.[29] The private type exists as a function of the information hiding feature of Ada. It allows only the name of the type to be accessible to the user. Only internal modules have knowledge of its properties. This is another way to ensure that external programs cannot corrupt local entities.[26]

CONCLUSION

The scope of this paper only allows a cursory look at some of the features of Ada which distinguish it from other higher-level languages. It is a complex language which is capable of a variety of applications. The fulfillment of the STEELMAN requirements appear to have created a language which will satisfy the broad spectrum of applications that exist at the Department of Defense.

The success of Ada relies ultimately on its acceptance by the programming community. The Department of Defense has expended a great deal of time and money in an effort to create a language that will meet its needs. In doing so, they have utilized worldwide resources throughout the computer industry. Participants have included, but were not limited to, universities, private industry, and US government, as well as foreign contributors. Private industry has been encouraged to participate in the process, since it will continue to be a major source of software. Currently, Zilog, Inc. and Litton Systems, Inc. are working on a program which will allow the Ada language to run on Zilog's 16-bit System 8000 and generate code for the 28000 microprocessor family.[27] As an indication of possible international acceptance, both the West German and British Ministries of Defense are developing Ada systems.

The fanfare with which Ada is being presented can be deceiving. Experience has shown that major changes are not quickly accepted. It is important to note, for instance, that the Department of Navy is not an enthusiastic supporter of Ada. One reason for this is that they have already committed themselves to another single language which meets their needs. In this case, the cost-effectiveness of any changes has to be fully considered before they will take place. This illustrates that Ada will not necessarily be accepted with open arms by all. Personal prejudices will have to be overcome.

Efforts are being made by the Department of Defense to combat these barriers. The importance of the Ada Support Environment has been recognized and the same amount of time and energy which has been poured into the language is being placed into its environment. The Department of Defense has realized that if the overall system is not compatible with the user, then the chance of success for the language is limited. In addition, a compiler validation program is being created in order to ensure that all compilers meet the rigid requirements. All proposed Ada compilers must pass this validation test before they will be accepted.

The language uses many software-engineering principles. The textual layout of the program units encourage modularity and top down design. They may be developed independently and then separately compiled. This is particularly useful in a team programming effort. After a common interface has been agreed upon, each programmer may develop, code, and compile his unit. The result would be an interacting program. The ability to hide information within a program unit is another important feature. The control over the access of a program's local variables and the implementation features for that control prevent both corruption of the variables and changes to the body of the program by external programs. Security of a program is also enhanced by the denial of access.

The language is problem-oriented. Various factors work together to shorten the distance between a program's conception and its implementation. This is especially true of the strong typing feature. A programmer must be aware of a variable's type at all times, because the compiler will flag type compatibility errors. Run-time errors may also be flagged because of the ability to place range constraints on variables. For example, if a number should fall out of bounds, the compiler will flag it.

Ada is a powerful language with the ability to perform complex mathematical functions. Current software-engineering principles have been utilized. Efficiency, readability, and maintainability of programs has been stressed. It is able to perform real-time and time-critical operations. Its abstraction facilities encourage the portability of programs. In essence, it fulfills the functional requirements of STEELMAN.

Ada's success, however, will be dependent upon the support environment. The beginning of this paper pointed out that various factors are considered when choosing a language. Most of them were related in some way to the training of the programmer. At present there is not a well-defined training program. The Ada Reference Manual is not adequate as a tutorial. It has been indicated that the Department of Defense is relying on the universities and private software firms to initiate teaching procedures. It is doubtful that this will be adequate.

The complexity of the Ada language is not the only thing working against it. Cost-effectiveness is an important factor. As was noted, the Navy is already firmly committed to its own standardized language. It has to be proved that Ada will be more effective for its purposes before any full commitment will be made. Another question arises when determining what costs are involved in implementing Ada and rewriting current software. The implementation schedule of Ada will affect how well Ada is received. A gradual introduction of Ada into the working environment may, in the long run, be more successful.

While Ada is a powerful language, it was written especially for embedded computer systems and is not all-encompassing. There will be instances where Ada will not be the best language for a given application. Judgments will still have to be made as to which language should be used. If the Ada language is successfully implemented, however, it will cut down on the proliferation of languages in the Department of Defense and, subsequently, cut down software costs.

Footnotes

1. Welke, Larry. "The Origins of Software." Datamation, Vol. 26, No. 12, December 1980. pp. 127-130.
2. Osterweil, Leon J. and Richard N. Taylor. "Notes on Software Engineering." (Lecture). Dept of Computer Science, University of Colorado, Boulder, Colorado.
3. Edwards, Perry and Bruce Broadwell. Computers in Action: Data Processing. California: Wadsworth. 1979.
4. Carlson, William E. "Ada: A Promising Beginning." Computer, Vol. 14, No. 6, June 1981, pp. 13-14.
5. Carlson, p. 14.
6. Buxton, John N. and Larry E. Druffel. "Requirments for an Ada Programming Support Environment: Rationale for STONEMAN." The IEEE Computer Society's 4th Annual International Computer Software and Applications Conference, 1980, p. 66.
7. Reference Manual for the Ada Programming Language. United States Department of Defense. July 1980.
8. Cohen, Paul M. "Ada: A Language and a Concept." 20th Annual Technical Symposium of the Washington, D.C. Chapter of the ACM Proceedings, June 1981, p. 59.
9. Hurwitz, Judith and Paul Kinnucan. "Ada Compiler Programs Get Under Way at Pentagon." MiniMicro Systems, Vol. 12, No. 12, December 1979, p. 35.
10. Cohen, p. 59.
11. Hurwitz, p. 35.
12. Cohen, p. 59.
13. Ibid.
14. Hurwitz, p. 35.
15. Cohen, p. 59.
16. Ibid.
17. Reference Manual.
18. Wegner, Peter. "Programming with Ada: An Introduction by Means of Graduated Examples." SIGPLAN Notices, Vol. 14, No. 12, December 1979, p. 15.
19. Reference Manual, p. D-2.
20. Reference Manual; Wegner, Peter. "A Self-Assessment Procedure Dealing with the Programming Language Ada." Communications of the ACM, Vol. 24, No. 10, October, pp. 647-677; _____, Programming with Ada, pp. 1-46.
21. Wegner, A Self-Assessment Procedure, pp. 670-671.
22. Reference Manual, p. D-3.
23. Evantoff, William, Gary Anderson, Ronald Price, and Irving Rabinowitz. "Ada: A Significant Software-engineering Tool." Mini-Micro Systems, Vol. 14, No. 4, April 1981, p. 224.
24. Wegner, Programming with Ada, p. 11.
25. Ichbian, J.D., J.C. Heliard, D. Roubine, J.G.P. Barnes, B. Krieg-Brueckner, and B.A. Wichmann. "Rationale for the Design of the Ada Programming Language." ACM Sigplan Notices, Vol. 14, No. 6, June 1979, pp. 6-1 - 6-10.
26. Reference Manual; Wegner, A Self Assessment Procedure, pp. 647-677. Wegner, Programming with Ada, pp. 1-46.

MANAGING OUR SYSTEMS FOR PERFORMANCE:

ARE WE GETTING WHAT WE DESERVE? (U)

by [   ] T3

P.L. 86-36

by [   ] T3

P.L. 86-36

Psychologists tell us that students who are graded perform better than students who are not graded. While it is not generally realized, the same holds true for computer systems: measuring and reporting their performance induces better performance--not because computers are somehow people-like, but because their managers are.

Yet over the years, the benefits of grading computers--of formulating relevant processing objectives and then scoring system performance against these objectives--have been largely overlooked.

In an attempt to promote objective-oriented performance management practices across our large installations and subsystems, this paper takes a twofold approach:

[] A strong case is presented for the implementation of installation and subsystem performance reporting which is effective, reporting which accurately portrays the ability of the installation to satisfy customer needs and which is sufficiently comprehensive to aid in the ongoing management and administration of the installation.

[] Within the framework of a large, general-purpose Agency installation, the mechanics and pragmatics of implementation are discussed and demonstrated.

Over and above informing and explaining, this paper seeks to move Agency thinking--to persuade and convince Agency managers and implementors that performance reporting can provide a sizable payoff and that the obstacles to reporting performance within a complex processing environment can be surmounted.

MEASURES OF PERFORMANCE

Any serious attempt to report or even discuss system performance requires that the basic components of performance be separated out into measurable dimensions.

[] Availability is normally expressed as percent of a specified time period during which the system was available to the customer. Since the "system" may actually represent the intersection of several subsystems, the determining and reporting of availability in a way that is relevant to the customer is essential.

[] Responsiveness, the time taken by the system to perform a service, is best measured over groupings of like transactions. Typical metrics of responsiveness include:
● average response time per Class Z transaction,
● percent of Class A jobs completed within Z minutes.

[] This writer has found it exceedingly useful to report on a hybrid performance dimension, dependability, representing the intersection of availability and responsiveness: percent of time during which the system was up and reasonably responsive.

[] Productivity describes the amount of product processed through the system over a time period of interest and is normally expressed in terms such as:
★ transactions per hour,
★ jobs per shift.

[] Reporting the utilization of installation resources and devices fulfills dual purposes:
■ Breaking down all work transactions performed by the system into meaningful work categories or "workloads," and then describing the consumption of

installation resources in terms of
these workloads, enables the alloca-
tion of resources and the relative
cost of each workload to be reviewed.
■ The aggregate utilization of installa-
tion resources across all workloads
gives an indication of reserve capa-
city to support crisis loads in the
present and expanded services in the
future.

ENSURING PERFORMANCE REPORTS
WHICH ARE RELEVANT

Most assuredly, the concept of reporting
performance is not a new one. But a survey
of our large installations and subsystems
would reveal that in many cases, performance
reports are conscientiously generated but rou-
tinely ignored--because management finds them
irrelevant. For other installations, perfor-
mance reporting is just not practiced, perhaps
because a past history of irrelevant perfor-
mance reporting has cast the concept into
benign neglect, or even disrepute. Perfor-
mance reports fail to be relevant when they
are inconsistent with customer perceptions;
those which fail to explain or account for
irate customers are just not worth the bother
of reading. Performance reports achieve
relevance when they:
[]   focus on meaningful customer objectives;
     and
[]   report on the system's degree of success
     in attaining them.

Customer objectives may be estimated or
negotiated; but no matter how they are arrived
at, they should represent real and perceived
customer needs. Specifically, customer objec-
tives:
    should address activities and services
    of concern to a large cross section of
    customers...

    over time periods deemed critical by the
    customer (for example, the prime
    shift)...

    in terms of those performance dimensions
    which the customer deems critical.

EXAMPLES

System Availability over Prime Shift > 95%
Average Response Time of Transaction < 5 sec.
Percent Class A Jobs Completed in 10 min > 80%

The successful identification of customer
objectives virtually guarantees relevant
reporting--reporting which documents user
satisfactions and dissatisfactions, suggesting
the cause of the latter or at least hinting at
its origins.

ENSURING RELEVANT REPORTS GET READ

As a rookie journalist quickly learns, an
interesting and relevant story is apt to go
unread unless it has been headlined and organ-
ized to attract and retain the attention of
the reader. For just this reason, the daily
performance report should resemble stylisti-
cally a newspaper article, telescoping the
data from most important to least important,
from summarizations to specifics.

Analogous to the headline is an executive
shift summary, scoring total installation per-
formance against objectives and rendering a
rapid determination as to whether and where
attention (and further reading) is required.

Intermediate level report segments should
break down shift productivity and utilization
into categories that are meaningful to manage-
ment. Lower level report segments should
refine problem areas both in detail and by
time slice, hinting at suspected problem
areas.

Processing exceptions should conclude the
report, with inordinate wait times noted and
inordinate resource consumption--sometimes the
cause of systemwide degradation--attributed.

BENEFITS OF EFFECTIVE PERFORMANCE REPORTING

Routine performance reporting which is:

[]   directed at relevant processing objec-
     tives,
[]   sufficiently comprehensive to support
     troubleshooting, and
[]   organized so as to attract and not deter
     the reader,

offers many benefits to installation manage-
ment, tuners, planners, and operators. Yes,
operators!

The degree to which an installation is ful-
filling quantifiable processing objectives
becomes a source of pride, or concern, to its
staff. Concern leads to concerted action.
But pride or concern, when commonly shared,
enhances the esprit de corps of an organiza-
tion.

Performance reporting exposes the existence
of performance bottlenecks which, once
uncovered, are not apt to be ignored.

Performance reporting suggests focal points
for system tuning, and ultimately enables sys-
tems personnel to determine if their tuning
efforts have improved (or worsened) the per-
formance of the system.

The reporting of performance exceptions
enables installation personnel to spot
resource consumption "hogs" and to take
appropriate action. (An offending user is
often unaware of his culpability and grateful
to learn what he can do to improve his ser-
vice.)

Performance reporting enables management to
determine if the allocation of installation
resources among the different work classes is
as intended and to determine if priority ser-
vice is being accorded to priority work
classes.

Over the long run, a continuing history of
performance reporting enables installation
managers and planners to track growth by work-
load and to anticipate future capacity needs
on the basis of continuing trends.

In summary, performance reporting enables
management at all levels to accurately assess
the quality and quantity of service being
delivered, to determine if processing objec-
tives are being met, to evaluate the results
of intended corrective action, and to continu-
ally gauge reserve capacity with an eye to the
future.

A CASE IN POINT: CARILLON

High-sounding and idyllic as they may
sound, these concepts can be put into prac-
tice, even in a very complex environment. As
a case in point, we will focus upon CARILLON,
an Agency installation whose size and complex-
ity ambivalently discourage and demand the
management control which only effective per-
formance reporting can provide.
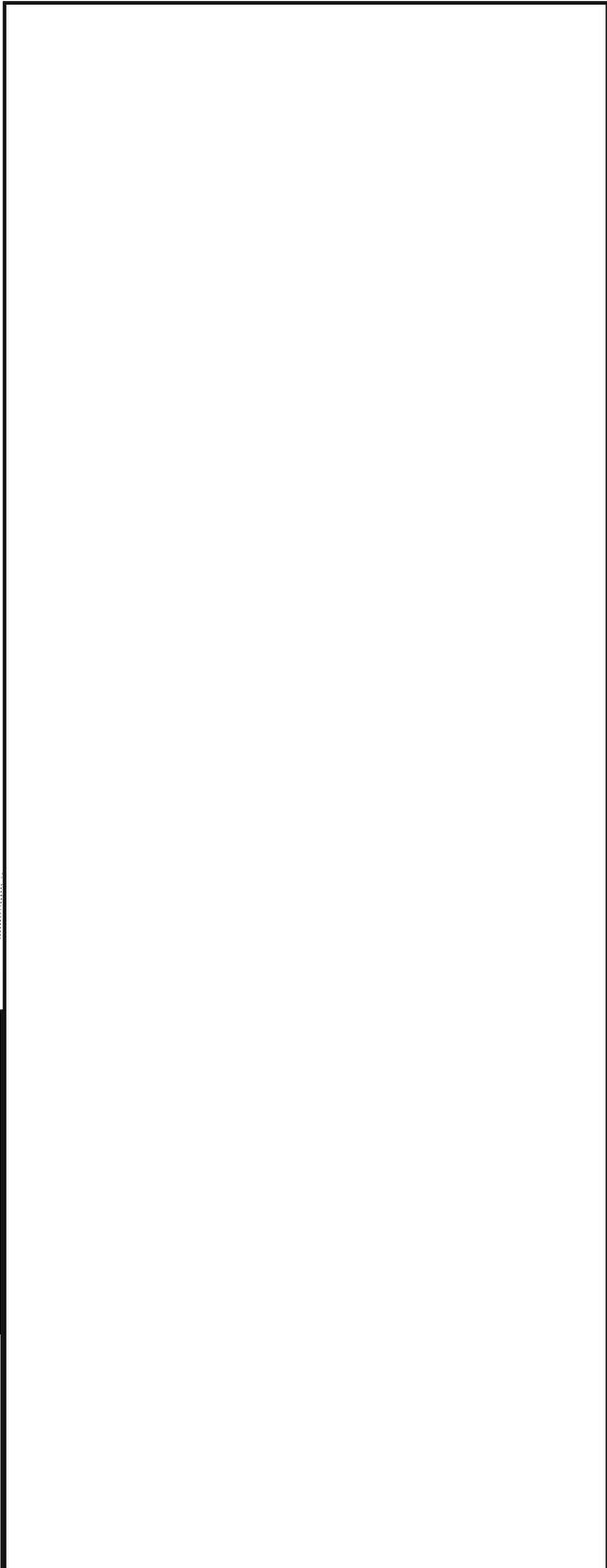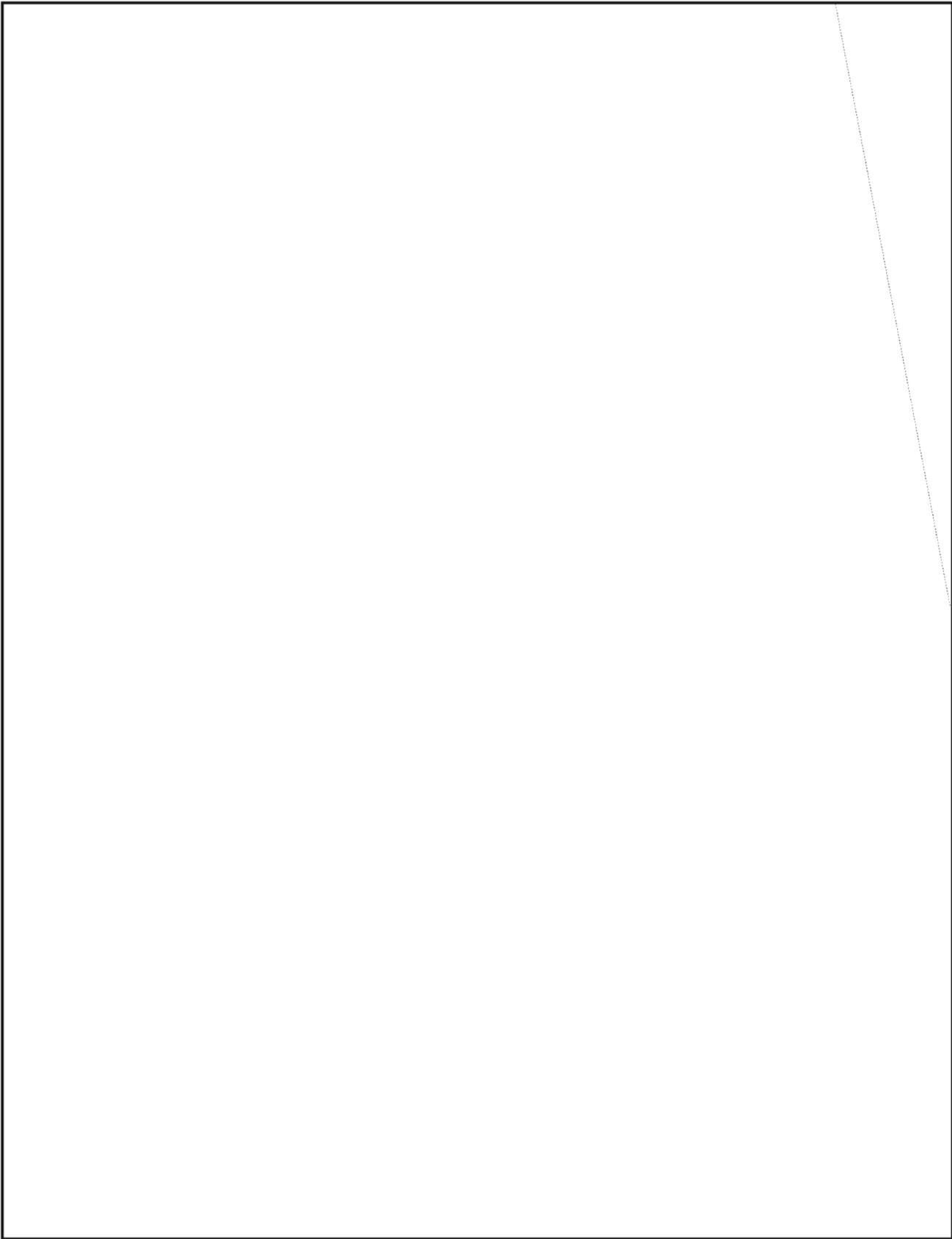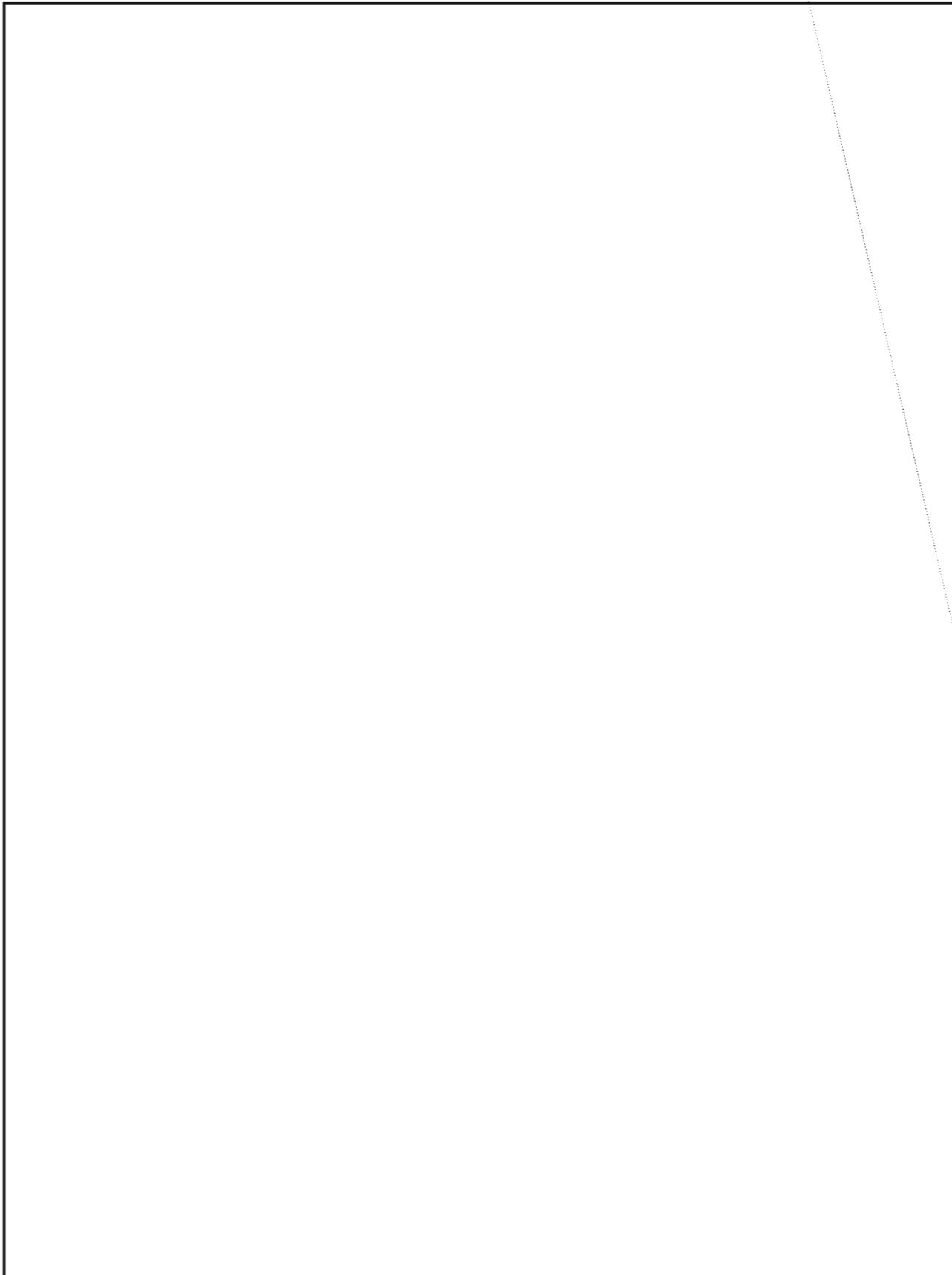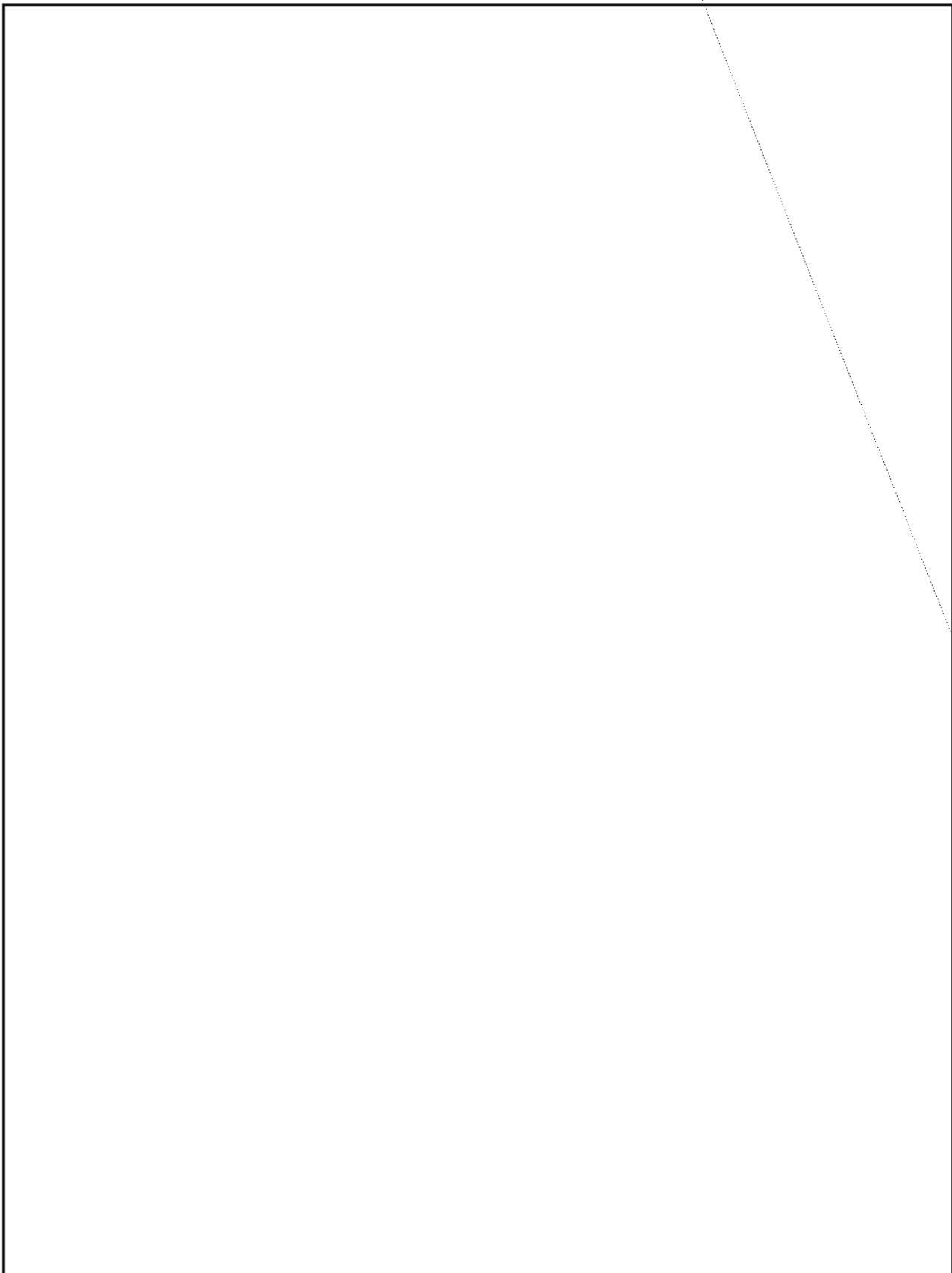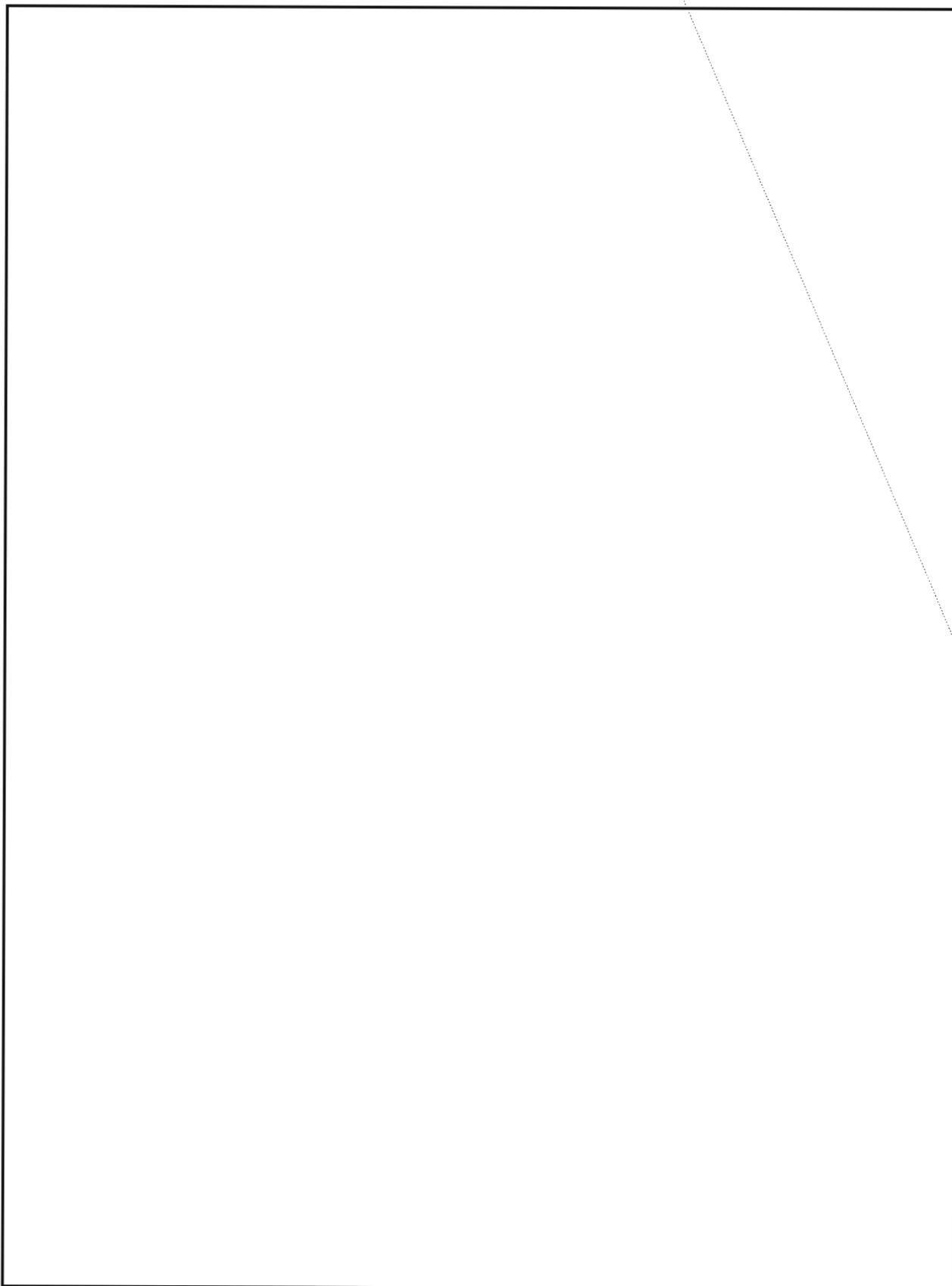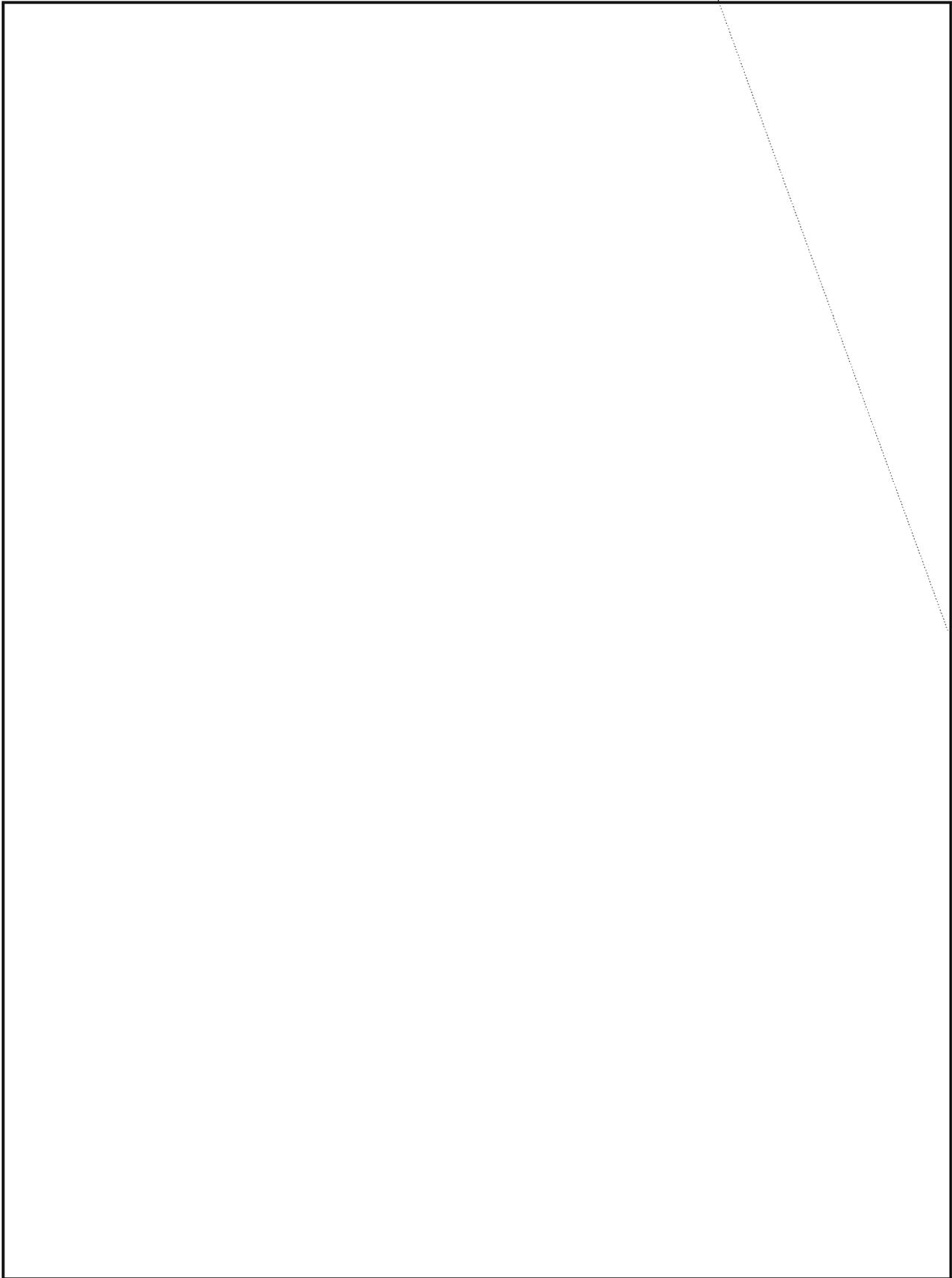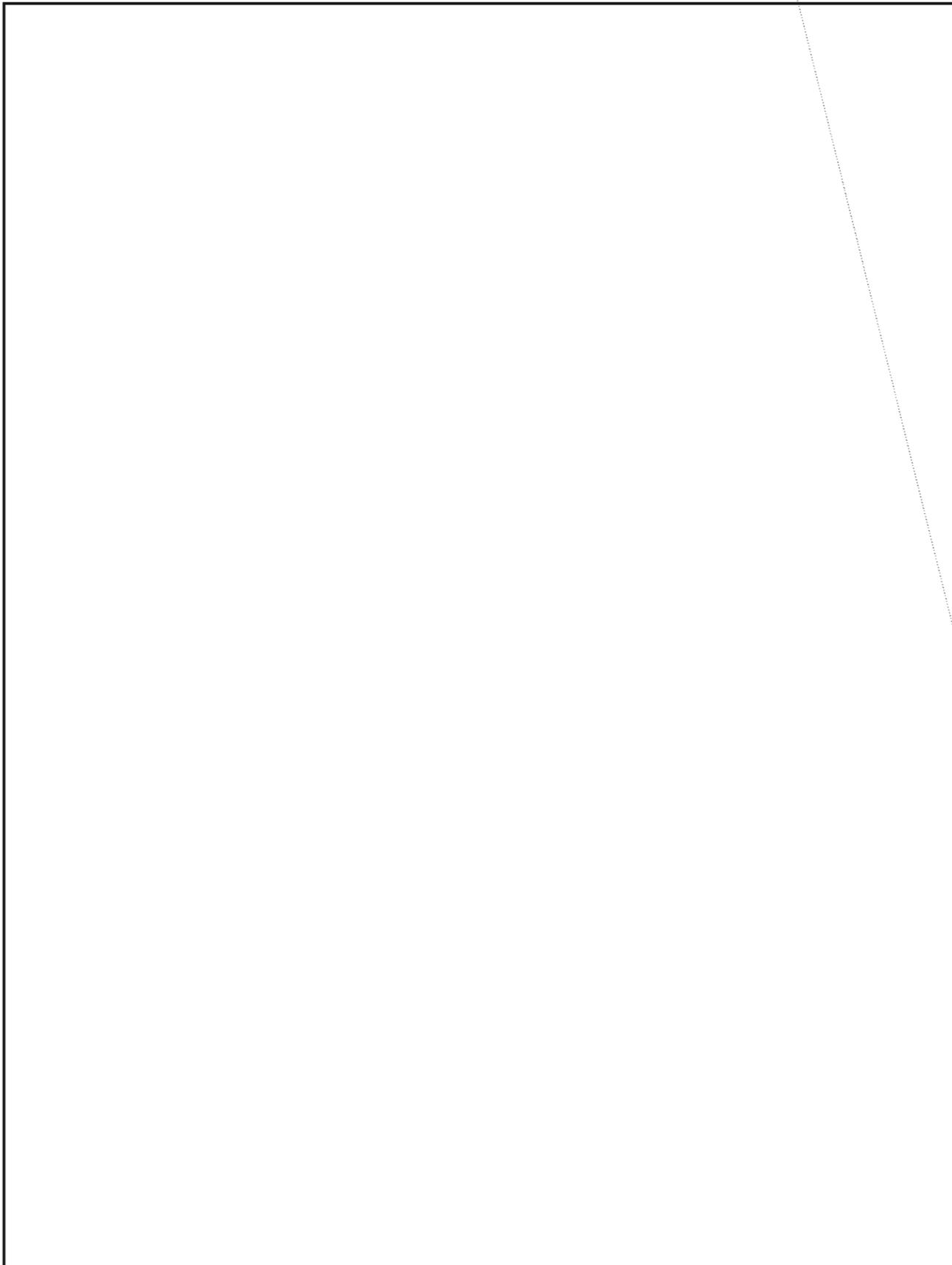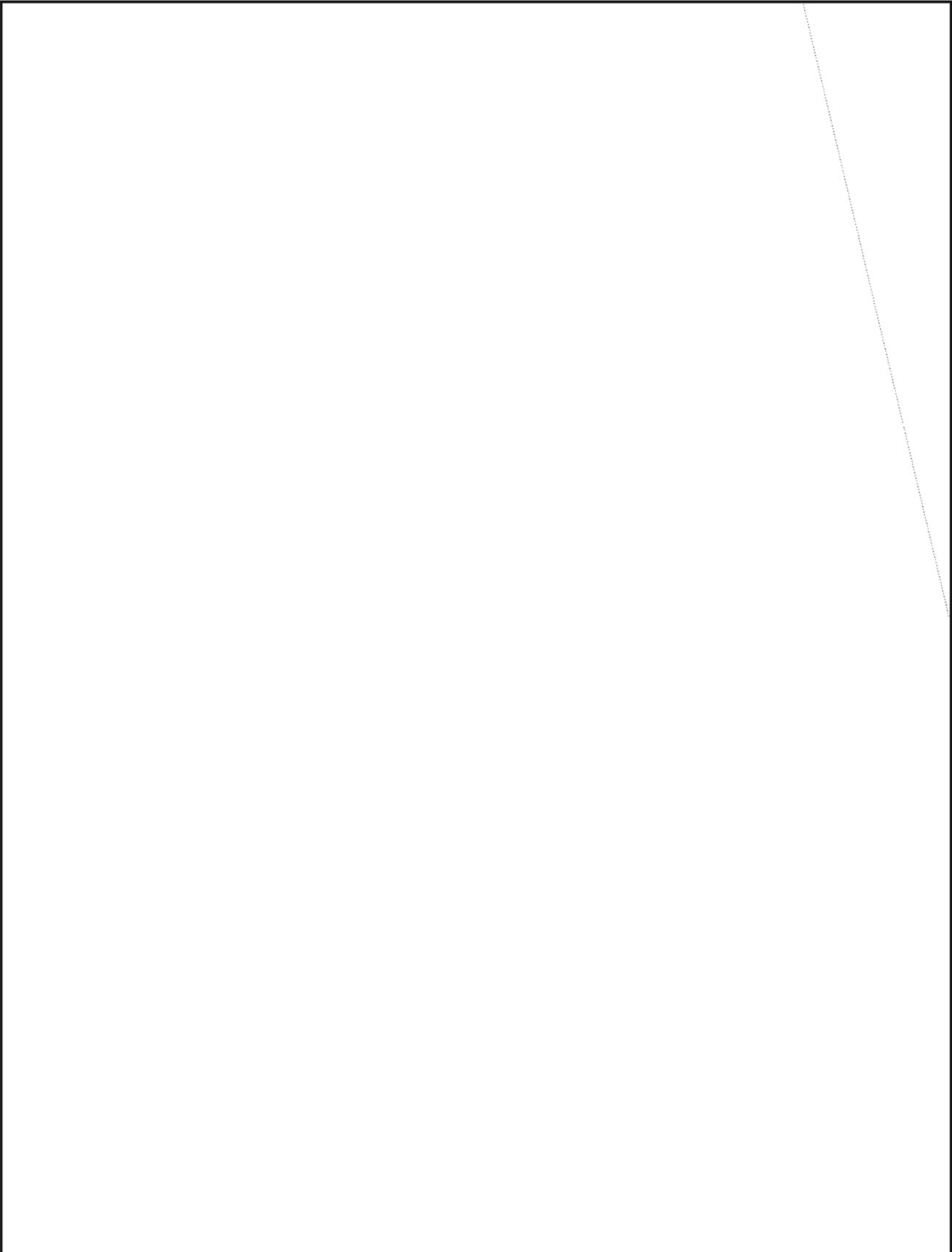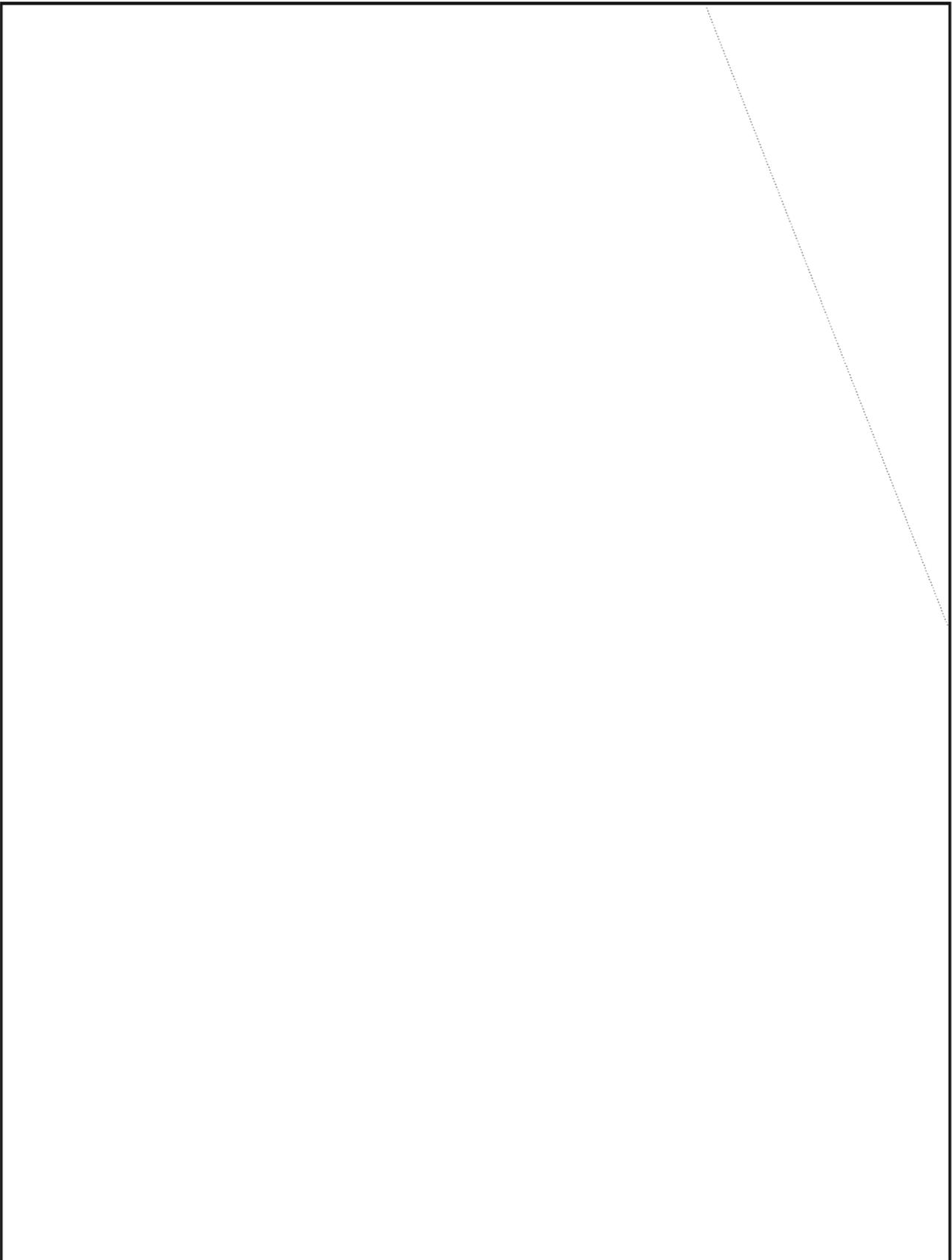
P.L. 86-36

P.L. 86-36

P.L. 86-36

GETTING

PERSONAL (U)

P.L. 86-36

by [   ] T52
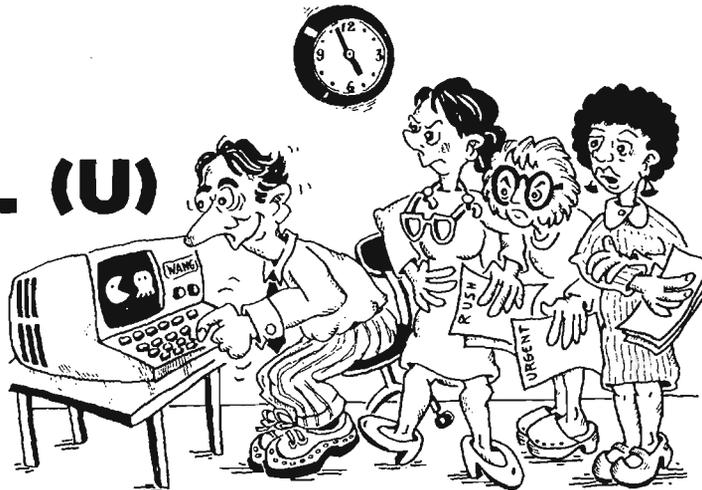
(U) Today, with the information explosion fully underway, information managers around the world find themselves trying to cope. NSA is no exception. And as technology marches on, NSA analysts (or "users," as we call them) cannot always keep pace. While discussing how new technology can help NSA manage its information resources, this paper will also examine the human factor: how people fit into the automated information systems picture.

(FOUO) Information--its acquisition, processing, dissemination, storage, and retrieval --permeates every phase of NSA operations. The majority of information handling, including research, takes place in the Agency's T5 (Information Resources Management) organization. Here, the battle with the information glut is waged daily.

(FOUO) As the need for fast, precise, and complete information has increased, T5 has begun to rely more and more on the speed and accuracy of commercially available information services. Designed for every phase of the information process, (from acquisitions and cataloging to electronic mail and report generation), the automated information system is billed as this era's panacea. The most popularly subscribed-to elixir today is the on-line information data base. Again, NSA is no exception.

(FOUO) In its role as Agency information manager, T5 maintains contracts with a myriad of these corporations. Central Research (T5211) alone, where the majority of NSA's

on-line information searching takes place, subscribes to five commercial data bases: Lockheed's DIALOG, System Development Corporation's (SDC) ORBIT, the subsystems of the New York Times Information System (NYTIS), the Bibliographic Retrieval Service (BRS), and Mead Data's NEXIS.

(U) Simply stated, the hosts for these multi-file information systems are accessed through leased hardware (terminal/printers), via commercial nonsecure telecommunications networks. Search results are retrieved by keyword and field strings in the appropriate query language, connected with Boolean operators. Results, most often in the form of bibliographic citations and/or abstracts, can be scanned and printed on-line at 1,200-baud rates, or can be ordered off-line for receipt by mail in 10-12 days. Beyond start-up and yearly contract fees, costs include per-hour connect rates for the telecommunications networks and the files, and range from $15.00 to $450.00 per hour, depending on the file accessed.

(U) At first glance these costs may seem high, but when balanced against the savings in person-hours for processing and scanning, storage space, and subscription fees, the commercial systems are actually cost-efficient. DIALOG, with its more than 170 separate information files, is a good example of this efficiency. Why subscribe to, process, store, and scan through the more than 750,000 sources that DIALOG indexes when you can pay a reasonable fee to access just what you need when you need it?

(U) But these systems are already "old hat." New technology can make what once seemed a revolutionary way of coping with the information explosion seem passe. Applying microcomputers (also referred to as personal computers) to these commercial information systems can greatly increase the speed, accuracy, and usefulness of their services.

(U) The most common application of a microcomputer to the commercial data bases is for reducing a four-minute or more, six-line log-on procedure to a one-keystroke stored operation. This can prevent the personal frustration that often develops when a seemingly simple log-on process turns into an aggravating 20-minute tug-of-war with the system. (That makes the research to follow begin with an already annoyed researcher: a volatile and often disastrous combination!) Since the searcher is already connected to the communications network when he/she is attempting to log on, the time spent is costing money. With the attachment of a personal computer to speed and ease the log-on process, cost efficiency results and mental stability is preserved.

(U) Another practical use for the personal computer attached to the commercial data bases is for storing frequently-run queries. Although the systems already have a query-saving (by number) capability available for a small monthly fee, there is no provision for cross-saving a search from one system to another. The query would have to be saved under a different number in each system checked. The personal computer could "can" the query and dump it into a file that could be consulted quickly before a query is run to make sure it hasn't already been searched on the systems. With some sophisticated programming, the personal computer could translate a query saved in the language of one system into that of the others and run the search against them as well. It could even be instructed to "weed out" the duplicate citations after they are retrieved.

(FOUO) Saving an NSA item of interest in our own terminal instead of the system sponsor's also seems a bit more private. Although all information on these commercial data bases is unclassified, it is wise not to call undue attention to our topics of interest by storing them out in plain view, under our sponsor-provided password.

(FOUO) The personal computer could also be programmed to remind the analyst when an update of a saved search should be run, to whom it should be sent when completed, etc.

When electronic mail becomes a more widespread reality at NSA, the personal computer could be networked to other microcomputers in other organizations. Search results could be delivered without hardcopy printing ever taking place.

(U) Most microcomputers on the market today can be programmed in a variety of languages, including BASIC, POGOL, PASCAL, PILOT, and LOGO. Many come with basic programs preset on floppy disks or offer "how-to" floppy diskettes (also called "floppies") that instruct the user in programming his/her own personal computer.

(U) The microcomputer can offer the professional information manager/searcher the opportunity to somewhat "polish" the product before supplying the retrieved information to the client. Headers and comments could be added, duplicates or "false drops" could be removed, typos corrected. In striving to fulfill a mission of providing the most accurate information available in a reasonable period of time and in its most usable form, editing capabilities like these could aid the information analyst.

(U) The hottest use yet, however, for microcomputers attached to commercial information data bases, is data downloading. "Downloading" is a process by which information being retrieved from one system is written onto a medium (most likely diskettes) of an attached microcomputer for future retrieval, editing, or storage. Once this data is written onto the disks, there is really no need to ever pay for retrieving that same information again. When a similar request is received later, the information is provided to the requestor without the costs incurred by rerunning a search.

(U) The diskette can be labeled and filed, or be indexed in a directory on the personal computer for future use. Floppies can be written over when their content has served its purpose. They have a long life, when stored properly, but their file integrity is somewhat susceptible to physical maladies such as scratches.

(U) Requestors could begin to scan their own search results instead of having the knowledgeable, but not omniscient, information specialist determine which retrieved items are pertinent and which are not. The client can determine which items he/she would like to have and print them out, or in the day of

electronic mail, receive, scan, and store the pertinent information in soft form at his/her own desktop terminal.

(FOUO) Audit and security trails are always of interest in our line of work. The microcomputer can record the various use and performance statistics monitored for these on-line data bases. Central Researchers currently record manually their on-line time and charges, summaries of requests, communications network accessed, and results on a systems logsheet. The microcomputer could easily monitor all this information automatically and also provide monthly statistical summaries, including audit trails and user performance charts. On some models, packages for producing color-coded graphics for this type of information are available. These statistics could then be employed in identifying problem areas, searchers who may need extra training, billing irregularities, and the general effectiveness of the information tool.

(FOUO) The microcomputer could contain the on-line tutorials crucial to properly accessing these systems with a minimum of aggravation. With more than 10 query languages to learn and employ in Central Research, the simplest of procedures can often become confusing! The microcomputer could provide logically organized menu screens and step-by-step instructions for the various systems. The searcher could refer to these instructions as he/she performed the search on-line. On-line time spent floundering around in "trial-and-error-land" is expensive and could be alleviated with this capability. No new technology is really helpful unless people can use it.

(U) The human factor—how people fit into this picture of automated information management—is and should remain a concern of systems designers and consumers. With much of the work force still experiencing "Technofear" (also known as "Technophobia," fear of technology), "Technomaniacs" must beware of so complicating an information specialist's tools that the information research process becomes the search for the Holy Grail!

(U) The increasingly frequent appearance of on-line tutorials, "help" screens, and logically layered menu screens indicates that some designers are already paying attention to the human needs of their consumers. The demands of executives, who do more and more of their own research and on-line report generation, were for easier-to-use working aids. In response, designers are moving away from voluminous technical hardcopy "how-to" system

manuals full of coded fields and technical lingo to "English-y" user-oriented on-line instructions and brightly-colored, almost tempting, keyboards.

(U) The "user-friendly" concept, an integral aspect of any attempt towards creating a healthy person-machine relationship, is well worth the effort devoted to countless studies on the subject. These studies show people much more inclined to use (and, as studies prove, use correctly) a system that talks to them, as if in conversation. The concept of on-line (usually an interactive session, like volleying in a tennis game) means that both participants (the user and the host computer) must do their part. Chances are that if a user is uncomfortable with a language or terminal, an input error will occur, and the host computer (sympathetic, but without human compassion) will reject the input. The result: user frustration and incomplete research. On-line tutorials and working aids provide the user with a convenient, quick place to look for the proper formats and phrases. Since logging off to look for instruction in some manual is impractical and rarely done, the on-line aid can solve many problems before they have the chance to develop.

(U) A correctly chosen personal computer can become almost a friend or interpreter to the on-line searcher. The importance of the proper training in how to use the microcomputer and concentration on what it can do to help (not usurp) the role of the information specialist should be emphasized when introducing users to new technology. Personal computers should be chosen with a number of things in mind, including: design and capabilities, naturally; compatibility with other systems; the selection and functions of its peripherals; "creature comforts"; and costs. Will information managers be inclined to sit at this terminal and use it for that purpose for which it was purchased? Or will they be so intimidated by the looks of it alone that they will avoid it like the plague or pray for an early retirement? Is it of a comfortable height? Is there ample legroom? Is the keyboard well-labeled? Is the color-type contrasted comfortably with the background screen? Is the screen designed to tilt at a natural reading angle? Is it noisy? Are the diskettes easy to load and remove? Is there room for growth? Will it still be cost-efficient in five years? Are there peripherals that can be purchased later to expand existing functions? These questions and many more must be asked before choosing a personal computer that many different people, including Technophobics, will have to access daily.

(U) Unfortunately, one of the reasons these human factors are often overlooked is because those who determine which equipment to procure are most likely neither Technomaniacs nor Technophobics--they are most often Bureaucrats. But a personal computer, or any technological device, chosen at least partially with its users in mind, can aid the employee in accomplishing his/her tasks, increasing productivity and success, and developing a healthy person-machine relationship.

(U) This paper has stressed both cost efficiency and the human factor in applying personal computers to the commercial data bases at NSA. Perhaps we should also take a look at the initial financial investment required to obtain and implement these ideas.

(U) Costs for the average microcomputer (see charts below) ranges from $600.00 to well over $5,000.00, depending on the capabilities desired in both the software and the hardware. With each personal computer marketed, compatible software packages are also sold at additional costs. Peripherals like CRT's, printers, modems, disk drives, and Tempest packages also add up. Adequate supplies of floppies for stored data and preprogrammed disks are reasonably inexpensive, but they are still an additional cost. It is safe to say, however, that with the proper forethought and investigation (including observing how these microcomputers perform in actual workcenter settings), and by considering the uses for and the users of the proposed microcomputer, the initial investment could easily repay itself within a 1-to-3-year period. Applying carefully selected and well designed technology to prevent waste, increase productivity, and improve morale is almost always cost-efficient.

FEATURES	COMMODORE 64	APPLE II+*	IBM* PC	TANDY TRS-80* III	ATARI 800*
Base Price*	$595	$1530	$1565	$999	$899
Advanced Personal Computer Features					
Built-in User Memory	64K	48K	16K	16K	16K
Programmable	YES	YES	YES	YES	YES
Real Typewriter Keyboard	YES (66 keys)	YES (52 keys)	YES (83 keys)	YES (65 keys)	YES (61 keys)
Graphics Characters (from Keyboard)	YES	NO	NO	NO	YES
Upper and Lower Case Letters	YES	Upper Only	YES	YES	YES
5¼" Disk Capacity Per Drive	170K	143K	160K	178K	96K
Audio Features					
Sound Generator	YES	YES	YES	NO	YES
Music Synthesizer	YES	NO	NO	NO	NO
Hi-Fi Output	YES	NO	NO	NO	YES
Video Features					
TV Output	YES	EXTRA	EXTRA	NO	YES
Input/Output Features					
"Smart" Peripherals	YES	NO	NO	NO	YES
Software Features					
CP/M* Option (Over 1,000 Packages)	YES	YES	YES	YES	NO

MAKE & MODEL	Victor 9000	IBM PC	Xerox 820	Apple III	Radio Shack TRS80 Model II
Processor Type	8088	8088	Z80A	6502	Z80A
Word Length	16 bits	16 bits	8 bits	8 bits	8 bits
Memory Size (Internal)	128-896KB	16-256KB	64KB	96-256KB	32-64KB
Storage Capacity on 2 Floppies	2400KB (5¼")	640KB (5¼")	160KB (5¼")	280KB (5¼")	960KB (8")
CRT Display Standard Format	80 x 25	80 x 25	80 x 24	80 x 24	80 x 24
Alternate Format	132 x 50	None	None	None	None
Graphics Resolution	800 x 400	640 x 200	None	560 x 192	None
Communications Built-in Serial Ports at no extra cost	2	0	2	1	2
Built-in Parallel Ports at no extra cost	1	0	2	0	1
Human Factors Keys on Keyboards	94-104	83	96	74	76
Detached Keyboard mechanism	Yes	Yes	Yes	No	Yes
Tilting Display mechanism	Yes	No	No	No	No
Swiveling Display	Yes	No	No	No	No
Desk Area Required (Approx. Square in. with 2 floppy disks)	310	420	470	381	500
Operating System Supplied Standard	CP/M-86* MS-DOS	None	None	Apple DOS	TRS DOS

NOTE: Chart based on manufacturer's information available as of April 4, 1982.
*CP/M is a registered trademark of Digital Research, Inc.

May I suggest...?
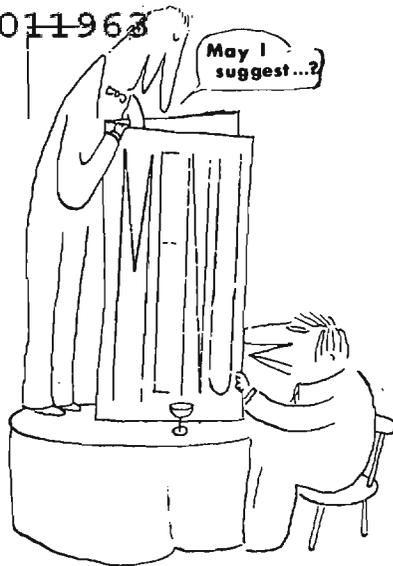
Menu Selection as a Tool

for Human/Machine Interaction (U)

by [   ] E53

During the past decade, system designers and applications programmers have come to realize that the most unpredictable component of an (U) interactive program in terms of performance is the user interface. By its very nature, an interactive program must deal with many factors that need not be considered when developing a conventional non-interactive program. Primarily, the applications programmer must consider the "human factors" aspects of the program-user interaction, since the success or failure of an interactive program depends mainly on its ease of use [1]. Because the user interface design has such a strong impact on program acceptability and usefulness, it becomes very important to design good interactive user interfaces. One technique being used more frequently to promote good human/machine communications is menu selection.

(U) A menu consists of a list of items, usually displayed along the screen border, from which a user makes a selection. These items can be represented numerically, graphically, or by text. In each case, the user can choose an item using one of several techniques, e.g., pushing a numbered key corresponding to the number of the choice or positioning a light pen over the choice as displayed. There are other ways of interacting with a computer display: command languages, function buttons, and transaction codes. However, these methods do not provide an interactive interface for the naive user which is as user-friendly as menu selection. By definition, a naive or casual user is a person who has little, if any, training in computer science and is unfamiliar with the internal workings of the computer. This person's job description does not mandate the use of the computer. However, if the computer can get the job done better and faster,

without losing data and can be operated without extensive training, the casual user will be more likely to utilize this resource [2].

(U) A well-written menu utility is the perfect interactive tool for both the naive and sophisticated user. With a menu, the user does not have to remember commands or syntax and is less likely to make errors with all the valid choices displayed. Since making a selection can be as simple as pushing a button, the amount of time spent typing commands and studying documentation is reduced. In addition, a selection menu provides more flexibility than fixed keys when changing or updating functions. Although a well-written menu utility provides both types of users with a good interface to the computer, a poorly designed utility can have the opposite effect. A poorly designed menu utility could force the user to page through a large number of levels to perform an operation. This is annoying, especially when the response time between selections is very slow. A system which provides no feedback to the user that selections are being received and processed is another example of poor design. The next section of this paper will focus on the features necessary to design and implement a menu utility which provides a good user interface.

MENU SELECTION IMPLEMENTATIONS

(U) A well-written menu utility offers both the sophisticated and naive user a good way to interface with an interactive graphics system. Menus usually consist of a number of choices listed for the user from which he or she may choose an operation or command to be performed. This type of selection may be implemented very simply by typing the desired

response on a keyboard or by selecting the desired option via touch panel, light pen, mouse, or any other interactive input device. The options may be presented in a list across the bottom of the screen or along one side, or they may be represented graphically as icons. (An icon is a pictorial symbol used to represent an idea or entity.) The menu may only be displayed at the user's request, in which case a movable menu could appear close to the cursor to minimize movement.

(U) In more advanced systems, such as SMALLTALK [3], the screen may be divided into several windows, each with its own menu. Only one window at a time can be interacted with; it is indicated as active and in the foreground. A different window can come into the foreground by activating its menu. Thus, by designing a system which simulates how someone might work at a desk with various papers spread out on top of it, an interface is created which is close to the actual working conditions of the user. By choosing from different menus, the user can manage several simultaneous tasks and contexts, alternating between them at his or her convenience. Though menu selection may be implemented using many different methods and formats, the advantages offered by a menu utility are similar from one implementation to another.

MENU SELECTION : ADVANTAGES

(U) Although a good menu utility offers many advantages as a user interface, perhaps the most important is that little training is required of the user. In all of the graphic systems investigated and reported in a later section of this paper, the user was found to be a professional in a non-computer-related career field. A menu utility could save the casual user from extensive training time by presenting a range of alternatives from which to choose. Since only valid choices are included in the menu, the user is protected from making an invalid selection. This may also serve as a security feature if the user is shown only the names of operations he or she is authorized to access.

(U) Because the user is prompted with a repertoire of commands, he or she is relieved of the burden of remembering command names. By giving the user the ability to point to material currently displayed, this material can be treated as input, drastically reducing the amount of typing to be done, saving time and preventing errors [4]. There are only a limited number of choices in a menu, so that

designing effective user aids such as a "HELP" key should be easier than when the user has an unlimited number of options. The programmer should have less complicated error handling and documentation to write than with a system which requires the user to know what each command does and when to type it [5]. A menu utility is generally more flexible with changes in the application or requirements, since it is easier to change or add menu items than it is to change or add new function keys. For the many reasons cited above, menu selection is becoming a popular tool for those tasked with designing interactive systems.

MENU SELECTION : FEATURES

(U) Once the decision has been made to write a menu utility, the designer must be aware of the many features it should include to be a successful tool. While some of these features were promoted by most of the listed references, the usefulness of others may be more dependent on the particular application. Generally it was felt that menus should always be numbered starting with one instead of zero, since that is how people begin to count. The menu should be kept short using submenus if necessary. However, these submenus should not be nested more than three or four levels deep. An option for longer menus that cannot fit on one screen is a scrolling mechanism to allow the user to browse through the menu which has choices listed alphabetically.

(U) A very important feature is the use of feedback to provide reassurance to the user that the selected operation is acceptable and is being processed. The user should be able to respond as soon as the choice appears and should be informed if the system is slow and the process will take a long time. According to Miller [6], both experienced and novice users may tend to become bored and frustrated if an interactive system takes more than 15 seconds to respond to their query or command. Schneiderman agrees that long delays are usually disruptive and disturbing, but he says that "the variance of response time may be as critical as the mean response time." [7] He suggests that performance and satisfaction may improve if responses are delayed to minimize variance and also suggests informing users of the estimated waiting time as another tactic. Another way of giving feedback to the user is by highlighting or increasing the brightness of the menu item chosen [8]. Feedback could further be provided by blinking the item on and off, drawing a box around it, or changing colors to indicate an "ON" state [9].

(U) Another important feature of a menu utility is the provision of special keys or functions to aid the user. Martin says that a "help" key or option can be a psychologically valuable device to minimize the potential confusion and hostility of the unskilled operator [10]. Once the user chooses the "HELP" option, a "HELP" conversation could begin with a menu screen giving possible categories where help is provided. After being helped, the user must have access to a "CONTINUE" key or function to return the system to its point of interruption. The casual user may feel less intimidated in using the system if a key is provided which will "undo" any operation he or she has already started. Having a "modify" feature to correct any mistakes in building a picture might also be convenient for the user. In a system described by Teitelman at the Xerox Palo Alto Research Center, if the interactive input device button on the mouse is held down instead of lightly pressed, the system is instructed to tell the user what it "would" do if the chosen operation were actually performed [11]. Being able to see the result of an operation before it is actually performed would save the user time in correcting errors and in redoing the operation.

(U) The user should always be provided with the opportunity to exit the menu sequence at any point and return to previous menus. Being able to continue a computation following an error would be useful when the error occurs following a significant amount of computation. This facility would be essential for good interactive debugging, although not all graphics applications would require a tremendous amount of computation. In designing any features for the users' convenience, it is important for the designer to remember that there should be consistency of choices to minimize the users' difficulty in learning to operate the system. A "HELP" option, for example, should work at any level of the menu, and any operation should produce the same result whenever or in whatever context it is performed.

(U) In an interactive system using a menu generator developed by Logicon [12], the designers' rationale was that the users would not use the system if they did not like it, and one reason they would dislike a system would be because of the unfriendly environment it provides. To keep the novice user from becoming anxious and the experienced user from becoming bored at the terminal, their command menu interpreter writes a joke or witticism on the bottom of each menu or command page. They provided two different environments, a verbose and a terse mode, to provide more explanation for new users and to reduce this information

after they have had a chance to become more experienced. Other researchers suggest tailoring the menu to the user, which might be helpful if there are different applications on a given system [13]. Lastly, a profile might be saved for each user so that when he or she logs into the system, he or she will be provided with the appropriate menu or can continue with a process. Use of these techniques should enable any designer to create a well-written menu utility for an application using interactive graphics. In the following section, some NSA graphics systems will be investigated to show the versatility of menu utilities as they are implemented in different applications.

EXISTING NSA SYSTEMS USING MENU UTILITIES

(U) In the following pages, several NSA graphics systems are described which have implemented menu selection in some form to improve their human/machine interface. Although the systems surveyed do not include all NSA graphics systems which have menu utilities, all three types of display technologies (raster, calligraphic, and storage tube) are represented. These systems are then compared, not by the type of graphics display, but on the basis of the characteristics of the menu utility implemented on each system.

CALMA GDS2

(U) The CALMA GDS2 of S27 is a medium-resolution color graphics system used for designing chips, pattern generation, design rule checks, and plotting. The cost of the system was about $500,000, with $100,000 of special software. It consists of an Eclipse computer with 256K of memory (i.e., 262,144 bytes), disks and tape drives, a color display controller, a Lexidata color monitor, and a black-and-white Hazeltine terminal. There are two workstations which consist of a graphics terminal and an alphanumeric terminal mounted in a movable hood with a keyboard and tablet on a desk below. As the user moves a pen across the tablet, a cursor is moved across the screen.

(U) A menu is displayed on this system as a series of small rectangular boxes, four rows across the top and four columns down the side of the screen in an "L" shape. Each box has a symbol or word which denotes the operation it performs. The user makes a selection by moving the pen on the tablet until the cursor is over the box he or she wants to select and then presses the pen down on the tablet to

make the actual choice. The software allows for up to four different levels of menus with up to 300 different commands in ten categories, but only two levels of options are presently available. Selections made from the menu are reflected on the black-and-white monitor so the user can review previous selections he or she has made.

(U) This particular menu utility has many of the advantageous features mentioned in the previous section. For example, it has an option to highlight changes to be made before they become permanent and has a "wipe-out" key to undo the latest changes. The user can be working on ten different scratch areas of the screen at once and can get a summary of the characteristics of each area by typing a command. The menu has four different options for filling figures, along with a scale that pops up to measure distances. Although there are up to 64 layers on a chip that can be made selectable for viewing, there is never a delay of more than a few seconds before a chip is displayed or a command is given a response by the system.

(U) The users, who are engineers, can take a two-week training seminar to learn the different commands and how to run jobs and seem to be quite satisfied with the system. In general, the menu utility provides a very good user interface because it simplifies what could have been a complicated system to learn.

QUICKER

(U) The QUICKER system in R812 is a high-resolution graphics system used for designing circuit boards, floor plans, mechanical drawings, and software flowcharts. Each of the four workstations costs about $50,000 and consists of a Tektronix 4014 graphics device, driven by Computer Vision software, a separate alphanumeric terminal with a keyboard, and a tablet with a pen for selection. Special hardware and software purchased from Computer Vision makes this storage tube draw images more quickly than conventional storage tube technology.

(U) The menu utility on this system is implemented by overlaying on the tablet a sheet of paper containing several rows of rectangular boxes with symbols or words designating the operations to be performed. This menu is unusual in that it is not displayed on the graphics screen at all. The user can point to a box with the pen and to a part of the circuit board displayed to delete or insert por-

tions of the drawing. Different sheets with different menu items can be overlayed for other applications. Since the menu software was not built into the system, R812 can change or add new functions to the 200 options already available.

(U) Although this system has the advantage that no space on the screen is used up by a menu, this can be a disadvantage since the user must constantly move his or her eyes from the screen to the tablet to make selections. With the CALMA system mentioned above, the user can focus his or her entire attention on the screen while making menu choices. Despite the fact that the QUICKER system is faster than most storage tubes, it might be uncomfortable for the user to have menu selection from the screen since the entire picture would have to be redrawn each time the menu level changed and a new one appeared. Some of the most useful features of a menu utility (highlighting, special keys, etc.) are impractical to implement on QUICKER because of the bright flash which occurs when the screen is cleared. Though the QUICKER system's users seem fairly satisfied with their system in general, a better technology to use for interactive user interfaces would be raster or calligraphic.

(U) R812 is now developing a new graphics system by Megatek which uses medium-resolution raster technology and costs $50,000 per device. This system receives input from a keyboard with twelve function buttons, a tablet with a pen, and a joystick with a button. A menu resembling the twelve function buttons is displayed on the screen and the user can use any of the named input devices to make a selection. The menu can be moved to different parts of the screen or made invisible, and the user can push one of the function buttons to move to different menu levels. Although only 512 by 512 pixels are displayed, this represents only one out of eight pixels which are stored so that as raster technology improves, a higher resolution may be obtained. In the meantime, added detail is gained by using a zooming feature. The Megatek contains fifteen microprocessors to do hardware fill and transformations and will eventually be connected to a mini-computer enabling parallel transmission.

menus offering options of geometric shapes and military symbology to superimpose on maps created from World Data Bank II data, as well as menus for adding, deleting, editing, or moving text. A scroll feature will be provided for all those flags or designs which cannot fit into the menu area at once. In addition, a "modify" feature is proposed for the analyst to correct mistakes made in constructing his or her symbol. Planning the user interface in the design phases of the system, instead of adding features as an afterthought, should make the A21 prototype system a good model for those planning to develop interactive graphic systems in the future.

P.L. 86-36
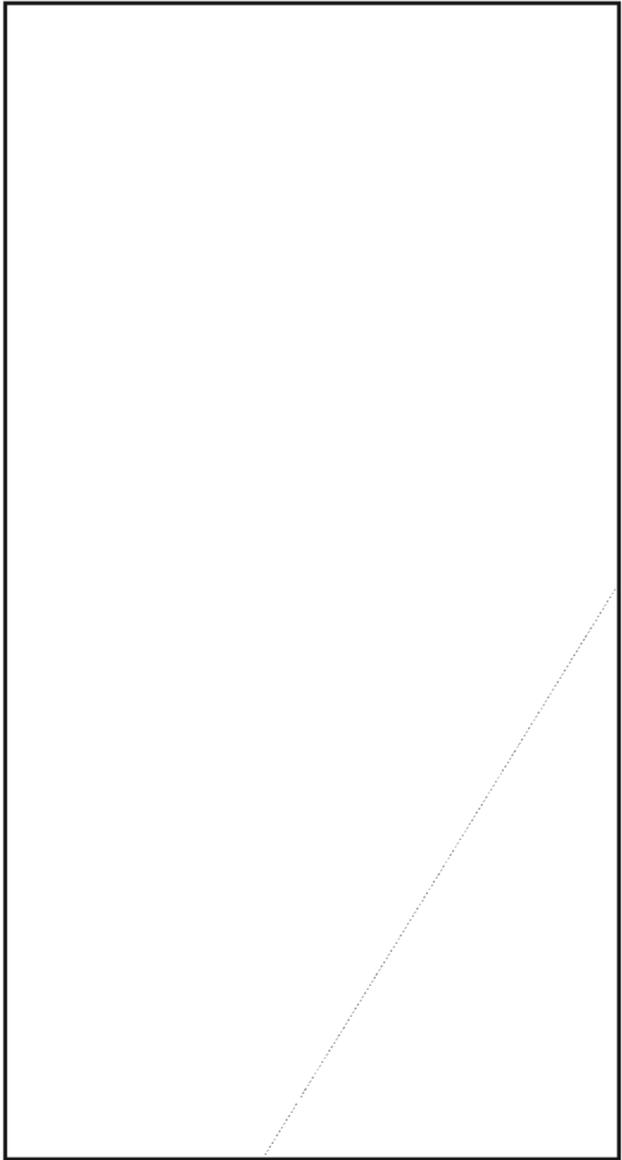
(U) Since [   ] is a fairly old system using electrostatic deflection, each display station is about the size of a 5 foot cube. Only 2,500 to 3,000 vectors and place-names can be displayed and images are blurred around the edges of the screen. [   ] menu utility is very rudimentary because of its age and lacks most of the features implemented in the newer systems such as the CALMA. A2 will eventually change its display stations to [   ] calligraphic devices to handle a mixture of signals at once with a 65K microprocessor, instead of the 4K of storage available from the current system. The menu utility could be updated to provide a better user interface with this newer, more powerful graphics system.

P.L. 86-36

A21 REPORTING PROTOTYPE

(C-CCO) A new high-resolution raster system with keyboard and an interactive device proposed by A21 will aid the analyst whose function is reporting. Instead of using wall-sized maps and slide presentations, the analyst will be able to view maps with special symbols and place-names, timelines, and business/management data on a graphics device. Hardcopies can be made of the display file for in-house presentations or the time-sensitive displayed data may be transmitted to other agencies. The user will interact with the system through displayed menus. There will be

SECRET

DOCID: 4011963

COMPARISON

(U) As new interactive graphic systems are being developed (like the CALMA GDS2 raster system for engineers and the A21 reporting prototype), more emphasis is being placed on developing a good user interface to make these systems better tools for the casual user. Both older systems surveyed, [ ] have added menu software to improve the user interface. Yet the original design makes the system more complicated for the analyst to learn. Besides deciding on the appropriate choice from the older systems' menus, the user must learn numerous function buttons and other input devices like the trackball and light pen in order to effectively complete an operation.

(U) [ ] menu utility shows no indication of the particular level the user is at and he or she may become easily trapped in a mode or level. Although the [ ] system's menu levels are not confusing, there is no legend to remind the user what the symbols or linestyles he or she has chosen represent, and no help in prompting the user for fill-in-the-blank responses. Neither system has a method for the user to wipe out previous choices and begin again, whereas the newer CALMA system has such an option to enable the user to make a mistake without the fear that the system may crash. In addition, the CALMA system highlights the user's creation or change of a design enabling the user to study the update before making it permanent.

(U) Although the QUICKER and CALMA systems physically implement menu selection differently, the former having a menu overlay sheet on a tablet and the latter with menus on the screen, both utilities are designed with the flexibility to add new menu items. Whereas the QUICKER system has a much higher resolution for electronic details than the CALMA, the CALMA system utilizes color to distinguish between layers of a design. Viewing the menu and drawings in color should not only be more pleasing to the user's eye, but also should keep the user interested longer than when watching a black-and-white display. Both the CALMA and [ ] systems allow the user to point to an item on the screen and receive more detailed information about the item. Additionally, the CALMA system is sophisticated enough to allow the user to type a query on the monitor to show the status of the 10 different scratch areas of the screen he or she can be working on simultaneously.

(U) As newer systems are being developed, like the CALMA GDS2, the user interface to the computer continues to improve over older models because human factors are being considered. when designing a system, instead of being implemented as an afterthought.

MENU SELECTION TECHNIQUES
USED BY R53 IN THE IOMS SYSTEM

(U) In this section the IOMS interactive graphics system, developed by R53 in conjunction with T333 and A512, will be described. This system is concerned with the user interface from the design stages and, in addition, is using a graphics standard to develop device- and machine-independent menu software.

(U) An R53 technical report published by David Nation in April 1980 [14] found that there is an unexpectedly large requirement for geographical computer graphics across many DDO organizations. Based on Nation's survey and given that similar geographical display capabilities are required by many NSA organizations, it was evident that development costs of geographical display systems could be reduced by designs which take advantage of a set of reusable geographical display software. In order to write reusable geographical display software, a standard graphics software package was needed. R531 had been interested in using a commercial implementation of the ACM SIGGRAPH CORE System, which is a proposed standard developed by the ACM/SIGGRAPH Graphics Standards Planning Committee to promote application program portability and device-independence [15]. R531 has installed one such system, DI-3000, on the MYCROFT PDP 11/70.

(S-CCO) In order to test the theory that graphics software developed on one system could be easily implemented on another by using a CORE graphics package, a prototype had to be chosen which would meet a real Agency requirement and at the same time illustrate the usefulness of the CORE System. R531 worked with the Graphics Investigation Group (GIG) in selecting a prototype application [16]. The Initial Operations Management System (IOMS) in A512 was chosen to be the prototype graphics application.

P.L. 86-36
EO 1.4.(c)
P.L. 86-36

Jan 83 * CRYPTOLOG * Page 80

SECRET   HANDLE VIA COMINT CHANNELS ONLY

IOMS GS general screen format

```
                     1111111111222222222233333333334444444444555555555566666
           12345678901234567890123456789012345678901234567890123456789 01234

 1  |Comms Nets        Comms Paths        Unidentified |                    |  1
 2  |                                                  |                    |  2
 3  |      (Optional lists of displayed data)          |                    |  3
 4  |                                                  |                    |  4
 5  |                                                  |                    |  5
 6  |                                                  |                    |  6
 7  |                                                  |                    |  7
 8  |                                                  |                    |  8
 9  |                                                  |                    |  9
10  |                                                  |                    | 10
11  |           G R A P H I C S                        |     M E N U        | 11
12  |                                                  |                    | 12
13  |                                                  |                    | 13
14  |               A R E A                            |     A R E A        | 14
15  |                                                  |                    | 15
16  |                                                  |                    | 16
17  |                                                  |                    | 17
18  |                                                  |                    | 18
19  |                                                  |                    | 19
20  |                                                  |                    | 20
21  |                                                  |                    | 21
22  |                                                  |                    | 22
23  |                                                  |                    | 23
24  |                                                  |                    | 24
25  |                                                  |                    | 25
26  |                                                  |                    | 26
27  |                                                  |                    | 27
28  |                                                  |     (Standard      | 28
29  |                                                  |      Functions)    | 29
30  |  (Optional map scale line)                       |                    | 30
31  |                                                  |PRINT       HELP |  | 31
32  |      xxx NM at center                            |                    | 32
33  |<----------------------->  (Classification area)  |RETURN      EXIT |  | 33
34  |                                                  |                    | 34
35  |                                      |                               | 35
36  |    P R O M P T S   A N D             |       O P T I O N A L         | 36
37  |                                      |                               | 37
38  |                                      |    M A P   L E G E N D         | 38
39  | S Y S T E M   M E S S A G E S        |                               | 39
40  |                                      |        A R E A                | 40
    |
                     1111111111222222222233333333334444444444555555555566666
           12345678901234567890123456789012345678901234567890123456789 01234
```

Figure 1: Mock-up of a Typical Display Screen

SECRET 1.4.(c)
P.L. 86-36

(C) In order to design a system which would
met the requirements of the A512 analysts and
have a good interface for users in a non-
computer-related profession, R531 had to learn
A512's terminology and understand the types of
data from which they derived information about
collection targets. Mock-ups illustrating
various scenarios, similar in format to the
one shown in Figure 1, were used to aid the
user in knowing what could be expected from
the graphics software. A menu generator was
designed to communicate with the user in the
simplest manner with a limited amount of com-
puter training required (see Figure 2). Fig-
ure 3 shows a hierarchical breakdown of the
different menu levels.

(U) The graphics process is initiated at a
Delta Data alphanumeric terminal. Thereafter,
the user makes all responses to menu queries
using the keyboard of the graphics display for
input. The user is prompted in the message
box of the display when it is time to make a
selection from the menu and receives feedback
in the form of a message indicating that his
or her request is being processed. Figure 4
shows an example which begins with the highest
menu level and continues to three lower levels
as the user presses the number of the menu
option on the keyboard.

(U) Each menu has been designed to indicate
if submenus exist for a given choice by having
a dash placed after the selection number (see
Figure 4). The user may return to previous
menus by typing an R for return or may exit
the process at any level by typing an E.
Other special features planned, but not yet
implemented, include a "HELP" key to display
documentation on the Delta Data and an
"ESCAPE" key to take the user back more than
one menu at a time. To further increase the
flexibility of the utility, the user will have
the option of picking a displayed data item
with the joystick and receiving more detailed
information about the item. Currently, the
user has the option of replaying one or more
sessions, redisplaying data already processed
without requerying the data base. The user
has a "CLEAR" option if he or she does not
want to save the data displayed and a full
screen feature to expand the map display and
erase menu areas. The user may also type text
onto the map to provide annotated hardcopy for
analytic use or presentations.

(C CCO) The colors of the menu selections
match the color of the item as it is drawn.

If a river is shown in blue on the menu area,
it is drawn in blue on the map. Likewise,
when communications networks are displayed,
the links between cities are shown in the same
color as the signal types they represent in
the legend area. As additional feedback to
the user, menu items are brightened from gray
to white to indicate that these features are
"on." Eventually, a command mode feature will
be added to allow the more experienced user to
type in options like "projection = Mercator"
without having to browse through several menu
levels to the menu which sets projections.
One of the most useful menu features is a file
that is created at the end of each session
which saves all the parameters the user has
set, such as environment, map scale and coor-
dinates, map details, type of projection, etc.
Given this parameter file, the user does not
have to reset these options anew each time a
new session begins.

(U) The IOMS graphics subsystem was
developed and tested on the R53 MYCROFT PDP
11/70. Delta Data 7000 alphanumeric terminals
and AED 512 color raster displays are used for
display and interaction. The software was
written using the DI-3000 implementation of
the CORE System (DI-3000 is a product of Pre-
cision Visuals, Inc.). R53 has transferred
the IOMS graphics subsystem to A512's system
with little difficulty, except in synchroniz-
ing operating systems. There are now plans to
transfer the software from the PDP 11/70 to a
VAX system to document the effort and the
number of code changes needed for the software
to run on a different host system. Since a
different computer and different graphics de-
vices will be used, this effort will be an
even better test of the CORE System's trans-
portability feature. Another future test
being considered is to use the same basic menu
generator, altered somewhat to meet a related
but different application. This would be done
on a third type of host system.

CONCLUSIONS

(S CCO) The purpose of this paper was to
show that a well-written menu generator makes
a good user interface for interactive graphic
applications because of the many advantages it
offers the casual user. The most important
advantages are feedback, the need for little
training, ease of use, choice of alternatives,
and special help keys. After the features and
advantages of menu selection were discussed,
several case studies involving NSA graphics
applications which utilized menu tools were
presented to illustrate the various ways menu
utilities have been implemented. Finally,
R53's implementation of menu selection using

SECRET   HANDLE VIA COMINT CHANNELS ONLY

Communications
paths over
Geography

1 Add communi-
cations net

2 Add communi-
cations path

3 Clear/Reset
data display

4 Draw/redraw
current disp

5-Alter disp.

6 Amplify data
on display

7 Update data

Select
Criteria

9-Comms net/
Comms path

PRINT   HELP

RETURN   EXIT

2500 Minutes at center

Select menu item.

EO 1.4.(c)
P.L. 86-36

SECRET

EO 1.4.(c)
P.L. 86-36

the CORE System [   ] was presented, not only to show the use of a menu utility as a good user interface in a geographical application, but to illustrate the concept that properly designed software can be transportable to other future systems with related applications by using a graphics standard such as the CORE System.

(U) Although there are still a few refinements to be made to the IOMS graphics software, the reaction of A512 analysts using the graphics subsystem has been very favorable. R53 has been successful in developing menu software for an interactive graphics application that is both convenient and user-friendly. This is due in part to the large effort made by the design team to consider what the user wanted from the system and how he or she would interface with it before any coding ever began. Even after the first implementation of the graphics software, the R53 team continued working closely with the users to determine how the menu utility could be further upgraded or changed to aid the analysts. R53's success can also be credited to the fact that many of the features mentioned above as being desirable in a menu utility were implemented in the menu utility designed for IOMS. From the IOMS project, R53 has demonstrated that menu selection is an invaluable tool for creating a good user interface in an interactive graphics environment. Further research and future projects should show that by using a graphics standard such as the CORE, the success of R53 can be replicated in other applications without having to replicate the entire programming effort.

REFERENCES

[1] Bergerson, Bono, and Foley, "Graphics Programming Using the CORE System," ACM Computing Surveys, (10)4 Dec,1978 p.400-425.

[2] Lecture notes from Computer Graphics MP-413 class, Oct-Dec, 1981.

[3] Tesler, Larry, "The SMALLTALK Environment," BYTE Publications, Inc. August, 1981 pp.90-147.

[4] Tesler, Larry, "The SMALLTALK Environment," BYTE Publications, Inc. August, 1981 pp.90-147.

[5] Schneiderman, Ben, "Software Psychology,," Winthrop Publishing Company, Cambridge, Mass., (1980).

[6] Miller, Robert, "Response Time in Man-Computer Conversational Transactions," Fall Joint Computer Conferences 1968, p.267-277.

[7] Shneiderman, Ben, "Improving the Human Factors Aspect of Database Interactions," ACM Transactions on Database Systems, Vol.3,No.4, December 1978, p.417-439.

[8] Ellis, T., "Interactive Man-Machine Communications," The RAND Corporation, DAHC15-67-C-0141-Arpa, January,1971.

[9] Newman and Sproll, "Pinciples of Interactive Computer Graphics," McGraw-Hill,Inc., (1979).

[10] Martin, James, "Design of Man-Machine Dialogues," (1973).

[11] Teitelman, Warren, "A Display Oriented Programmer's Assistant," Int J. Man-Machine Studies (1979)11, p.157-187.

[12] Logicon Inc., "Design Considerations for the Man-Machine Relations User Interface," San Diego, Calif. July, 1979.

[13] Mehlman, Marilyn, "When People Use Computers," Prentice-Hall, Inc., Englewood Cliffs, New Jersey. (1981).

[14] Nation, David A., "Survey of NSA/CSS Computer Systems Using Graphics," TR-R53-02-81, August 15, 1981.

[15] Newman and Van Dam, "Recent Efforts Towards Graphics Standardization," Computing Surveys, (10)4, Dec 1978 p.365-379.

[16] Graphics Investigation Group, "Interim Report," September,1980.

SECRET   HANDLE VIA COMINT CHANNELS ONLY

SECRET

IOMS Graphics System

Menu Hierarchy
State transition diagram

Figure 3: A hierarchical breakdown of the IOMS system

Figure 4: A logical menu sequence with selected options (*)

EO 1.4.(c)
P.L. 86-36

SECRET   HANDLE VIA COMINT CHANNELS ONLY

A TUTORIAL ON

COLOR THEORY & HUMAN COLOR PERCEPTION

for the Computer Graphics Programmer (U)
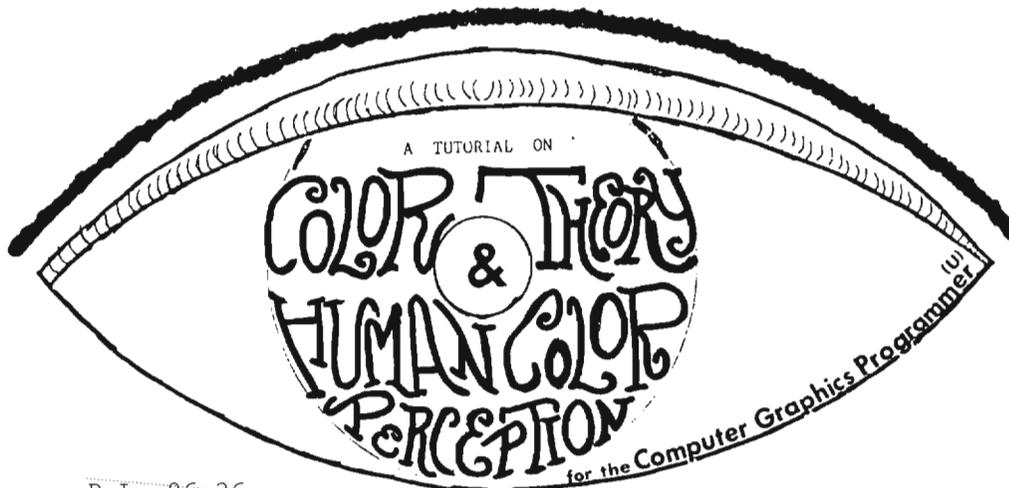
P.L. 86-36

by [   ] B62

INTRODUCTION

Psychological studies have found that color graphic systems (compared to monochrome displays) significantly improve the presentation of complex data: data are easier to recognize, exceptions and errors are easier to detect, and user satisfaction is increased.* The purpose of this paper is neither to support nor to dispute these views. Color graphics does indeed represent the computer state of the art more than any other single machine discipline.

A user receives hundreds of times more information from his eyes than from all his other senses; moreover, a user thinks graphically. The user sees signs or pictures instead of spots of light; he reacts to patterns of stimuli, usually with little awareness of the parts composing the pattern. Given a black and white photograph, the human eye can only distinguish the difference between thirteen levels of gray from black to white. Under carefully controlled conditions, adults with normal color vision can discriminate from 120 to 150 color differences across the visible spectrum. If saturation and brightness are added, the number reaches into the millions[2]. Consequently, color extends the amount of perceptable information that can be injected into, or extracted from a visual image. The intrinsic nature of color adds a new dimension to computer graphics techniques. The varied character of color provides the potential for creating unlimited effects with color that can accurately delineate various aspects of a visual image.

The challenge, then, is to create displays which are not just aesthetically pleasing. The graphics programmer must start with what, when viewed closely, appears to be a meaningless collection of colored dots to create a total impression with predictable properties from organized stimuli. In order to effectively utilize color in the visualization of ideas, information, or concepts, the aspects of human color perception as well as basic color theory will be explored in detail. The conclusions presented following this discussion are intended to be used as a guideline

---

* In one such study[18], four different types of CRT display formats were evaluated in the context of a computer-based telephone line testing system (see figures 1[18] and 2[18]). The subjects, eight Bell System employees (ages 25-50, all high school graduates, one college graduate) had normal visual acuity and color vision. The formats considered were narrative, which used complete words and phrases; structured, which used a tabular format; black and white graphics, which used a schematic of the telephone line; and color graphics, which also used a schematic but added color coding. The evaluation measured speed and accuracy of the subjects' interpretation of the displays of the test results. Accuracy did not significantly vary with format, but speed did. Response times for both graphic formats were consistently shorter than those for the narrative format. With additional practice, however, response times for the structured format were just as short. These findings are in agreement with earlier studies by Richard Christ and his associates [18] who concluded that while color is often an effective code, it neither improves nor detracts from human performance when compared to achromatic schemes.

Figure 1

Advantages and Disadvantages of Test Results Formats as Stated on Questionnaire

Format	Advantages	Disadvantages
Narrative	"You get a lot of information on the screen."	"A bit difficult to read." "Not as clear for a person with (little) experience." "Too much initial concentration." "Not as easy to read—time consuming." "In a busy office . . . could become tedious." "Recognition factor slower than graphics." "Time needed to read the test results." "Difficult to read and use."
Structured	"Summary stands out." "Information is simpler to absorb initially, strictly from the way it is . . . segmented in contrast to the narrative system." "Very good—are familiar with all areas of failures." "A blend of the qualities needed."	". . . would rather have all categories of data in one row." "In a busy office . . . could become tedious."
Black-and-white graphics	"Easy to read. More accurate, less time consuming" "Time saver" "Summary stands out." "Quick recognition of trouble and location." "Fast to use during busy days." "Clear—easy to read."	"Could be hard on the eyes if used constantly—unless toned down." "Broken lines tend to confuse at times."
Color graphics	"Pleasant, accurate. Good color selection, faster trouble decision." "Summary stands out." "Color adds to what might otherwise be a boring task." "Quick recognition of trouble and location." "Clearest for person with no mechanical background." "Color highlights important parts. High speed and very clear."	"Color could be hard on the eyes."

Figure 2
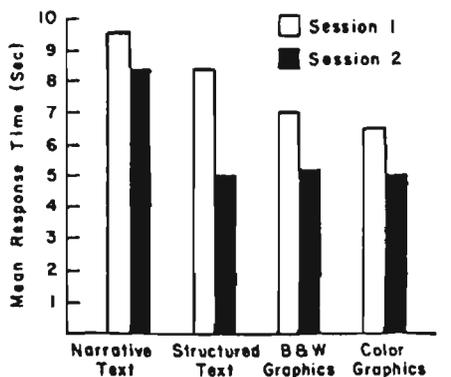
Mean response times to test questions for the four formats, by session.

for the graphics programmer to enhance graphics displays (when it has been determined that color applications will accomplish a particular job more effectively) without overstimulating the viewer.

COLOR THEORY

The vast diversities found in color result from the fact that colors vary on three dimensions--hue, brightness, and saturation. No matter how an image is created, be it electronic, photographic, or otherwise, it is perceived by a human observer only according to these three visual perception parameters[3].

Hue, which is determined by the length of the light wave, is defined as the perceived "color". Hue refers to the specific color family such as red, blue, green, or yellow. Brightness (value) corresponds to the perceived intensity of the light and varies with the amount of energy in it. Saturation (chroma) refers to the amount of color and is dependent upon the degree to which the wavelength is diluted by white light.

Further discussion of hue, brightness, and saturation is dependent upon a basic understanding of the relationships among and between the colors found on the color wheel. Sir Isaac Newton discovered that the spectral colors can be wrapped around in their natural order about the circumference of a circle (see figure 3[7]), allowing room between the red and violet ends of the spectrum for the purples not found on the spectrum[5]. If properly spaced, colors opposite each other on the circle will be complementary; that is, if the lights of these colors are mixed in proper proportions, they disappear to a neutral gray (see figure 4[5]). Some colors appear to be more elementary than others; they appear to be composed of a single hue. These elementary colors are called psychological primaries-- red, yellow, green, and blue. Between them are secondary colors in which the components are still identifiable--orange between red and yellow, the yellow-greens between yellow and

green, the blue-greens between blue and green, and the purples and violets between blue and red.* Another set of primaries is called the color-mixture primaries. Any three widely spaced colors on the spectrum can be used to provide all the other colors by additive mixture (see figure 4[5]); the three usually chosen are red, green, and blue.

Using the color wheel as a reference, colors can be chosen for their varying degrees of similarities or differences to meet the needs of the image. Warm colors, reds and oranges, tend to advance in space and take a frontal position in a composition. On the other hand, cool colors, blues and greens, tend to recede in space and take a background position. Spatial effects can be created with warm/cool contrasts (forms may appear larger when they are colored red as opposed to green or blue). Bright reds and oranges seem to radiate warmth and light and are thus appropriate colors for depicting light, warm spaces or objects. Blues and greens project a sense of coolness or a cool temperature. These cool colors are passive and seem to imply a comfortable stability. Reds and oranges are very dynamic, active colors and seem to energize spaces and objects.

As one continues to explore the similarities and differences of chosen colors, it becomes increasingly apparent that hue, brightness, and saturation can be used to clarify elements in a composition as well as to provide more detailed information. Gradual changes in hue create a subtle blending in which boundaries are obscured; depending on the degree of similarity or difference between the hues, three-dimensional spatial effects can be created. Brightness, a matter of appearance (whatever the composition of the color may be), can be reduced by mixing gray with a hue or by increasing (or decreasing) the value of a pure color. Brightness contrasts can be used to create spatial effects (light areas in a composition imply a source of light or lightness while dark areas create a sense of depth or weight). Saturation can be reduced by adding a gray of the same

---

* This set of color categories is generally recognized by Western peoples. Other culture groups see colors in ways that seem curious to a Westerner. The Hanunoo, a Stone Age People, for example, describe four basic classes of colors: dark colors such as black; light colors such as white; "dry" colors including red, orange, and yellow; and "wet" colors including light green, green-yellow, and brown. To the Hanunoo, these categories make sense because they help to describe the vegetation that supplies their food.
Regardless of the terms used to describe colors, all people appear to perceive the

divisions in the spectrum in about the same way. Despite differences in the words used by various languages to describe colors, people generally agree on what colors are the best examples of focal colors (what adults identify as the purest example of each color category, such as "blue" or "red"): black, white, red, yellow, green, blue, and so on. Similarly, people say nonfocal colors (for example, blue-green or red-orange) are like focal colors rather than focal colors are like nonfocal ones (i.e., pink is almost red, but not that red is almost pink)[2].

CID: 4011963

brightness creating a very drab color or by adding a complementary color of similar value. The latter method is preferred since a richer color is maintained as the saturation is reduced. Variations in saturation can be used to differentiate elements as well as to create three-dimensional figures in space.

Harmony is a sense of continuity that is created by establishing a relationship between the compositional elements. In choosing harmonious color sets, colors should relate to one another in a given manner. Two techniques for choosing harmonious color sets are:

1. the use of a series of adjacent colors on the color wheel, and

2. the use of a pair of opposite (complementary) colors.

Adjacent or neighboring colors, such as red and orange, have close similarities, while complementary colors, such as blue and orange, exhibit a high degree of contrast in hue. Another technique for choosing harmonious color sets is the selection of colors that are found at equal intervals on the color wheel. Harmony can also be achieved within a set of colors by maintaining equal brightness and/or saturation levels.

If a person stares at a red circle and then looks at a plain gray rectangle, he is likely to see a green circle on it. The viewer experiences a negative afterimage--negative because green is the complementary color of red (see figure 5[5]). Seeing the complementary color is common, but not always the case. After staring at a very bright light, one would probably see a whole succession of colors rather than the predicted complementary color. Still another exception is worth noting: usually, dark surroundings make a light area seem lighter, and light surroundings make the enclosed area seem darker. Under some

conditions, there is what is called a spreading effect so that dark areas make neighboring portions appear darker, and light areas make neighboring portions appear lighter[5] (see figure 7[5]).

THE TRANSITION FROM COLOR THEORY TO HUMAN COLOR PERCEPTION

Light is a continuous mixture of all the wavelengths across part or all of the visible spectrum. The normal human visual system is sensitive to visible light from violet with a wavelength of 400 nanometers to deep red with a wavelength of 700 nanometers.* Two kinds of light-sensitive bodies in the retina, rods and cones, catch the photons of visible light and in effect count them, thereby triggering a complex series of chemical and neural events[10]. The rods function mostly in very dim light to which the cones are insensitive. At normal light levels, virtually all visual information is provided by the cones. (In reality, individual cone cells distinguish color no better than rod cells; additional neural "wiring" is required before discrimination can be made solely on the basis of color[10].) Three types of these receptors, each sensitive to different wavelengths, together are responsible for color discrimination. Although the sensitivities of the three receptors overlap, one is sensitive particularly in the blue area of the spectrum, one in the green, and one in the red.

Any three wavelengths of light can be mixed in varying proportions to create many different colors, but the particular characteristics of the three human receptor systems make it impossible to duplicate all colors. The three primaries which can be mixed to produce the greatest number of colors are particular wavelengths of red, green and blue. For this reason, most color display systems are based on three light sources which are as close to an rgb component system as possible[8].

---

* There are two curious aspects as to how the progression across the visible spectrum is perceived. First, there is a simple, continuous change in a single quantitative property of light--the length of the light wave. This change in a physical quantity leads to a changing perceptual quality, namely hue. Reds do not look longer than yellows, greens do not look longer than blues. Instead, they all look different.
Second, though one can change the physical wavelength of light in a continuous fashion, from short to long, the changes in color are themselves rather discontinuous, and the colors seem to group themselves into sets or

categories. Blue, green, yellow, and red have for many people a unique appearance, and seem to characterize rather stable portions of the spectrum. A physical change in wavelength entirely within the blue or green region may be hardly noticeable, or may seem to be a change from one kind of blue or green to another. The same physical change in wavelength between categories (a change across the blue-green boundary, for example) can be striking, appearing to be a change from mostly blue to mostly green. It may be that humans have a built-in mechanism that partitions the continuous physical spectrum into a small set of color categories[2].

HUMAN COLOR PERCEPTION

All experiences of objects and events take place within a framework of space and time. Vision, being the user's preferred spatial sense, provides some of the most complex patterns of these perceptual experiences.

During the last century, two major theories of color vision vied for supremacy. The physicist Herman von Helmholtz proposed that there were three color receptors in the eye, each giving its own color sensation--violet, green, and red. According to von Helmholtz, these primitive sensations combine to form the entire gambit of color experiences--for instance, sensations of red and green together produce the color yellow.

The other theory was that of the physiologist Ewald Hering, who argued that there were three pairs of colored processes--a black-white process, a red-green process, and a yellow-blue process. Hering's view was an "opponent-process" theory, in that the two elements in each pair are antagonistic to each other--a color can have some yellow or some blue, for example, but cannot be both yellow and blue in the same place at the same time.

Recent studies strongly support Hering's view. Work by Dorothea Jameson and Leo Hurvich of the University of Pennsylvania [2] have extablished that although it is true that the eye contains three color receptors, these receptors do not give sensations directly. Instead, in a complex fashion, the three receptors feed into a set of nerve cells that work by opponent processes. It is the pattern of activity of these opponent cells that appear to underlie the experiences of color.

Russel DeValois and his associates at Berkeley have identified three main classes of nerve cells that are specifically sensitive to variations in light wavelengths[2]. Generally, it is possible to identify these three kinds of cells as the embodiments of white-black, red-green, and yellow-blue opponent processes.

The above mentioned color categories match up quite well with the action of these three main classes of color-sensitive cells in the brain. For example, when a person sees yellow, opponent cells that become more active in response to yellow, and less active in response to blue, are excited. When one sees green-yellow, both green (in a green-red cell) and yellow (in a yellow-blue cell) are excited simultaneously. The existence of color cells that can give sensations of black (versus white) presumably makes it possible for a person to see surfaces with dark colors, such as navy (black plus blue), maroon (black plus red), and brown (black plus orange).

INTEGRATING COLOR INTO THE VISUAL IMAGING PROCESS

In the natural world, color comes free; in the graphics world, the advantages of color must be great enough to offset the greater cost, lesser resolution, or other disadvantages[15].* Use of color, therefore, is recommended as a coding agent (to indicate to a user the category into which data being displayed falls) and as a formatting aid (whereby color helps the user in understanding the logical structure of the data on the screen). Unfortunately, computer programmers often have a low A.Q. The conclusions which follow should be useful to computer graphics programmers who are now designing screen displays for the application of color graphics in today's world.

CONCLUSIONS

The appearance of color is relative and is dependent upon the background color which can influence the appearance of foreground color(s) in various ways (see figure 6[5]). The most obvious change in the character of a color occurs with the hue. If the foreground colors are opposite one another on the color wheel, they take on characteristics of the complement of the background. The brightness of the background color can also have an effect on the appearance of foreground color. Blue lines on a black background are hard to distinguish and will tend to deemphasize data fields whereas white, with the opposite effect against the dullness of the screen, can be used for emphasis. Yellow lines are hard to distinguish on a white background. Neutral backgrounds help set off full color. A dark background makes a color tend to appear lighter than it really is and a light background makes a color appear darker. In order

---

* Much has been written or said about the number of "colors" a system is capable of displaying[3]. Some manufacturers claim the ability to create and display several million color combinations. In an additive color system with three 8-bit digital to analog converters, the number of possible combinations is 16,777,216. The vast majority of the

combinations are redundant and cannot be perceived. It is of no use for the range of values displayed to exceed the limits of visual perception; the observer wants only to be able to discern each independent gradation of intensity, hue, or saturation as a minimally detectable change.
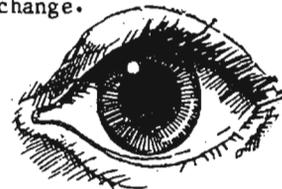
for these color changes to occur, the background area must be considerably larger than the foreground figure. Generally, a small area of intense color is balanced by a larger area of less intense color. Also, a smaller proportion of a light color, such as yellow, is balanced by a larger area of dark color, such as purple.

Patches of light colors will seem to be larger than those of dark colors. This is because brightness is more stimulating than darkness. Bright colors appear brighter on a dark background, and dark colors appear still darker on a bright background. In other words, predominant components of an image can be given visual dominance by displaying them in a bright intense color on a duller, less bright background. For example, a gray arrow shown on a white background looks darker than a gray arrow exactly like it shown on a black background (see figure 8[7]).

When blue rays of light enter the user's eye, they bend sharply and are focused at a point in front of the retina. Red colors are bent less, and focus at a point behind the blue rays. Therefore, the human eye cannot focus red and blue at the same time to form one image. Looking at alternate red and blue bands, or lines, will tend to "hurt" the user's eyes and make the colors glisten and vibrate (see figure 9[7]).

Color harmony is created with opposite colors since the mixing of complementary colors intensifies each other (blue or violet flowers often have orange or yellow centers). Each pure color physiologically demands its complement. If the opposite color is not present, the eye simultaneously produces the complementary color.

The colors on the right side of the color wheel, from yellow to red-violet are generally called warm colors. They suggest sunlight and the flames of fire. The colors on the left side of the wheel, from yellow-green to violet, are cool colors as seen in nature, the sky, and water. Greatest cool/warm contrasts are achieved using orange-red and blue-green. All other colors appear cool or warm depending upon their placement with warmer or cooler values. Red and red-orange are seen by most persons as the colors of greatest excitement. Blue and blue-violet are colors that seem peaceful and subdued. Green and yellow-green are the most neutral and tranquil. Yellow is the most cheerful.

High contrasts clearly delineate shapes by creating sharp, clearly discernable boundaries (see figure 10[7]). High contrasts imply a difference between elements. In situations where false coloring is used to differentiate

previously unclear gradation, the technique of using high contrast is successful. Low contrast in colors is used in continuity. Spatial effects can be created by employing different levels of contrast.

With similar objects of unknown but equal size, the brighter appears to be closer. If two objects at indeterminate distance are changed in both size and brightness, apparent motion toward or away from the subject will be enhanced if the cues cooperate.

Color edging enhances polygons (in general, any border pulls a graphics display together).

Other general rules will be found by studying the Birren color triangle (see figure 11[7]). Colors naturally fall into tints, shades, and tones. Tinted colors look well together and are the chief colors of spring. When colors are viewed under tinted light, the dominant tint tends to draw a group of colors together by introducing a mellow tone. Pure hues are the colors of midsummer. Shades are the colors of fall. Grayed tones are the colors of winter. On the triangle, any colors connected by a straight line also go well together. A hue may be safely harmonized with various values and chromas of itself; pure hues harmonize with tints and white; pure hues also are pleasing with shades and black, or with tones and gray. Other harmonies are found with combinations of tint, tone, and black, as well as of shade, tone, and white.

The above list is neither exhaustive nor applicable to every situation. The graphics display programmer is faced with the choice of:

1. using only colors and techniques that he knows are pleasing as well as effective,

2. using only the colors and design specifications indicated by the user, or

3. creating a subtle and harmonious blending of the first two options.

The last alternative is probably the most successful and certainly the most difficult solution. Therefore, when any doubt remains, the programmer should keep in mind the following two considerations:

Most people like blue best, then other colors in the following order: red, green, violet, orange, and yellow[2]. Appreciation, by the user, of nonfocal colors such as turquoise, rose, chartreuse, beige, and tan, requires many years of familiarity with color[7].

Above all, one should avoid using too

much color; overstatement or "business" will only confuse the user.

SUMMARY

Because of the benefits offered by the visual image, the projection by experts in the computer industry that computer technology in the '80's will emphasize graphics in addition to alphanumeric displays is already being realized. Computer graphics is fast becoming the medium of visual communication between man and machine, allowing improved data interpretation, higher productivity, and complex problem solution. A properly designed and used graphics display can transfer many types of information from machine to user much more rapidly and efficiently than can a verbal or numerical description. In a black and white picture, distinctions are made between elements according to the visual cues of size, shape, position, orientation and gray value. These visual cues allow the viewer to organize and comprehend the image. Carefully used, the addition of color adds an entirely new dimension to a composition and provides information beyond that encoded in the normal visual cues. Color expands the capability of the visual composition in communicating ideas by providing more detailed information.

It is essential that the choice of color and its application to various areas be used effectively in order to present clear, accurate and well organized information in a visually pleasing manner. An awareness of the behavior of color in composition and the impact it has on the aesthetics of an image combined with a firm understanding of how humans perceive and interact with color (in other words, a high A.Q.) will clearly benefit the programmer to effectively and sensitively develop and enhance color graphics displays.

BIBLIOGRAPHY

[1] Baeverstad, Jr., Harold and Bruderer, Clark C. "Display System Designed for Color Graphics." HEWLETT PACKARD JOURNAL, Vol. 31, No. 12, December 1980, pp. 25 – 31.

[2] Bornstein, Marc H. and Marks, Lawrence E. "Color Revisionism." PSYCHOLOGY TODAY, Vol. 16, No. 1, January 1982, pp. 64 – 73.

[3] Buchanan, Michael D. and Pendergrass, Richard. "Digital Image Processing." EOSD ELECTRO-OPTICAL SYSTEMS DESIGN, March 1980.

[4] Calew, Fred. "Enhancing Comprehension with Color." MINI-MICRO SYSTEMS, Vol. XIV, No. 8, August 1981, p. 139.

[5] Hilgard, Ernest R. INTRODUCTION TO PSYCHOLOGY. New York, New York: Harcourt, Brace, and World, Inc., 1962, pp. 186 – 193, 207 – 208, 228 – 232.

[6] Itten, Johannes. DESIGN AND FORM THE BASIC COURSE AT THE BAUHAUS. New York, New York: Reingold Publishing Company, 1963, pp. 42 – 43.

[7] Itten, Johannes. WORLD BOOK ENCYCLOPEDIA. Chicago, Illinois: Field Enterprises Educational Corporation, 1960, pp. 658 – 667.

[8] Joblove, George H. and Greenburg, Donald. "Color Spaces for Computer Graphics." COMPUTER GRAPHICS, Vol. 12, No. 3, August 1978, pp. 20 – 25.

[9] Kaplan, Alan R. "Special Report." MINI-MICRO SYSTEMS, Vol. XIII, No. 12, December 1980, pp. 68 – 69.

[10] Levine, Joseph S. and MacNichol, Edward F. "Color Vision in Fishes." SCIENTIFIC AMERICAN, Vol. 246, No. 3, February 1982, pp. 140 – 149.

[11] Meyers, Ware. "Computer Graphics: The Need for Graphics Design--Part One." COMPUTER, Vol. 14, No. 6, June 1981, pp. 82 – 92.

[12] Meyers, Ware. "The Need for Computer Design – Part Two." COMPUTER, Vol. 14, No. 7, July 1981, pp. 82 – 88.

[13] Pratt, Warren C. "A Precision Color Raster-Scan Display for Graphics Application." HEWLETT PACKARD JOURNAL, Vol. 31, No. 12, December 1980, pp. 19 – 24.

[14] Rossiter, Frank. "A Close Look at IBM's Color Graphic Offerings." MINI-MICRO SYSTEMS, Vol. XIII, No. 12, December 1980, pp. 116 – 128.

[15] Sorenson, Keith. "Making Color Affordable." HEWLETT PACKARD JOURNAL, Vol. 31, No. 12, December 1980, p. 152.

[16] Tektronix, Inc. TEKTRONIX 4027 OPERATORS MANUAL, pp A-1 – A-3.

[17] Truckenbrod, Joan. "Effective Use of Color in Computer Graphics." COMPUTER GRAPHICS, Vol. 15, No. 3. Siggraph 1981 Conference Proceedings, August 1981, pp. 83 – 90.

[18] Tullis, Thomas. "An Evaluation of Alphanumeric, Graphic, and Color Information Displays." HUMAN FACTORS, Vol. 23, No. 5, October 1981, pp. 541 – 550.

[19] _____. "User's Report: Terminal Turns Executive into Color." INFO SYSTEMS, Vol. 27, No. 11, November 1980, p. 93.
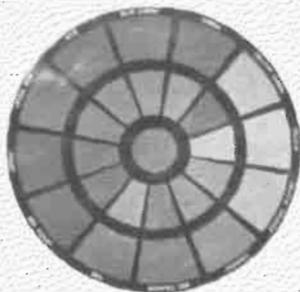
Figure 3

The Color Wheel
All major colors are shown in the outer cir-
cle. The grayed colors in the inner circle
are made by mixing colors that are directly
opposite each other in the outer circle.

Figure 4
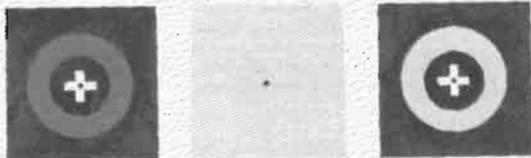
Figure 5

Negative afterimages
Look steadily for about 20 seconds at the dot
inside the blue circle; then transfer your
gaze to the dot inside the gray rectangle.
Now do the same with the dot inside the yellow
circle. What do you see?
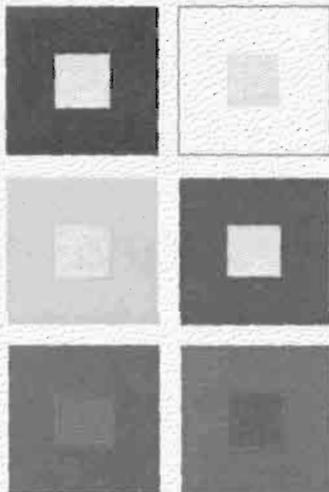
Figure 6

Simultaneous contrast
Note the darkening effect on the gray patch
when it is against white; the same patch of
gray against black looks much lighter. A gray
patch against a colored background tends to
take on the complementary hue. With colors
that are approximately complementary (as in
the red and green patches), there is an
enhancement through contrast.

Figure 7

The spreading effect
The same red is used throughout the strip,
but the red with black looks darker than the
red with white.

Figure 8
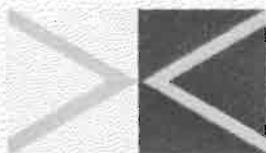
Figure 9

Background Makes a Difference
in the appearance of color. The two arrows
shown below are the same tone of gray, but the
one on black seems to be lighter than the one
on white.

Figure 10

Legibility of Color
The letters on the left are easy to read, be-
cause of the color contrast. The letters in
the center panel are not so easy to read be-
cause the colors clash. The edges of the
letters seem to vibrate when we look at them
closely. The panel on the right is not as le-
gible as the one on the left because there is
not enough color contrast.
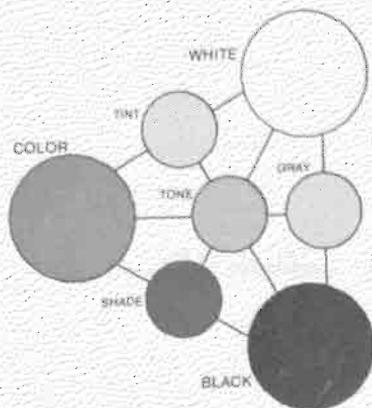
Figure 11

Color Triangle
designed by Faber Birren classifies all color
sensations in seven forms. Mix pure color
with white to get a tint; mix it with black
and get a shade. Black and white make gray;
black, white, and color produce a tone.

Computer Graphics to Enhance Collection Management (U)

by [   ] R53
and [   ] R53

INTRODUCTION

(U) The declining cost of hardware is making small interactive computer systems with graphic capabilities practical for small groups and individuals. Although "turn-key" systems are available in areas such as CAD/CAM and management graphics, many applications do not lend themselves to a standard solution. Individually tailored software to drive these systems is not generally obtainable without considerable investment in software development.

(U) There can be much similarity between systems having related applications. What often happens during normal system development is the "reinvention" of basic functions for each one. Software and data exchange are often difficult because of differences in hardware or software implementation.

(U) This paper will show that software designed using a standard set of graphics functions can be moved between mainframe systems or graphics devices and applied to similar applications.

(U) It was demonstrated that a working version of an interactive system could be developed in a short period of time. The first version was running on the target system three months after software design began. An additional three months was spent working closely with the users in the refinement of the interactive approach and in the incremental addition of functions.

(C) A recent survey of computer systems using graphics at NSA revealed numerous different computer and graphics device technologies. On closer examination of the graphics applications, many similarities were discovered. Most systems had charting and graphing applications, many had applications involving the creation of graphic pictures by the operator, and a significant number involved the display of geography and the overlay of various kinds of data on geography.

(U) A subsequent study was made of requirements for applications involving geographics and management graphics. Many of the geographically related requirements were the same. Typically a map was needed with geographic features (e.g. coastlines, political boundaries) shown as line segments. Various scales and projections were appropriate for each application. The primary differences between the requirements were the types of overlay data and the method of storing or creating these data.

OVERALL OBJECTIVES

(U) A new approach was needed in the development of interactive graphics software. It was apparent from the survey that most systems were developed "from scratch" even though existing systems had similar requirements. This happened for a variety of reasons. Early systems were written in assembly language and were not portable. Other systems were based on software packages that support a specific

graphics device. Changes in the graphics device or other hardware required software modifications to maintain equivalent functionality.

(U) The overall objectives in the development of software to improve this situation were:

1. Portability,

2. Rapid prototyping,

3. User friendly systems, and

4. Reusable software.

(U) Portability is concerned with the transfer of items between two or more environments. In this context, portability was applicable in three areas. First was the transfer of software between environments without extensive modifications and without compromising functionality. Currently it is difficult to use improvements in hardware technology, such as new host computers or graphics equipment, because of the software conversion expense.

(U) Secondly, the geographic reference data needed to be flexible to conform to specific application requirements and environmental constraints. Different systems had similar geographic data requirements with emphasis on specific areas of the world, and different environments would not accommodate the same size data sets. There was a need to dynamically "customize" data for each new system, based on the requirements and system configuration, from a centralized source.

(U) Finally, the finished graphic picture needed to be portable to permit recreation in another environment. Because of differences in system configurations, the method to accomplish picture portability required efficient storage utilization. It was impractical to retain every vector to draw a map with data overlays. It was feasible to retain the logical map specification and overlay data coordinates that were necessary to recreate the picture.

(U) A problem existed in the specification of requirements for new systems. It was often difficult for potential graphics users to adequately communicate their requirements for an interactive system. Problems in specification were encountered when some of the capabilities offered by the new system did not exist in the non-automated system. A need was recognized for the capability to quickly demonstrate the behavior of a proposed system during requirement definition, or "rapid prototyping." The appearance of the graphics produced and the

means of interaction would be clear to the end user and to the designer before implementation. This method reduced risk of major revisions late in development when the cost of such changes is much higher.

(U) A primary objective of the prototype development was to make the system appear friendly to the new user. It was considered essential for the success of this effort to provide for a user totally inexperienced in programming or the operation of interactive systems. Features oriented to experienced users will be included later in the development process.

(U) The concept of reusable software was important for subsequent system developments. Reusable functions were specified as autonomous subroutines that could be called with a minimum number of parameters. Hidden values, such as those sometimes found in FORTRAN common blocks, were avoided in the specification of parameters. The product was a subroutine or set of subroutines that could be used without alteration for similar applications with minimal knowledge of the software environment.

APPROACH

(U) The following approach was taken to satisfy these objectives:

1. Use a graphics lab with a variety of computer systems and graphics equipment,

2. Use a detailed worldwide geographic database,

3. Use a standard graphics interface,

4. Build additional layers of software to implement common functions,

5. Develop a High Level Display File format for exchange of pictures,

6. Develop one or more prototypes for real applications,

7. Develop a quick look capability to experiment with new applications, and

8. Build future interactive graphics systems using software components from prototypes.

(U) It was essential to have several different computer systems with varying capabilities to demonstrate the portability of the software and data. Graphic output devices representing various technologies made it possible to compare the appearance of the same picture.

DOCID: 4011963

EO 1.4.(c)
P.L. 86-36

(U) The primary geographic database chosen was the World Data Bank II (WDBII) [1] because of its high level of detail on a worldwide basis. It represents geographic features such as coastlines or railroads as line segments. Each line segment is composed of a set of geographic locations defined in terms of latitude and longitude. The WDBII consists of approximately 47,000 line segments with over eight million of these latitude/longitude pairs. The capability exists to extract data from the WDBII with specified characteristics for individual user's needs.

(U) The CORE system [2] was the primary basis for portability and reusability of the basic graphics software. It consists of a set of subroutines the functionality of which was defined by a committee of the Special Interest Group on Graphics of the Association for Computing Machinery (SIGGRAPH-ACM). The CORE system has been proposed as the basis for a national standard for graphics programming. The functions defined are independent of the particular graphics device being used, the host computer or the language used to implement the software. Software developed on this basis may be used with little or no modification on a variety of graphics devices and computer systems. Additional functions were added using CORE primitives that would expand the utility of the CORE software for common graphics requirements.

(U) The development of the prototype is detailed below. We considered the implementation of a set of requirements as an appropriate approach to the demonstration of these techniques. Another prototype is currently being designed. It will utilize software implemented for the first one, and add new functional capabilities for future software developments.

(U) A High Level Display File (HLDF) format is being developed to facilitate the exchange of pictures. Typical functions to be performed include drawing a map with certain characteristics, drawing a bar graph or using one of a standard set of graphic markers or "icons" to indicate the location of data.

(U) Software components developed for the prototype are intended to be used for experimentation in interactive techniques and to aid in the specification of requirements for proposed systems. The software itself is intended to be used as the foundation for new systems.

P.L. 86-36
EO 1.4.(c)

THE PROTOTYPE SYSTEM

Application

(C-CCO) [ ] is designed to convert [ ] records, received from [ ] over the PLATFORM network, into significant information for the System Operations Manager (SOM). The SOM has interactive tools that provide access to several online databases (kept current by background processing of the [ ] records), automatic notification of special conditions as they occur and interactive color graphics to enhance the analysis of collection activities.
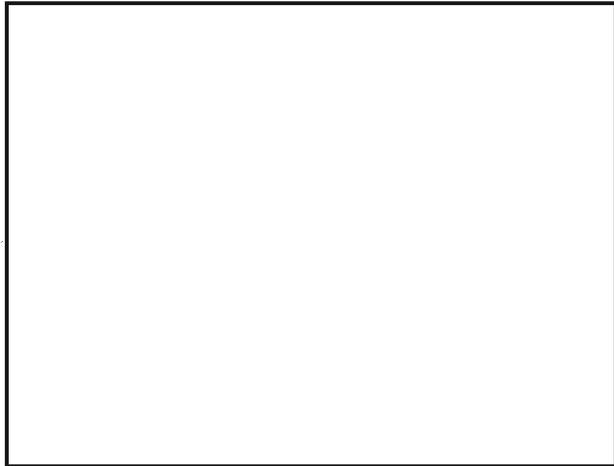
Environment

(U) The prototype was developed using a PDP 11/70 running the UNIX\PWB operating system (version 6). The display device was the AED512 raster color terminal with 8 memory planes and joystick. The target system was not available for the initial prototype development; consequently all software was completed on a second homogeneous (same architecture and configuration) system. This second system had been used for previous graphics research and was able to support the significant development effort required. When the initial system was completed it was transferred to the target system and installed with minimal effort. Enhancements were installed on the development system and periodically moved to the operational system reducing the impact on the operational system.

(U) A third heterogeneous system was available to enable the testing of software portability and device independence. This system was a VAX-11/780 with the VMS operating system. In addition to the AED512 graphics terminal, the VAX had a RAMTEK 9400 color raster system with 9 memory planes and a MEGATEK vector system.

(U) The graphics applications software was written using a FORTRAN 77 compiler developed and implemented under UNIX at Bell Labs. The Precision Visuals Inc. DI-3000 CORE graphics software package was used to generate all graphics displays. The installation of DI-3000 on the UNIX operating system and the PDP 11/70 was performed in-house and is not directly supported by Precision Visuals Inc. However, this installation does not affect the graphics software and therefore did not affect software portability. Certain application dependent and environmental constraints required the implementation of functions in the C programming language. These segments did not affect the portability of the major graphics components to a heterogeneous environment.

(U) Several databases were developed to support the dynamic map generation features and the application requirements. The map data was obtained from WDBII. Two additional map related files were also used. The first permits map selection by area name (e.g., country names) and the second allows the water areas of a map, such as oceans, to be differentiated from land areas and consequently permits "filling in" these areas on the display. The ocean fill data were created interactively using a modified version of the software described.
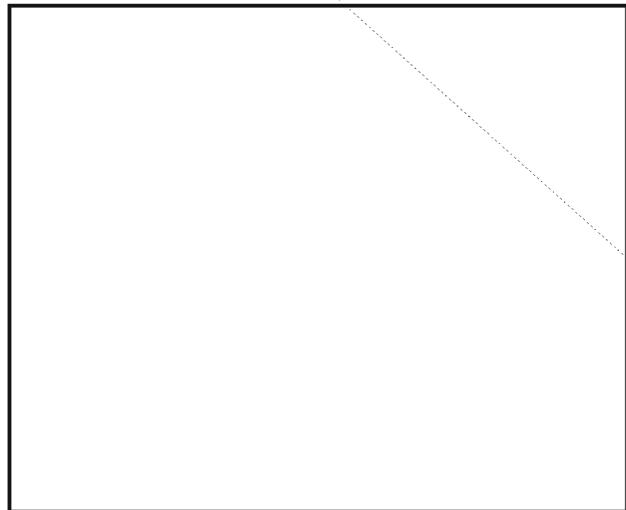
Software structure

(U) The extent of the prototype and the size of the commercial graphics software package exceeded the available program space on the 16-bit PDP 11/70. A multiprocess environment was established with UNIX "pipes" invoked to enable interprocess communication. The "pipes" were were used to pass data between processes and to synchronize the sequencing of operations.

(U) There are eight processes which comprise the current software structure: three to support DI-3000 and five to support the application (see figure 1). The The processes are organized by function with the interprocess communication kept to a minimum for performance reasons. The main process (main FORTRAN program) is responsible for initiating the other non-DI-3000 processes, and for controlling the interaction between the software and end-user. All menu generation and processing, all data input and database query generation and all parameter file input and output is coordinated by the main process. The main process "spawns" several other processes: the Database Query Process(DQP), the High Level graphics Display Process (HLDP) and the FFIX process. The DQP generates query strings based on the SOM input values, performs the database query and interprets the results. Successful queries require interpretation and secondary queries to obtain all data required to display the network(s) or paths. It also starts the Database Display Process (DDP). The DDP is notified of the query results and displays the application dependent graphics overlays. It does this by sending graphics commands to the HLDP. Query statistics and overlay display data are returned to the main program from the DDP.

DOCID: 4011963
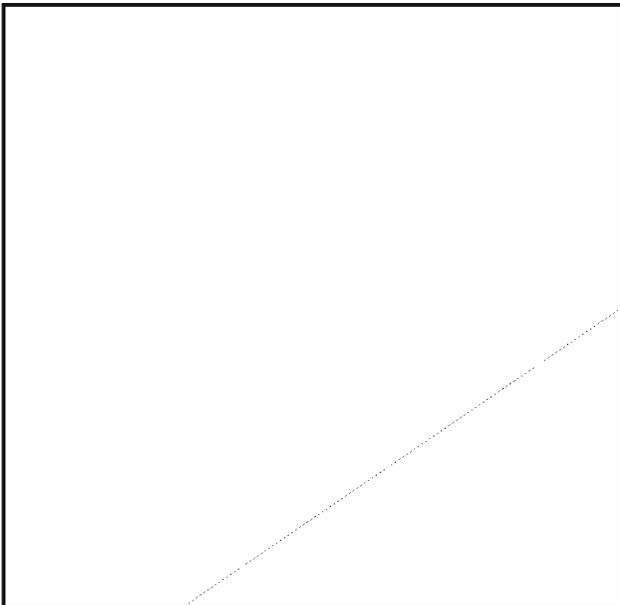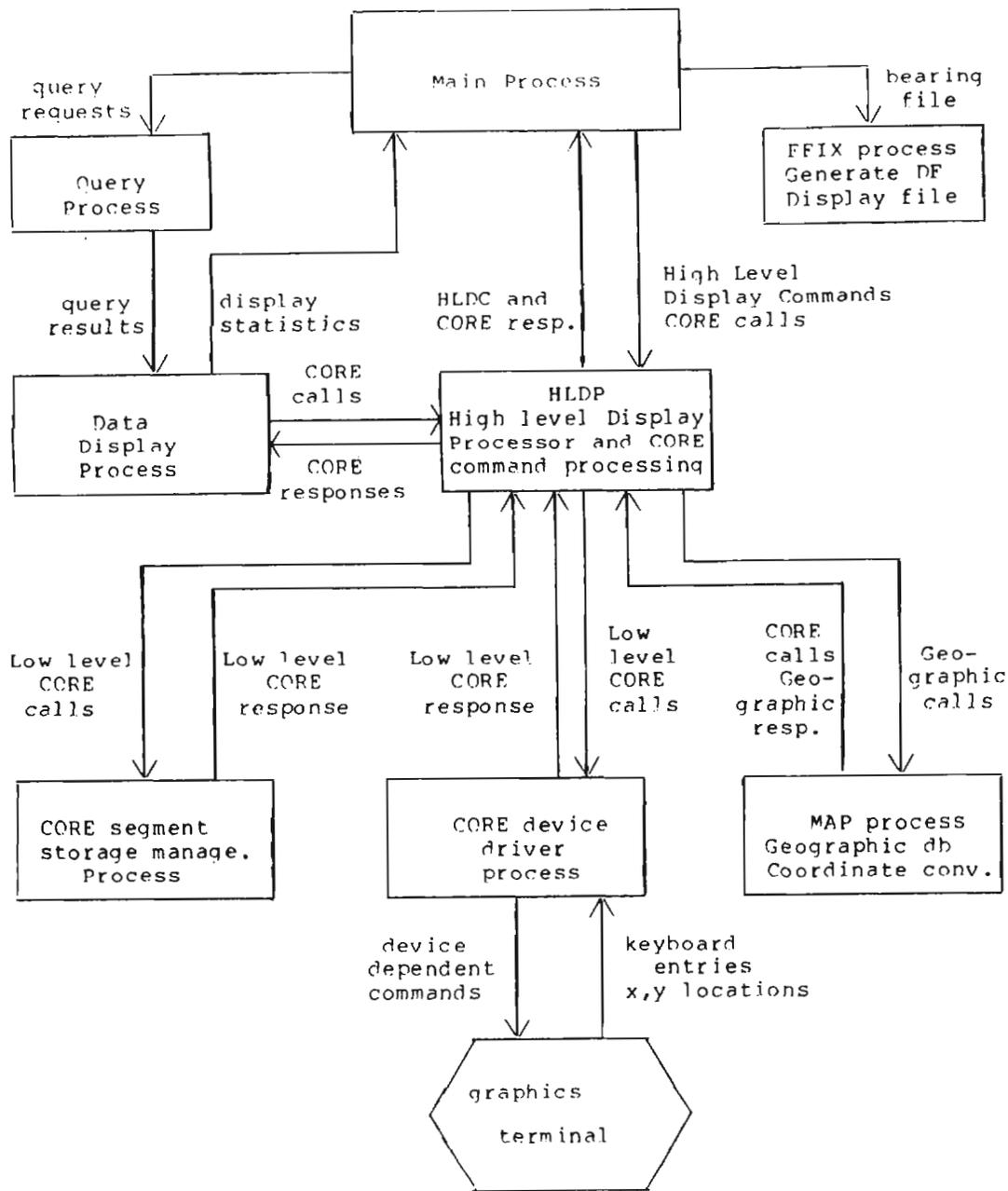
Figure 1.

(C CCO) Geographic Process Interaction Diagram

(U) The HLDP is the focal point for all graphics operations. It provides the interface between the application program and the graphics software. This process receives commands from the pipe which it translates to specific DI-3000 subroutine calls or to other high level graphics commands, such as the map generation command. When necessary it will initiate additional processes to do certain functions.

(U) The map generation process is initiated by the HLDP whenever geography based displays are requested. This process performs all access to the geographic data files, converts data to graphics oriented coordinates, and generates commands to the HLDP to invoke the necessary DI-3000 graphics subroutines. It receives parameters which determine what features are to be displayed, the geographic extent of the display and the projection to be used in the conversion process.

(U) The FFIX process [3] is initiated only when needed to generate a Best Point Estimate (BPE) and 95% confidence ellipse from a set of bearings or "flash." This process reads a file containing a set of digraphs and angles. The digraphs represent Direction Finding (DF) sites whose locations are stored in a separate file. The FFIX process creates an intermediate file containing information necessary to display the BPE, ellipse, and lines of bearing on a map. This file is used by the Map process to display the current DF data if that option is active on any map that is drawn.

(U) The remaining two processes, the CORE device driver and segment storage process, are part of the DI-3000 installation on the PDP 11/70. Each is initiated by the DI-3000 software when required, with the device driver designed to perform the device dependent functions and the segment storage process designed to manage retained segment usage.

(U) The collection resource utilization program requires one process to manage the user interaction and a second process whenever the collection resource database is queried (see figure 2). It also uses the HLDP and AED512 device driver for all graphics output.

(U) The multiprocess environment was necessitated by the small address space of the PDP 11/70. The implementation of the software structure was performed transparent to the user and in most cases to the application programmer. The interprocess structure can be collapsed without modifying the application software by linking the program with a different library. This makes the majority of the software portable to larger machine environments.

User Interface

(U) The prototype was implemented for inexperienced computer graphics users. The user interface had to be easy to use and "friendly" to avoid user alienation. It also required minimal user interaction to produce useful displays. There are two components to the effective user interface; the graphics screen layout, and the interaction required to produce displays.

(U) The screen was divided into five functional viewports: graphics, menu, legend, system message area, and access time (see figure 3). The intention was to provide the necessary information on the screen to enable the user to perform a task without extensive hardcopy documentation. The arrangement of the viewports is different for the collection resource display and the communications network displays. This was necessitated by the type of data contained in each display. In each case, the largest area is reserved for the user's data. The menu area provides the options available to the user. The menu options are specified in user terminology to enhance the ease of use. Items are preceded by a digit from 0 – 9 indicating the button which will initiate the function. A special menu area is defined to permit certain functions at any processing level. These functions include an online Help facility, a hardcopy Print facility, a Return function and a system Exit function. The collection resource display has an Update function which will requery the collection resources database to obtain the current collection status.

[] The Help facility provides documentation on how the graphics functions work;

[] the Print facility enables users to obtain a hardcopy print of the current display on a TRILOG color printer;

[] the Return function displays the previous menu; and

[] the Exit function terminates graphics mode after verification.

Each of these operations is invoked by pressing the key with the first letter of the function. The optional legend area permits user symbols to be defined to aid in interpreting the display.

[] The communications network displays use the legend to color code signal types and define line styles as environments.

[] The collection resource displays use the legend to show the color definitions for the collection data. It has relevance

```
                                  ┌─────────────────────┐
                                  │      Resource       │
                                  │    Main Process     │
    query                         │                     │
    requests ↓                    └─────────────────────┘
          ┌─────────────────┐          ↑ │
          │    Resource     │          │ │
          │     Query       │          │ │
          │    Process      │          │ │
          └─────────────────┘     HLDC and    High Level
                                  CORE resp.   Display Commands
                                               CORE calls
    query                              │ │
    results ↓                          │ ↓
          ┌─────────────────┐  CORE  ┌─────────────────────┐
          │    Resource     │  calls │        HLDP          │
          │     Data        │ ─────→ │  High level Display  │
          │    Display      │        │  Processor and CORE  │
          │    Process      │        │  command processing  │
          └─────────────────┘        └─────────────────────┘
                                          ↑ │
                                          │ │
                            Low level    Low
                            CORE         level
                            response     CORE
                                         calls
                                          │ ↓
                                  ┌─────────────────────┐
                                  │    CORE device      │
                                  │      driver         │
                                  │     process         │
                                  └─────────────────────┘
                                       │      ↑
                                  device      keyboard
                                  dependent   entries
                                  commands
                                       ↓      │
                                  ⬡ graphics ⬡
                                  ⬡ terminal ⬡
```

(C CCO) Resource Status Process Interaction Diagram

Figure 2

```
                 1111111111222222222233333333334444444444555555555566666
        12345678901234567890123456789012345678901234567890123456789012  34

  1  | Comms nets       Comms paths        Unidentified |        |   1
  2  |                                                  |        |   2
  3  |     (Optional lists of displayed data)           |        |   3
  4  |                                                  |        |   4
  5  |                                                  |        |   5
  6  |                                                  |        |   6
  7  |                                                  |        |   7
  8  |                                                  |        |   8
  9  |                                                  |        |   9
 10  |                                                  |        |  10
 11  |             G R A P H I C S                      | M E N U|  11
 12  |                                                  |        |  12
 13  |                                                  |        |  13
 14  |                 A R E A                          | A R E A|  14
 15  |                                                  |        |  15
 16  |                                                  |        |  16
 17  |                                                  |        |  17
 18  |                                                  |        |  18
 19  |                                                  |        |  19
 20  |                                                  |        |  20
 21  |                                                  |        |  21
 22  |                                                  |        |  22
 23  |                                                  |        |  23
 24  |                                                  |        |  24
 25  |                                                  |        |  25
 26  |                                                  |(Standard| 26
 27  |                                                  |Functions)| 27
 28  |                                                  |        |  28
 29  |                                                  |_____|  29
 30  | (Optional map scale line)                        |        |  30
 31  |                                                  |PRINT   HELP|31
 32  |    xxx NM at center                              |        |  32
 33  |<--------------------->  (Classification area) |RETURN  EXIT|33
 34  |_____|              |_____|  34
 35  |                                  |                          35
 36  | P R O M P T S   A N D            |                       |  36
 37  |                                  |   M A P   L E G E N D  |  37
 38  | S Y S T E M   M E S S A G E S    |                       |  38
 39  |                                  |       A R E A         |  39
 40  | A C C E S S    T I M E           |                       |  40
     |_____|_____|
                 1111111111222222222233333333334444444444555555555566666
        12345678901234567890123456789012345678901234567890123456789012  34
```

Figure 3

(C-CCO) OMSGS general screen format

only in the communications network display where there is more user interaction with the software. The user is always aware of the current system status by the message displayed in this area.

[] The time area reflects the date and time that the database was accessed to produce the current display. The SOM can determine how old the current display is by examining this field.

(U) The user workstation consists of one alphanumeric terminal for data entry and display (Delta Data 7000), and one AED512 graphics terminal for all graphics displays. The user selects either Collection Resource Display or Communications Network Display from a menu on the alphanumeric terminal. This action will bring the initial graphics display to the AED512 and request input from the user. All graphics interaction is provided through the graphics terminal. The alphanumeric terminal is available to perform other tasks.

(U) The user selects a function from the on-screen menu by pressing the key corresponding to the number beside the menu option. The function selected will either result in a new menu being displayed, a request for additional user input, the change in color of a menu option (turning the parameter on or off), or a change to the main display area. Figures 4 – 8 illustrate a typical sequence of interactions and the resulting displays for communications networks. The system message area will always reflect the current system state to keep the user informed. Invalid selections are trapped and the user is informed of the extraneous input before requesting the next selection.

(U) The collection resource program initiates a database query when the menu item is selected and when the SOM requests an updated status while examining an existing set of resource data. Figure 9 gives an example of a collection resource display.

(U) The user interface has several additional benefits. First, the screen image remains stable at all times. Whenever a state change requires a display alteration, such as displaying a new menu, only the affected viewport is redrawn. The remaining viewports are not affected and the user can page through menus without affecting the legend or main display area. Second, the use of color is effective in showing "on/off" states for system parameters in the menu area. The user can easily recognize the current status of certain parameters by the color in which they appear. The prototype employed yellow for "on" states and gray for "off". The parameter change is reflected immediately in the menu display.

Third, the user may request to have the main graphics area redrawn using the full screen area by pressing the 'F' key. This is particularly useful when a hardcopy print of a display without the other four viewports is desired. This full screen mode is available at any level of processing. (You need not be at a specific menu to invoke full screen mode.)

(U) The user has several local files that can optionally be created to save display parameters and overlay data, to alter the color scheme, and to alter the arrangement of the viewports on the display. These files are stored in ASCII format that individuals may alter using any file editor. The files are transportable between homogeneous and heterogeneous systems which enhances their usefulness. The prototype provides the mechanism for users to dynamically override the system default definitions by specifying an alternate filename. The color table specifications are defined by hue, lightness and saturation, and can be easily modified to user desires. The prototype provides a means for users to save display parameters in named files which are then used to regenerate the same display at a different time. The display is regenerated from the user specified filename. The system always saves the current system parameters on exit so a user may resume processing when reentering graphics mode.

(U) The final user interface feature is an online help facility. The user may request documentation on the currently displayed menu options by pressing the 'H' key. The system will display the document on the user's alphanumeric terminal. The user may browse through the document using the alphanumeric keyboard buttons to do paging and scrolling operations.

(U) The prototype provides a simple, easy to understand, and flexible user interface. It attempts to encourage inexperienced users without alienating them and at the same time it has the flexibility to enable experienced users to adapt the system to best fit individual needs. Despite the multiprocess configuration, the graphics software is reasonably efficient in processing and response time. Users are always aware of the current processing state and may alter system parameters through menus or files.

Geographic Software

(U) One objective of the prototype was to provide common geographically oriented displays. This requirement resulted in a map software package that is easy to use and adaptable to general purpose mapping

EO 1.4.(c)
P.L. 86-36
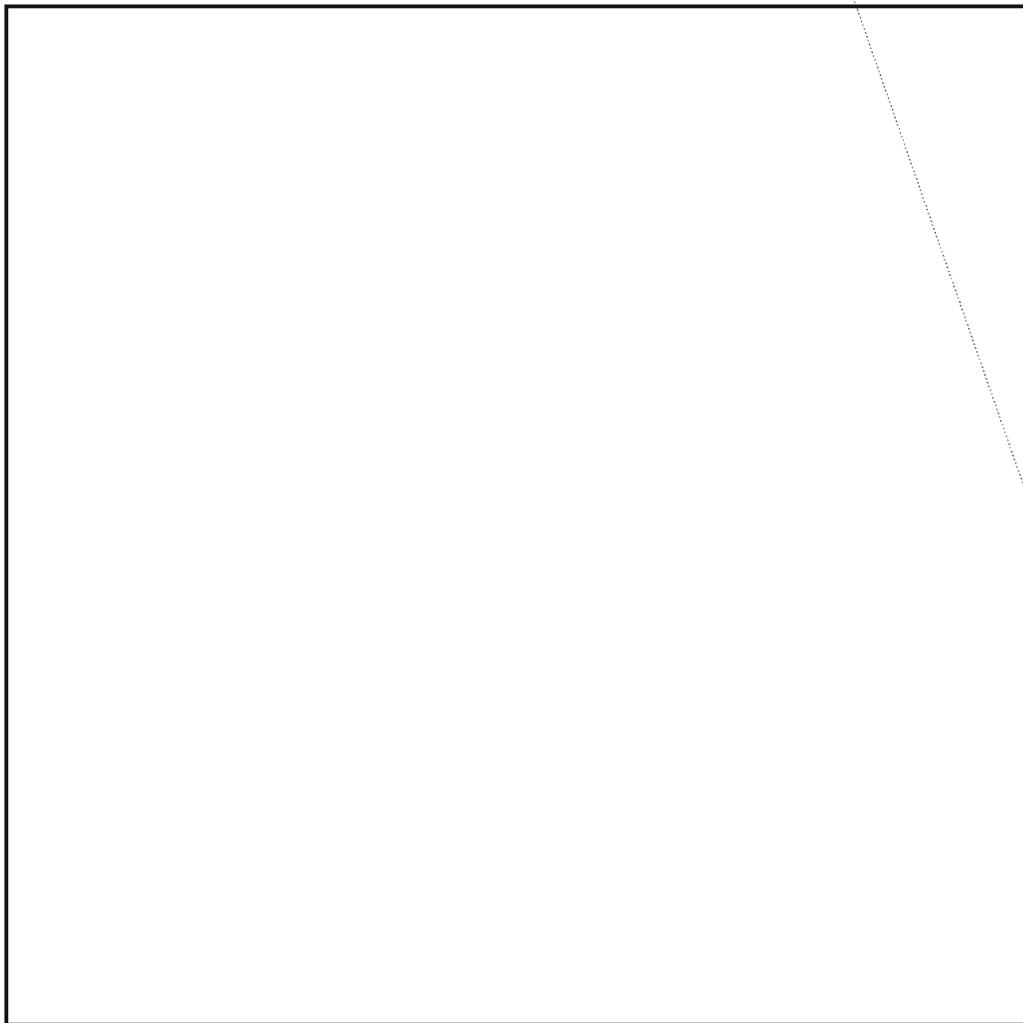
EO 1.4.(c)
P.L. 86-36
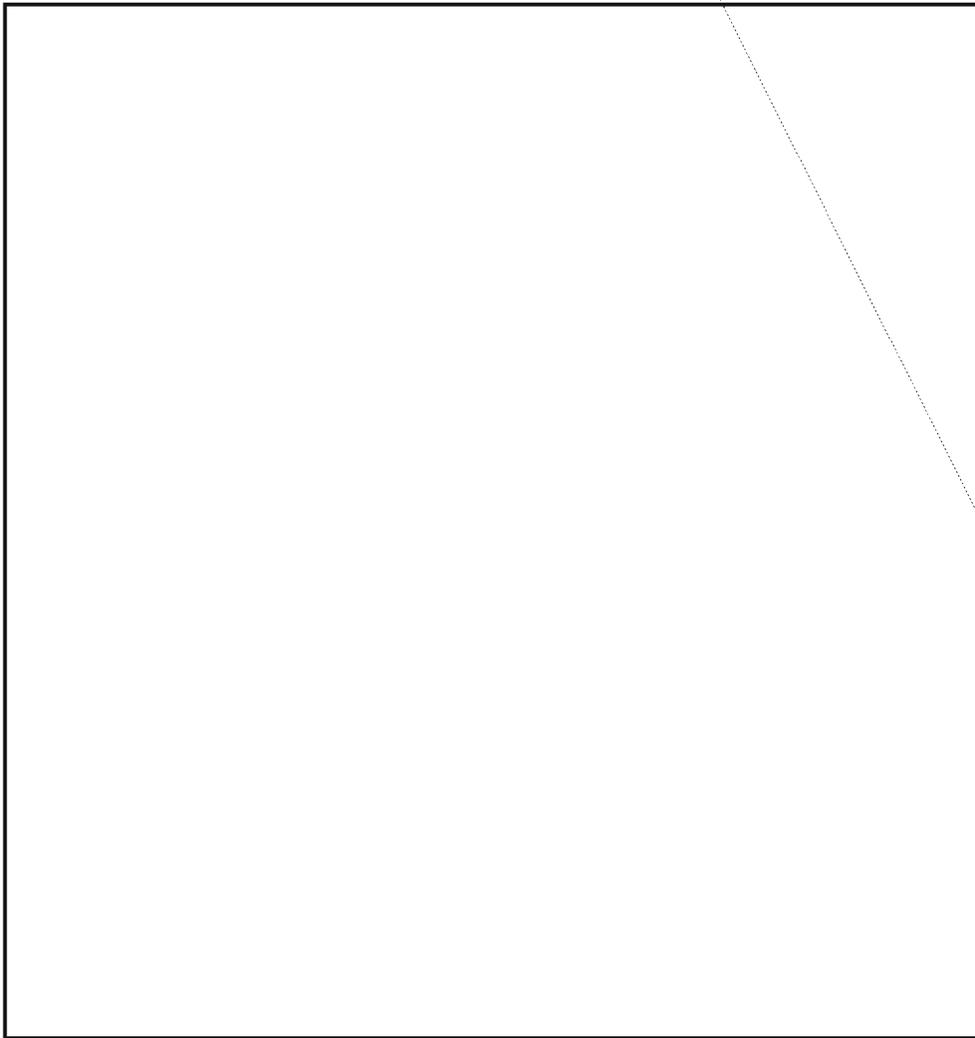
EO 1.4.(c)
P.L. 86-36
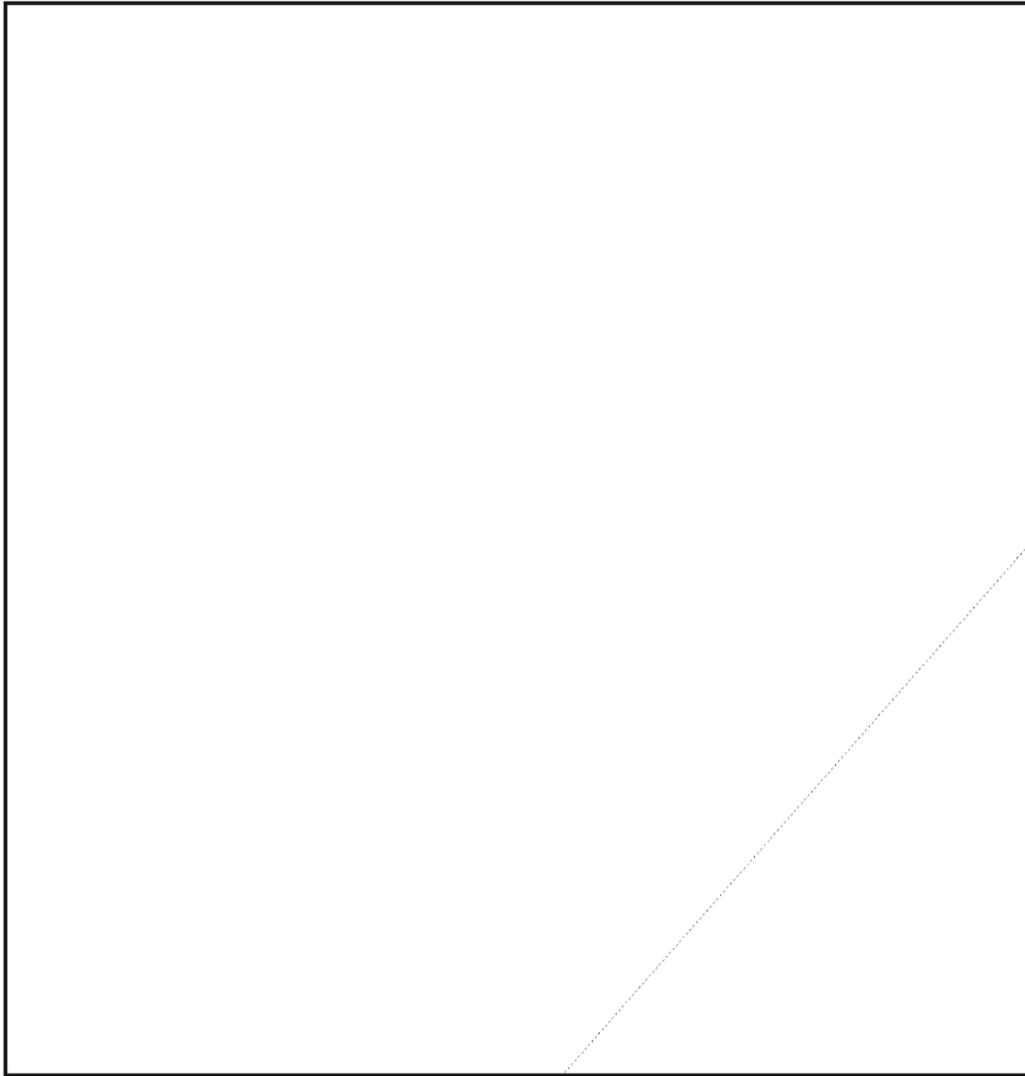
EO 1.4.(c)
P.L. 86-36

(U) The "Drawing map ..." message is displayed in the system message area while the map is being drawn.

EO 1.4.(c)
P.L. 86-36

DOCID: 4011963
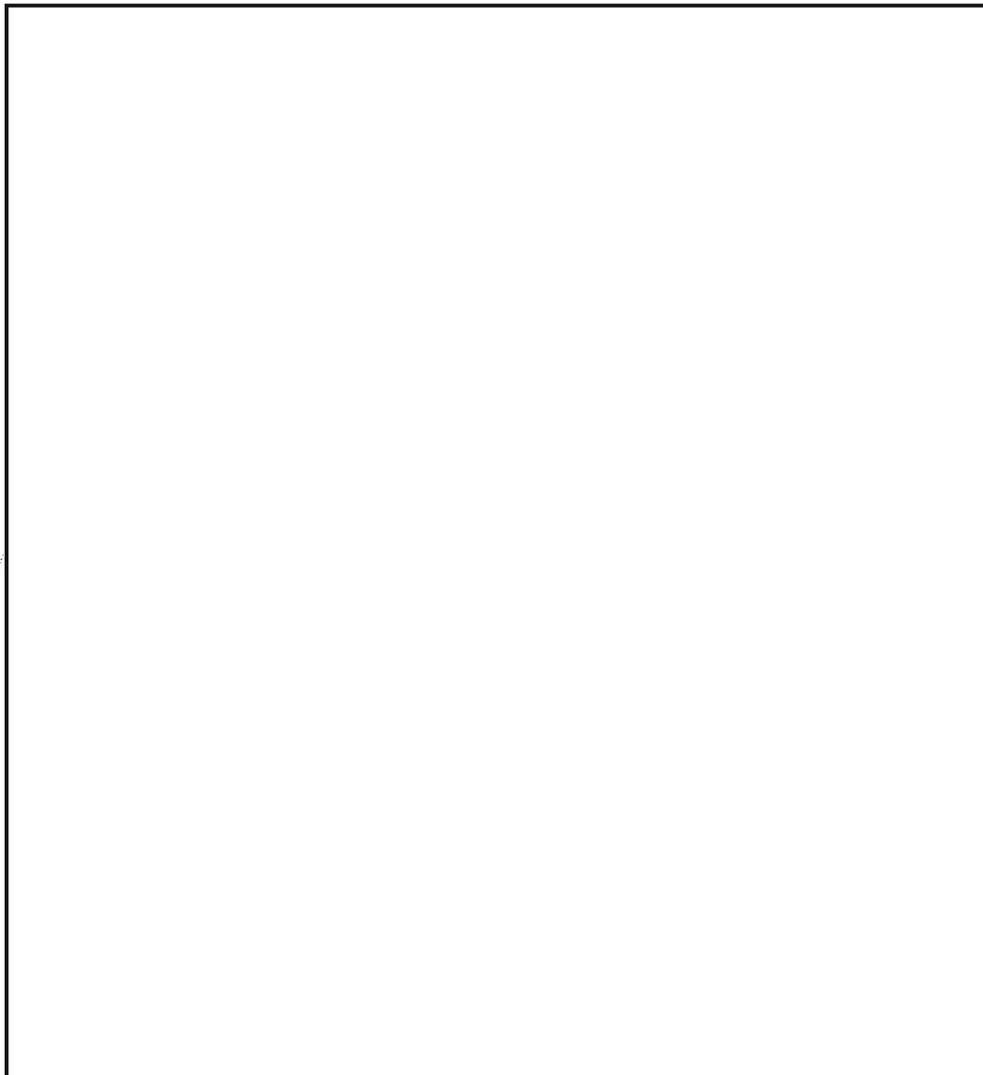
EO 1.4.(c)
P.L. 86-36

EO 1.4.(c)
P.L. 86-36
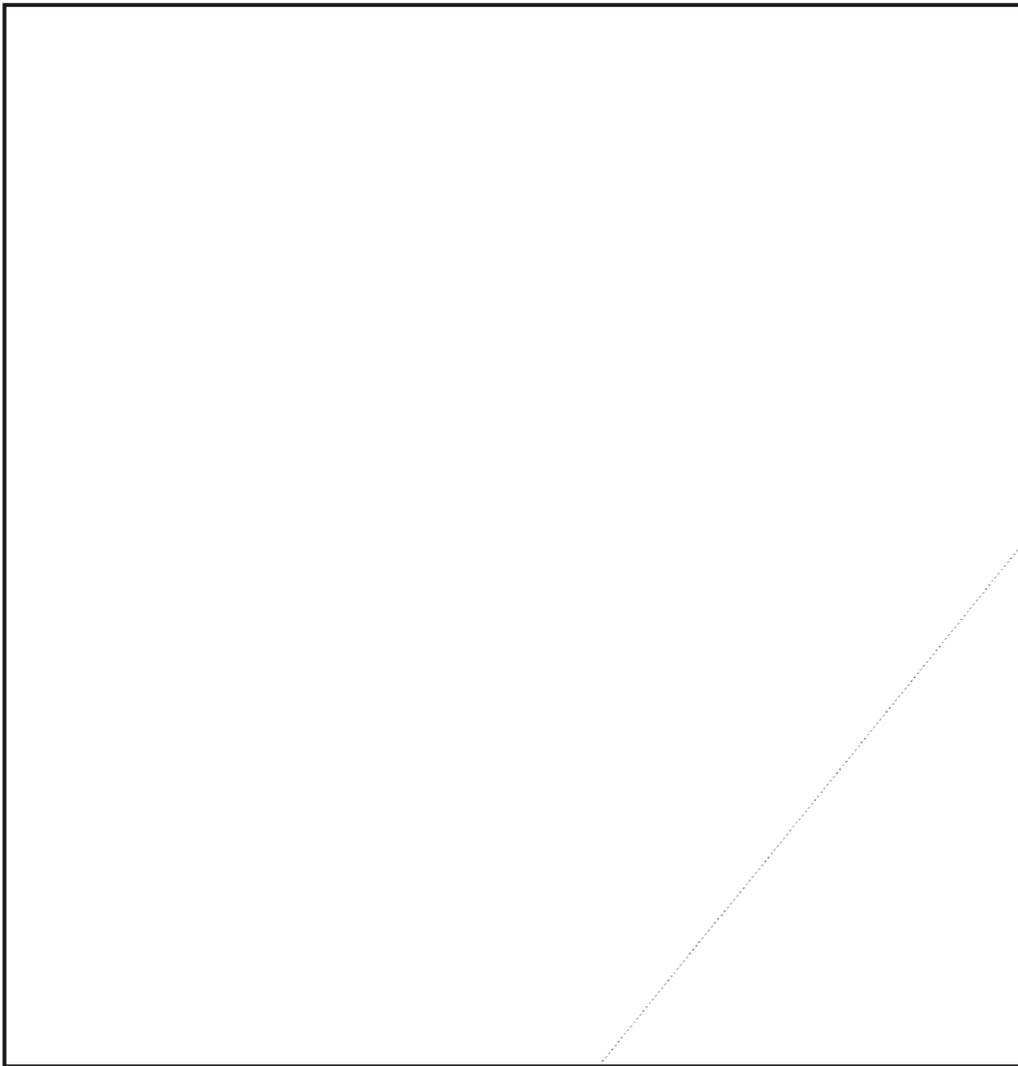
(C-CCO) The communications net can be drawn in proper geographic position without the pres-
ence of a map. All map features can be "turned off" using the "map features" menu. Subsequent
net diagrams will appear more quickly due to the absence of the geographic database access. Map
features can then be added when appropriate.

EO 1.4.(c)
P.L. 86-36

TOP SECRET   HANDLE VIA COMINT CHANNELS ONLY

DOCID: 4011963
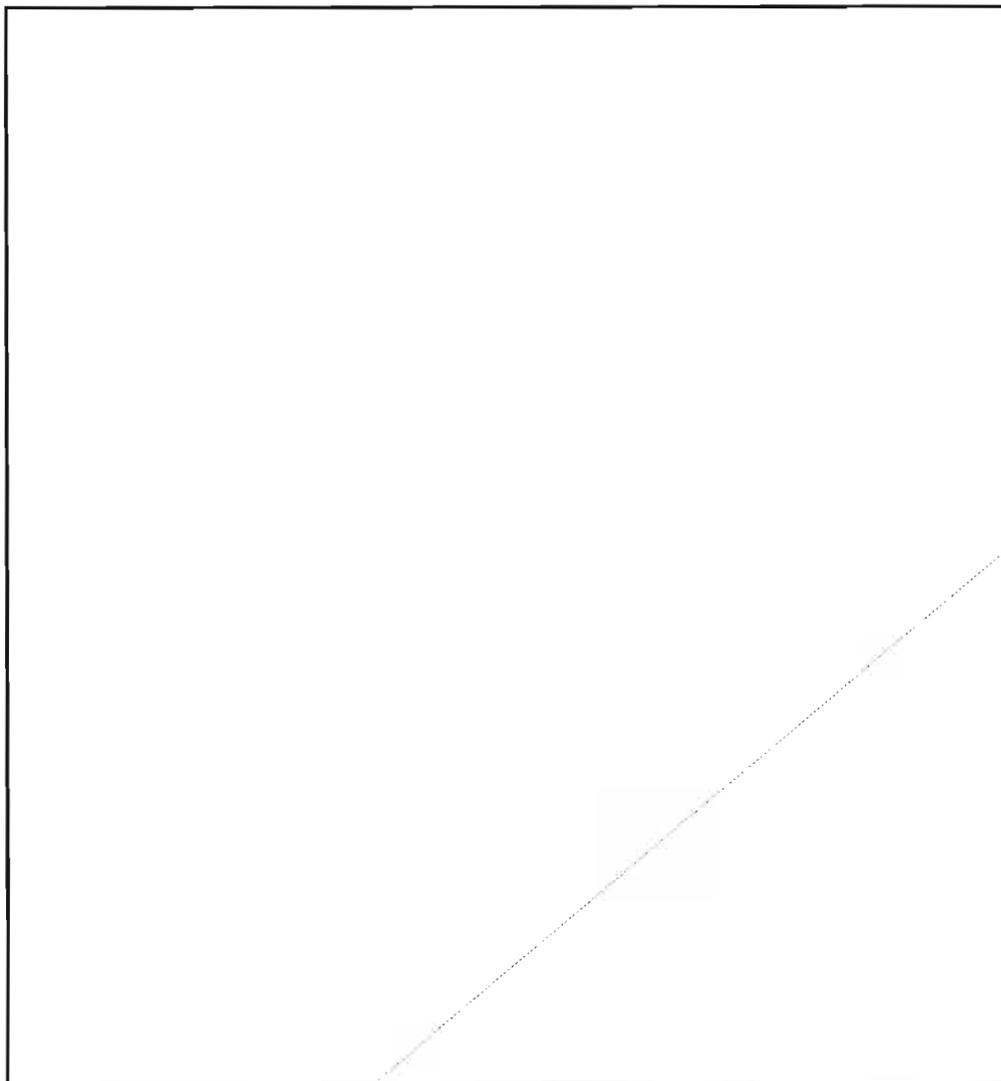
(U) This is an example of the screen used to display collection resource status information. The legend area has been relocated to the lower left and the menu is in the lower right.

EO 1.4.(c)
P.L. 86-36
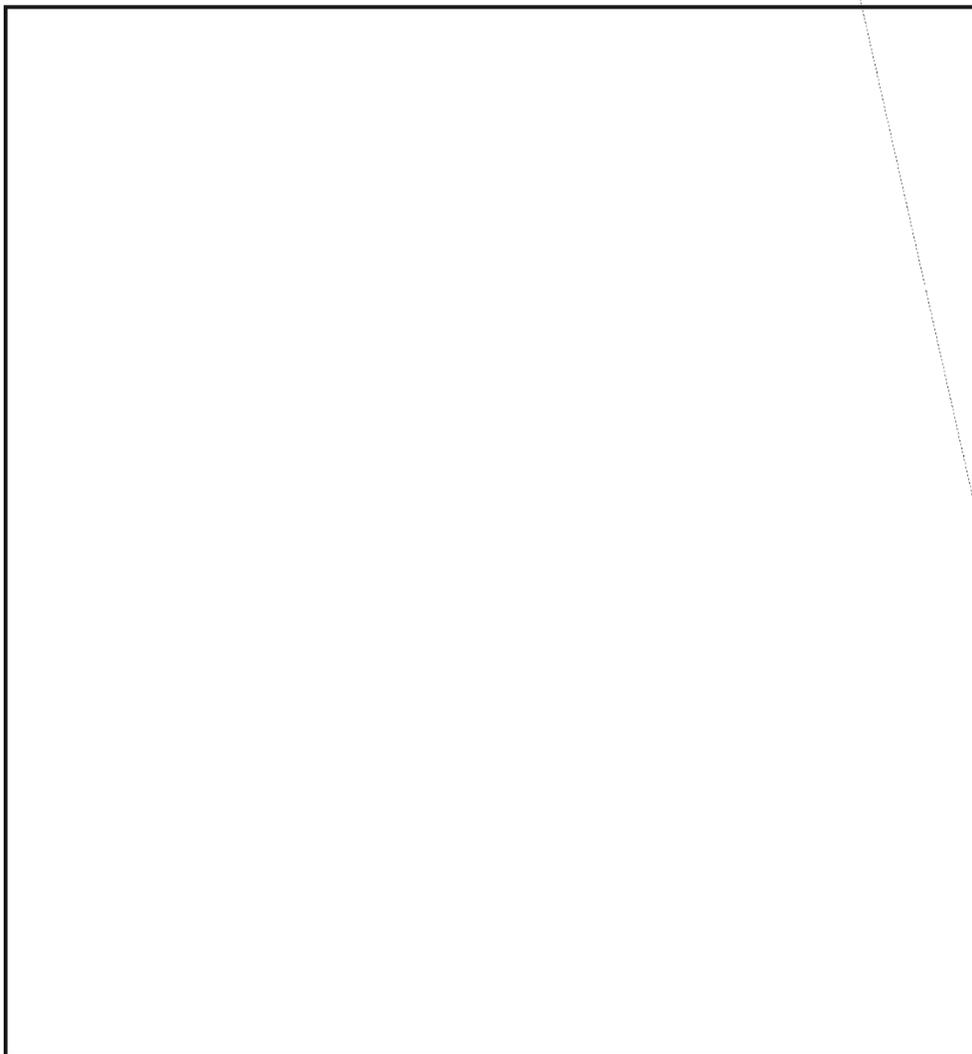
EO 1.4.(c)
P.L. 86-36

EO 1.4.(c)
P.L. 86-36

applications. The programmer draws a map by invoking one subroutine which has a series of arguments to define the map characteristics. The arguments can be set via assignment statements in the program or by end users through the menu interface.

~~(C)~~ The options include:

a. seven projections;

    []   mercator,

    []   gnomonic,

    []   perspective,

    []   orthographic equatorial,

    []   orthographic polar,

    []   orthographic,

    []   equirectangular,

b. four (or more) geographic database resolutions;

c. twelve (or more) geographic feature options;

    []   coastlines, islands, lakes,

    []   political boundaries,

    []   internal political boundaries,

    []   rivers,

    []   railroads,

    []   roads (S.E. Asia),

    []   military district boundaries,

    []   DF data,

    []   latitude/longitude grid lines,

    []   earth outline (orthographic projections),

    []   water fill,

    []   world area names,

d. specification of colors for graphic features;

e. center point of map;

f. map scale (scale indicated by distance across viewport);

(U) The map displays are generated from a standard geographic database, the WDBII. Special access techniques are employed to directly address subsets of the data and reduce data access time for arbitrary area maps. One of four versions of the WDBII data is chosen automatically based on the scale of the map being drawn. These subsets were created by selecting "1 out of N" points from the original data. This method reduces the amount of data that is read and processed when minimal detail is sufficient. The most detailed version of the map data on the operational system uses 1 out of 32 points.

(U) There are several additional features the geographic software provides. First, the selection of the map area can be dynamically determined based on the user overlay data. The user may select the data to be displayed and the software will provide the correct map based on the area covered by the data. The burden of producing the proper geographic region is not on the user. Second, the user may select a geographic area or country by name and the map software will select the map center and scale. Finally, the geographic software provides calculations to compute distances between two user specified points, azimuth calculations, and inverse projections. The current geographic parameters are preserved at the end of a user session to enable continuation in the next session.

Software: Collection Resource Status
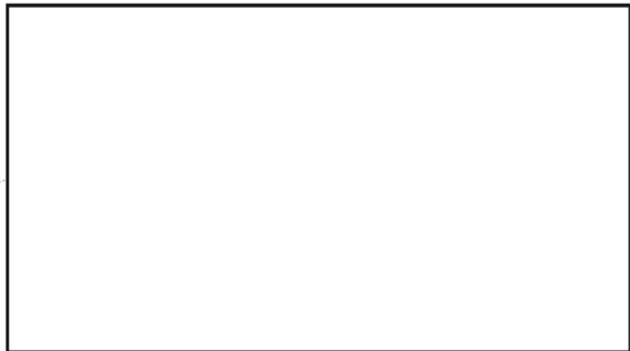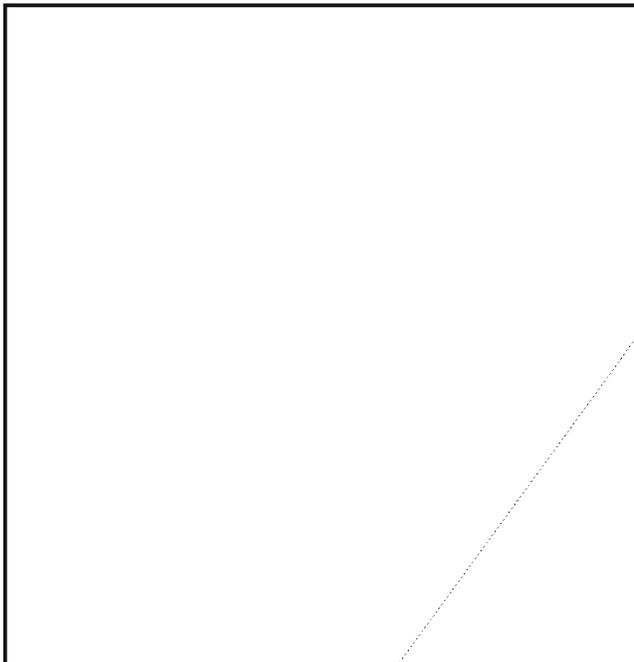
(U) The SOM can manipulate the display by scrolling up or down, paging up or down, proceeding directly to the beginning or end of the display, or requerying the file to obtain the latest information. These are invoked by pressing the corresponding key number next to the option or a 'U' for the update. The last page of data contains statistics on information being collected but not forwarded and counts of the receivers used for the collection.

P.L. 86-36
EO 1.4.(c)

(C-CCO) Future considerations include selective query for either GOOTEE 1 or GOOTEE 2 lines and options for displaying partial records and summaries.

IOMS Software: Communications Network Displays

P.L. 86-36
EO 1.4.(c)

DOCID: 4011963
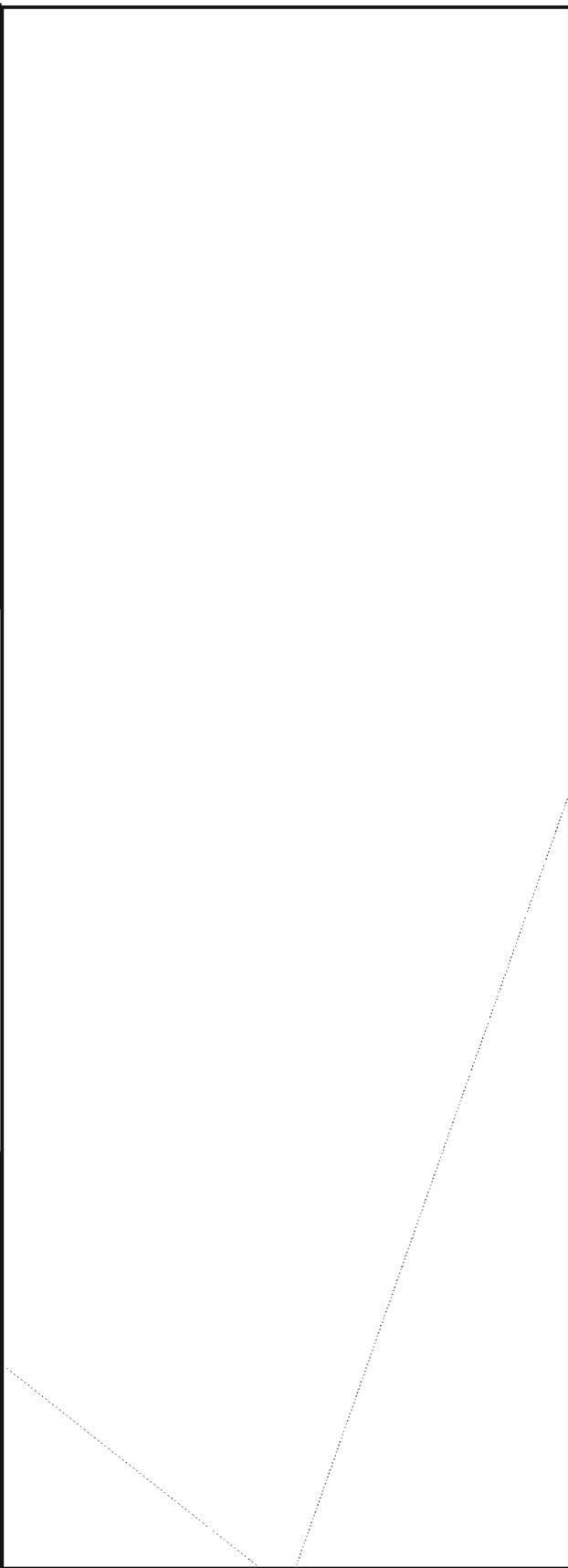
(U) Not all information contained in the network data records can be conveyed using the techniques described above. A menu option was provided to enable the SOM to enter "amplify mode." This feature permits the SOM to position the cursor (using a joystick) over a marker, PNAB or line segment, press any key and have the data record(s) corresponding to that item displayed in an enlarged system message area. This is especially useful to investigate display conflicts.

(U) Portability was demonstrated in the transfer of software between 16- and 32-bit machines, and the use of several graphics output devices without significant change in the application software. The WDBII and other geographic data files were also transferred without change. The named map parameter and data files provided an initial version of the capability to store and transfer finished pictures.

(U) The quickly developed initial capability and subsequent evolutionary growth were valuable in keeping the system visible to the users. There were no major surprises, and the feedback from the users helped in "fine tuning" the system to meet their needs.

(U) New users considered it easy to learn. Additional features for the experienced user are planned including a command mode consisting of short English sentences to allow direct control of the system without menu interaction.

(U) Many of the geographic and menu functions are applicable to other systems without change. The technique developed to transfer CORE commands between processes on the 16-bit machine has already been used to transfer commands between two other computer systems.

CONCLUSION

(U) The prototype was a significant step toward a coherent approach to the development of new graphics software at NSA.

(U) The value of color graphics was demonstrated in an operational environment on important Agency concerns. Color graphics provides an effective, dynamic method of conveying information to an analyst to affect near-real-time decision making. It proved effective in reducing or eliminating:

[] time examining black and white alphanumeric listings to isolate events requiring analysis,

[] volumes of multiple hardcopy alphanumeric listings required to present the same "picture" to an analyst, and

[] the documentation in comparison to written summary detailing the collection management events.

(U) CORE as a basis for the development of interactive graphics software has been shown to be effective. The addition of modular graphics functions using CORE primitives was important in enhancing the value of the CORE system for the development of applications software with similar requirements.

BIBLIOGRAPHY

[1] Delmar E. Anderson, James L. Angel and Alexander J. Gorny, "World Data Bank II: Content, Structure and Application", Harvard Papers on Geographic Information Systems, Vol 2, 1978.

[2] "Status Report of the Graphic Standards Planning Committee", Computer Graphics 13(3), August 1979.

[3] W. Jones, A. Hockenbery, P. Peters, C. Lempke, "Standardized DF Algorithm for Target Location using Bearing Information", NSA/CSS paper, 1 March 1976.

THE NSA HIGH-LEVEL DISPLAY FILE

Requirements

and Basic Design (U)

P.L. 86-36

by [   ] **R53**
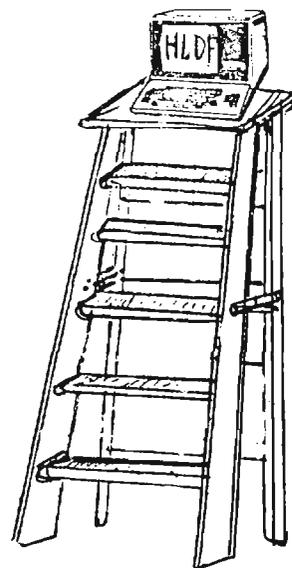
Introduction

(U)   he NSA High-Level Display File (HLDF) is a high-level applications data structure to be used in the modern NSA graphics system. The fundamental idea behind the HLDF is to unambiguously describe the graphics used both in the analytic processes at NSA and in NSA end product. While a simple file format for such a graphic (and a conceptually adequate one) would include only primitive geometric constructs such as lines, polygons, markers, etc. (in the manner of the Core System metafile [1], the GKS metafile [2], and the Tektronix Graphic Model Exchange Format [3], and others [4, 5, 6]), the HLDF is an application data structure that contains not only these primitives but also contains high-level application-specific constructs. The particular high-level constructs included in the HLDF have been especially chosen to reflect NSA's peculiar graphics needs and include entities such as maps, timelines, pie charts, etc. The set of constructs and primitives chosen for the HLDF is certainly not logically minimal (because a map, for instance, can be expressed as a sequence of line primitives) but it is a logically complete set because of the inclusion of the more primitive geometric constructs. These primitive constructs provide a "safety net" ensuring that any graphics image can be encoded in an HLDF. The design of the HLDF is also constructed so that the set of HLDF primitives and constructs can easily be extended to include, at an equally high level, other graphic formats should they become desired.

(U) One use of the extensibility feature of the HLDF can be seen in the inclusion of what is traditionally not considered to be a part of graphics: digital images. This "other side of the coin" of displayed pictures is included in the HLDF in a logically consistent manner

with that of traditional graphics, and its inclusion represents a possible future trend for analytic graphics at NSA.

(U) The HLDF is NSA specific but it is not
a repeat of the NSA designed and developed
programming languages such as BETA [7], GOTS,
or POGOL with their corresponding support,
acceptance, and transportability difficulties
but rather is a data structure, and more
specifically, a file format. Such an NSA
specific format definition would be needed if,
among other things, NSA product is to be
shipped electronically to NSA customers. The
need for such a specific format and its pre-
cise definition is independent of the particu-
lar programming language chosen for the vari-
ous applications programs. There are commer-
cially available graphics file formats ([1, 2,
3]), but these include only the most primitive
graphics entities that would be common across
the myriad commercial applications of graph-
ics. The encoding of a map, for example,
using these schemes would be voluminous and
would not allow later manipulation at the log-
ical level--for example changing the map pro-
jection. For this reason the design of an NSA
HLDF was undertaken.

Design Goals

(U) The design goals of the High Level
Display File are:

√ to contain all the information necessary
to reconstruct a given display;

√ to provide for the "logical" manipulation
of a display generated from an HLDF (the
so-called "Out to Lunch" problem; i.e.,
the problem of the construction of a com-
plex graphic by a computer-naive analyst
with interruptions of several hours or
several days);

√ to be machine- and graphics device-
independent;

√ to be unseen and transparent to the end
user;

√ to serve as an NSAwide (and perhaps Com-
munitywide) graphics metafile for picture
storage, transmission, and regeneration;

√ to be easily interpreted; and

√ to be as compact and space-efficient as
possible for most of NSA's graphics needs,

without conflicting with the other design
goals.

(U) Peripherally, it is also desired that
this data structure promote a modular design
of the graphics system using it. This goal
can be achieved by a system in which the HLDF
is:

◆ constructed by one routine of the system,
thus localizing the need to understand the
details of the HLDF structure; and

◆ "executed" by another routine, whose lower
level routines implement the actual graph-
ics algorithms that construct the displays
encoded by the HLDF.

(U) These goals imply that the current HLDF
(i.e., the HLDF corresponding to the current
display):

▷ store the structure of the current display
(e.g., at the level of map, circle, text,
histogram, etc.) and not just store
separate and logically "disconnected" line
segments or pixels;

▷ be updated (i.e., by the addition, the
modification, or the deletion of an entry)
with each and every modification to the
graphics portion of a display; and

▷ explicitly store all possible parameters,
e.g., line color and style, text font, map
projection, scale, etc. Thus, the HLDF
itself must not rely on any system
defaults since they can be different not
only from system to system within the
Agency, but also from user to user within
any system.

(U) Also the design of the HLDF:

◆ will resolve conflicts between space-
efficiency and ease of use (in construc-
tion, execution, and debugging) in favor
of ease of use:

◆ will not enable the end user to differen-
tiate between a display built "from
scratch" from one recalled from secondary
storage. Thus any operation possible at
any point in a graphics system must be
equally possible for a restored display
(created in a previous session on that
system) and a display built entirely at
that session;

◆ will guarantee that a transmitted picture
will be redisplayed exactly as it was
created (e.g., same colors, line styles,
marker positions, etc.), though the
receiver will be able to modify the
display at the logical level (e.g., change

the color of some entity or the back-
ground, move a marker or icon, add addi-
tional lines, text, or markers, redesign
legends, use a different map projection,
add or remove inserts, etc.);

► is the "glue" that holds together many of
the modules in the modern NSA graphics
system. It compactly represents the
display at any point during the user's
session with the system. It must effi-
ciently encode displays associated with
geographics, management graphics, briefing
slides, and dependency charts, but must
also be capable of use in other applica-
tion areas such as signals analysis, VLSI
design, ideographic language processing,
etc.

(U) The use of this HLDF for the trans-
mission of graphics is not dependent upon the
particular CPU of the receiver, the graphics
hardware, or even the graphics systems
software of the receiving system. The receiv-
ing system need only possess software which
can interpret the HLDF format. In addition,
the receiving system must have appropriate
geographic data   the NSA
Icon file (a file now being designed which
will store frequently used icons, e.g., mis-
sile outlines, military unit designators,
mushroom clouds, etc.), and the font data for
the various character sets to be used.

Proposed Design

(U) The following is a design for an HLDF
that purports to satisfy all of the above
design goals. In the course of constructing
this design, two major implementation deci-
sions were made--decisions whose impact and
effect on the HLDF design are second only to
the major design goals enumerated above. The
first of these implementation decisions is
that this HLDF design does not allow for the
direct encoding of three-dimensional data, but
only for data in two dimensions. This deci-
sion is motivated by two concerns:

● the extremely small use of three-
dimensional graphics in graphics applica-
tions in the Agency and

● the great increase in complexity that
would be incurred in designing and imple-
menting a three-dimensional HLDF as
opposed to a two-dimensional one.

There is little to be gained in delaying the
use of a two-dimensional HLDF for the amount
of time that it would take to design and
implement a complete three-dimensional HLDF.
For those few applications that may require a

three-dimensional HLDF, such an HLDF could be
designed at a later date and equipped with a
three-to-two-dimensional HLDF translator. A
three-dimensional HLDF-based system equipped
with such a translator would allow a three-
dimensional display to be stored locally with
the ability for later logical manipulation of
three-dimensional entities and for the trans-
mission of such a display in a manner that
would allow redisplay by anyone capable
of processing a two-dimensional HLDF. The
receiver of such a transmission would not,
however, be able to modify this transmitted
display at the logical level, since the
three-dimensional entities that make up this
display are not directly represented in the
translated and transmitted HLDF; they are
represented by their two-dimensional projec-
tions and thus the modification process would
not be possible at the logical level.

(U) The second major implementation deci-
sion for the proposed design which follows is
in the choice of coordinate systems. In this
proposed design, an HLDF can contain data
expressed in either one of only two coordinate
systems: the traditional latitude/longitude
coordinates for geographic plots and an
integer Cartesian coordinate system with a
range from 0 to $2^{16} - 1$ for each dimension
(hereafter referred to as the integer virtual
coordinate system). Systems that use any
other type of coordinate system (e.g. polar
coordinates, normalized device coordinates,
etc.) are responsible for transforming that
data into one of these coordinate systems for
inclusion into an HLDF. These two coordinate
systems are meant to be used exclusively for
any given HLDF, though with some disclaimers
concerning the results of subsequent zooming
operations. These two coordinate systems can
be used in one HLDF.

(U) Given these two implementation deci-
sions, there are at least two different
approaches to the design of this format. One
approach is that of a generative grammar
(used, for example in the Caltech graphics
metafile for VLSI circuits [8]) and another is
an assembly language model with op-codes and
arguments. Because the graphics used in NSA
product typically do not have the intricate
internal structure found in circuit layouts
and thus cannnot easily avail themselves of
the expressive power of a generative grammar,
the assembly language model was chosen as the
basis for the HLDF design. At the highest
level, a general format of an HLDF which will
satisfy the stated design goals is:

```
OP_CODE[1]          ARGUMENT_LIST[1]
OP_CODE[2]          ARGUMENT_LIST[2]
       .
       .
       .

OP_CODE[n]          ARGUMENT_LIST[n]
```

where each ARGUMENT_LIST is of variable length. However, since on some systems (i.e., machines and/or operating systems) this variable-length structure is difficult (if not impossible) to construct and process, this format may be further specified as

BEGIN_HLDF_TOKEN

```
OP_CODE[1]        BEGIN_ARG_LIST_TOKEN

                  ARGUMENT[1],

                  ARGUMENT[2],

                  ARGUMENT[3],

                         .
                         .
                         .

                  ARGUMENT[n]

                  END_ARG_LIST_TOKEN

OP_CODE[2]

      .
      .
      .
```

END_HLDF_TOKEN

giving it fixed-length records. This second approach of using an assembly language model for the design of the format of the HLDF itself has two alternatives:

▶ the op-codes can be fixed length encodings, e.g., a three-long character string; or

▶ they could be encoded using a Huffman-like encoding giving the more frequently-used op-codes shorter encodings and the less frequently used op-codes longer encodings.

A Huffman scheme can significantly reduce the size of the data set to be transmitted, but for our purposes here it is felt to be an unnecessary complication to the basic design. Accordingly, the proposed encoding for the op-codes will be fixed length. Note that this does NOT prevent any particular installation from using such a Huffman scheme for trans-mission, nor does it prevent a later modifica-

tion to the detailed design for the HLDF from using such a scheme.

(U) The op-code is a three-long character string representing the graphics entity being constructed or some information necessary for the reconstruction of the display. The elements of the ARGUMENT_LIST are separated by commas and not all arguments are mandatory.

(In the detailed description of the individual op-codes that follows, optional arguments are surrounded by square braces. The only other use of braces in that description is for array indices. Context easily distinguishes these two uses.)

The order of the op-codes in an HLDF is important to the structure of the display it represents in two ways:

▶ the overlaid structure of the display may depend on the order in which particular entities are displayed (the so-called "two-and-one-half" dimension hidden surface processing which allows pseudo-three-dimensional pictures to be constructed by judiciously choosing the order in which two-dimensional objects overlap and are displayed); and

▶ some control information necessary for the display of some entities is contained in op-code entries that precede the actual definition of the entity.

(U) The op-codes that have been defined to date are given below, separated into the logical categories of General Control, Geographics, Management, and Basic Graphics. Following these lists are detailed explanations of the individual op-codes.

● General Control

[ ]   Begin

[ ]   End

[ ]   Comment

[ ]   Viewport

[ ]   Plot Size

[ ]   Classification

● Geographics

[ ]   Map Controls

[ ]   Map

[] Geo_Marker

[] Geo_Line

[] Geo_Conic

[] Geo_Text

[] Geo_Polyline

[] Geo_Polygon

[] Geo_Military Symbol

● Management

[] Histogram

[] Pie Chart

[] Bar Chart

[] Timeline

[] Line Graph

● Basic Graphics

[] Text

[] Rectangle

[] Line

[] Marker

[] Polygon

[] Fill

[] Axes

[] Polyline

[] Circle

[] Arc

[] Image

[] Military Symbol

General Control Op-Codes

Op_Code: BEG   (Begin Segment)

Argument List: [Label, Line_Color, Line_Style, Line_Width, Character Attributes (Font, Text_Color, Character_Direction (x,y), Character_Path, Character_Size, Position_Precision)]

(U) Comments: The op-code 'BEG' allows for an Algol-like block structure to be imposed on the HLDF. Within the scope of a particular block (named with the 'Label' argument), one can set values for the current color, current line style, current line width, current character attributes, etc. These values will be used unless specifically overridden by different values in one of the op-codes within the block, and then these new values will be used only for that particular op-code. The initial 'BEG' of an HLDF, i.e., the 'BEG' of the outermost block must contain values for all the optional arguments in order to set the initial default values for this HLDF, since one of the design goals was that the interpretation of an HLDF not depend on the defaults for a particular system or user. Subsequent 'BEG' op-codes may contain only those arguments that are desired.

(U) Note: 'BEG' must be matched with the next most closely nested 'END'.

(U) The possible values for the various parameters are:

Label	Any text string
Line_Color	An HLS triple PLUS a Line, Style value. The Line Style value included here will be used ONLY IF the displaying system does not support color and, in this case, this Line Style overrides the normal Line Style. (See Appendix III or original report.)
Line_Style	{Solid, Dashed, Dotted, Dot-Dash}
Line_Width	An integer from 1 to 99. 1 corresponds to the narrowest line and 99 to the widest.
Character Attributes	The display attributes of any text string. (See Appendix IV of original report for details. This parameter has a number of possible subparameters explained below.
Font	{Roman, Italics, Roman Bold}
Text_Color	An HLS triple
Character_Direction	The direction vector for the writing direction of the text. Both X and Y must be integers from 0 to $2^{16} - 1$. Note that a

character direction of (0, 0) is not valid, since this vector has no direction.

Character_Path   {Right, Left, Up, Down} (See Appendix IV of original report for a description of how the character attributes work.)

Character_Size   An integer from 1 to $2^{16} - 1$ where $2^{16} - 1$ corresponds to a character filling the entire current viewport.

Position-Precision   (High, Medium, Low}

(U) Note: From this point on in the HLDF proposal, none of the individual character attributes will be explicitly named in the op-code description. These parameters will be referred to by the group name "Character Attributes". See appendix IV for further details.

Op_Code: END   (End Segment)

Argument List: [Label, Checksum]

(U) Comments : This op-code delineates the extent of a picture block. The optional, though strongly encouraged, argument must match the 'Label' of the corresponding 'BEG' op-code.

(U) The 'Checksum' argument provides a further level of redundancy and error detection for an HLDF file than is normally given to any other alphanumeric file by the system. If present, this argument should contain a checksum of the bytes in the concluded HLDF. If the receiving system finds a discrepancy between this count and the count of the received block, then that HLDF should be marked as defective and the transmitter should be notified to retransmit.

(U) It is possible that there would arise a situation where an HLDF would be constructed on one graphics system and then transferred via magnetic tape to another system (possibly with a different CPU and/or operating system) for transmission. This situation is important because in the writing to magnetic tape the HLDF may undergo some seemingly trivial modifications, such as replacing EOLs or <CR>s by <CR><LF>s, addition or deletion of nulls and of trailing spaces, conversion of TABs into spaces, addition of spaces/CRs at either end of the HLDF, and the like. It is desired that any checksum algorithm proposed for use with

the HLDF for the computation of the 'Checksum' argument be invariant under such transformations.

(U) This checksum provides only an additional level of error detection not error correction. It is modeled after (more properly, stolen from) the checksum proposed for the Caltech Intermediate Form (CIF) format for the transmission of VLSI designs over the ARPAnet.[9] The computational method proposed there may even be useful for the HLDF checksum and this checksum algorithm is presented in Appendix II of original report.

Op_Code: COM   (Operator Comment)

Argument List: String

(U) Comments : This non-graphic, non-executable op-code allows the originator of the ·HLDF to communicate a small amount of alphanumeric information to the displayer of the HLDF. This text string could contain such information as the originator's name, the classification (admittedly redundant, but not a bad idea anyway; see CLA below), etc. The displaying program may, for example, merely echo the 'String' to the associated alpha-numeric "device" when the HLDF is processed.

(U) Note that there is no theoretically imposed or a priori limit on the size of the 'String' argument, but a limit of 100 characters seems consistent with the intent of this op-code.

Op_Code: VPT   (Screen Viewport)

Argument List: Lower_Left_Corner (x,y), Upper_Right_Corner (x,y) (in the integer virtual coordinate system)

(U) Comments : This defines the portion of the display surface in which the picture corresponding to this HLDF is to be displayed. The default (that is, the value if this op-code is not present) is for a full-screen viewport. This op-code, if present, governs the display of all entities in its block. Accordingly, it must preceed all op-codes which construct graphic entities.

(U) This use of a 'VPT' op-code allows one HLDF to contain more than one viewport and allows the user to logically manipulate this image in terms of these viewports. The user could, for example, make only one of several viewports appear on the screen or could even change the viewport extent.

(U) This control op-code allows the user both to place several non-overlapping pictures

on one screen, as well as to specify a portion of the display for an inset (i.e, overlapping pictures). Because of this, the virtual coordinate system used in this block (with the exception of the 'VPT' op-code) is the full coordinate system, not a subset, and is such that its full extent is displayable in the viewport window. That is, the coordinate system inside any viewport is the same. A viewport using the integer virtual coordinate system uses the entire coordinate system regardless of the size of the viewport or its aspect ratio. This will allow for subsequent zoom opertations to be accurately displayed.

(U) The VPT op-code will be used for, among other things, legend boxes and small map inserts displayed at a different scale than the main map.

(U) Note that such an inset viewport may "lie on top of" a larger picture and the end user will be able to "look under" the inset by removing it. The picture underneath will not be affected by this use of a viewport.

Op_Code: SIZ (Plot Size)

Argument List: X_Extent (in hundredths of an inch), Y_Extent (in hundredths of an inch), String

(U) Comments: This op-code allows the originator of an HLDF to request a certain plot size when hardcopies are made. An 8.5" x 11" plot, for example, would be requested with the arguments (850, 1100). The 'String' argument allows for the passing of a small amount of alphanumeric information from the originator to the plotter operator.

Op_Code: CLA (Classification)

Argument List: Major_Classification, [Position, Character Attributes], [Codewords (Codeword[1], Codeword[2], ... , Codeword[n]), Caveats (Caveat[1], Position[1], Caveat[2], Position[2], ... , Caveat[n], Position[n], Character Attributes)]

(U) Comments: The classification for use by the receiver of the HLDF, for hardcopies, etc. The classification is set in this op-code in order to make it easily accessible by the receiving system. Some have suggested that the system not allow a graphic hardcopy to be made of any HLDF file that does not possess a CLA op-code somewhere in its structure.

(U) The possible values for the various parameters are:

Major Classification   {None, Unclassified, Secret, Top Secret}. Note that 'None' is the null classification; i.e, it will not appear on the graphic when the HLDF is interpreted.

Position   (x,y) in the integer virtual coordinate system or a lat/long value.

Character Attributes   (See 'BEG' op-code and/or Appendix IV of the original report.) (For Major Classification and Codewords only.)

Codewords   String. These strings will be appended to the Major Classification and will appear on the same line with it.

Caveats   String. These strings will be diplayed at their individually specified positions, typically on separate lines. One example of a caveat is "EYES ONLY DIRNSA". No maximum size is given for these caveats, but 50 seems reasonable.

Character Attributes   (See 'BEG' op-code and/or Appendix IV of the original report.) (For Caveats only.)

Geographic Op-Codes

Op_Code: MPC (Map Controls)

Argument List: Center (lat/long), Radius (nautical miles), Projection, Grid_Switch, [Grid_Height, Grid_Width,] Map_Line_Resolution, Altitude_Viewpoint

(U) Comments: This op-code must precede any of the geographic op-codes, and its parameters govern the display invoked by all subsequent geographic op-codes. At most one MPC op-code may be present within the scope of a single viewport (VPT), regardless of whether the VPT op-code is explicitly present or if it is implied for an entire HLDF by its absence. The 'Projection' argument takes its values from the set of cartographic projections, a set which currently includes Mercator, Gnomonic, Orthographic, Polar, Perspective, etc. The 'Grid_Switch' argument determines

whether or not a lat/long grid is to be displayed, and 'Grid_Height' and is displayed, in degrees (in DDDMMSS format). The 'Map_Line_Resolution' parameter gives the user some control over the precision of the map background. This parameter takes its values from {rough, medium, detailed}, with 'detailed' providing the largest amount of map detail. The value of 'detailed' for this parameter may be appropriate for very detailed maps of a small region or for the final high-quality plotting of maps for inclusion in product. The 'rough' value would be appropriate for a schematic of a large area or for a "quick and dirty" map for some types of preliminary analytic work. It is anticipated that the user-perceived response time for a graphics system to draw a detailed map will be significantly larger than for a rough map. The 'Altitude_Viewpoint' is a measure in nautical miles of the altitude of the viewpoint used in perspective cartographic projections.

Op_Code : MAP

Argument List : Feature_Level_of_Detail,
    (Feature_Name[1], Color[1]),
    (Feature_Name[2], Color[2]), ...,
    (Feature_Name[n], Color[n]),

(U) Comments : This op-code is the encoding for the map background data. The feature names are present only if they are to be displayed and are chosen from the following set : {International Boundaries; Coastlines, Islands, and Lakes; Rivers; Internal Boundaries; Military Districts; Air Corridors; Railroads; etc.} Any given feature name can be present in the argument list at most once. The 'Color' subparameters are chosen in accordance with the usual HLDF color encoding. (See the 'BEG' op-code and/or Appendix III of the original report.) and a particular "color" may be present in the argument list more than once. The 'Feature_Level_of_Detail' subparameter takes its values from the set {major, intermediate, all} with the value 'major' providing only the most prominent features and the value of 'all' providing all of that feature present in the map data base.

Op_Code : GMK   (Geographically_based Marker)

Argument List : Position (lat/long),
    Marker_Number (in the NSA Icon File),
    Scale, Orientation (as measured in degrees
    from the equator), [Mirror_Image_Flag,
    Text_String[1], Text_String_Position[1]
    (x,y), Character Attributes[1],
    Text_String[2], Text_String_Position[2]
    (x,y), Character Attributes[2], ... ,
    Text_String[n], Text_String_Position[n]
    (x,y), Character Attributes[n]]

(U) Comments : This op-code allows for the placement of marker symbols (e.g., circles, triangles, military unit symbols, icons of antennas or missiles, etc.) on a map. The 'Mirror_Image_Flag' takes its values from {on, off} and when 'on' causes the icon displayed to be a mirror image of the stored icon image. Since some of the symbols have a varying textual component, the 'Text_String' allows a user to associate this string with the marker. The position of the text string is expressed in the integer virtual coordinate system but with respect to the position of the marker. That is, the position of the text string is relative to that of the marker. Its character attributes are determined by the optional parameters, if present, or the current character attributes otherwise. Note that the Character_Direction is with respect to the coordinate system of the marker; i.e., if the icon is turned 5 degrees with respect to the equator, then the Text_String will be too if the horizontal Character Direction is used. The horizontal Character_Direction is (1, 0).) To angle just the Text_String with respect to the marker the Character_Direction parameter should be changed accordingly. The 'Scale' parameter controls the size of the displayed icon relative to its definition size. The values for this parameter range from 0 to 99, with the scale of the icons as stored in the NSA Icon file defined to be size 30. Some samples depicting the effect of these various parameters on the display of icons are shown in Figure 1.

(U) Note that (1) the application programs that change the size of a marker will probably also want to change the size of any attached text, but that this function is outside of the HLDF itself; (2) the display of any associated text string is not affected by the value of the 'Mirror_Image_Flag'; and (3) there is no 'Color' parameter to this op-code since the Icon file itself is assumed to store multicolored icons.

Op_Code : GLN   (Geographically-based Line)

Argument List : Endpoint[1], Endpoint[2], (Both
    expressed in lat/long), Direction_Flag,
    [Line_Style, Color, Width]

(U) Comments : This op-code allows for the placement of geographically-based lines. The 'Direction_Flag' is set to TRUE if the line direction (from Endpoint[1] to Endpoint[2]) is to be indicated (e.g., with arrowheads). Other optional arguments allow for the temporary override of the current color, line style, and width. Note that this line will not, in general, be a straight line on the display; but rather will be seen by the viewer as the result of a cartographic projection of

Appendix III of the ori-
ginal report for details.

Number_of_Title_Lines   [0 .. 4]

Title[1:Number_of]   Character string

X_Axis_Ticks, $[0 .. (2^{16}.)]$
  Y_Axis_Ticks

X_Axis_Label[1:n],   Character string
  Y_Axis_Label[1:n]

(U) Note that the actual data from which
the histogram was constructed is <u>not</u> stored in
the HLDF, but rather only the data necessary
to reconstruct the display.

Op_Code: PIE (Pie Chart)

Argument List: Number_of_Sectors, Percen-
tage[1], Percentage[2], ...,
Percentage[Number_of_Sectors], Label[1],
Label[2], ..., Label[Number_of_Sectors],
Label_Position[1], Label_Position[2], ... ,
Label[Number_of_Sectors],
Explode((Sector[1], Distance[1]), (Sec-
tor[2], Distance[2]), ... , (Sector[m],
Distance[m])),
Percentage_Label_Position[1],
Percentage_Label_Position[2], ... ,
Percentage_Label_Position[Number_of_Sectors],
Radius, Color[1], Color[2], Color[3], ... ,
Color[Number_of_Sectors], Pattern[1], Pat-
tern[2], Pattern[3], ... ,
Pattern[Number_of_Sectors], Starting_Angle,
Edge_Style

(U) Comments: An encoding of a standard pie
chart allowing for exploded sectors and two
types of labels. The "Starting Angle" parame-
ter is the angle from the horizontal for the
start of sector 1. This is required in order
to insure that the pie chart will be displayed
in exactly the same orientation on different
systems. All other parameters are self-
explanatory.

Op_Code: BAR (Bar Chart)

Argument List: Style, X_Axis_Label,
Y_Axis_Label, Number_of_Families,
Number_of_Bars, Amount1[1], Amount1[2],
... , Amount1[Number_of_Bars], [Amount2[1],
Amount2[2], ... , Amount2[Number_of_Bars],
Amount3[1], Amount3[2], ... ,
Amount3[Number_of_Bars]], Color[1], Size
(Left_Top_Corner, Right_Bottom_Corner),
[Color[2], Color[3], Color[4],
Number_of_Title_Lines, Title_Line[1],
[Title_Line[2], Title_Line[3]],
Legend_Position, Legend_Name[1],
[Legend_Name[2],

Legend_Name[Number_of_Title_Lines]],
Legend_Style, X_Axis_Tic_Mark_Labels[1:n],
X_Label_Position,
Y_Axis_Tic_Mark_Labels[1:n],
Y_Label_Position

(U) Comments: (To be determined)

Op_Code: TML (Timeline)

Argument List: (To be determined)

(U) Comments: (To be determined)

Op_Code: LNG (Line Graph)

Argument List: Number_of_Families,
X_Axis_Label, Y_Axis_Label, X_Range
(X_High, X_Low), Y_Range (Y_High, Y_Low),
Family[1] (Number_of_Line_Segments, (X[1],
Y[1]), (X[2], Y[2]), ... ,
(X[Number_of_Line_Segments],
Y[Number_of_line_Segments])), [Family[2]
(Number_of_Line_Segments, (X[1], Y[1]),
(X[2], Y[2]), ... ,
(X[Number_of_Line_Segments],
Y[Number_of_line_Segments])), ... ,
Family[Number_of_Families]
(Number_of_Line_Segments, (X[1], Y[1]),
(X[2], Y[2]), ... ,
(X[Number_of_Line_Segments],
Y[Number_of_line_Segments])),] Color[1],
[Color[2], ... ,
Color[Number_of_Families],]
Number_of_Title_Lines, Title_Line[1],
[Title_Line[2], Title_Line[3],]
X_Axis_Tic_Mark_Labels[1:n],
X_Tic_Mark_Interval,
Y_Axis_Tic_Mark_Labels[1:n],
Y_Tic_Mark_Interval

(U) Comments: (To be determined)

Basic Graphics Op-Codes

Op_Code: TXT (Text)

Argument List: String, Position, [Character
Attributes]

(U) Comments: (None. See Appendix IV of the
original report for details.)

Op_Code: RCT (Rectangle)

Argument List: Lower_Left_Corner,
Upper_Right_Corner, [Fill_Color,
Fill_Pattern, Orientation]

FOR OFFICIAL USE ONLY

(U) Comments: "Orientation" is measured in degrees from the X-axis direction. Possible "Fill Pattern" values are shown in Figure 6 of Appendix IV (in the original report).

Op_Code: LIN (Line)

Argument List: Endpoint[1], Endpoint[2], [Width, Line_Style, Color]

(U) Comments: (None)

Op_Code: MRK (Marker)

Argument List: Position (x, y), Marker_Number, Scale, Orientation (as measured in degrees from the X axis), [Mirror_Image_Flag, Text_String[1], Text_String_Position[1] (x,y), Character Attributes[1], Text_String[2], Text_String_Position[2] (x,y), Character Attributes[2], ... , Text_String[n], Text_String_Position[n] (x,y), Character Attributes[n]]

(U) Comments: (Same as Geographically based Marker, GMK.)

Op_Code: PLG (Polygon)

Argument List: Number_of_Points, Point[1] Point[2], ..., Point[Number_of_points], Attributes (Interior_Style, Interior_Color, Edge_Color, Edge_Style, Edge_Width, Vertex_Color, Vertex_Size)

(U) Comments: Point[1] is both the starting and ending point of the polygon. 'Vertex_Size' is the size of the dot used for the vertex. The meaning of the other arguments is self-explanatory. The values for the Attribute subparameters are:

Interior_Style	(To be determined)
Interior_Color	HLS value PLUS Fill Pattern
Edge_Color,	HLS value PLUS Line Style
Vertex_Color	HLS value
Edge_Style	(Solid, Dashed, Dotted, Dot-Dash)
Edge_Width	{Narrow, Normal, Wide}
Vertex_Size	[0 .. 99]

Op_Code: FIL (Fill)

Argument List: Interior_Point (x,y),

Fill_Color, Fill_Pattern

(U) Comments: This command will be used to fill an area of the screen not defined as a polygon (using the PLG command above), such as the region bounded by three mutually inter-secting lines. Fill_Color is an HLS value PLUS a Fill Pattern. See Appendix III of ori-ginal report.

Op_Code: AXS (Axes)

Argument List: X_Lower_Bound, X_Upper_Bound, Y_Lower_Bound, Y_Upper_Bound, [X_Axis_Tick_Mark_Labels[1 m], X_Tick_Mark_Interval, Y_Axis_Tick_Mark_Labels[1 m], Y_Tick_Mark_Interval, Line_Style, Width, Color]

(U) Comments: (To be determined)

Op_Code: PLN (Polyline)

Argument List: Number_of_Points, Point[1] (X, Y), Point[2], ..., Point[Number_of_Points], [Line_Style, Width, Color]

(U) Comments: (None)

Op_Code: CIR (Circle)

Argument List: Center (X, Y), Radius, [Fill_Style, Fill_Color], Border_Style, Border_Color]

(U) Comments: (None)

Op_Code: ARC (Arc)

Argument List: Center (X, Y), Radius, Start_Angle, Stop_Angle, [Fill_Style, Fill_Color], Border_Style, Border_Color]

(U) Comments: The parameters 'Start_Angle' and 'Stop_Angle' are measured in degrees from the horizontal.

Op_Code: MIS (Military Symbol)

Argument List: Position (x, y), Flag, Symbol_Size, [Primary_Duty, Secondary_Duty, Staff, Base, Associated Text (Text_String[1], Text_String_Position[1], Character Attributes[1], Text_String[2], Text_String_Position[2], Character Attri-butes[2], ... , Text_String[n], Text_String_Position[n], Character Attributes[n]), Size, Color, Interior_Color]

FOR OFFICIAL USE ONLY

(U) Comments: These entities denote the positions of particular types of military units. They are derived from the "standard" military symbols. For further details, see Appendix V of original report.

Op_Code: IMG (Image)

Argument List: (To be determined)

(U) Comments: A pixel array that is to be placed within a certain portion of the current viewport. It is through this image op-code that digital images can be placed in the background of an HLDF with the possibility of adding additional graphics "on top of" the digital image.

Operations involving an HLDF

(U) The operations which are to be performed on a data structure (and the frequency with which they will be performed) can be used to guide the decisions in its implementation. The operations which will be performed on the HLDF are (roughly in order of frequency):

▶ Execute (i.e., construct the display represented by some HLDF, thus making that HLDF the current HLDF.) The HLDF to be executed will usually be stored on disk.

▶ Add a new graphics entity.

▶ Store (on disk).

▶ Transmit (over PLATFORM).

▶ Search for some entity by op-code and some parameters.

▶ Modify some entity (e.g., change color, edit text).

▶ Search for some entity based on the values of some parameters (no op-code).

▶ Delete some entity.

▶ Overlay the display represented by a given HLDF on the current display, thus concatenating the given HLDF to the current HLDF.

??? UNRESOLVED QUESTIONS ???

(U) There are two approaches to the inclusion of management graphics in the HLDF design. One is to try to parameterize the seemingly infinite varieties of such charts (e.g., the MANY possible styles of line graphs). This is the approach that was taken,

with perhaps some success. This approach can be termed the "All things to all people" approach. The other approach is to standardize a couple of line graph formats, a couple of bar chart formats, etc. These would then be the ones compactly encoded by an HLDF. If an end user had to have a different format, it could still be encoded using the basic graphics safety net, albeit with some loss of later functionality with regard to later modifications of that HLDF. In this second approach, there would be a FAMILY of op-codes for, say, line graphs: LGA, LGB, LGC, ... Implementation would probably be simplified. This approach can be called the "Family of Fixed Formats" approach. Which approach is better in the long run for the HLDF?

Appendix I: HLDF Examples

(U) Following are some sample HLDF encodings. Each has been done "by hand" from a given display in order to simulate the HLDF that would be output, so the reader should not be concerned about the third or fourth decimal place for the numerical values, but rather the overall structure. All examples closely resemble Agency product but all are taken from unclassified open-source material. In order to increase the readability of these examples, the format used will be "Variable = VALUE" where the "Variable" is some argument name and "VALUE" is the value for that argument. Most probably this will not be the format employed for an actual HLDF implementation and, accordingly, the values given for the sizes of these example HLDFs will be character counts for the VALUEs only.

(U) Unless otherwise indicated, all the figures in this section are from [10].

Example 1: Geographic HLDF with Icons and Inset
(Shown in Figure 1)

(Size of this HLDF is 4,152 characters)

OP_CODE	ARGUMENT LIST
BEG	Label = EXAMPLE1, Line-Width = 1, Character Attributes (Font = ROMAN, Text_Color = (0, 0, 0), Character_Direction = (1,0), Character_Path = RIGHT, Character_Size = 500, Position_Precision = HIGH.
COM	String = "BLACK AND WHITE GRAPHIC, HIGH PRECISION. UNCLASSIFIED"
CLA	Major_Classification = NONE

Figure 1: Geography with Icons and Legend

MPC   Center = (354000N,1341200E), Radius = 893, Projection = MERCATOR, Grid_Switch = OFF, Level_of_Detail = DETAILED, Altitude_Viewpoint = 1000

MAP   (COASTLINES, SOLID), (INTERNATIONAL BOUNDARIES, DOT-DASH), (MAJOR ISLANDS, SOLID), (MINOR ISLANDS, SOLID), (MAJOR LAKES, SOLID)

GMK   Position = (410000N, 1412000E), Marker_Number = 122, Scale = 30, Orientation = -5, Text_String = "Misawa," Character Attributes (Font = ROMAN BOLD, Position = (600, 100), Character_Size = 700)

GMK   Position = (434000N, 1404200E), Marker_Number = 3, Scale = 30, Orientation = 0, Text_String = "Sapporo," Character Attributes (Position = (300, 0), Character_Size = 200)

GMK   Position = (424200N, 1321000E), Marker_Number = 3, Scale = 30, Orientation = 0, Text_String = "Vladivostok," Character Attributes (Position = (100, 300), Character_Size = 200)

GMK   Position = (372000N, 1265200E), Marker_Number = 62, Scale = 30, Orientation = 0, Text_String = "Pyongyang," Character Attributes (Position = (300, 0), Character_Size = 200)

GMK   Position = (365200N, 1271000E), Marker_Number = 39, Scale = 30, Orientation = -5, Text_String = "Camp Casey," Character Attributes (Font = ROMAN BOLD, Position = (400, 0), Character_Size = 700)

GMK   Position = (364100N, 1263200E), Marker_Number = 39, Scale = 30, Orientation = -5, Text_String = "Yongsan," Character Attributes (Font = ROMAN BOLD, Position = (-2048, -300), Character_Size = 700)

GMK   Position = (363700N, 1265200E), Marker_Number = 62, Scale = 30, Orientation = 0, Text_String = "Seoul," Character Attributes (Position = (200,50), Character_Size = 700)

GMK   Position = (360200N, 1273300E), Marker_Number = 122, Scale = 30, Orientation = -5, Text_String = "Osan," Character Attributes (Posi-

tion = (300, 50), Character_Size = 700)

GMK   Position = (354200N, 1264600E), Marker_Number = 122, Scale = 30, Orientation = -5, Text_String = "Kunsan," Character Attributes (Font = ROMAN BOLD, Position = (-2000, 0), Character_Size = 700)

GMK   Position = (353841N, 1285500E), Marker_Number = 122, Scale = 30, Orientation = -5, Text_String = "Taegu," Character Attributes (Font= ROMAN BOLD, Position = (200, 50), Character_Size = 700)

GMK   Position = (354000N, 1400000E), Marker_Number = 62, Scale = 30, Orientation = 0, Text_String = "Tokyo," Character Attributes (Position = (-1024, 200), Character_Size = 200)

GMK   Position = (353000N, 1382216E), Marker_Number = 122, Scale = 30, Orientation = -5, Text_String = "Atsugi," Character Attributes (Font = ROMAN BOLD, Position = (-2000, 100), Character_Size = 700)

GMK   Position = (344225N, 1413000E), Marker_Number = 164, Scale = 30, Orientation = -5, Text_String = "Yokosuka," Character Attributes (Font = ROMAN BOLD, Position = (300, -75), Character_Size = 700)

GMK   Position = (344016N, 1293511E), Marker_Number = 3, Scale = 30, Orientation = 0, Text_String = "Pusan," Character Attributes (Position = (-100, -40), Character_Size = 200)

GMK   Position = (335750N, 1321047E), Marker_Number = 122, Scale = 30, Orientation = -5, Text_String = "Iwakuni," Character Attributes (Font = ROMAN BOLD, Position = (-2500, 100), Character_Size = 700)

GMK   Position = (332712N, 1295842E), Marker_Number = 164, Scale = 30, Orientation = -5, Text_String = "Sasebo," Character Attributes (Font = ROMAN BOLD, Position = (-1500, 100), Character_Size = 700)

GMK   Position = (290502N, 1281116E), Marker_Number = 122, Scale = 30, Orientation = -5, Text_String = "Fatanma," Character Attributes (Font = ROMAN BOLD, Position = (175, 0), Character_Size = 700)

GMK   Position = (281942N, 1273000E), Marker_Number = 122, Scale = 30, Orientation = -5, Text_String = "Kadena," Character Attributes (Font = ROMAN BOLD, Position = (-900, 300), Character_Size = 700)

GMK   Position = (360103N, 1402213E), Marker_Number = 122, Scale = 30, Orientation = -5, Text_String = "Yokota," Character Attributes (Font = ROMAN BOLD, Position = (600, 100), Character_Size = 700)

GTX   String = "Soya Strait," Position = (442020N, 1395216E), Character Attributes (Font = ITALICS, Character_Size = 200)

GTX   String = "Kuril," Position = (450311N, 1452011E), Character Attributes (Character_Size = 800)

GTX   String = "Islands," Position = (451016N, 1441022E), Character Attributes (Character_Size = 800)

GTX   String = "U.S.S.R.," Position = (444517N, 1332037E), Character Attributes (Character_Size = 800)

GTX   String = "CHINA," Position = (442002N, 1264712E), Character Attributes (Character_Size = 800)

GTX   String = "Lake," Position = (441001N, 1323000E), Character Attributes (Font = ITALICS, Character_Size = 500)

GTX   String = "Khanka," Position = (435716N, 1322511E), Character Attributes (Font = ITALICS, Character_Size = 500)

GTX   String = "Hokkaido," Position = (434123N, 1403719E), Character Attributes (Character_Size = 800)

GTX   String = "Tsugaru," Position = (413000N, 1382051E), Character Attributes (Font = ITALICS, Character_Size = 200)

GTX   String = "Strait," Position = (411000N, 1383116E), Character Attributes (Font = ITALICS, Character_Size = 200)

GTX   String = "Sea of Japan," Position = (400000N, 1331627E), Character Attributes (Font = ITALICS, Character_Size = 800)

GTX   String = "NORTH," Position =

(402632N, 1263023E), Character Attributes (Character_Size = 800)

GTX   String = "KOREA," Position = (392518N, 1263023E), Character Attributes (Character_Size = 800)

GTX   String = "Pacific," Position = (395756N, 1420106E), Character Attributes (Font = ITALICS, Character_Size = 800)

GTX   String = "Ocean," Position = (384016N, 1370000E), Character Attributes (Font = ITALICS, Character_Size = 800)

GTX   String = "JAPAN," Position = (384016N, 1370000E), Character Attributes (Font = ROMAN BOLD, Character_Size = 1200)

GTX   String = "Demilitarized," Position = (383012N, 1290037E), Character Attributes (Font = ROMAN BOLD, Character_Size = 700)

GTX   String = "Zone (DMZ)," Position = (380217N, 1290037E), Character Attributes (Font = ROMAN BOLD, Character_Size = 700)

GLN   Endpoint[1] = (383012N, 1290000E), Endpoint[2] = (381612N, 1283000E), Width = 50

GTX   String = "SOUTH," Position = (371024N, 1294500E), Character Attributes (Font = ROMAN BOLD, Character_Size = 1200)

GTX   String = "KOREA," Position = (371024N, 1294500E), Character Attributes (Font = ROMAN BOLD, Character_Size = 1200)

GTX   String = "Honshu, Position = (373012N, 1383026E), Character Attributes (Character_Size = 800)

GTX   String = "Tsushima (Korea) Strait," Position = (333243N, 1262247E), Character Attributes (Font = ITALICS, Character_Size = 200, Character_Direction = (1, 1))

GTX   String = "Cheju-do," Position = (244019N, 1335247E), Character Attributes (Character_Size = 700)

GTX   String = "Shikoku," Position = (332143N, 1324237E), Character Attributes (Character_Size = 800)

GTX   String = "Kyushu," Position =

(325947N, 1303017E), Character Attributes (Character_Size = 800)

GTX   String = "East," Position = (320627N, 1252742E), Character Attributes (Font = ITALICS, Character_Size = 800)

GTX   String = "China," Position = (313000N, 1251000E), Character Attributes (Font = ITALICS, Character_Size = 800)

GTX   String = "Sea," Position = (310941N, 1251000E), Character Attributes (Font = ITALICS, Character_Size = 800)

GTX   String = "RYUKYU," Position = (300000N, 1263047E), Character Attributes (Character_Size = 750)

GTX   String = "ISLANDS," Position = (294738N, 1263000E), Character Attributes (Character_Size = 750)

GTX   String = "Philippine," Position = (300000N, 1330000E), Character Attributes (Font = ITALICS, Character_Size = 800)

GTX   String = "Sea," Position = (293041N, 1341258E), Character Attributes (Font = ITALICS, Character_Size = 800)

GTX   String = "Bonin," Position = (295907N, 1413137E), Character Attributes (Character_Size = 700)

GTX   String = "Islands," Position = (294251N, 1410939E), Character Attributes (Character_Size = 700)

GTX   String = "Iwo," Position = (280944N, 1421033E), Character Attributes (Character_Size = 700)

GTX   String = "Jima," Position = (280000N, 1421033E), Character Attributes (Character_Size = 700)

BEG   Label = INSET1, Character Attributes (Font = ROMAN BOLD, Character_Size = 500)

VPT   Lower_left_Corner = (12739, 1024), Upper_right_Corner = (28221, 14327)

GTX   String = "Legend:," Position = (4096, 61440) Character Attributes (Font = ITALICS, Character_Size = 300)

MRK   Position = (8192, 50462),

Marker_Number = 122, Scale = 30, Orientation = 0, Text_String = "Airfield," Character Attributes (Position = (8000, 0))

MRK   Position = (8192, 43253), Marker_Number = 164, Scale = 30, Orientation = 0, Text_String = "Naval Facility," Character Attributes (Position = (8000, 0))

MRK   Position = (8192, 35389), Marker_Number = 39, Scale = 30, Orientation = 0, Text_String = "Army Command Headquarters," Character Attributes (Position = (8000, 0))

LIN   Endpoint[1] = (6000, 5400), Endpoint[2] = (50000, 5400)

TXT   String = "0," Position = (6000, 6100), Character Attributes (Font = ROMAN, Character_Size = 175)

TXT   String = "0," Position = (6000, 4800), Character Attributes (Font = ROMAN, Character_Size = 175)

TXT   String = "300 Kilometers," Position = (32768, 6100), Character Attributes (Font = ROMAN, Character_Size = 175)

TXT   String = "300 Miles," Position = (49000, 4700), Character Attributes (Font = ROMAN, Character_Size = 175)

END   Label = INSET1

END   Label = EXAMPLE1, Checksum = ****

Example 2: Geographic HLDF
with Arrows as Geo-Polygons
(Shown in Figure 2)

(Size of this HLDF is 2,847 characters)

OP_CODE   ARGUMENT LIST

BEG   Label = EXAMPLE2, Line_Color = ((0, 0, 0), SOLID), Line_Style = SOLID, Line-Width = 50, Character Attributes (Font = ROMAN, Text_Color = (0, 0, 0), Character_Direction = (1,0), Character_Path = RIGHT, Character_Size = 700, Position_Precision = HIGH.

COM   String = "BLACK AND WHITE GRAPHIC,   MAP   (COASTLINES, SOLID), (INTERNA-
         HIGH PRECISION. UNCLASSIFIED"   TIONAL BOUNDARIES, DOT-DASH),
                                                           (MAJOR ISLANDS, SOLID)

CLA   Major_Classification = NONE
                                                  GTX   String = "SCHLESWIG-," Position =
VPT   Lower_Left_Corner = (4000, 4000),   (544026N, 0084512E)
         Upper_Right_Corner = (61000, 44500)

                                                  GTX   String = "HOLSTEIN," Position =
MPC   Center = (504900N,0100000E),   (542216N, 0092000E)
         Radius = 310, Projection = ORTHO-
         GRAPHIC, Grid_Switch = OFF,   GTX   String = "EAST," Position =
         Level_of_Detail = ROUGH,   (534500N, 0123000E), Character
         Altitude_Viewpoint = 1000   Attributes (Character_Size = 1000)

Figure 2: Geography with Arrows

GTX   String = "GERMANY," Position = (533000N, 0104000E), Character Attributes (Character_Size = 1000)

GTX   String = "WEST," Position = (484848N, 0104101E), Character Attributes (Character_Size = 1000)

GTX   String = "GERMANY," Position = (482701N, 0091216E), Character Attributes (Character_Size = 1000)

GTX   String = "NETHERLANDS," Position = (514200N, 0055500E), Character Attributes (Character_Direction = (1,2))

GTX   String = "(NORTHAG)," Position = (511000N, 0064200E), Character Attributes (Character_Size = 1100)

GTX   String = "(CENTAG)," Position = (492302N, 0072847E), Character Attributes (Character_Size = 1100)

GTX   String = "BEL.," Position = (501500N, 0053616E), Character Attributes (Character_Direction = (3, 1))

GTX   String = "LUX.," Position = (494218N, 0055732E)

GTX   String = "CZECHOSLOVAKIA," Position = (493912N, 0120246E), Character Attributes (Character_Direction = (3, 1))

GTX   String = "FRANCE," Position = (483606N, 0054232E), Character Attributes (Character_Direction = (2, 1))

GTX   String = "SWITZERLAND," Position = (470319N, 0064500E)

GTX   String = "AUSTRIA," Position = (470330N, 0105739E) Character Attributes (Character_Direction = (4, 1))

GMK   Position = (534016N, 0083127E), Marker_Number = 16, Scale = 20, Orientation = 0, Text_String = "Bremerhaven," Character Attributes (Font = ITALICS, Position = (500, 0), Character_Size = 400)

GMK   Position = (494201N, 0093522E), Marker_Number = 16, Scale = 20, Orientation = 0, Text_String = "Wurzburg," Character Attributes (Font = ITALICS, Position = (-5000, 100), Character_Size = 400)

GMK   Position = (485226N, 0085814E), Marker_Number = 16, Scale = 20, Orientation = 0, Text_String = "Stuttgart," Character Attributes (Font = ITALICS, Position = (500, 0), Character_Size = 400)

GMK   Position = (500214N, 0082237E), Marker_Number = 16, Scale = 20, Orientation = 0, Text_String = "Frankfurt," Character Attributes (Font = ITALICS, Position = (-5000, 0), Character_Size = 400)

GPL   Number_of_Points = 9, Point[1] = (534016N, 0083127E), Point[2] = (533022N, 0083923E), Point[3] = (523000N, 0093410E), Point[4] = (515000N, 0093520E), Point[5] = (510400N, 0092214E), Point[6] = (513516N, 0092400E), Point[7] = (494201N, 0093522E), Point[8] = (492314N, 0090416E), Point[9] = (485226N, 0085814E), Line_Style = DOT-DASH, Width = 50

GTX   String = "MAJOR LINE OF," Position = (533314N, 0083127E), Character Attributes (Character_Direction = (1, -1))

GTX   String = "COMMUNICATION," Position = (532742N, 0082214E), Character Attributes (Character_Direction = (1, -1))

GMK   Position = (524000N, 0123316E), Marker_Number = 23, Scale = 20, Orientation = 0, Text_String = "Berlin," Character Attributes (Font = ITALICS, Position = (500, 0), Character_Size = 400)

GMK   Position = (504227N, 0070123E), Marker_Number = 23, Scale = 20, Orientation = 0, Text_String = "Bonn," Character Attributes (Font = ITALICS, Position = (300, 200), Character_Size = 400)

GTX   String = "XXXX," Position = (520000N, 0081000E) Character Attributes (Character_Direction = (1, 1), Character_Size = 600)

GPL   Number_of_Points = 7, Point[1] = (515842N, 0105847E), Point[2] = (512036N, 0090415E), Point[3] = (511627N, 0083057E), Point[4] = (510243N, 0081057E), Point[5] = (504000N, 0073000E), Point[6] = (503229N, 0070000E), Point[7] = (502234N, 0061000E), Width = 50

GPG   Number_of_Points = 24, Point[1] =

```
              (533742N, 0115103E), Point[2] =
              (523758N, 0113000E), Point[3] =
              (522903N, 0113000E), Point[4] =
              (522000N, 0075537E), Point[5] =
              (522600N, 0075426E), Point[6] =
              (521000N, 0072800E), Point[7] =
              (515800N, 0075530E), Point[8] =
              (520008N, 0075530E), Point[9] =
              (520900N, 0113428E), Point[10] =
              (514000N, 0120000E), Point[11] =
              (504216N, 0080937E), Point[12] =
              (504937N, 0080812E), Point[13] =
              (503000N, 0075116E), Point[14] =
              (502616N, 0081422E), Point[15] =
              (502811N, 0081152E), Point[16] =
              (511923N, 0115211E), Point[17] =
              (492437N, 0095723E), Point[18] =
              (492540N, 0095103E), Point[19] =
              (490023N, 0094728E), Point[20] =
              (491316N, 0102347E), Point[21] =
              (491522N, 0101833E), Point[22] =
              (512000N, 0122116E), Point[23] =
              (511826N, 0123142E), Point[24] =
              (512631N, 0124106E), Width = 70

GTX      String = "NORTH GERMAN PLAIN,"
         Position = (520101N, 0075916E),
         Character Attributes
         (Character_Direction = (15,1), Font
         = ROMAN BOLD, Character_Size =
         1000)

GTX      String = "FULDA GAP," Position =
         (504223N, 0090400E), Character
         Attributes (Character_Direction =
         (4,1), Font = ROMAN BOLD,
         Character_Size = 1000)

GTX      String = "HOF CORRIDOR," Position
         = (493316N, 0103000E), Character
         Attributes (Character_Direction =
         (1,2), Font = ROMAN BOLD,
         Character_Size = 1000)

END      Label = EXAMPLE2, Checksum = ****
```

References

[1] Graphic Standards Planning Committee, "Status Report of the Graphic Standards Planning Committee," Computer Graphics, Volume 13, Number 3, August 1979 (Special Issue).

[2] International Standards Organization, Information Processing Graphics Kernel System (GKS), Draft International Standards Organization Standard (ISO/DIS) (ISO TC97/SC5/WG2 N117) (ANSI X3H3/82-10), 14 January 1982.

[3] Tektronix, Graphic Model Exchange Format Functional Description, GCS Software Standard, 5 November 1980.

[4] Enderle, G., I. Giese, M. Krause, and H. P. Meinzer, "The AGF Plotfile towards a Standardization for Storage and Transportation of Graphics Information," Computer Graphics, Volume 12, Number 4, (December 1978), pp. 92-113.

[5] Warner, Jim, "Device-Independent Intermediate Display Files," Computer Graphics, Volume 13, Number 1, (March 1979), pp. 78-109.

[6] Teus Hagen, Paul J. W. Ten Hagen, Paul Klint, and Hans Noot, "The Intermediate Language for Pictures," Information Processing 77, (B. Gilchrist, Ed.) International Federation for Information Processing and North-Holland Publishing Company, The Hague, (1977), pp. 173-178.

[7] National Security Agency, BETA--A Cryptologically Oriented Advanced Programming Language, May 1970, S-196,383.

[8] Mead, Carver, and Lynn Conway, Introduction to VLSI Systems, Addison-Wesley, New York, 1980, pp. 115-127.

[9] Cohen, Danny, A Proposed End/End Check-Sum Option for CIF Files, USC/ISI Unpublished manuscript, 28 April 1978

[10] Collins, John M., U.S.--Soviet Military Balance--Concepts and Capabilities 1960-1980, (New York: McGraw-Hill, 1980).

[11] American National Standards Institute Committee on Computer Graphics Languages (ANSI X3H3), American National Standard Functional Specification of the Programmers's Minimal Interface to Graphics, Working Document ANSI X3H3 / 82-15, 19 February 1982.

OCID: 4011963

A SURVEY OF PARALLEL SORTING (U)

EO 1.4.(c)
P.L. 86-36

by

E4

Donald Knuth [6] says that computer manufacturers estimate that over 25 percent of the running time on their computers is devoted to sorting when (C) all customers are taken into account. No one in the Agency is as willing to estimate the amount of sorting that we do, though informal estimates put our sorting use at about five percent. This paper will review several parallel sorting algorithms, particularly in the context of sorting very large files.

(U) Thus our file of tapes functions as a massive random access memory. To locate a data item, we first must determine the tape it is in, and then run through the tape to the desired word. Even with the efficiency gained by sorting the tapes, access in this data base is a lengthy process. The access time includes the mounting of the tape and the time to run through (on the average) half of the tape—about two minutes total.

(S CCO) Some cryptanalytic programs on our various computer systems are limited as to the amount of data they can handle. By presorting the data into pockets, we can produce small data sets that can be handled by the programs. The remaining data can then be used for secondary testing.

(U) Generally speaking, sorting N words requires $O(N \log N)$ word comparisons. On a standard serial computer, $\log N$ passes over the N data items usually produces a sorted list. Parallel processing enables us to make some comparisons simultaneously. The parallel algorithms discussed below can decrease the running time from about $N \log N$ to $\log^2 N$ or $\log N$, but at the expense of making a larger total number of comparisons.

(U) Note that throughout this paper we use $\log N$ to stand for the base 2 logarithm of N. Thus $\log 32 = 5$.

PARALLEL ALGORITHMS

(U) The acronym SIMD stands for single instruction stream multiple data stream. An SIMD array processor is well suited to take advantage of parallelism in many algorithms. Both ILLIAC IV and Staran are SIMD machines and this is the type of parallel processor most seriously considered for construction. If array processors become commercially available, they will probably be of the SIMD design. For this reason, it is important to study and understand algorithms for various interconnection networks in SIMD array processors.

(U) An SIMD machine usually has a central controller, N processors, N memories, and an interconnection network. The central controller issues a single instruction to all processors which in turn operate on their memories (multiple data) as required. A mask register permits certain processors to be inactive during an instruction, but only one operation is performed at one time by each active processor. The interconnection network allows the processors to communicate with each other. In general, each processor is connected to between $O(1)$ and $O(\log N)$ other processors.

HIRSHBERG'S BUCKET SORT ALGORITHM

(U) D.S. Hirshberg of Rice University has developed a parallel sorting algorithm for SIMD machines [4] which requires time $O(\log N)$. His SIMD model assumes that there is a common memory which can be accessed by all processors. Simultaneous access to a memory location is not allowed for stores, but may be allowed for fetches.

(U) Hirshberg's algorithm is a parallel version of the "bucket sort."[7] Assume that the numbers to be sorted C_i, are from $0, \ldots, M-1$ where $M \leq N$. In the common memory there is a "bucket" devoted to each processor. If there are no repetitions among the C_i, and if each processor p_i (which has been temporarily assigned to C_i, the ith number to be sorted) places the value i in bucket B_j, where $j = C_i$, then bucket i contains the address of the jth sorted word. An example of this for $N = 8$ is given in Figure 1.

i	C_i	B_i
0	3	1
1	0	5
2	6	3
3	2	0
4	7	7
5	1	6
6	5	2
7	4	4

Figure 1

(U) In general, there may be repetitions in the C_i's, and this could cause store conflicts if more than one processor tried to write to its address in a bucket. We can avoid this by assigning N locations per bucket so that each processor has a unique place in each bucket to make conflict-free marks (only a mark, not the processor number i, is now written.) An example of this is shown in figure 2, where $N = 16$ and $0 \leq C_i \leq 11$. Consider specifically the case of bucket 1. Processors 5, 8, 9, and 15

have made their marks in their respective portions of the bucket.

Figure 2

```
C_i  i    0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

4   0    | |*| | | | | | | | | |  |  |  |  |  |
0   1    | | | | | | |*| | |*|*|  |  |  |  | |*|
2   2    | | |*| | | | | | | |  |  |  |  | |*| |
7   3    | | | | | | | |*| | |  |  |*|  |  |  |
9   4    |*| | | | | | | | | |  |  |  |  |  |  |
1   5    | | | | | | |*| | | |  |  |  |  |  |  |
5   6    | | | | | | | | | | |  |*|  |  |  |  |
3   7    | | | |*| | | | | | |  |  |  |*|  |  |
1   8    | | | | | | | | | | |  |  |  |  |  |  |
1   9    | | | | |*| | | | | |  |  |  |  |  |  |
6  10    | | | | | | | | | | |  |  |  |  |  |  |
11 11    | | | | | | | | | | |  |  |*|  |  |  |
```

(U) If each bucket comprises one memory location, we must avoid multiple accesses to a bucket. We need a technique to allow processors to "sense" the presence of other processors in the bucket and deactivate themselves if necessary, leaving one active processor in each bucket.

(U) We will use the "buddy system," analogous to the buddy system for dynamic memory allocation.[6] If $\oplus$ represents bit-by-bit mod 2 addition, and processor i is marking location j, then the kth buddy of processor i is $j \oplus 2^k$. Each processor determines whether its buddy is active within the same location. If so, then the processor with higher rank (numerical value) will deactivate. If the buddy is not active, or if the buddy is of higher rank than the location the processor is marking, the processor continues. The processor shifts its mark to the smaller of its mark and that of its buddy. After log N iterations, only one processor remains active in each bucket. This process is illustrated below.

Bucket 1

```
0 1 2 3  4  5 6 7  8  9 10 11 12 13  14  15

| | | | | |P_5| | |P_8|P_9| | | | |   |P_15|
| | | | |B_5| | |B_9|B_8| | | | |B_15|  |
```
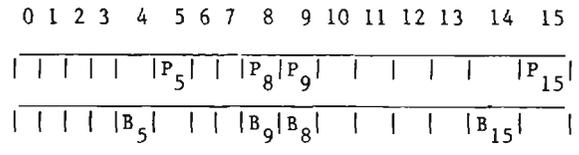
k = 0

Figure 3

(U) Figure 3 shows bucket 1 and the locations marked by the active processors (all four are active initially). The locations of each processor's 0th buddy is also marked. Processors 5 and 15 check the location of their buddies, and find no processor has that location marked. Because their buddies are smaller than the locations they are marking, they will move their marks to their buddies' locations. Processors 8 and 9 check their buddies' locations and simultaneously discover that they are both active. Processor 9, being of higher rank than processor 8, will deactivate. Because the 0th buddy of processor 8 is larger than the location marked by processor 8, processor 8 will not move its marker.

Bucket 1

```
0 1 2 3  4 5  6 7 8   9 10 11 12 13  14 15

| | | | |P_5| | |P_8|P_9*| | |  |P_15| |
| | | | | |B_5| | |B_8| |B_15| | |  |
```
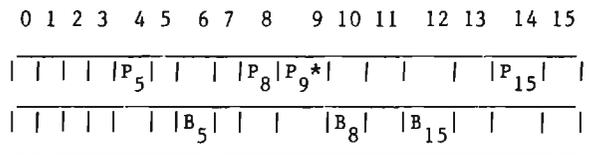
k = 1

Figure 4

(U) Figure 4 shows processor 9 deactivated (though it continues to mark its last active location) and the markers of processors 5 and 15 moved to their new, lower locations. Each remaining active processor finds no mark in the location of its first buddy, so each remains active. Processors 5 and 8 do not move their markers because their buddies are at locations higher than their markers. Processor 15 moves its marker because its first buddy is in a location lower than its marker.

Bucket 1

```
  0 1 2 3  4 5 6 7   8   9 10 11  12 13 14 15
| | | | |P₅| | | |P₈|P₉*| |  |P₁₅| | | |
|B₅| | | | | | | |B₁₅| | |  |B₈| | | |
```
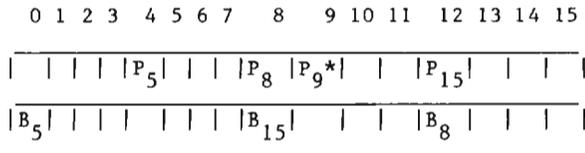
k = 2

Figure 5

(U) Figure 5 shows the locations of the second buddies. Processors 8 and 15 detect each other's presence simultaneously, and processor 15 deactivates. Processor 5 moves its marker to the location of its second buddy.

Bucket 1

```
  0 1 2 3 4 5 6 7  8   9 10 11   12 13 14 15
|P₅| | | | | | | |P₈|P₉*| |  |P₁₅*| | | |
|B₈| | | | | | | |B₅| | | |  | | | |
```
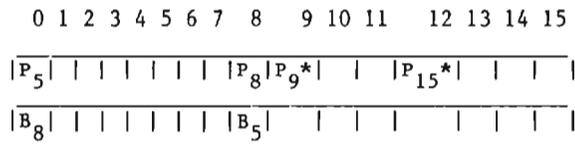
k = 3

Figure 6

(U) Finally, in Figure 6, the third buddies of processors 5 and 8 detect each other, processor 8 deactivates and processor 5 is left as the only active processor in bucket 1 with its marker in location 0. In general, the lowest ranked processor will remain active in each occupied bucket, with its marker at location 0.

(U) This algorithm sorts the list in $O(\log N)$ steps, but discards duplicates from the list. A modification of this algorithm will enable us to sort in $O(\log N)$ and keep the duplicates. Each processor keeps a running count of the number of processors greater than or equal to itself that are active in the bucket. Figure 7 illustrates this procedure within bucket 1, using a modified buddy system.

(U) The modified buddy system is much like the system described earlier. Whenever a processor's buddy detects an active processor of lower rank, the higher ranked processor deactivates (that is, it does not move its mark). However, its buddy assumes the value of the lower ranked processor's buddy, and the value of the buddy of any lower ranked processor it subsequently detects. Whenever a processor's buddy detects a processor of higher rank, it adds that processor's count to its count. Note that a processor can be detected by several processors simultaneously because we allow multiple fetches.

(U) At the end of log n steps, location 0 of each bucket contains the marker of the lowest ranked processor active in that bucket. The market will be equal to the number of processors in the bucket. Also in the bucket, at various locations, will be the markers of the other processors, each equal to the number of processors ranked greater than or equal to the processor.

(U) Algorithm 1.1 formally expresses these ideas. In the algorithm, we use the notation that $x<k>$ is the kth binary digit of x (note that we have digits 0 through $\log_2 N-1$), $\oplus$ represents bit by bit mod 2 addition, and e_k is the $\log_2 N$ digit number with a single 1 in the kth place.

Algorithm 1.1 – Parallel Bucket Sort
                    (N numbers, range $M \leq N$)
Input: $A[j,i] = 0$ $0 \leq i \leq N-1$
                            $0 \leq j \leq M-1$
        C_i $0 \leq C_i \leq M-1$
                    (the numbers to be sorted)

```
Concurrently for all i do
  mark_i <- i
  point_i <- i
  A[C_i, mark_i] <- 1
  flag_i <- 1
for k <- 0 step 1 until (log₂N-1) do
  buddy_i <- point_i ⊕ e_k
  if point_i<k> = 0 then
    A[C_i,mark_i] <- A[C_i,mark_i] + A[C_i,buddy_i]
  else if point_i<k> = 1 then
    point_i <- buddy_i
    if A[C_i,buddy_i] = 0 then
      if flag_i = 1 then
        A[C_i, buddy_i] <- A[C_i, mark_i]
        A[C_i, mark_i] <- 0
        mark_i <- buddy_i
      end (if flag_i = 1)
    else if A[C_i, buddy_i] ≠ 0 then
      flag_i <- 0
    end (if A[C_i, buddy_i] ≠ 0)
  end (else point_i<k> = 1)
end (for k loop)
end (Algorithm 1.1)
```

(U) We now have completed the first part of our algorithm and know how many elements are in each bucket. We next want a cumulative total so that bucket i contains the number of terms less than or equal to i. This is accomplished by transferring the data in $A[j,0]$ to $B[j]$ and totaling $B[j]$. Then with $D[i] \leftarrow B[C_i] - A[C_i, mark_i]$, $D[i]$ has the location

DOCID: 4011963


Figure 7

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
B	1	4	2	2	1	1	1	2	0	1	0	1	0	0	0	0
	B_1	B_0	B_3	B_2	B_5	B_4	B_7	B_6	B_9	B_8	B_{11}	B_{10}	B_{13}	B_{12}		

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
B	1	5	2	4	1	2	1	3	0	1	0	1	0	0	0	0
		$B_{2,3}$		$B_{0,1}$		$B_{6,7}$		$B_{4,5}$		$B_{10,11}$		$B_{8,9}$		$B_{12,13}$		

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
B	1	5	7	9	1	2	3	5	0	1	1	2	0	0	0	0
				B_{5-7}			B_{0-4}				B_{12-16}					

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
B	1	5	7	9	10	11	12	14	0	1	1	2	2	2	2	2
								B_{8-15}								B_{0-7}

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
B	1	5	7	9	10	11	12	14	14	15	15	16	16	16	16	16
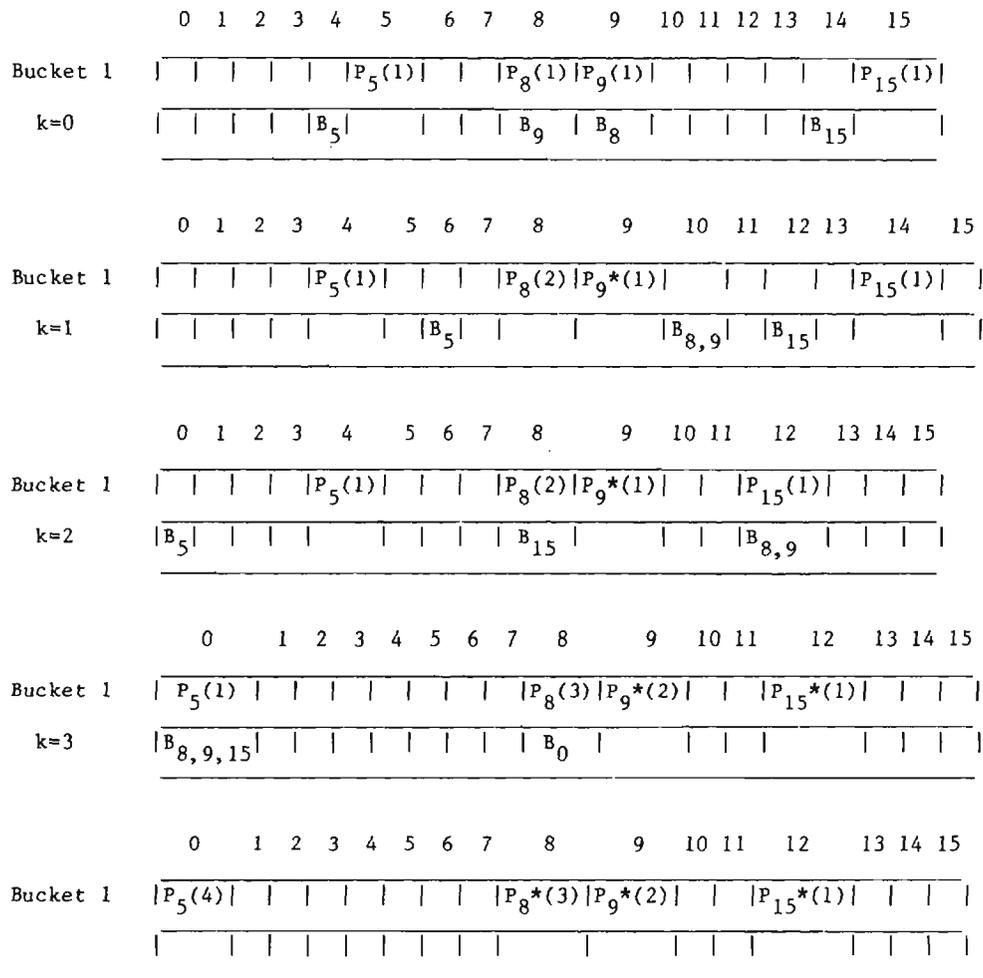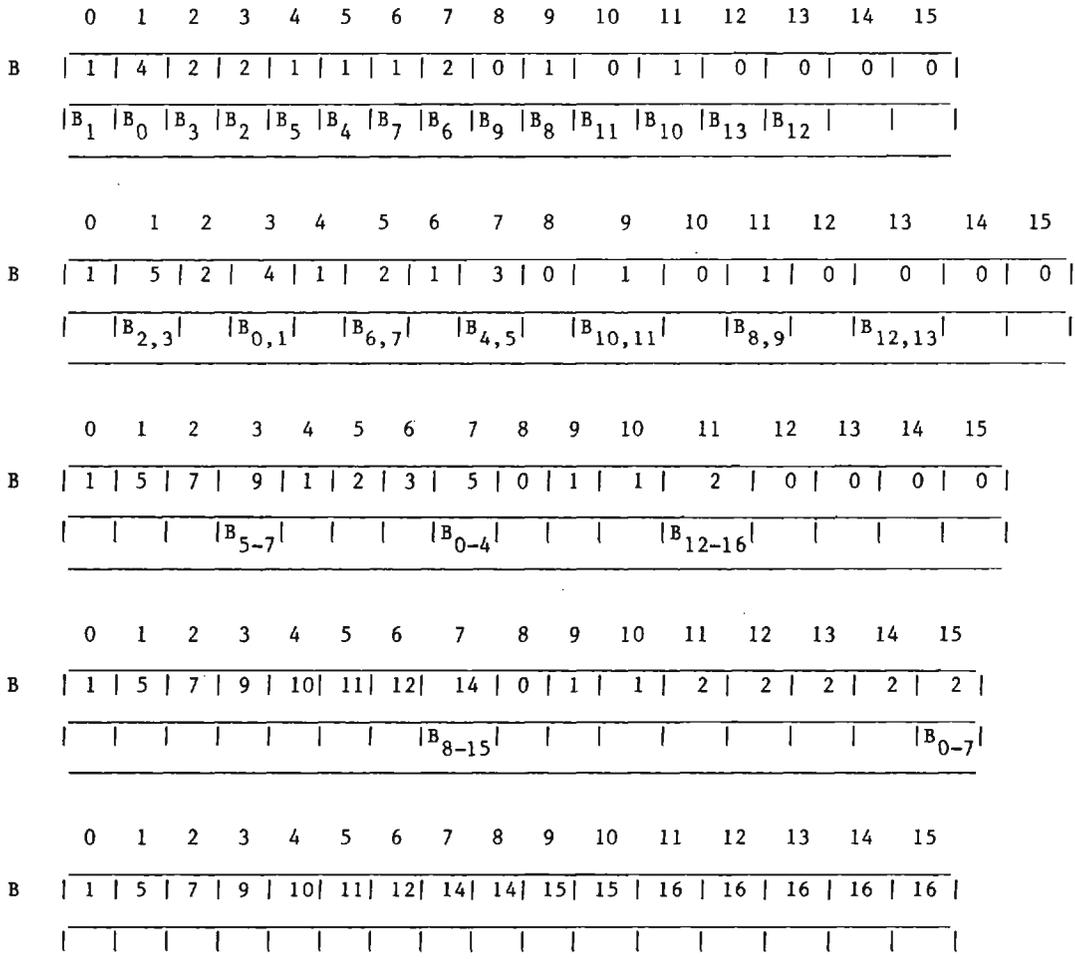
Figure 8

(in an N-long sorted list) which contains the first value i.

(U) Figure 8 shows the various steps of producing the cumulative total. Note again that we allow multiple fetches from the same memory location. The cells in Figure 8 represent $B[0]$ through $B[15]$, and are a continuation of the example begun in Figure 7.

(U) To conclude this algorithm, note what happens in bucket 1. Processors 5, 8, 9, and 15 have totals of 4, 3, 2, and 1 respectively. With only one element in bucket 0, C_5, which equals 1 and is assigned to processor 5, has final position $B[C_5] - A[C_5, \text{mark}_5] = B[1] - A[1,0] = 5 - 4 = 1$. Similarly, C_8, C_9, and C_5 have final positions 2, 3, and 4.

(U) Algorithm 1.2 concludes Hirshberg's parallel bucket sort.

Algorithm 1.2 – Parallel Bucket Sort
Input : from Algorithm 1.1
  B[i] = 0 $0 \leq i \leq$ N-1
  D[i] = 0 $0 \leq i \leq$ N-1

Concurrently for all i do
  $B[i] \leftarrow A[i,0], \quad 0 \leq i \leq$ M-1
  $\text{point}_i \leftarrow i$
  for k $\leftarrow$ 1 step 1 until log N do
    $\text{buddy}_i \leftarrow \text{point}_i \oplus e_k$
    If point_i <k> = 0 then
      $\text{point}_i \leftarrow \text{buddy}_i$
    else point_i <k> = 1 then
      $B[i] \leftarrow B[i] + B[\text{buddy}_i]$
    end (if point_i <k> = 1)
  end (for k loop)
  $D[C_i] \leftarrow B[C_i] - A[C_i, \text{mark}_i]$
end (Algorithm 1.2)

(U) The algorithm produces D = (0, 1, 5, 7, 9, 10, 11, 12, 0, 14, 0, 15, 0, 0, 0, 0) from the initial data C = (4, 0, 2, 7, 9, 1, 5, 3, 1, 1, 6, 11, 3, 7, 2, 1), M = 12, N = 16.

VALIANT'S FAST MERGING ALGORITHM

(U) Leslie Valiant at the University of Leeds has developed a fast merging algorithm for SIMD machines [2] which in turn leads to a fast sorting algorithm. The merging algorithm is somewhat dissatisfying because of the vagueness of the data rearrangement techniques. To merge two lists of length M and $N \leq M$, Valiant requires $\sqrt{NM}$ processors and $O(\log \log N)$ time and to sort he requires $O(\log N \cdot \log \log n)$ time. His SIMD model

requires either that each processor has access to a common memory or that the processors have

a sufficiently robust interconnection to permit necessary data routing.

(U) To see that Valiant's merging algorithm requires time $O(\log \log N)$, we shall proceed recursively. We will reduce the problem of merging two lists of length N and M to that of merging $\sqrt{N}$ lists of length $\sqrt{N}$ and, at most, $2\sqrt{M}$.

(U) Let $X = (x_1, x_2, \ldots, x_N)$ and $Y = (y_1, y_2, \ldots, y_M)$ be the two sorted lists we want to merge with $1 \leq N \leq M$. First, mark the elements of X that are subscripted by $i\sqrt{N}$ and those of Y that are subscripted by $i\sqrt{M}$, for $i = 1, 2, \ldots$.

(U) Then compare each marked element of X with each marked element of Y. This will require at most $\sqrt{MN}$ comparisons, and can be done in unit time with $\sqrt{MN}$ processors. We now can pinpoint the segment between marked elements of Y into which each marked element of X must be inserted. Now, compare each marked element of X with every element in the segment Y into which it must be inserted. As each segment of Y has $\sqrt{M}$ elements, at most $\sqrt{MN}$ comparisons are needed, and these can be done in unit time.

(U) After these comparisons, we know exactly where each marked element of X should be inserted into Y. X is subdivided into $\sqrt{N}$ segments of length $\sqrt{N}$, X_i, between the marked elements of X; and Y is subdivided into $\sqrt{N}$ segments, Y_i, between the $\sqrt{N}$ inserted elements of X. Further, we know which segments of X and Y must be merged.

(U) Before we can recursively use our algorithm to merge the disjoint segments $|X_i|$ and $|Y_i|$ we must determine if we have enough processors. We will need $\sqrt{|X_i|+|Y_i|}$ processors to merge each pair (X_i, Y_i). Now, we clearly have $\Sigma|X_i| \leq N$ and $\Sigma|Y_i| \leq M$, so by Cauchy's inequality,

$$\Sigma \sqrt{|X_i| \times |Y_i|} \leq \Sigma \sqrt{|X_i| \cdot |Y_i|} \leq \sqrt{MN}.$$

Thus we have enough processors to handle merging each pair of disjoint segments.

(U) In two time units we have reduced the problem from merging two lists of size M and N to that of merging several pairs of size $\sqrt{N}$ and at most $2\sqrt{M}$. Our cutdown varies on the larger of the two lists but is quite regular on the smaller: from $2^{\log N}$ to $2^{(\log N)/2}$ to $2^{(\log N)/4}$ and so on. The merging is finished when we have reduced the problem to merging several lists of length 1 with other larger

lists, and this requires about $2 \log \log N = O(\log \log N)$ time units.

(U) A few comments on Valiant's merging algorithm are in order here before proceeding. Our estimations of timing have allowed for one comparison to be made per unit time. We have not directly addressed the time needed to determine the insertion location nor the time needed for assigning data to processors. While both of these operations may be able to be done in a constant amount of time, the time required could still be overwhelming. Further, the requirement of $\sqrt{MN}$ processors is itself somewhat staggering. Finally, each processor must have some sort of broadcast capabilities to allow a word to be simultaneously compared with several other words (which might be handled by memory access capability). However, Valiant's techniques are nevertheless worth considering, if only for their elegance.

(U) This merging technique can easily be adapted to a sorting algorithm. Consider the problem of sorting $N = 2^n$ words with 2^{n-1} processors. At time 1 we have 2^{n-1} sorted lists of length 2^1. We can merge 2^{n-i-1} pairs of these lists by assigning 2^i processors to each pair. At stage i, the time for merging is about $2 \log \log 2^i = 2 \log i$. Thus the total time for sorting the N words is about

$$\sum_{i=1}^{\log N} 2 \log i \;\le\; \sum_{i=1}^{\log N} 2 \log \log N = 2 \log N \cdot \log \log N$$
$$= O(\log N \cdot \log \log N).$$

PREPARATA'S ALGORITHM

(U) Franco Preparata at the University of Illinois used Valiant's fast merging to design a parallel sorting algorithm for SIMD machines.[14] Recall that Valiant required 2^{n-1} processors to sort $N = s^n$ words in $O(\log N \cdot \log \log N)$ time. Preparata uses N log N processors to sort in $O(\log n)$ time. Note that with $\sqrt{N} \cdot \log n$ times more processors, he was able to reduce the time by a factor of only log log n.

(U) Preparata's algorithm, as well as Hirshberg's, are examples of what Knuth [7] terms "enumeration sorting." In these methods, each word is compared with all others, and the number of smaller keys determines the word's final position. In this instance, we will merge two sorted lists to count the number of terms less than a given word. As an example, consider two sorted lists, X and Y,

and consider the word in X in position i, x. After merging X and Y with a stable merge algorithm to produce Z, X is now in position q; that is, $z_q = x$. This means that there must be q - i words in Y that are less than or equal to x.

(U) Preparata's algorithm proceeds as follows: Let $N = 2^n$ and let $A = \{a_i | i=0,\ldots,N-1\}$ be our list to be sorted. We assume that for fewer than N words, at most $N \log_2 N = n2^n$ processors are needed to implement the sort, and we will use induction to prove it for N. Divide A into n subarrays, each of size N/n, $A_0, A_1, \ldots, A_{n-1}$, and recursively call on the sort algorithm to sort the subarray. By the inductive hypothesis, each subarray requires at most (N/n) log (N/n) processors to be sorted. With n subarrays, the total number of processors needed to sort the subarrays is

$$
\begin{aligned}
n(N/n) \log (N/n) &= N \log (N/n) \\
&= N \log N - N \log \log N \\
&\le N \log N .
\end{aligned}
$$

If it requires T(k) time to sort K words, then this step requires T(N/n) time. Associated with the lth word of the sorted subarray A_i is the label (i,l).

(U) Next, we copy our sorted subarrays A_i into longer arrays: $S_{i,j} = A_i, A_j$ for i<j. Note that there are (n/2)(n-1) arrays $S_{i,j}$ and each requires 2N/n words (including their labels). If we assign one processor to each word to be written into $S_{i,j}$, we will need $(n/2)(n-1)(2N/n) < N \log_2 N$ processors. This copying can be done in one time unit and will require simultaneous fetches of some words (for example, A_i is copied into N - 1 arrays).

(U) We now use Valiant's technique to merge each $S_{i,j}$. This will require $\sqrt{N/n \cdot N/n} = N/n$ processors for the $(n/2)(n-1)S_{i,j}$, or a total of $(n/2)(n-1)(N/n) < N \log_2 N$ processors. The time required is $O(\log \log (N/n))$

(U) An array $R[i;j;k]$ $0 \le i,j < n$, $0 \le k$ N/n is set up to hold the count information of each word, prior to the final rank computation. $R[i;j;k]$ will equal the number of words in A_j that are less than or equal to the kth word of A_i.

(U) The $R[i;j;k]$ are determined as follows. Let (x,ℓ) be the label associated with the word in the qth position of $S_{i,j}$, $S_{i,j}[q]$. (Recall that (x,ℓ) is associated with the ℓth word of A_x, so x = i or x = j.) If x = i, then $R[i;j;\ell] = q - \ell$ or if x = j, then $R[j;i;\ell] = q - \ell$, and $R[x;x;\ell] = \ell$. There are

$n \cdot n \cdot N/n = N \log_2 N$ words to be copied into R, and, as before, we use one processor per word to accomplish the copying in one time unit.

(U) Now to determine final rank of $A_i[\ell]$, we must sum up the number of words less than or equal to $A_i[\ell]$ in all other subarrays.

Thus,

$$rank\ (A_i[\ell]) = \sum_{j=0}^{n} R[i;j;\ell],$$

and this computation requires $n/2$ processors for each $A_i[\ell]$ (compare this with the buddy method in Hirshberg's Parallel Bucket Sort) and $\log \log n$ time. The total number of processors used here is $N \log N$. Finally, we complete our work by

$$A[rank(A_i[\ell])] = A_i[\ell].$$

(U) To conclude our analysis, note that none of the steps used more than $N \log N$ processors. The time required to sort N words, $T(N)$, is: $T(N/\log N)$ to sort the subarrays A_i; one time unit to copy A_i and A_j into $S_{1,j}$; $O(\log \log (N/n)) < O(\log \log n)$ time for merging the $S_{1,j}$; one time unit to copy partial data into R; $\log \log N$ time to compute rank $(A_i[1])$; and one time unit to make the final rearrangement. Thus

$$T(N) \cong T(N/\log N) + C_1 \log \log N + C_2,$$

and solving this recurrence, we have

$$T(N) = O(\log N).$$

Thus, Preparata using an SIMD model similar to Valiant's model is able to sort in time $O(\log N)$ as compared to $O(\log N \log \log N)$. However this faster method uses $N \log N$ processors, and it also requires a considerable amount ($2 N \log N$) of extra storage to keep up with the various arrays.

(U) Preparata in [14] also presents a generalized version of the preceding algorithm. This generalized algorithm avoids memory fetch conflicts, and uses $N^{1+\alpha}$ processors to sort in time $(1/\alpha)\ C \log N + O(\log N)$, for $0 < \alpha \le 1$ and C a constant. However, as elegant as this result may be, or that of Hirshberg or Valiant, they all presuppose a multiple-access memory, and this is not a likely or sound basis for design. Even with the elimination of fetch conflicts, this memory design is not

likely to be seen in the near future. These three algorithms fall in the class of the theoretically important but practically unworkable.

MESH-CONNECTED COMPUTERS

(U) A more practical avenue of research is to consider an SIMD model where each processor has its own memory and can communicate directly with other processors with which it is connected. This fixed interconnection network in turn becomes the dominant factor to be considered in the design of efficient algorithms.

THOMPSON AND KUNG'S ALGORITHM

(U) Thompson and Kung [21], and after them Nassimi and Sahni [12], have produced algorithms for a mesh-connected computer. The ILLIAC IV computer is an 8 x 8 example of this type of architecture. Figure 9 shows the two-dimensional array of processors, each denoted by P, and their interconnections. The processors are placed in a square array and each one is connected to all of its neighbors. Processors at the perimeter have two or three rather than four neighbors--there are no "wrap-around" connections, as found on the ILLIAC IV.
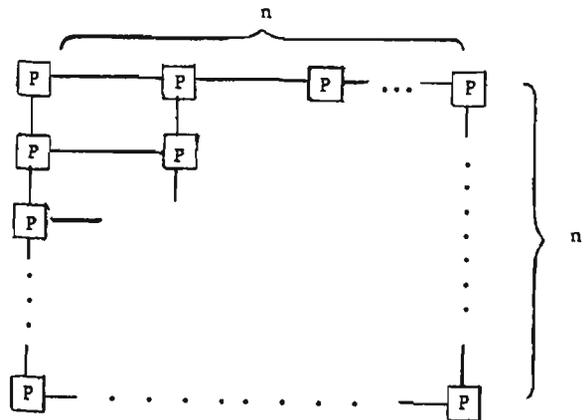
Figure 9.

(U) Before describing the algorithm, it will be necessary to decide upon an indexing scheme. Such a scheme is dependent upon how the sorted words will be used and upon the algorithm chosen. Figure 10 shows two indexing methods: row-major and snake-like row major. While in general in an SIMD machine every processor executes the same instruction or transmits data in the same direction, it would not be difficult to allow one special

```
 0  1  2  3          0  1  2  3

 4  5  6  7          7  6  5  4

 8  9 10 11          8  9 10 11

12 13 14 15         15 14 13 12
```

Row-major   Snake-like
  indexing   row-major
                        indexing
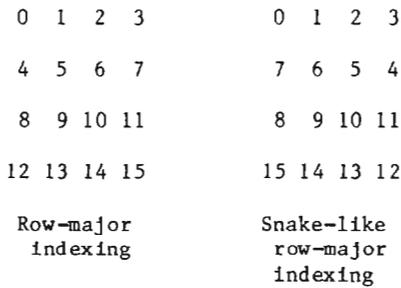
Figure 10

instruction that permits data to be trans-
mitted along some fixed path, such as the
snake-like row-major indexing. Thompson and
Kung use the snake-like row-major indexing for
their sort and Nassimi and Sahni use row-major
indexing for their scheme.

(U) In our mesh-connected SIMD model, we
shall define t_r to be the time required to
route data one unit distance in any direction,
and t_c to be the time to perform the compari-
son of two words within one processor. Any
number of simultaneous data moves can be made
in any one direction at one time, and any
number of simultaneous comparisons can be made
at one time. Thus a comparison-exchange step
between two items in horizontally adjacent
processors can be done in time $2t_r + t_c$ (route
left, compare, route right).

(U) Regardless of the algorithm selected
for our model, it is possible to get an abso-
lute lower bound on the time. For any index-
ing scheme, there are situations where it will
be necessary to exchange the words in opposite
corners of the processor array. For this
movement, shown in Figure 11, $2(n-1)$ routing
steps are needed to move a up and to the
right, and $2(n-1)$ routing steps are needed to
move b down and to the left. Thus, at least
$4(n-1)$ routing steps are needed. For this
model then, no algorithm can sort n^2 words in
time less than $O(n)$.
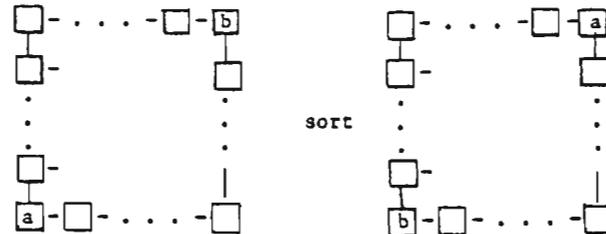
Figure 11.
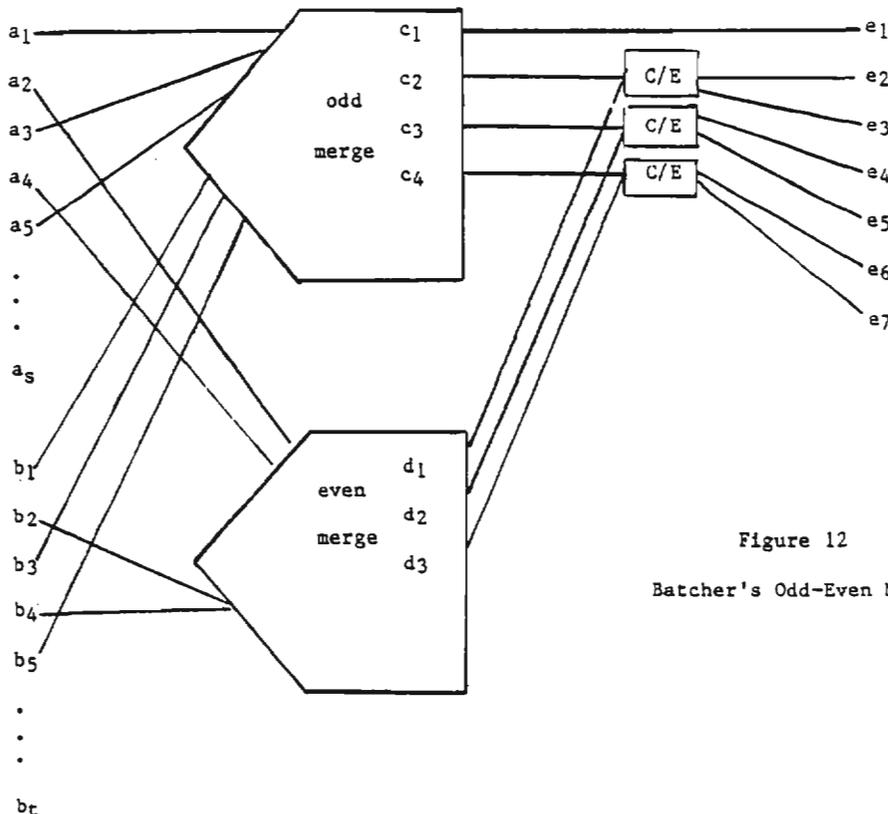
Figure 12

Batcher's Odd-Even Merge

ODD-EVEN TRANSPOSITION SORT

(U) Before proceeding with a description of the Thompson-Kung algorithm, we must discuss two other algorithms: the odd-even transposition sort and the Batcher odd-even merge. The odd-even transposition sort is described in Knuth [7] and is a straightforward, if slow, sort. To sort N words, the first step is to compare words $2i$, and $2i+1$, for $0 \le i \le N/2$, and exchange (or transpose) them if needed so that the larger is now in position $2i+1$. Then for the second step, compare $2i-1$ and $2i$ and exchange them so that the larger is in position $2i$. After alternating $N/2$ of step 1 with $N/2$ of step 2, the words will be sorted, with the smallest in position 0.

(U) With this algorithm, it does not matter if we start with step 1 or step 2, as long as they alternate. When N is odd the beginning step will be executed once more than the second step. The algorithm requires $N^2/2$ comparison exchanges. Because both steps 1 and 2 compare disjoint pairs of elements, $N/2$ processors in parallel could execute the algorithm in $O(N)$ time.

BATCHER'S ODD-EVEN MERGE

(U) The Batcher odd-even merge [1, 7] is a technique for merging two sorted lists (of possibly different lengths). The two lists are "unshuffled" so that all the odd terms are together, as are the even terms. The odd terms and the even terms are merged (by a recursive call to the odd-even merge), the two merged lists are shuffled together, and a final step of $N/2 - 1$ comparison-exchanges completes the merge. Figure 12 illustrates Batcher's merge.

(U) The Thompson-Kung algorithm [21] assumes that for our N x N mesh-connected computer, N is a power of 2. We first consider an odd-even transposition sort on a $j \times k$ subarray, were both j and k are powers of 2, and the processors are indexed by a snake-like row-major order. As discussed earlier, the time required for a comparison-exchange on a pair of words (or several similarly oriented pairs) is $2t_r + t_c$.

(U) Let $T_{oe}(j \cdot k)$ be the time required for an odd-even transposition sort on a $j \times k$ subarray. If $j = 1$ or $k = 1$, then our model reduces to a linearly connected SIMD machine and $T_{oe}(j,k) = jk(2t_r + t_c)$. If $k = 2$, the step 1 comparison-exchanges are all made horizontally and the step 2 comparison-exchanges are all made vertically, and $T_{oe}(j,k) = jk(2t_r + t_c)$. These cases are

illustrated in Figure 13, with an arrow indicating a comparison-exchange.

Figure 13

step 1   step 2   step 1   step 2

k = 1   k = 2
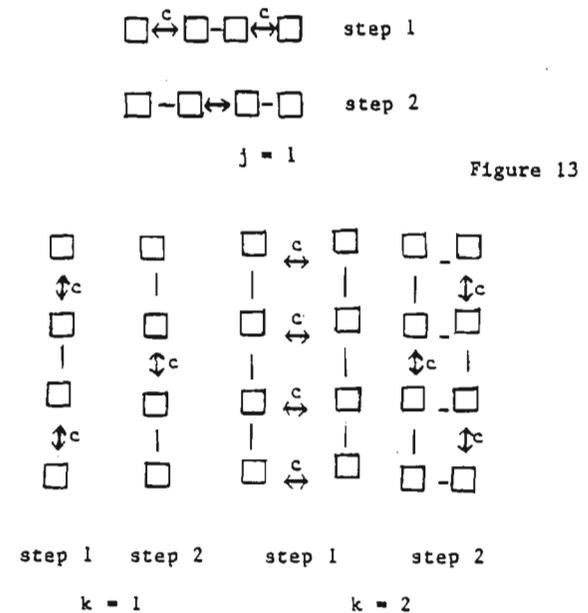
(U) In the case where $j > 1$ and $k > 2$ the times for steps 1 and 2 become different. Step 1 still requires $2t_r + t_c$ time because all of its comparison-exchanges are made horizontally; however, step 2 now requires $4t_r + t_c$ time, because some of its comparison-exchanges are made horizontally and some vertically. (Recall that data movements must all be in the same directions. Thus, for step 2 we perform: route left, route up, compare, route right, route down.) Thus for $j > 1$, $k > 2$

$$T_{oe}(j,k)=1/2jk[(2t_r+t_c)+(4t_r+t_c)]=jk(3t_r+t_c).$$

This is illustrated in Figure 14 for $j = k = 4$.

Step 1   j = k = 4   Step 2

Figure 14
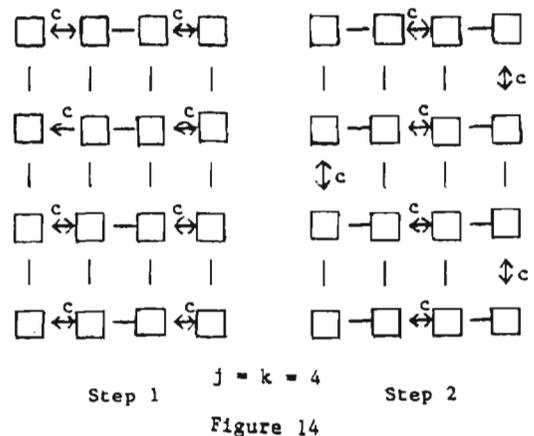
(U) By these timing arguments, our initial array of $N = n^2$ words could be sorted with N processors by the odd-even transposition sort in time $N(3t_r + t_c)$. We will now describe a variation of Batcher's odd-even algorithm for merging two $j \times k/2$ sorted arrays to produce a $j \times k$ sorted array. This will lead to a divide-sort-merge strategy that will sort in $O(\sqrt{N} \log \sqrt{N})$ rather than $O(N)$. First note that the two halves of a $1 \times k$ can be shuffled (or unshuffled) in time $(k-2)t_r$. This is illustrated in Figure 15. The case for merging two $j \times 1$ lists will be treated separately.

```
a - a - a - a<->b - b - b - b

a - a - a<->b - a<->b - b - b

a - a<->b - a<->b - a<->b - b

a - b - a - b - a - b - a - b
```
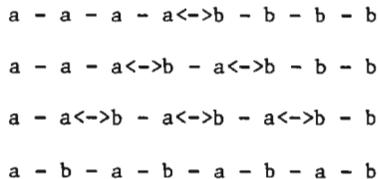
Figure 15

Algorithm 2.1: Thompson-Kung $j \times 1$ Parallel Merge $(M(j,2))$

Input: Two adjacent $j \times 1$ sorted arrays in a mesh-connected computer

M_1 - Move all even terms from the right column to the left column, and all odd terms from the left column to the right column. (This "unshuffles" the terms.)
Time: $2t_r$

M_2 - Sort each column with an odd-even transposition sort.
Time: $j(2t_r + t_c)$

M_3 - Interchange on odd rows. (This "shuffles" the terms back together.)
Time: $2t_r$

M_4 - Compare-exchange words $2i - 1$ and $2i$ of the $j \times 2$ array.
Time: $2t_r + t_c$.

Figure 16 illustrates M(4,2). The merge M(j,2) takes time $T(j,2) = (2j + 6)t_r + (j + 1)t_c$.
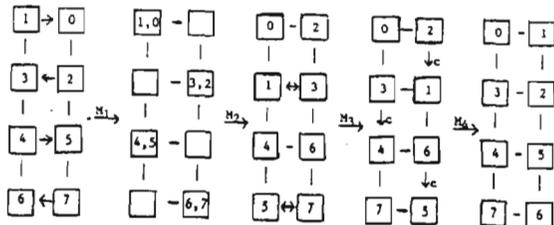
Figure 16.

Algorithm 2.2: Thompson-Kung $j \times k$ Parallel Merge $(M(j,k))$

Input: Two adjacent $j \times k/2$ arrays in a mesh-connected computer, each sorted in snake-like row-major order.

M_1 - Interchange words $2i$ and $2i+1$ on odd rows so that each column contains either all evens or all odds.
Time: $2t_r$

M_2 - Unshuffle each row so that the first quarter of the columns are now the odd columns from the first half of the columns.
Time: $(k-2)t_r$

M_3 - Merge each half with $M(j,k/2)$.
Time: $T(j,k/2)$

M_4 - Shuffle each row.
Time: $(k-2)t_r$

M_5 - Interchange words $2i$ and $2i+1$ on odd rows.
Time: $2t_r$

M_6 - Compare-exchange words $2i+1$ and $2i$.
Time: $4t_r + t_c$.

Figure 17 illustrates M(4,4).

```
2 - 4 - 0 - 1
|   |   |   |
8<-> 5 - 6<-> 3     M_1
|   |   |   |
10 - 13 - 7 - 9
|   |   |   |
15<->14 - 12<->11


2 - 4<-> 0 - 1
|   |   |   |
5 - 8<-> 3 - 6     M_2
|   |   |   |
10 - 13<-> 7 - 9
|   |   |   |
14 - 15<->11 - 12


2 - 0 - 4 - 1
|   |   |   |
5 - 3 - 8 - 6     M_3
|   |   |   |
10 - 7 - 13 - 9
|   |   |   |
14 - 11 - 15 - 12
```

```
   0 - 2<-> 1 - 4
   |    |    |    |
   5 - 3<-> 8 - 6    M4
   |    |    |    |
   7 - 10<-> 9 - 12
   |    |    |    |
   14 - 11<->15 - 13


   0 - 1 - 2 - 4
   |    |    |    |
   5<-> 8 - 3<-> 6    M5
   )    |    |    |
   7<-> 9 - 10<->12
   |    |    |    |
   14<->15 - 11<->13

              c
   0 - 1 - 2 - 4
   |    | c  |   |c
   8 - 5 - 6 - 3    M6
   |c   |c  |    |
   7 - 9 - 10 - 12
   |    | c |   |c
   15 - 14 - 13 - 11
```
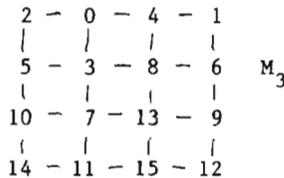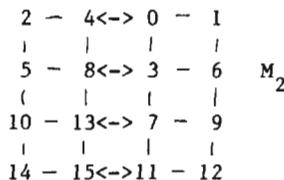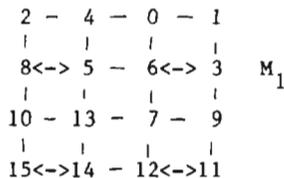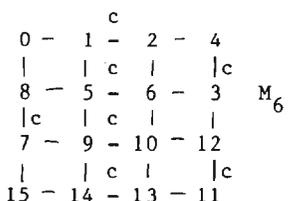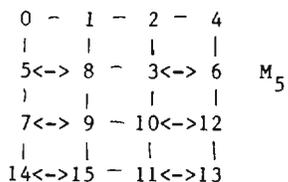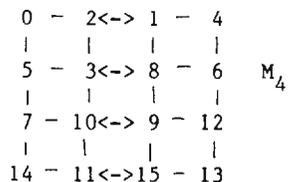
Figure 10

(U) The time required to perform M(j,k) is given by

$$T(j,k) = (2k + 4)t_r + t_c + T(j,k/2),$$

and we can find that

$$T(j,k) \le (2j + 4k + 4\log k)t_r + (j + \log k)t_c.$$

In particular, $T(n, n/2^i) = 0(n)$. Now to obtain our sorting algorithm, we first sort each column in time $T_{oe}(n.1) = n(2t_r + t_c)$ and then successively call on M(n,2), M(n,4), ..., M(n,n). The total time required for the sort, S(n,n), is then

$$S(n,n) = T_{oe}(n,1) + \sum_{i=1}^{\log n} T(n,2^i) = 0(n \log n).$$

Thus we can sort $N = n^2$ in time $0(\sqrt{N} \log \sqrt{N})$.

(U) One further refinement of the algorithm will give us an improvement in the time to sort N words from $0(\sqrt{N} \log \sqrt{N})$ to $0(\sqrt{N})$. Rather than merge sorted columns of length n horizontally, we shall merge columns of length $n/2^i$ both horizontally and vertically. The algorithm M(j,k) merges two horizontally adjacent subarrays of size j x k/2. The same steps allow us to merge vertically adjacent subarrays of size k x j/2 in exactly the same time.

(U) Thus if M'(j,k) is an algorithm that merges four square adjacent subarrays of size j/2 x k/2, then

$$T'(j,k) = T(j/2,k) + T(j,k).$$

The case of particular interest to us yields

$$T'(j,j) \le (11j + 8 \log j)t_r + (3j/2 + 2 \log j)t_c.$$

This means that we can sort an n x n array by M'(2,2), M'(4,4),...,M'(n,n); that is, we first merge 4 square adjacent 1 x 1 arrays, then 4 square adjacent 2 x 2 arrays, and so on. The total time for our sort, S'(n,n), is then given by

$$S'(n,n) = \sum_{i=1}^{\log n} T'(2^i, 2^i)$$

$$\le \sum_{i=1}^{\log n} (11j + 8 \log j)t_r + (3j/2 + 2 \log j)t_c$$

$$\le (22n + 8 \log^2 n)t_r + (3n + 2 \log^2 n)t_c$$

$$= 0(n).$$

That is, we can sort $N = n^2$ in time $0(\sqrt{N})$.

(U) Thompson and Kung use a slightly different square adjacent merge that has a time of about one-half that of ours. They also have an s x s square adjacent merge whose linear term is 6N, which is quite close to the optimal time of 4N. However, all of these algorithms are 0(N), regardless of the value of the linear term.

(U) In the same paper [21], Thompson and Kung present an algorithm for performing a bitonic sort (discussed later) on a mesh-connected computer. They again require the data to be in snake-like row-major order and the sort time is 0(N). In particular, the linear term is 14N for the bitonic sort versus 6N for the s x s merge sort. However, for small values of N, $N \le 2^{18}$, the bitonic sort is faster, under the assumption that $t_c \le 2t_r$. Nassimi and Sahni [11] present a bitonic sort algorithm for a mesh-connected computer that runs in about the same time as Thompson and Kung's algorithm, but it sorts in row-major order, rather than snake-like row-major. Finally, Thompson and Kung state that for a j-dimensionally mesh-connected computer, a bitonic sort on N words can be done in time $0(N^{1/j})$.

RING-CONNECTED COMPUTER

(U) An important factor to be considered in choosing an algorithm for an SIMD machine is a good match of the machine's interconnection network with the data routing of the

algorithm. For example, the odd-even trans-
position sort requires comparison-exchanges
between adjacent words, so a mesh-like connec-
tion seems naturally suited for this algo-
rithm. After discussing some sorting networks
later, we will discuss further SIMD networks
that are well suited for sorting. Figure 18
shows a ring connection and how an odd-even
transposition sort "naturally" fits the net-
work.

(U) A good measure of the effectiveness of
a parallel sorting algorithm is the ratio of
time to the number of processors. It is well
known that for a serial processor, this ratio
is asymptotically N log N. With k processors,
the best speed that we can hope for is
(N log N)/k. This is so, because otherwise we
could make a serial machine run k times faster
and then beat N log N time.

SORTING NETWORKS

(U) The several algorithms previously dis-
cussed have been designed for various models
of SIMD machines. These involved several
memories and processors with the capability of
choosing among interconnection paths, and a
master control unit. If our primary use for
this configuration is sorting, then we have
much more hardware than necessary. We will
now develop a series of processors whose only
purpose is to sort and which achieve a more
efficient use of their component parts.

(U) Figure 19 shows the basic processor
unit that we will use: a 2-sorter (or a
comparison-exchange module or a comparator).
The 2-sorter accepts two words, one on each of
its input lines A and B, compares them and, if
necessary, exchanges them so that the larger
exits on the line marked "high" and the
smaller on the line marked "low." The com-
parator would probably be built to accept
words in bit serial order, most significant
bit first. This would require the least
amount of hardware, though a parallel design
is feasible.

(U) To sort more than two words, we will
seek some connection of comparators that will
produce the desired result. Figure 20 shows a
3-sorter and a 4-sorter. In general, it is
difficult to determine if such a network
sorts. Knuth [7] has shown that if a network
sorts all 0 - 1 sequences, then it will sort
any sequence. Clearly testing 2^N sequences is
easier than N!, but it is practical for small
N. The solution is to find a recursive con-
struction which guarantees a network that
sorts, at perhaps the expense of less than an
optimal design.

(U) If we have a technique for sorting N
words, it is quite easy to extend it to an
(N + 1)-sorter. Figure 21 shows the technique
of insertion. After sorting the N words, the
$n + 1^{st}$ word is compared with the smallest of
the sorted list and exchanged if necessary.
The larger of these two is compared with the
next word, and so on. The $n + 1^{st}$ word "bub-
bles" up to its final position.
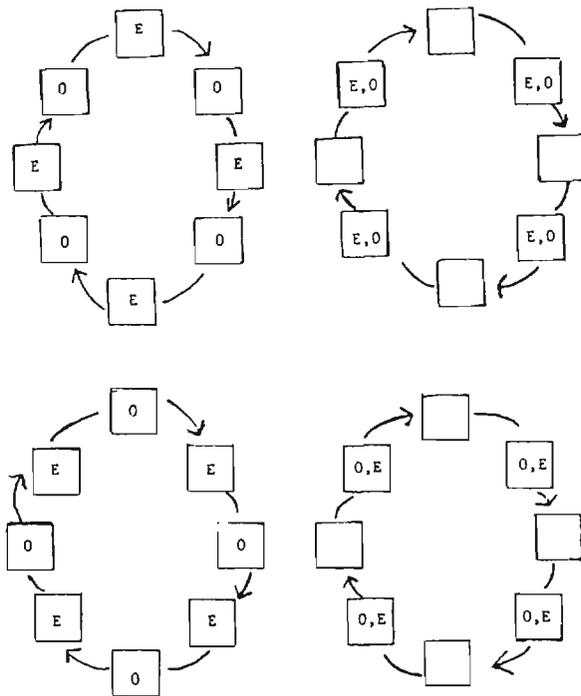
Figure 18.

(U) The repeated application of this inser-
tion principle produces the network shown in
Figure 22. To sort N words, this algorithm
requires (N - 1)N/2 comparators and a delay of
2N - 3. The product of time and processors
here is $O(N^3)$, and this seems decidedly nonop-
timal. One apparent inefficiency of the net-
work in Figure 22 is that line E is only
involved with one comparison. We shall soon
present elegant constructions that exceed this
naïve approach.

BATCHER'S BITONIC NETWORK

(U) Before continuing, it is necessary to
define what Kenneth Batcher [1] calls a
bitonic sequence, and then state his basic
theorem about bitonic sorting.

Figure 19

A —
B —
CLOCK   RESET

$\overline{A>B}$
$\overline{B>A}$
HI
LO

3-Sorter (a)

4-Sorter (b)

4-Sorter (c)

4-Sorter (d)
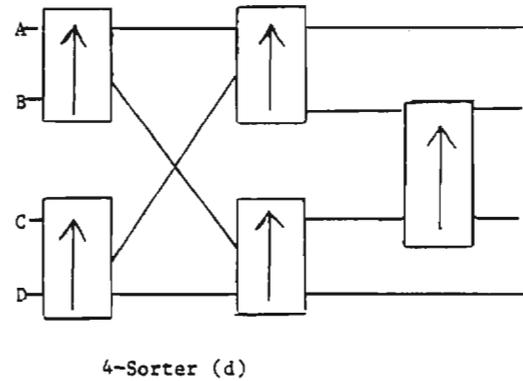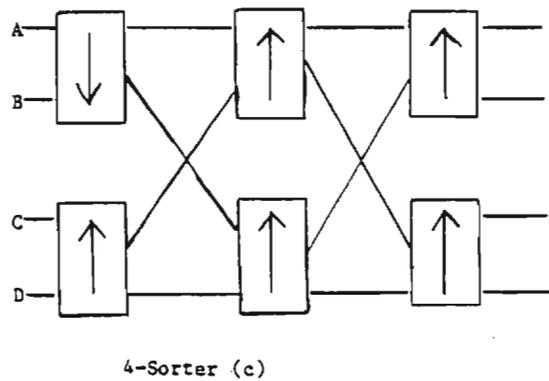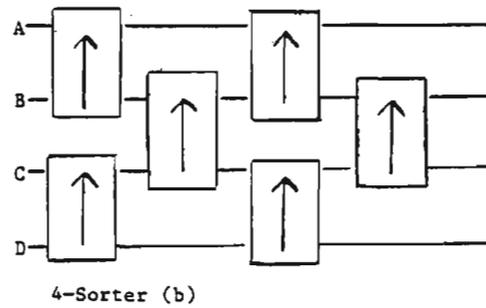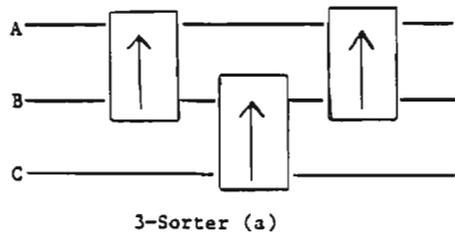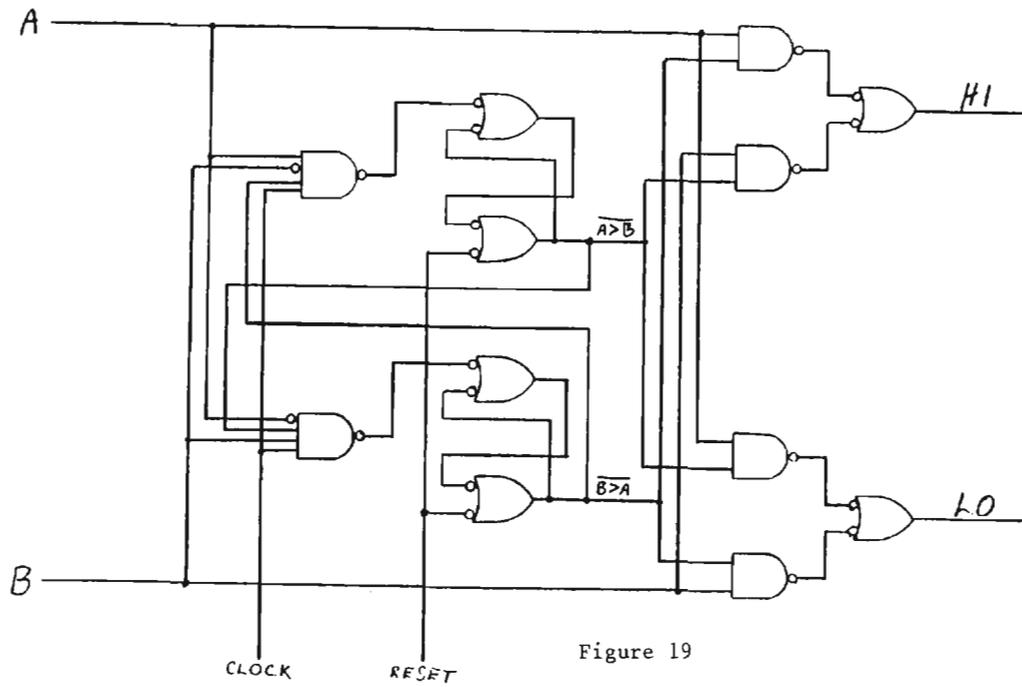
Figure 20.

Definition: A sequence of numbers $\{a_i|i=1,N\}$ is said to be bitonic if either

1) there is a j, such that $a_i \leq a_j$, if $i \leq j$ and $a_i \geq a_j$, if $i \geq j$, or

2) the sequence is an end-around shift of a sequence satisfying 1).

Theorem: let $a_i|i = 1,\ldots,N$ be bitonic, and define

$$b_i = \max(a_i, a_{i+N/2}), \; i = 1,\ldots,N/2$$

and

$$c_i = \min(a_i, a_{i+N/2}), \; i = 1,\ldots,N/2.$$

Then $\{b_i i = 1,\ldots,N/2\}$ and $\{c_i i = 1,\ldots,N/2\}$ are both bitonic, and $b_i \leq c_j$ for all i and j.

(U) The importance of Batcher's Theorem is that it enables us to take a bitonic list of length N, and with N/2 simultaneous comparison-exchanges divide the list into the smallest and largest elements, with each of these two sublists itself bitonic. It is easy to see that log N of these subdivisions will sort the initial bitonic list of length N.

(U) The initial list is obtained in a recursive manner. First, two words are sorted with one comparator and combined with another sorted 2-list to form a bitonic 4-list. The 4-list is then sorted by using Batcher's Theorem, and then paired with another sorted 4-list to make a bitonic 8-list, and so on. This procedure is illustrated in Figure 23.

(U) A few calculations will show that the delay for sorting N words is $1/2 \log_2 N(\log N + 1)$ and the number of comparators needed is $1/4 N \log N(\log N + 1)$. As clever as Batcher's recursive construction is, it does not produce the network with the least delay or the smallest number of comparators. For example, an 8-bitonic sorter requires 24 comparators and 6 delays (see Figure 23(d)), while the network in Figure 24 needs only 19 comparators, though still 6 delays (which is minimal in time and comparators). Networks in general and the Batcher network in particular (as illustrated in Figure 23) have certain drawbacks: a small number of comparison-exchanges are made at each stage, but the interconnection varies from stage to stage. Further, once built, the network can only be used to sort a fixed size list. We will present techniques that overcome both of these disadvantages.

STONE-BATCHER NETWORK

(U) Harold Stone [19] first observed that the perfect shuffle interconnection could be use as a fixed interconnection. He noted that Batcher's bitonic sorter for $N = 2^n$ first compared words that differed in the l's digit; then the 2's digit and the l's digit; then the 4's digit, the 2's digit, and the l's digit; and so on. The perfect shuffle permutation on $N = 2^n$ elements has $i \rightarrow 2i \pmod{2^n - 1}$. It can be visualized as the rearrangement caused by "cutting" the elements into halves, like a deck of cards, and "shuffling" them together. The permutation has the property of bringing together words that differ first in their most significant digit, then the next most significant, and so on.

(U) A sorting network that utilizes this fixed interconnection between stages is shown in Figure 25(a). Note that some of the comparators are inactive and simply allow data to pass through. By utilizing a three-state comparator (sort up, sort down, and pass through), it is possible to reduce a bitonic sorter to a single stage of comparators with a feed-around interconnection, as shown in Figure 25(b).

(U) The Stone-Batcher sorter requires only N/2 comparators as compared to about $1/2 N \log^2 N$ for a bitonic sorter, though its time is about twice as slow: $\log^2 N - \log N + 1$ versus $1/2 (\log^2 N + \log N)$. While Stone's configuration does dramatically decrease the hardware needed at only a slight increase in time, it will not allow data to be pipelined through it. If it is necessary to sort several lists of length N, the Stone-Batcher network requires that each list be sorted before the next one can be processed. A full network, on the other hand, can begin processing a new list as soon as the first stage of comparison-exchanges of the first list is completed.

(U) Another problem with sorting networks is that, once built, they cannot handle lists of a larger size, and they can only handle lists of a smaller size by "padding" them with $+\infty$ or $-\infty$. This can be overcome by expanding on the Stone-Batcher configuration shown in Figure 25(b) by replacing the storage register with first-in-first-out memories. These could be realized with disks, tapes, or similar media.

FASB SORTER

(U) Morris [10] first considered this design and called it the FASB Sorter, which is shown in Figure 26. Each box labeled M_i is a

OCID: 4011963

(a)

Figure 25.

Storage

(b.)

Figure 21.

Sorted 2-list
(a)

bitonic 4-list
(b)

Sorted 4-list
(c)

5-Sorter

Figure 22

Batcher's Bitonic 8-Sorter (d)
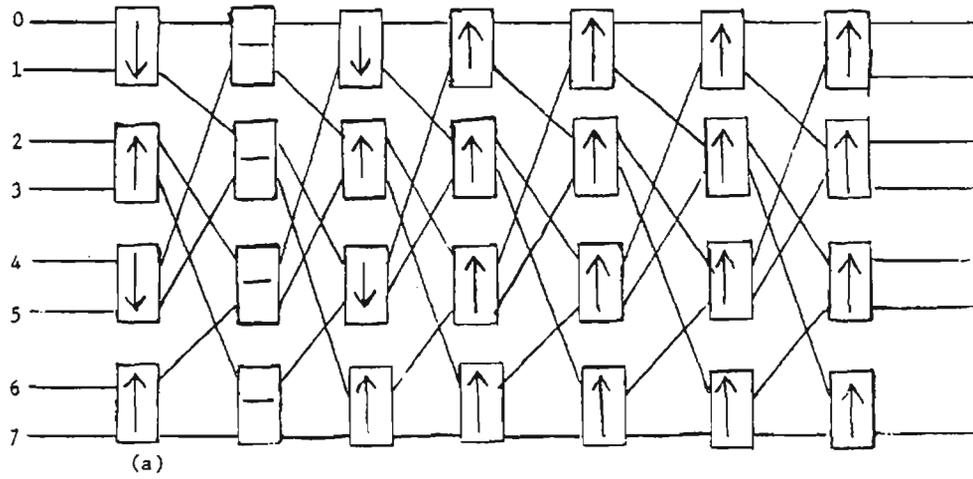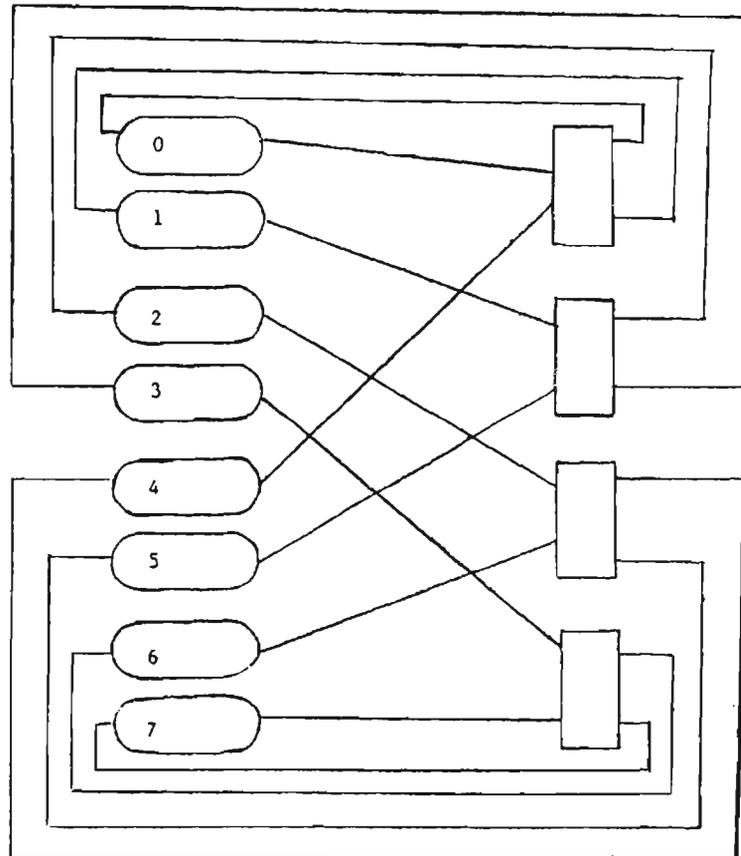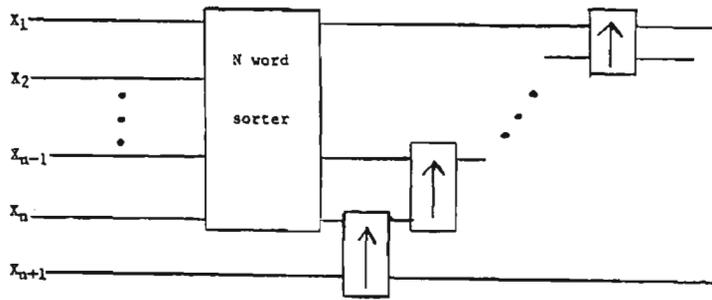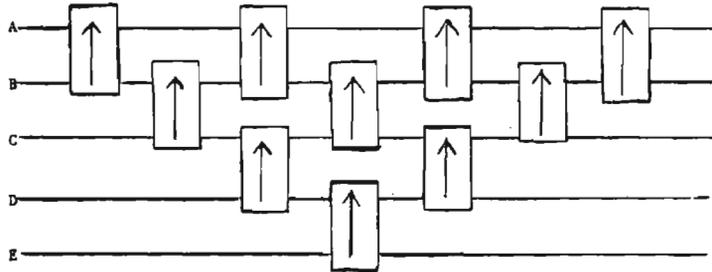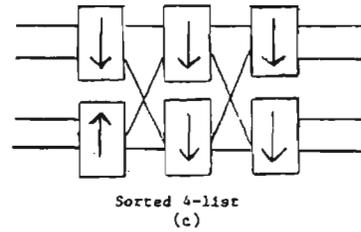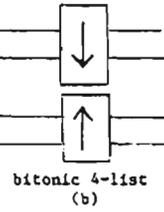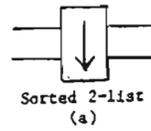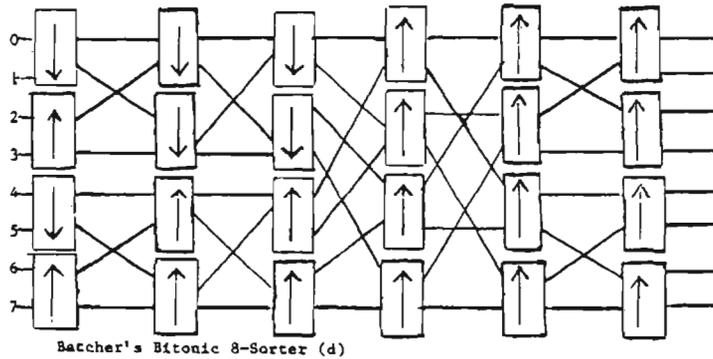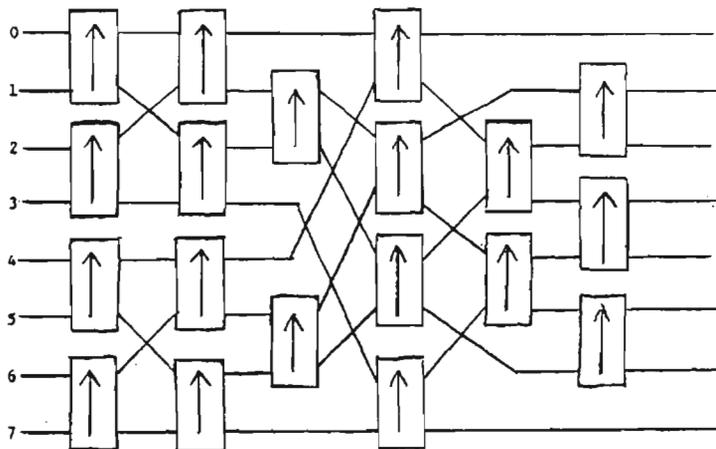
Figure 23.

8-Sorter

Figure 24

serial memory and each box labeled C/E is a three-state comparator. Note that we have a special switch for each even-odd pair of memories that allows two words to be written simultaneously into one memory or another. Because one word is written out while two are written in, the memories must effectively work at half-speed (unless radically redesigned).

Morris-Wisniewski Sorter
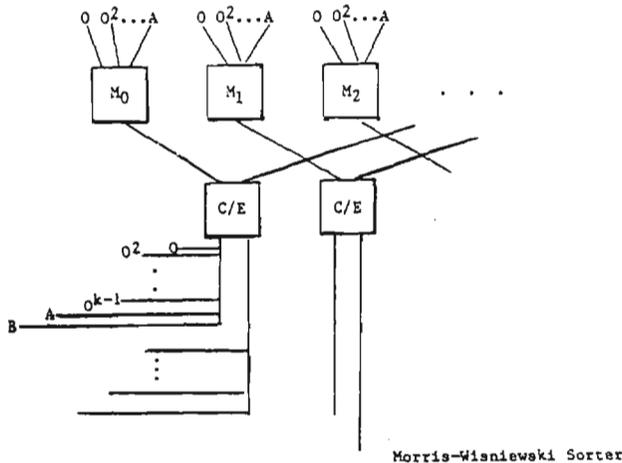
Figure 27.

MORRIS-WISNIEWSKI SORTER

(U) An improvement was made in this design by Morris and Wisniewski [11]. Rather than have three-state comparators that were idle nearly half of the time, they substituted a multiple interconnection scheme--in effect trading more wires for less time. The interconnections needed are all nontrivial powers of the perfect shuffle; 0, 0^2, 0^3, ..., and two permutations called A and B. Permutation A takes the outputs of the comparators and routes them in pairs to the first half of the memories and B routes the outputs in pairs to the second half of the memories. For 2^k elements, $0^k = id$, thus for 2^k memories, $k - 1 + 2 = k + 1$ interconnections are needed, rather than 1. Figure 27 shows a Morris-Wisniewski sorter.

(U) The Morris-Wisniewski sorter requires a more complicated control mechanism than the FASB or the Stone-Batcher because of the $k + 1$ data paths to choose from. However, for a configuration with 2^k memory modules, it can sort $N = 2^n$ words in time $2^{n-k-1} \log N (\log N + 1)$, about one half the *time of* a Stone-batcher sorter if $k = m$. The Morris-Wisniewski sorter can be expanded to handle any lists up to memory capacity.

## SINGLE LADDER ODD-EVEN TRANSPOSITION SORTER

(U) The last two sorting machines are highly specialized processors and would be useful only when considering the sorting of massive files. If, for example, it was necessary to serially read the data out before it could be used, then the I/O would dominate our considerations. For the latter situation, IBM [3] has proposed a special-purpose sorting processor that would be attached to a serial computer. Most of the sorting time would be hidden by the I/O time.

(U) The device is called a single ladder odd-even transposition sort (SLOETS). It consists of a series of comparators that can be set to allow straight-through or exchange data flow. The modules are linked by loops into a "ladder." Figure 28 shows a four-stage ladder in operation. The words are routed in serially so that each loop is filled with one word. Then as the words circulate through the loops, alternately odd-even and even-odd pairs are brought together in the comparators. As the data is routed in, some comparisons can be made before the ladder is full and data can be routed out before the sort is entirely finished. The net result is that all but about 20% of the sorting time is overlapped with I/O.

(U) By having two ladders, with the first sorting while the second is being filled, it is possible to overlap all but 6% of the sorting time with I/O. After sorting the two separate lists, they are merged as they are read out. With three or four ladders, the non-overlapped sorting time is negligible.

SORTING MASSIVE FILES

(U) During the editing of this paper, a paper appeared containing an analysis of merge and bitonic sorting in the context of massive files. The paper, "A Comparison of Merge and Bitonic Sorting," R51/MATH/05/81, shows that when the distribution of data is known and uniform on all subsets of tapes then merge sorting is superior.
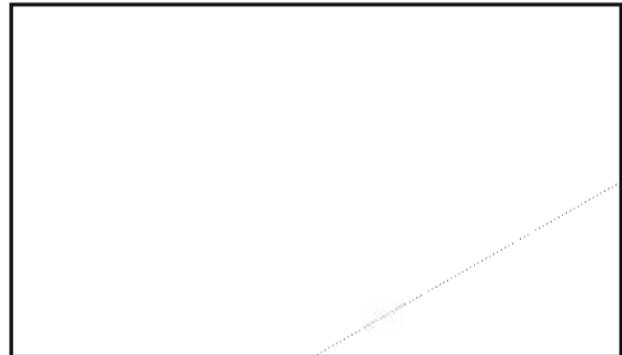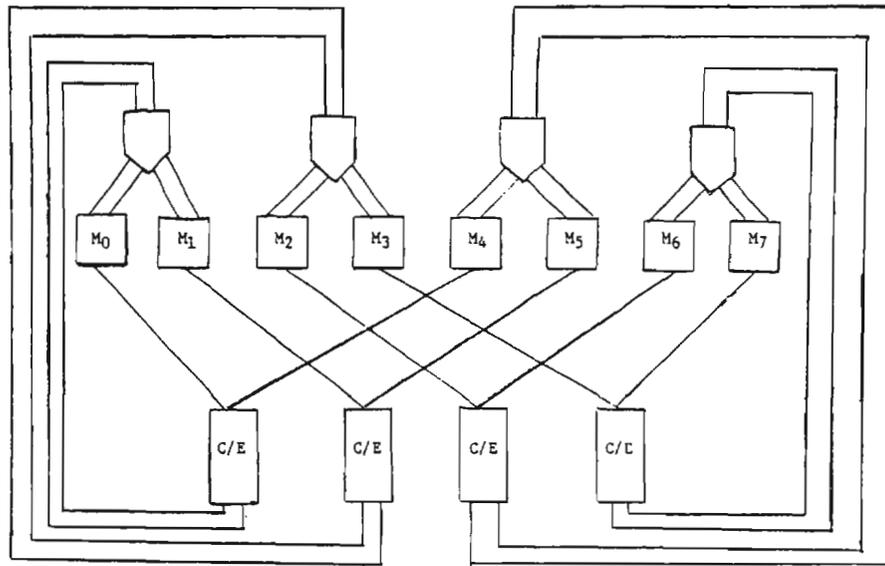
FASB Sorter

Figure 26.
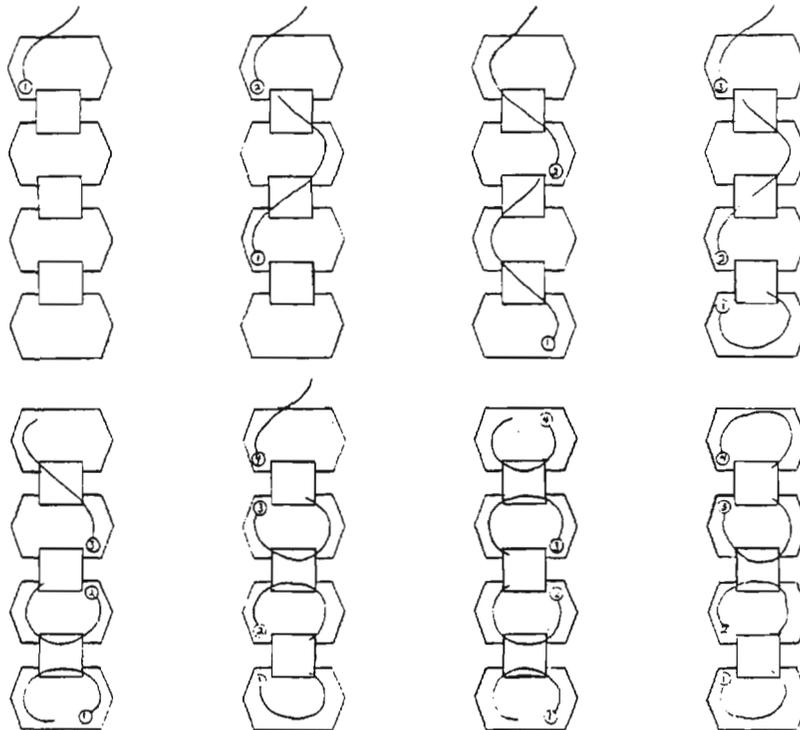
Single Ladder Odd-Even
Transposition Sort

Figure 28.

DOCID: 4011963
P.L. 86-36

(U) A standard tape reel has 2,400 feet of usable tape. Current high-density tape has 6,250 bits per inch and eight tracks (plus a ninth track for parity). Thus, one reel can hold about 2.25×10^7 64-bit words. If we allow for record gaps, we have perhaps 2×10^7 64-bit words. One of these tape reels can be sorted on a computer with sufficient disk storage in 20 minutes. Two sorted tapes can be merged at essentially the rate at which the data can be written out, about eight minutes per merge.

(U) One final note on sorting networks should be made. Batcher's bitonic sort of $N = 2^n$ elements in time $1/2 \, n^2 + 1/2 \, n$ is not the fastest known network. David van Voorhis [23] has developed a recursive construction that sorts in about time $1/2 \, n^2 + 1/4 \, n$, and the van Voorhis construction requires about $1/4 \, n^2 2^n - .37 n 2^n$ comparators versus $1/4 \, n^2 2^n + 1/2 \, n 2^n$ for a bitonic sort. However, the van Voorhis network is not as easy to analyze as is Batcher's network. For example, it is a simple task to determine which elements are being compared at any stage of a bitonic sort while the same cannot be done for an van Voorhis network. This means that for actual construction of a small-size network, van Voorhis' techniques would be used, though Batcher's networks will continue to be used to obtain performance estimates.

(U) The very pertinent question, "What is the best way to sort in parallel?" can best be answered: "It depends!" For a truly massive sort, some type of merging technique using tapes and many processors seems best. A parallel processor, though, is another matter. The instruction set, timing, and individual quirks are probably more important than the theoretical complexity of any algorithm. The best strategy, in advance of knowing the particular machine and its peculiarities, is to have a broad knowledge of the available algorithms.

EO 1.4.(c)
P.L. 86-36

APPENDIX

A Comparison of Some Algorithms in this Paper

Hirshberg's Bucket Sort:

Words: N
Processors: N
Interconnections: Each processor is connected to a common memory.
Time: $O(\log N)$

Valiant's Fast Merge:

Two Lists: N M
Processors: $\sqrt{NM}$
Interconnections: Each processor is connected to a common memory, or the connections are "sufficiently robust."
Time: $O(\log \log N)$

EO 1.4.(c)
P.L. 86-36

Preparata's Sort:

Words: $N = 2^n$
Processors: 2^{n-1}
Interconnections: Each processor is connected to a common memory.
Time: $O(\log N)$

Thompson and Kung's Sort:

Words: $N = n^2$
Processors: N
Interconnections: Mesh-connected, without end-around paths.
Time: $O(\sqrt{N})$

Batcher's Bitonic Sorter:

Words: $N = 2^n$
Processors: $N \log^2 N = n^2 2^n$
Interconnections: As required by the algorithm.
Time: $\log^2 N = n^2$

Stone-Batcher Sorter:

Words: $N = 2^n$
Processors: $N = 2^{n-1}$
Interconnections: Perfect Shuffle
Time: $\log_2 N = n^2$

Morris-Wisniewski Sorter:

Words: $N = 2^n$ in 2^k memories
Processors: $1 \quad 2^{k-1} \quad 2^{n-1}$
Interconnections: Multiple perfect shuffles
Time: $2^{n-k-1} \log^2 N = 2^{n-k-1} n^2$

BIBLIOGRAPHY

1. Batcher, Kenneth E., "Sorting Networks and Their Applications," Proc. AFIPS Spring Joint Comp. Conf. Vol 32, 1968, pp. 307-314.
2. Baudet, Gerard, and David Stevenson, "Optimal Sorting Algorithms for Parallel Computers," IEEE Trans. Comp. Vol. C-27, No. 1, January 1978, pp. 84-87.
3. Chen, T. C., K. P. Eswaran, V. Y. Lum, and C. Tung, "Simplified Odd-Even Sort Using Multiple Shift-Register Loops," Internat. J. Comput. and Inform. Sci., Vol. 7, No. 3, 1978, pp. 295-314.
4. Hirshberg, D. S., "Fast Parallel Sorting Algorithms," Com. ACM, Vol. 21, No. 8, August 1978, pp. 657-661.
5. [redacted] "A Merging Sort Machine," SCAMP Working Paper No. 18/78, IDA, September 1978.
   P.L. 86-36
6. Knuth, Donald E., The Art of Computer Programming, Vol. 1, Fundamental Algorithms, Reading, MA: Addison Wesley, 1973.
7. -----, The Art of Computer Programming, Vol. 3, Sorting and Searching, Reading, MA: Addison Wesley, 1973.
8. Martin, William S., "Sorting," Comput. Surveys, Vol. 3, No. 4, December 1971, pp. 147-174.
9. [redacted] "Costing Massive Sorts," R51/MEMO/06/80, 13 February 1980.
10. -----, "The FASB Sorter," S61 Informal Note No. 381, February 1976, S-218,292.
11. [redacted] "A Fast Parallel Sorting Processor," R51/MATH/17/78, S-216,947. P.L. 86-36
12. Nassimi, David, and Santaj Sahni, "Bitonic Sort on a Mesh-Connected Parallel Computer," IEEE Trans. Comp., Vol. C-27, No. 1, 1 January 1979, pp.2-7.
13. -----, "Parallel Permutation and Sorting Algorithms and a New Generalized Connection-Network," Technical Report 79-8, April 1979, University of Minnesota, (to appear in JACM).
14. Orcutt, Samuel Ellis, Jr., "Computer Organization and Algorithms for Very High Speed Computations," Ph.D. dissertation, Stanford University, September 1974.
15. Preparata, Franco R., "New Parallel-Sorting Schemes," IEEE Trans. Comp. Vol. C-27, No. 7, July 1978, pp. 669-673.
16. Schwartz, J. T., "Ultracompilers," Comp. Sci. Dept., New York University, 1979.
17. [redacted] "Shuffle Sorting with the Odd-Even Merge," SCAMP Working Paper No. 13/78, IDA, October 1978. P.L. 86-36
18. Smith, Burton J., "An Analysis of Sorting Networks," Sc.D. thesis, Dept. of E.E., M.I.T., August 1972.
19. Stone, Harold S., "Parallel Processing with the Perfect Shuffle," IEEE Trans. Comp., Vol C-20, No. 2, February 1971, pp. 153-161.
20. -----, "Sorting on STAR," IEEE Software Eng., Vol SE-4, No. 25, March 1978, pp. 138-146.
21. Valiant, Leslie G., "Parallelism in Comparison Problems," SIAM J. Comput. Vol. 4, No. 3, September 1975, pp. 348-354.
22. Van Voorhis, David C., "An Economical Construction for Sorting Networks," Proc. AFIPS Nat. Comp. Conf., 1974, pp. 921-927.
23. -----, "Efficient Sorting Networks," Ph.D. dissertation, Comp. Sci., Stanford University, December 1971.

ADA NEWS(U)

IRNSA has designated T303 as the Office of Primary Responsibility for Ada planning and transition activities at NSA. For information on the Agency's current and planned Ada technology utilization, contact [   ] or [   ] on 923-3227 or your local Cadre member.

The Ada Cadre is intended to serve as points of contact for the dissemination of information concerning Ada in addition to advising and assisting T303 as to Ada planning and requirements.

P.L. 86-36

Current Cadre members are:
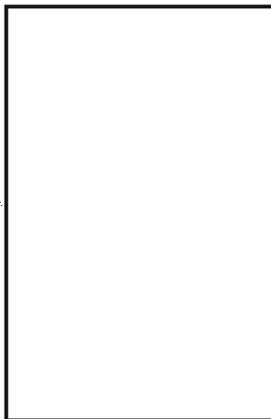
A043	963-3177
B62	963-4398
C31	968-7155
E09	968-8953
G33	963-4301
P309	963-5621
P372	963-4604
P5	963-3043
R09	968-7381
R623	968-8485
S352	972-2346
T303	963-3227
T303	963-3227
W09	963-3401

WE ARE ALWAYS
LOOKING FOR

ARTICLES, COMMENTS,
NOTES, LETTERS,

THAT WOULD BE
OF INTEREST TO
OUR READERS

SOLUTION TO NSA-Crostic No. 45

A. J. Salemme, "I Remember SPELLMAN," CRYPTOLOG, Jul-Aug 1978
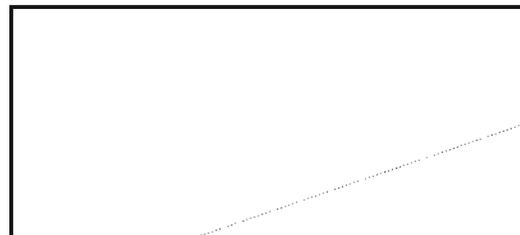
EO 1.4.(c)
P.L. 86-36

NSA - CROSTIC No. 46

P.L. 86-36

P.L. 86-36