AD-A083 854 NAVAL POSTGRADUATE SCHOOL MONTEREY CA F/G 14/5

A 16 - LEVEL COLOR DISPLAY SYSTEM FOR TWO DIMENSIONAL ULTRASONI--ETC(U)

DEC 79 E L MOON

UNCLASSIFIED NL

1 OF 2

AD A083854

# NAVAL POSTGRADUATE SCHOOL

## Monterey, California



# THESIS

A 16 - LEVEL COLOR DISPLAY SYSTEM FOR TWO DIMENSIONAL ULTRASONIC DATA

by

Eugene L. Moon

December 1979

Thesis Advisor: J. P. Powers

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

# REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER

2. GOVT ACCESSION NO. AD-A083 854

3. RECIPIENT'S CATALOG NUMBER

4. TITLE (and Subtitle)
A 16 - LEVEL COLOR DISPLAY SYSTEM FOR TWO DIMENSIONAL ULTRASONIC DATA

5. TYPE OF REPORT & PERIOD COVERED
Master's Thesis, Dec 1979

6. PERFORMING ORG. REPORT NUMBER

7. AUTHOR(s)
Eugene L. Moon

8. CONTRACT OR GRANT NUMBER(s)
National Science Found.
ENG 77-21600

9. PERFORMING ORGANIZATION NAME AND ADDRESS
Naval Postgraduate School
Monterey, California 93940

10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS

11. CONTROLLING OFFICE NAME AND ADDRESS
Naval Postgraduate School
Monterey, California

12. REPORT DATE
Dec[redacted]79

13. NUMBER OF PAGES
100

14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)
Naval Postgraduate School
Monterey, California 93940

15. SECURITY CLASS. (of this report)
Unclassified

15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)
Approved for public release; distribution unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES
This material is based on work supported by the National Science Foundation under Grant No. ENG 77-21600

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)
acoustical holography, computer display techniques, acoustical imaging

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)
A 16 - level color display computer program for use with an ultrasonic acoustic imaging system is presented. This display system is the second half of a two part ultrasonic imaging system built at the Naval Postgraduate School. The RAMTEK GX-100A display terminal and a PDP 11/50 digital computer were used as the display system. The software implementation and C language are discussed. Photographs of actual display using an experimental data tape and the computer program are also included.

DD FORM 1473, 1 JAN 73 (Page 1) EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-014-6601

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

Approved for public release; distribution unlimited

A 16- Level Color Disolay System for Two Dimensional Ultrasonic Data

by

Eugene L. Moon

Lieutant, United States Navy

B.S.C.S., University of Utah, 1972

Submitted in partial fullfillment

of the requirements for the

degree of

MASTER OF SCIENCE IN ENGINEERING SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL

December 1979

Author ______________________

Approved by: ______________________

Thesis Advisor

______________________

Second Reader

______________________

Chairman, Department of Electrical Engineering

______________________

Dean of Science and Engineering

# ABSTRACT

A 16 - level color display computer program for use with an ultrasonic acoustic imaging system is presented. This display system is the second half of a two part ultrasonic imaging system built at the Naval Postgraduate School. The RAMTEK GX-100A display terminal and a PDP 11/50 digital computer were used as the display system. The software implementation and C language are discussed. Photographs of actual display using an experimental data tape and the computer program are also included.

TABLE OF CONTENTS

## LIST OF TABLES

LIST OF FIGURES

## ACKNOWLEDGEMENTS

This material is based on work supported by the National Science Foundation under Grant No. ENG 77-21600.

Any opinions, findings and conclusions or recommendations expressed in this publication are those of the author and do not necessarily reflect the views of the National Science Foundation.

# I INTRODUCTION

This thesis deals with the development of a 16-level color display computer program, for use in a computer aided acoustic imaging system. The basic imaging system consists of a data acquisition system [1], a computer program for converting the data and controlling the display hardware, and the hardware for displaying the output.

The hardware display portion utilizes a RAMTEK GX-100A color raster scan display device that is controlled by a PDP-11/50 digital computer. This allows for real-time computer controlled displays to be presented.

Reference [1] discusses in depth the specific methods used for data collection and will not be elaborated upon in this thesis.

Section II provides bachground information about acoustical imaging and the computer program will be discussed in Section III of this thesis. Actual program listings are given in Appendix D. The equipment utilized for the displaying of data will also be discussed in Section III of this thesis.

In order to confirm the system realiability a data set was read into the PDP-11/50 and was used as the development tape.

# II BACKGROUND

## A. ACOUSTICAL IMAGING [3]

By using coherent ultrasonic waves of sound instead of a beam of coherent light to "illuminate" an object one can create acoustical holograms that become three-dimensional pictures when illuminated by laser light.

In order to produce an acoustical hologram the scene is "illuminated" with a pure tone of sound. The objects disturb the sound waves and produce interference patterns. This procedure is discussed in detail in Ref. [2] The pattern can be recorded in various ways. Once recorded the acoustical hologram can be reconstructed with a laser beam exactly as if it were an optical hologram. Other techniques such as computer processing can also be used to reconstruct the objects.

The interaction of sound with solids and liquids is different than the interaction of electro-magnetic radiation. Sound can travel a considerable distance, through dense, homogenous matter and lose little energy and yet it will lose a significant amount of energy when it passes through an interface.

This loss is due to reflection at the interface, which is converse to the energy losses that occur in electro-magnetic radiation Therefore sound can be significantly effective in medical diagnosis, in non-destructive testing and in seeing underwater and underground because it is mostly the discontinuities of the internal organs tumors, flaws, submerged objects or subterranean strata rather than the bulk matter, that is of interest to the observer.

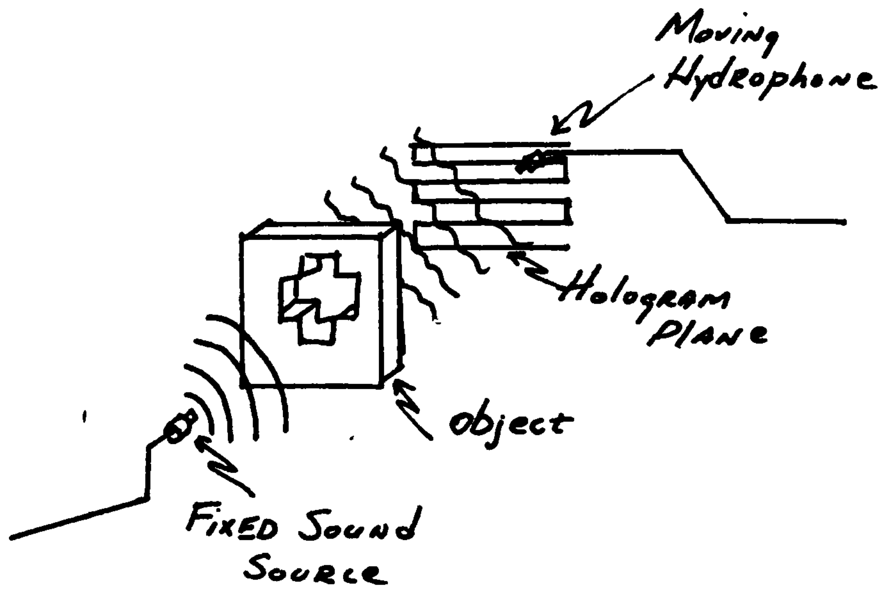
Acoustical imaging is not new; there are sonar devices that produce pictures similar to those on a radar screen, and can be used for prospecting for oil and minerals. Similar scanning methods are also in use by physicians for the detection of brain tumors and for examing the unborn child. In the latter examples the sound is a frequency of between one and ten megahertz. Another technique employs an acoustical "camera". In this method sound waves bounced off an object once focused with an acoustical lens onto an image converter that translates the patterns of sound intensity into a pattern of visible light.

The limiting feature of both these conventional methods is that the images show only two dimensions. They are two-dimensional because the methods detect only the intensity (the square of the amplitude) of the sound waves in the sound images. What the conventional methods are unable to record is the phase information, that is, the arrival time of the crest of the wave from the object with respect to the arrival time of the crest of a reference wave of the same frequency.

The most powerful feature of holography is that phase information as well as intensity information is retained in the hologram and can be subsequently "played back" in the optical image. Thus in acoustical holography there is a total transfer of information from the acoustical wave field to the visible optical wave field.

One type of acoustic holographic system which has been investigated is that which uses a scanning, detecting transducer moving through a raster pattern in the holographic plane. A typical system is illustrated in Fig. 1. The procedure involved was first described in an elementary form by Thurstone in 1966 [5]. Normally water is used as the medium for the acoustic propagation. A sound source is placed on one side of the object and a scanning hydrophone on the other. The use of a reference beam may be eliminated by means of electronic simulation. The acoustic object beam alone is detected and the detector output is added electronically to reference signal coming from the same signal generator that supplies power to the irradiating sound source. The resulting signal can then be processed to measure amplitude and phase which are subsequently recorded. In one version the signal is used to modulate the brightness of a small lamp that travels through a raster pattern in synchronism with the scanning hydrophone. The lamp brightness as a function of postion can be recorded on photographic film.

What has been described thus far is the most conventional of the point-by-point scanning systems. However it is possible to scan in many other ways. One example is suggested by the fact that reciprocity holds between the illuminating transmitter and the sensor. In this method, the point transmitter is scanned over the hologram aperature while the sensor remains fixed.




Point by Point Scanning System

Figure 1

Another example allows both the transmitter and sensor to be ganged together and made to scan as a unit over the holographic aperture. And yet another method is to hold both the transmitter and sensor stationary and scan the object to obtain an equivalent hologram. One of the problems with all these approaches is that the scanning is inherently slow,many transits of the scanning transducer are involved for wide band holograms. In order to overcome this problem various modifications have been proposed,several of which involve the use of transducer arrays. The arrays may be either of one or two dimensions and may be sampled or scanned mechanically or .electrically. In general these systems operate faster but typically do not provide as much image information because of limations on the number and uniformity of the elements of the array.

As reported in Ref. [2], a preliminairy version of ultrasonic imaging system with capability of two dimensional coherent data processing and computer image processing has been built and tested at the Naval Postgraduate School (NPS).

Reference [2] also indicated that further work was required to develop more software for processing and displaying of images produced by the system. This thesis deals with efforts in this direction.

## B. DATA COLLECTION

The type of system from which the data for this thesis was created is described in detail in Ref. [2].

Introductory information is included herein. The object is insonified by a 6" (15.24 cm.) diameter gold coated quartz tranducer The transmitted or reflected wave is detected coherently (ie.,both amplitude and phase) in some portion of a diffraction plane located an arbitary distance from the object. Detection of the complex wave is accomplished by a rasterscan of a single .04" (1.01mm) diameter PZT-5 ceramic receiver with a fundamental frequency of 1.014 MHz. Piezoelectric receiving elements offer an ideal combination of linearity and sensitivity[6] for this application.

After wave detection , the amplitude and phase of the wave are sampled, digitized, and recorded. The recorded data is then entered in the computer processing portion of the system at a later time. As indicated the Naval Postgraduate School (NPS) system consists of two parts : the data acquisition section and the computer processing and display section. The emphasis of this thesis has been placed on the presentation of the computer program used to display the data that was produced in the system previously discussed.

# III DISPLAY SYSTEM

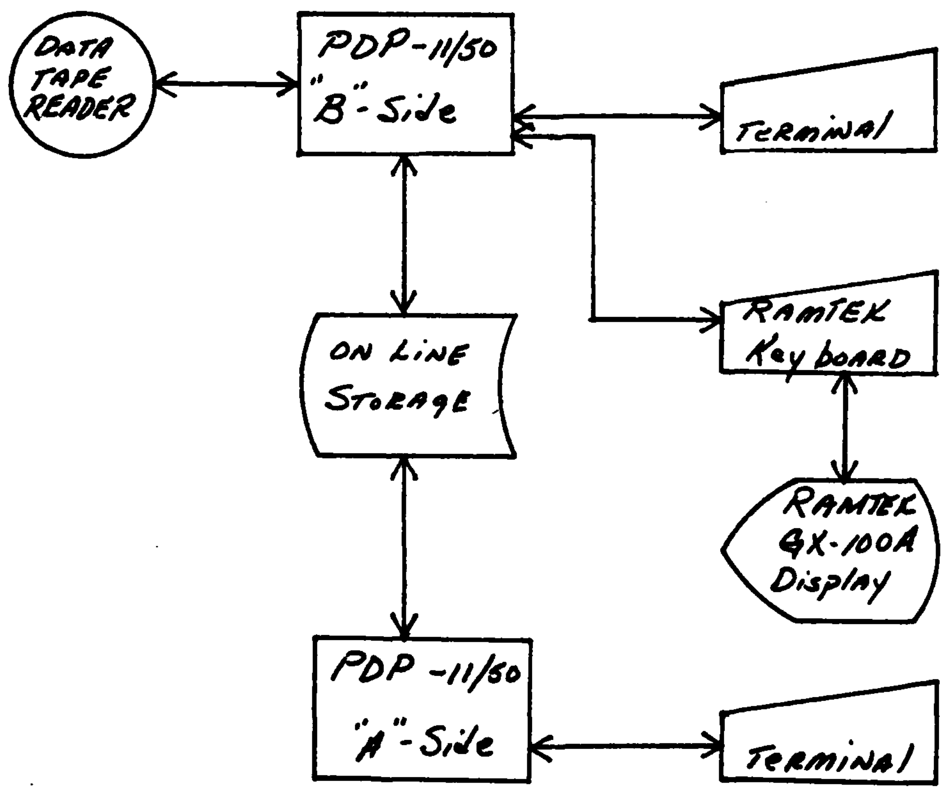
## A. GENERAL DESCRIPTION

The display system used is currently in operation at the NPS utilizing a PDP - 11/50 digital computer interfaced with a RAMTEK GX - 100A terminal and display screen, to generate the 16 - level color code displays of the digitized data.

Figure 2 shows a basic block diagram of the PDP - 11 and RAMTEK GX - 100A system as used.

The RAMTEK GX - 100A graphics display system utilizes a raster scan technique, with the display image data being extracted from an internal refresh memory. The RAMTEK system in operation at the NPS Computer Laboratory is hosted by the PDP - 11/50 computer and is accessed through this computer and its PWB/UNIX operating system, from one of the terminals in the lab.

The PWB/UNIX operating system is a library of executable statements that allow system operation to be controlled by simple call statements. The C language [7] programs discussed in this thesis are written for execution under the PWB/UNIX operating system and use many of these library calls.




Block Diagram of the Display System

Figure 2

## 1. RAMTEK Equipment

The RAMTEK GX - 100A is a color, raster scan display device, the heart of which is a color cathode ray tube, not unlike a home color television set. The method for generating its raster scan picture is identical to commercial cathode ray tubes. This is the only real similarity. The RAMTEK gets the information for its picture from a special video generator which reads the contents of a MOS refresh memory which contains all the information needed to produce an image on the screen.

It is not the intent of this thesis to discuss the details of how the electronics function, but to inform the user as to how to use the device to display the given data. A more detailed discussion of the functioning of the RAMTEK can be found in the RAMTEK GX - 100A [8] and RAMTEK GX -100B [9] programmers manual.

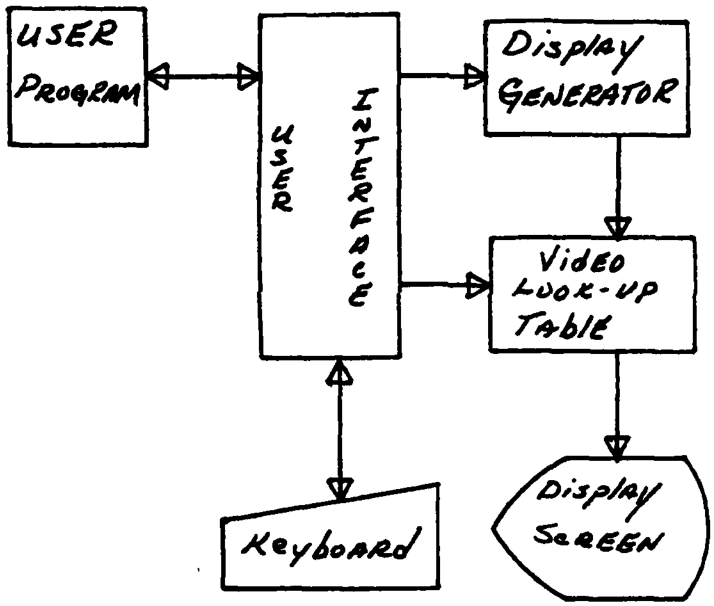
## 2. Operation

The RAMTEK system consists of a software interface to the user, a display generator, keyboard, video lookup table and a cathode ray tube. Figure 3 illustrates their relationship.

The user program executes on the PDP -11/50 computer. When the appropriate instructions are executed, the interface between the program and the display generator for the RAMTEK is activated. Information can be sent via the interface to the display generator from the user program or from the keyboard. Information can also be sent from the keyboard via the interface to the user program. The display generator interprets its instructions and constructs the display in its ROM memory. The image is generated on the CRT according to the colors contained in the video lookup table. Any image will remain ( unless over - written ), for an indefinite period of time in the ROM memory.

The RAMTEK graphics display system in the NPS Computer Laboratory utilizes a raster scan cathode ray tube with a screen resolution of 240 horizontal lines and 640 elements on each line. Each specific element is referred to as a pixel. This device has a high element resolution and a low line resolution. Therefore, the quality of certain images on the RAMTEK may be less than desirable. Lines that are not horizontal or vertical are drawn with a noncontinous, staircase effect. When utilizing the device these resolution factors should be kept in mind, espically when operating in one of the data modes.

Color is generated by mixing three primary colors: blue, green, and red each in varying intensities from 0 to 15.




RAMTEK Display System

Figure 3

a. Virtual Screen Addressing

The initialization of RAMTEK, which is accomplished by the execution of the instruction "ramtek()", sets various default values. The virtual screen is set to a standard cartesian coordinate system, with (0.0,0.0) at the lower left corner of the screen, and (100.0,100.0) at the upper right corner. This virtual screen can be redefined by execution of a "screen(x,y,x1,y1)" instruction, where x and y's are floating point numbers.

Upon execution of ramtek(), a shades of grey color table (table 0) is loaded, with color entry 15 of the table enabled for display purposes. It selects the alphanumeric control mode and opens the RAMTEK for reading and writing and finally erase the screen of any previous images or displays.

There are three modes of addressing the screen, absolute, indexed, and relative. The program in this thesis was done in the absolute mode, in which the user specifies a specific location on the user defined screen by issuing a "strtxy(x,y)" instruction. The 'x' and 'y' values establish the current operating position (COP) location. A discussion of the other modes of addressing can be found in Ref. [10].

b. Control Modes

The RAMTEK operates in any one of eight modes. Each mode has associated with it certain control flags which modify the operation of the specific instruction within each mode. Table I summarizes the control modes and control flags and their relationships.

TABLE I

CONTROL MODES AND CONTROL FLAGS

| MODE | | CONTROL FLAGS | | | | |
|---|---|---|---|---|---|---|
| number | name | IX | BK | AW | DW | FP |
| 0 | ALPHANUMERIC | X | X | X | X | |
| 1 | TRANSVERSE DATA | X | X | X | X | |
| 2 | RASTER DATA | X | X | X | X | |
| 3 | COMPLEX DATA | X | X | X | X | |
| 4 | GRAPHIC VECTOR | X | X | | | X |
| 5 | GRAPHIC PLOT | X | X | | X | X |
| 6 | GRAPHIC CARTESIAN | X | X | | | X |
| 7 | GRAPHIC ELEMENT | X | X | | X | |

definitions: IX - Index Addressing

BK - Reverse Background

AW - Additive Write

DW - Double Width

FP - Fixed Point

X - Control flag has an effect in this mode

(1) Alphanumeric Data Mode. The alphanumeric data mode is the default mode upon initialization of the RAMTEK. The COP is established by the execution of a strtxy(x,y) instruction. Following this instruction the instruction "strout(" character string ")" will allow a character string to be output to the RAMTEK screen. The character string can be no greater than 100 characters long beginning at the current operating position and continuing on the same line. After completion, an automatic line feed occurs which defines a new COP on the next line at the same starting point ( in x ) as the previous line.

When characters are drawn on the RAMTEK screen, the character appears in the designated color, while the rest of the screen is drawn in the background color. The description of how the color designation is accomplished will be discussed in Section III - c of this thesis.

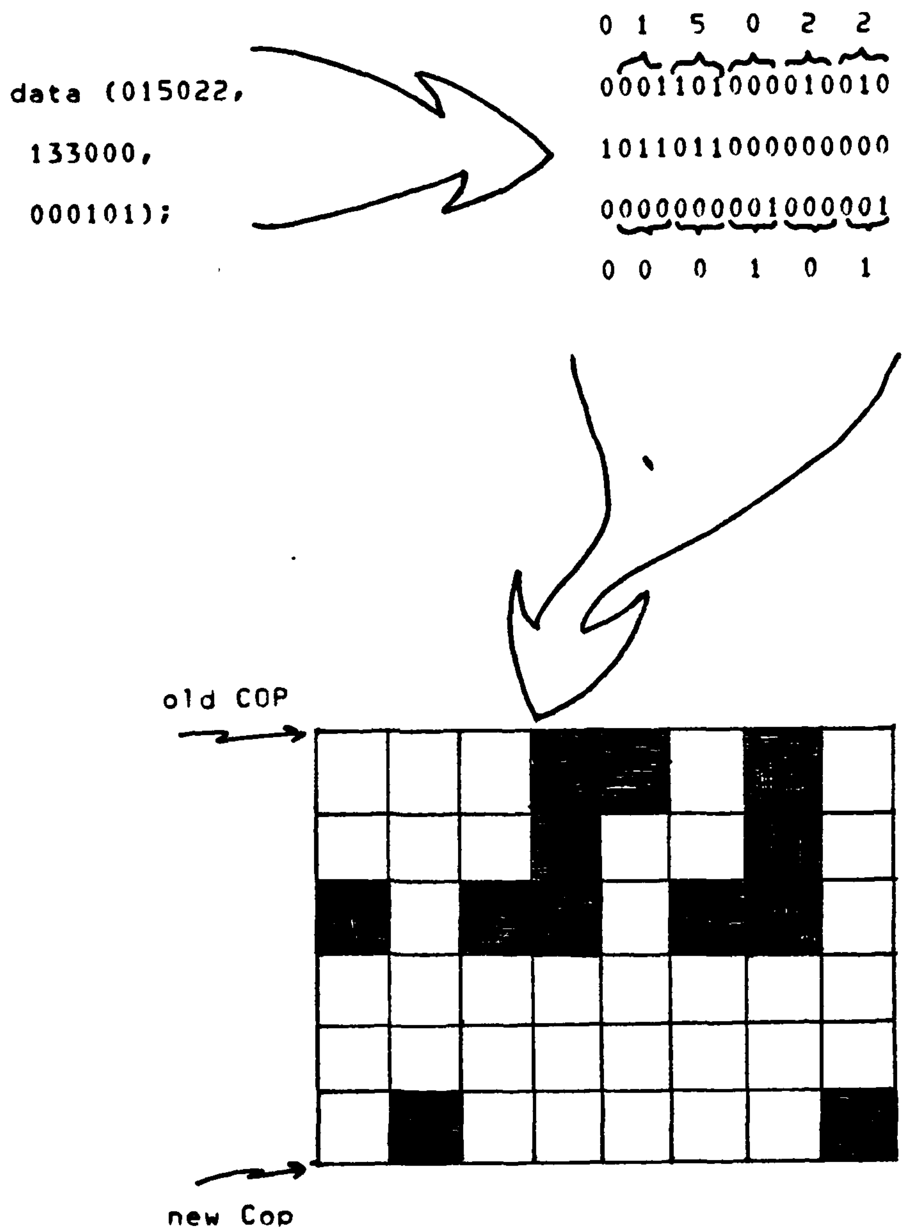
(2) Transverse Data Mode. The typical use of the transverse data mode is to define special symbols and characters which can not be found in the standard character set.

The various data modes all have one thing in common, namely the "data(name,m)" instruction. This instruction causes the raw data that is passed in a linear array pointed to by "name" to be displayed on the screen depending upon the current control mode. Here "m" is the number of bytes in the integer array named 'name'.

The execution of the data( name, m) instruction writes the information in raw format, that is each data byte is interpreted as a single bit per pixel description of eight consecutive pixels along a real screen line. As is illustrated in Fig. 4. execution of the instructuion:

```
data(015022,
     133000,
     000101);
```

causes each of the three words (015022, 133000, 000101) to be converted to binary representation. Eight binary digits (ones or zeros) are then used sequentially to produce an image with each pixel that corresponds to a binary one being intensified, while the zeros remain the background color.



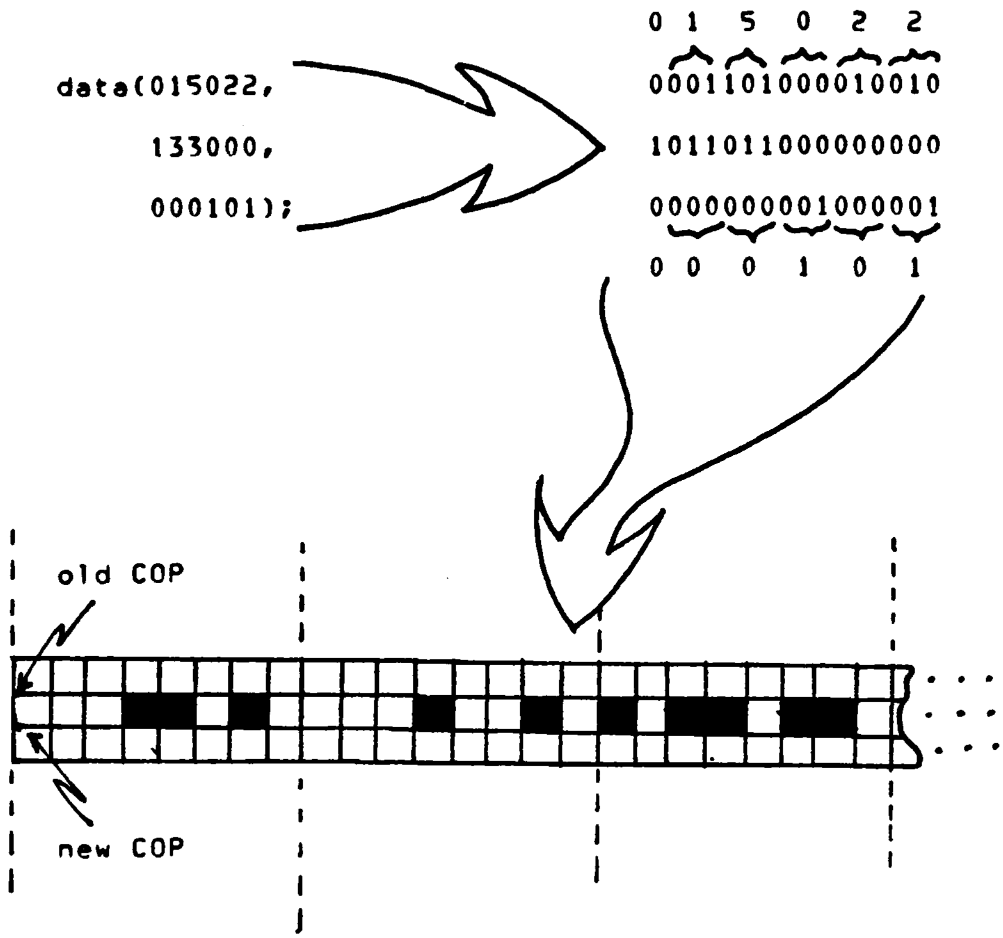

Transverse Data Mode

Figure 4

(3) Raster Data Mode. The raster data mode, except for the direction of the writing process is identical to the transverse data mode. The consecutive bytes transmitted by the data() instruction are written horizontally from left to right without starting a new line after eight bits are displayed. This is illustrated in Fig. 5 where the completion of eight bits (pixels) does not cause the COP to move to a new line.

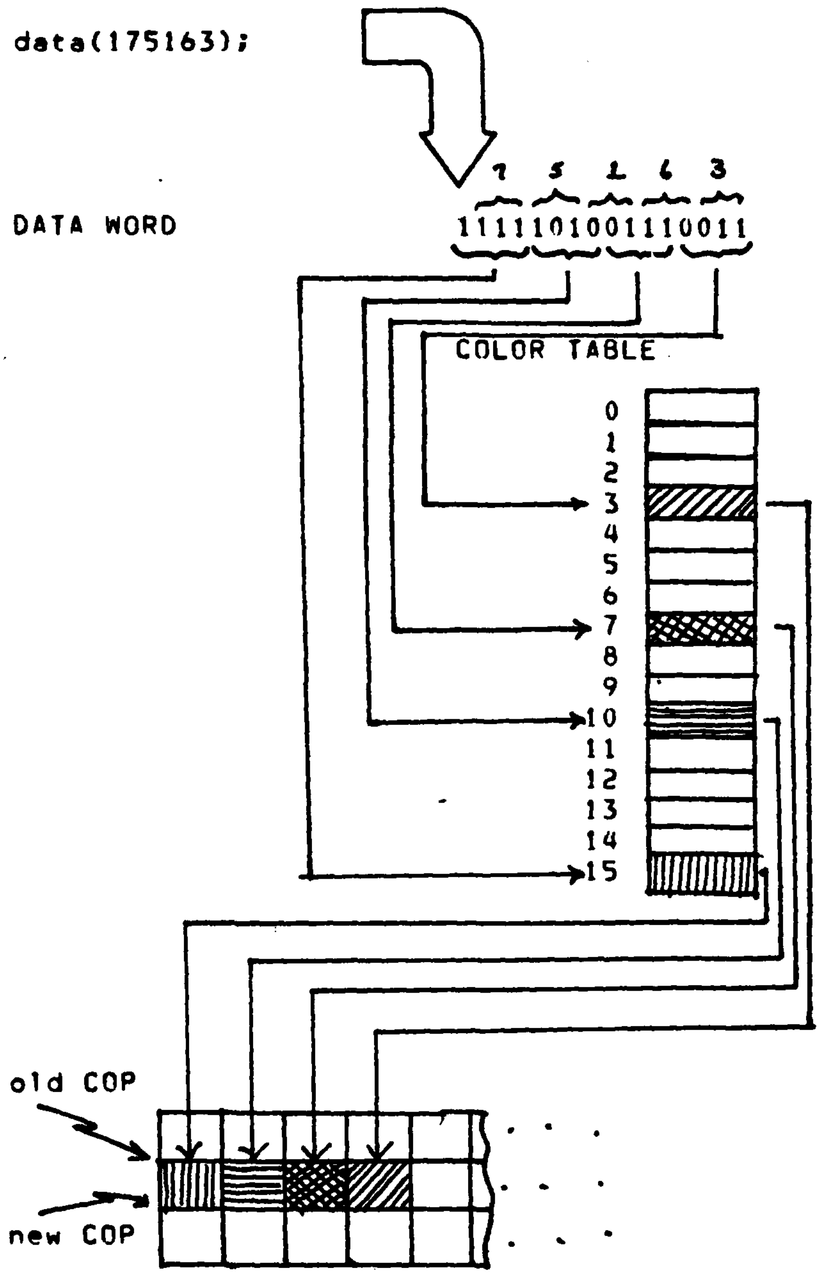
(4) Complex Data Mode. This mode writes data on the screen in the same manner that the raster data mode does. However each pixel is described by four bits of data instead of three. Thus, a single data word describes four pixels instead of two as in the previous data modes.

The color of each pixel being passed is defined by the four bits that are assigned to that pixel. This mode over-rides the color designated by the "color()" instruction. Unusual and unrealistic displays and images can be very easily achieved by the use of this data mode. Figure 6 illustrates how a data word is interpreted in this mode. The data word is converted to binary representation and each set of four bits represent an entry index of the current color table. The colors are drawn sequentially on the screen, using only four pixels per data word. The color() instruction will be described in Section III - c.




Raster Data Mode

Figure 5



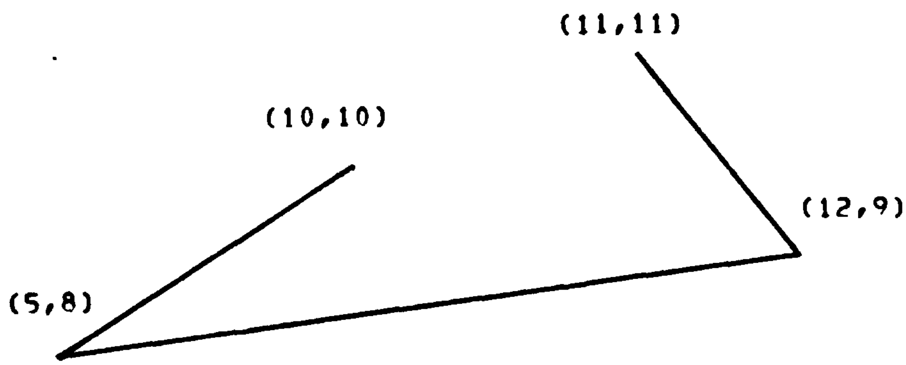

Complex Data Processing

Figure 6

(5) Graphic Vector Mode. The four graphic modes are used primarily for interactive drawing and plotting of data provided by the user. The graphic vector mode draws lines between arbitrary end points. The starting point is defined by the existing COP or it can be defined by the execution of strtxy(x,y), to establish a new COP. The end points can be defined by issuing either a "point(x,v)" or "pointr(x,y)" instruction. The first uses absolute or indexed addressing, depending upon the control flags condition, and the second uses relative addressing. The values of x, and y are real and must be exclusively different than the current operating point values. After execution, the COP is then the end point of the vector just drawn. Thus a linked line as shown in Fig. 7 can be drawn by issuing a strtxy() followed by successive point() instructions.

Instrutions:

```
setmode(4,0);
strtxy(10.0,10.0);
point(5.0,8.0);
point(12.0,9.0);
pointr(-1.0,2.0);    /* relative */
```

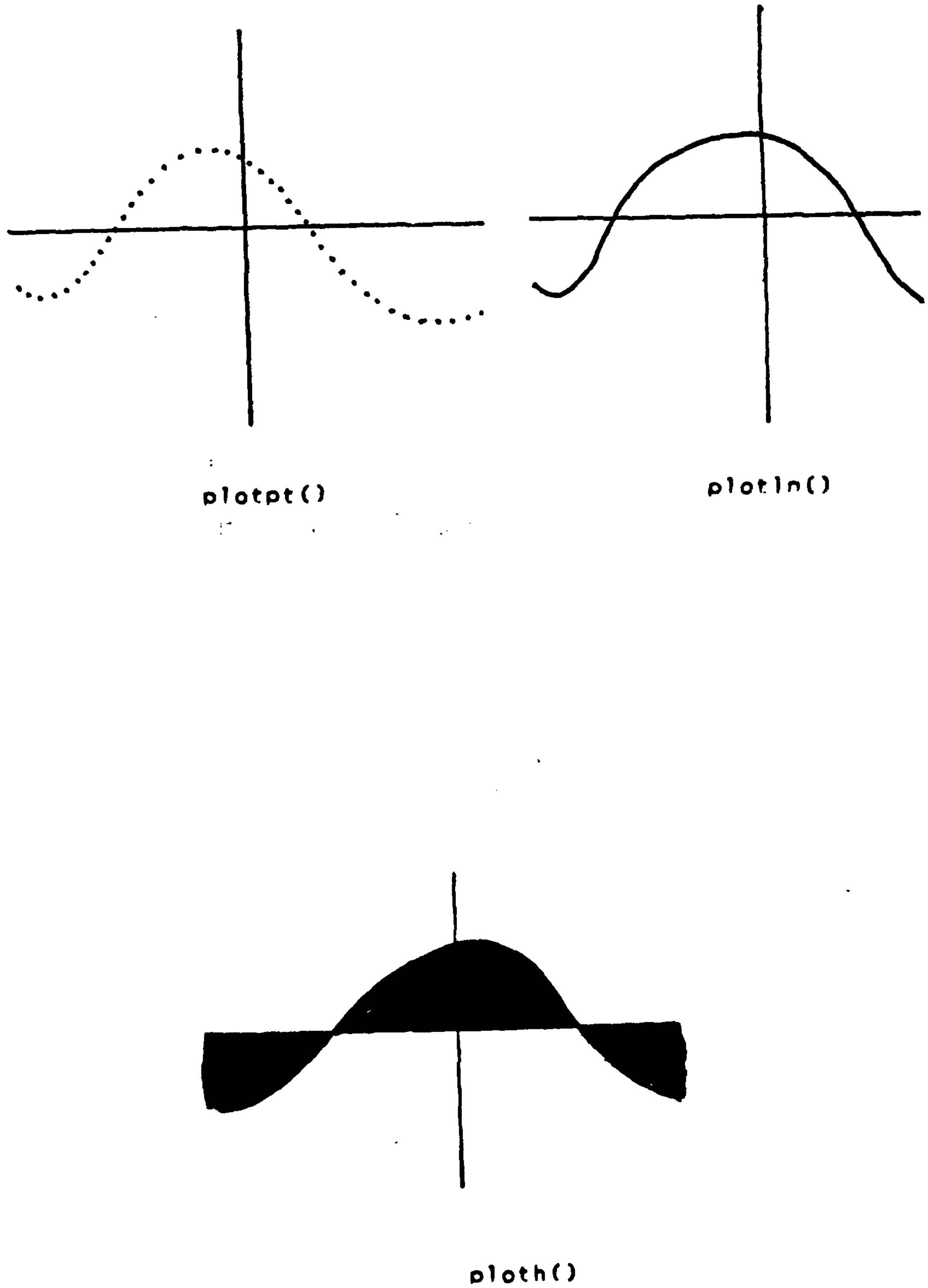



Drawing vectors in Graphic Vector Mode

Figure 7

(6) <u>Graphic Plot Mode.</u> The displays generated by the graphic plot mode have been implemented as a set of plot routines in the user interface. The three routines are "plotpt()" (plots points) "plotln()" (plots continious lines), and "ploth()", (plots a histogram). Each of the methods require the user to specify the points to be plotted. All three plotting methods are illustrated in Fig. 8. Detailed instructions on their use can be found in Appendix B of Ref. [10].

(7) <u>Graphic Cartesian Mode.</u> The graphic cartesian mode draws solid rectangles between arbitrary end points. The end points are defined as the lower left and upper right corners of a rectangle. The COP is the completion point of the drawing unless a new COP is defined with a strtxy() instruction. Second and subsequent rectangles can be drawn by the use of point() or pointr() instructions. With the fixed point flag set, a strtxy() instruction determines the common point for and between subsequent point() or pointr() instructions. Figure 9 illustrates these characteristics.

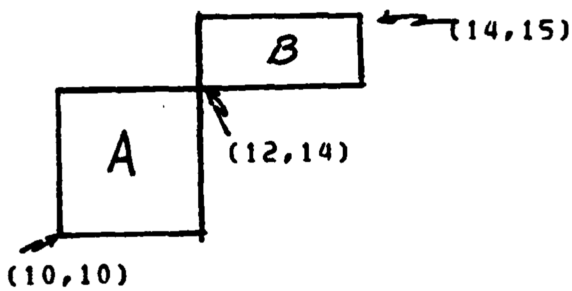


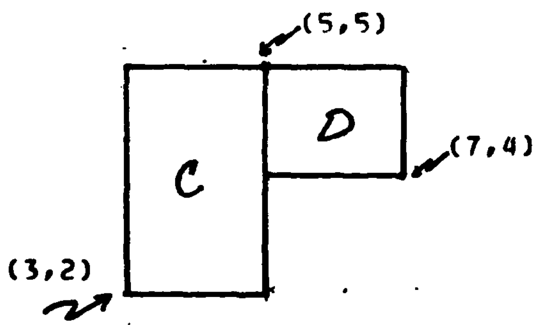

Graphic Plot Mode

Figure 8

Instructions:

```
setmode(6,0);
strtxy(10.,10);
color(1);
point(12.14.); /* Draw A in color 1 */
color(2);
point(14.,15.); /* Draw B in color 2 */
fixpt(1); /* set fixed point */
strtxy(5.,5.);
point(3.,2.); /* Draw C in color 2 */
color(3);
point(7.,4.); /* Draw D in color 3 */
```




Graphic Cartesian Mode

Figure 9

(8) Graphic Element Mode. The graphic element mode draws a single pixel on the screen as determined by a point() or a pointr() instruction. After execution, the COP is on the same real screen line and one screen element to the right.

c. Color Usage

All images that are displayed on the screen are displayed according to an entry in the video look-up table, which is commonly called the color look-up table.

(1) Defining Colors. The RAMTEK hardware defines colors by the combining red, blue and green in varying intensities to produce color. Since the color lookup table, (from which the CRT receives its instructions on how to mix the three colors,) stores information in digital format, the user is limited to 2↑12 (4096) possible color definitions. In order to define an entry in a color table, the instruction "triple(b,g,r)" is used to convert the three input parameters blue (b), green (g), and red (r) into an integer which is used for insertion into a color table entry. Each entry represents the code for a color in a twelve bit word. Each word is broken into three, four bit binary numbers which represents intensities of blue, green, and red to be mixed.

A color table that can be used for an image is limited to sixteen entries, these are the only colors that can be displayed at any one time on the screen. Entry zero of the color table is used as the background color, thus leaving the fifteen other colors for use in displaying images on the screen.

As one becomes familier with the RAMTEK display it will become obvious that many of the "colors" are simply shades of the same color, many of which appear to be almost black or almost white. It is extremely difficult for an ordinary person to distinquish between (0,0,0) , (0,0,1), (0,1,0), or (1,0,0) all of which are very 'close' to black.

The RAMTEK interface system permits the use of eighteen separate color tables, the first four tables ( 0 thru 3 ) are system reserved tables, and can not be modified in any way. The user may define as many additional color tables as desired up to fourteen others, as described in the following section. The system color tables are:

table 0 - shades of grey          table 1 - shades of blue

table 2 - shades of green          table 3 - shades of red

(2) Loading a Color Table. Loading a user generated color table requires a series of operations. First an integer array of sixteen words must be declared. Then it is necessary to load each entry of the array from zero to fifteen with the sixteen colors that are desired. In order to accomplish this, the triple(b,g,r) routine is used to code the ordered triples and then assign the returned codes to each entry in the integer array.

After the array has been loaded, the routine "clrtbl(n,name)" is used to load the array into the actual color table, numbered 'n', where n is the number of the table between four and seventeen, and named 'name'. An example of the code to load a color table is as follows:

```
int aa[16] ;              /* declare a 16 word array */
 .
 .
 .
aa[0] = triple(0,0,0);  /* color black */
aa[1] = triple(15,0,0); /* color blue */
aa[2] = triple(0,15,0); /* color green */
aa[3] = triple(0,0,15); /* color red */
aa[4] = triple(15,0,8)  /* color violet  */
aa[15] = triple (15,15,15)      /* color white */
 .
 .
 .
clrtbl(5,aa);             /* load color table 5 with
                              the array aa */
```

d. Programming the RAMTEK

In order to use the RAMTEK system it is necessary to place prior to any RAMTEK instructions in the C language program the ramtek() routine. This routine must be the first call made in the user program. It is the initialization routine for the user interface and it opens the RAMTEK device and keyboard.

In order to utilize the interface, when compiling a C language program for RAMTEK the system command is :

```
% ramtek filename.c
```

Appendix A contains a list of reserved words. These words should not be used within any user programs that utilize the user interface for the RAMTEK system.

## B. DISPLAY PROGRAM

### 1. C Language

This display system was implemented in the C programming language [7], and is hosted by a PDP - 11/50 computer in the NPS Computer Laboratory.

The RAMTEK routines were all previously written in the C language and were for the most part adequately documented in Ref. [10]. The C language itself is well documented in Refs. [7] and [11], and allows the user to write clear and concise programs. The C language was available under the PWB/UNIX operating system, which is discussed in Refs. [12] and [13].

A C program consists of one or more functions, which are similiar to the functions and subroutines of a Fortran program. "main()" is such a function; all C programs must have a main(). Execution of the program begins at the first statement of main, and main will usually simply call or invoke other functions, some which are user defined, others from libraries.

C language has four types of variables.

int - integer ( 16 bits )

char - one byte character ( 8 bits )

float - single-precision floating point

double - double precision floating point

There are also arrays and structures of these basic types, pointers to them and functions that return them. All variables in C must be declared and must preceed executable statements.

The basic conditional- testing statement in C is the if statement, the simplest form of an if statement is :

if ( expression ) statement

The expression is evaluated and if it is true, the statement is executed. Individual statements end with a semi-colon ( ; ). There can be used an optional else clause.

The Relational operators are:

== equal to

!= not equal to

> greater than

< less than

>= greater than or equal to

<= less than or equal to

Tests can be combined with "&&" ( AND ), "||" ( OR ), and "!" ( NOT ).

The basic looping mechanism in C is the while statement. The while statement is a loop whose general form is :

```
while ( expression ) statement;
```

Its meaning is :

(a) evaluate the expression

(b) if its value is true,

do the statement and go back to (a).

The arithemetic operators are the usual add '+', subtract (minus) '-', multiply '*', and divide '/' and the remainder or mod operator '%'. The else clause can be used to devise more elabroate programs such as :

```
if ( expression ) statement1 else statement2
```

or to construct logic that branches one of several ways :

```
if (....)
        {....}
else if (....)
        {....}
else if (....)
        {....}
else
        {....}
```

where the statements of the function are enclosed by the brackets {}.

In addition to the usual incrementing and decrementing methods, C has two other unary operators '++' ( increment ), and '--' ( decrement ). Where ++n is equivalent to n = n + 1 . The unusual feature of '++' and '--', is that they can be used before or after a variable. The value of ++k is the value of k after it has been incremented. The value of k++ is the value of k before it is incremented.

Arrays can be made as in Fortran, thus an array of ten integers is created by the following declaration :

```
int x[10] ;
```

square brackets [] are used for subscripting, and parentheses are used for function references . Array indices begin at zero, thus the elements of x are : x[0], x[1], x[2], ........x[9]. Multi-dimension arrays are provided, with the declaration similar to

```
int name[10][20] ;
```

name has 10 rows, and 20 columns; and the rightmost subscript varies fastest.

Text is usually kept as an array of characters. The statement:

```
printf("%d:\t%s",n,line);
```

will print the integer n, a colon, tab five spaces and print the characters stored in the array named line. The symbol "%d" indicates to print an integer, the symbol "\t", indicates to tab five spaces while the symbol "%s" indicates to print a character array. Each variable symbol must be identified in the description portion of the statement (that portion not enclosed by quotation marks).

Another method is to place the output between quotes as :

```
printf(" Mary had %d ducks\n", i) ;
```

with i equal to 7 , will print

```
Mary had 7 ducks
```

the symbol '\n' is the carriage return control character.

The for statement is a somewhat generalized while statement, that allows the user to put the initalization and increment parts of a loop into a single statement.

An example is:

```
for ( i = 0; i < k; i++ ) {
     statement1
     statement2
}
```

which executes statement1 and statement2 k times.

Global variables, variables common to all functions are declared outside of main(). Local variables are declared inside a function, while the declaration of passed variables is done between the function name argument list and the opening '{'. An example is shown on the following page.

```
int z, k ;                    /* global variable */
main()
{
        int x[10] ;
        count(sum,10) ; /* calls count routine */
        printf("The answer is %d0, x[i]) ;
        k = x[1] * x[2] + ( x[3] - x[4] )
count(total,size)
int total[], size ;                /* passed variables */
{
        int i, c ;                 /* local variables */
        for ( i = 0; i < size; i++ )
        {
        .
        .
executable statements
        .
        .
        }
        for ( z = k; z < c; z++)          /* z global */
        {
        .
        .
        }
        return ;
}
```

A pointer in C is the address of something. The unary operator '&' is used to produce the address of an object. Thus:

```
int a, b ;
b = &a ;
```

puts the address of a into b. If b is declared a pointer as :

```
int a, *b, c ;
b = &a ;
c = *b ;
```

b contains the address of a and c = *b means to use the value of b as an address ( as a pointer ).

An unusual feature of C is that normal binary operators like the '+' and the '-' can be combined with the assignment operator '=' to form new assignment operators. For example "x =- 10" uses the assignment operator '=-' to decrement x by ten, and x =& 0177 forms the AND of x and 0177. The space immediately following the equal sign is critical! X = -10 sets X to minus ten, while X =- 10 decrements X by ten.

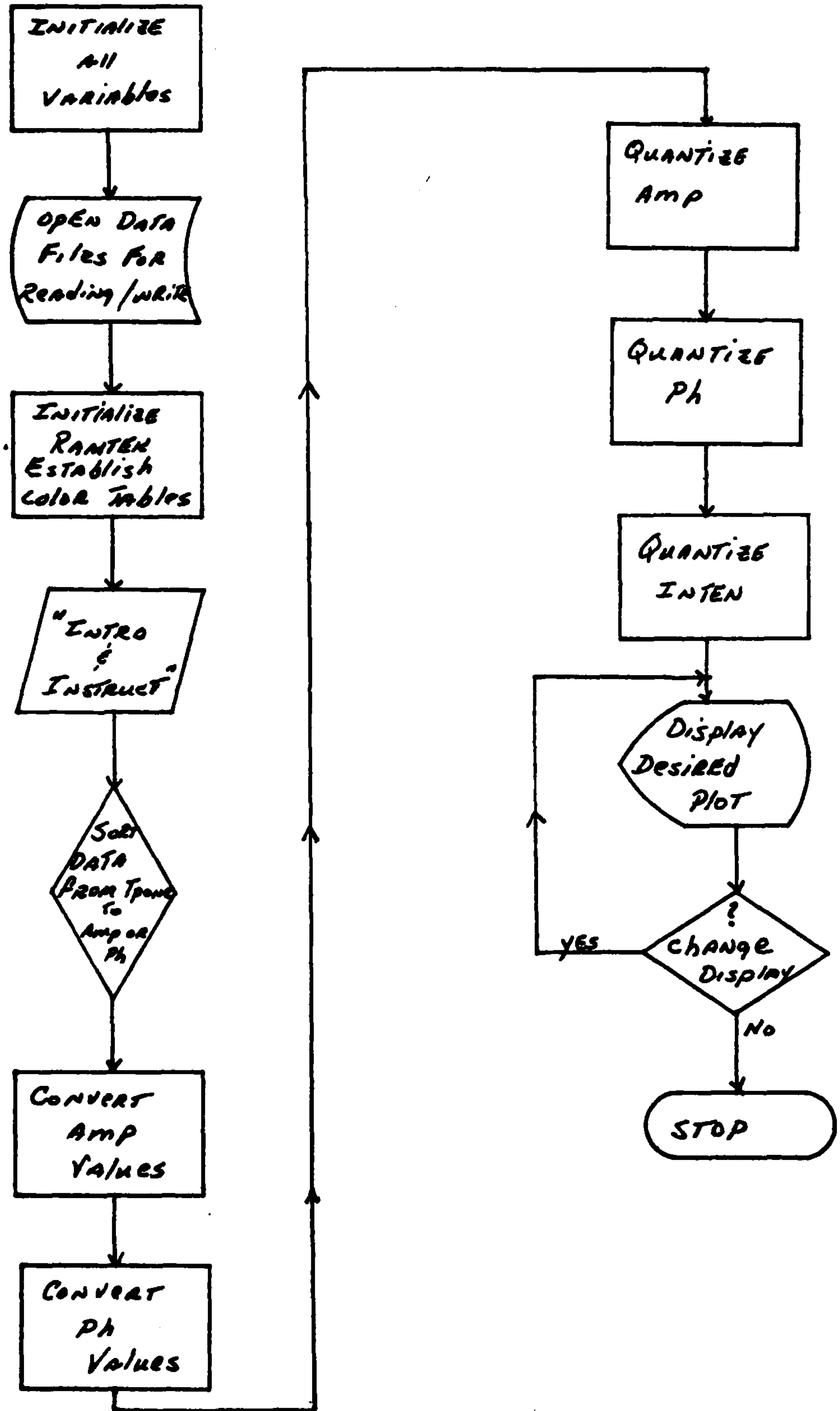
Appendix B lists key words in C language and may not be used otherwise. Additional information may be obtained from Refs. [7] and [11].

## 2. Program Description

A complete program listing of the display program can be found in Appendix D. The filename of this program is see3.c, and can be copied, moved, or edited using the routine system functions of PWB/UNIX listed in references [12] and [13] .

The purpose of this section is to discuss each routine individually, each routine's purpose (output) and its relationship with other routines in the program.

Figure 10 shows functional flow of the display program. The general purpose of the program is to read a data file into memory, sort the data into two separate files, and convert the two files which have relative values in them to absolute values. Next the program quantizes each file to 16 levels for use in displaying the image. During execution the RAMTEK system is initialized and the image is displayed with an option to change the selected display type. The basic flow is that the function main() calls each routine, that routine accomplishes a speciific function and returns to main for the next function call. Appendix C is a more detailed flow chart of the program and subroutines.




General Flow Chart

Figure 10

a. Declarations

The following variables are declared as global variables so that multiple declaration statements are not required.

| variable | function ( use ) |
|---|---|
| amp | filename to store 4096 (64x64) relative values of amplitude |
| ph | filename to store 4096 relative values of phase |
| inten | filename to store 4096 relative values of intensity |
| i, j, k | integer counters |
| m | ASCII conversion to integer of four bits of "tpone" |
| n | integer value for the selected color table |
| show | integer value for the selected display |
| flag | integer indicator as to whether the routine showclr() has been executed or not. If flag = 1 then erase the screen in the routine draw(). |
| fd1 | integer file descriptor for the opening of file tpone |
| fd2 | integer file descriptor for the opening of file amp |

| | |
|---|---|
| fd3 | integer file descriptor for the opening of file ph |
| fd4 | integer file descriptor for the opening of file mag |
| fd5 | integer file descriptor for the opening of file phase |
| fd6 | integer file descriptor for the opening of file inten |
| fd7 | integer file descriptor for the opening of file mag2 |
| aa[16], bb[16], dd[16], ee[16], ff[16] | 16 - word integer arrays to store various color tables |
| cc[4] | four character - character array to store four values of tpone either amp or ph |
| *pc | pointer to character pc |
| q | character, q = 'b' is a test value for determining that input from the keyboard has occured |
| z | character for labeling the color bar table |
| mag | filename to store 4096 actual values of magnitude |
| phase | filename to store 4096 actual values of phase |

| | |
|---|---|
| x, y | real counters for positioning the COP for displaying the data |
| maxa | maximum value of amplitude |
| mina | minimum value of amplitude |
| maxp | maximum value of phase |
| minp | minimum value of phase |
| maxi | maximum value of intensity |
| mini | minimum value of intensity |
| difa | difference between maximum and minimum amplitude |
| difp | difference between maximum and minimum phase |
| difi | difference between maximum and minimum intensity |
| mag2 | filename to store 4096 real values of intensity (magnitude squared) |

b. Main Program

The purpose of the main program is to call the various routines that are used to accomplish the purpose of the program. The flow of execution through main() is sequential. The compilation and the execution of this program causes specific data files to be filled with display data words. These files must be previously created by the user. One method is to use the "ed" (edit) command [12] such as:

%ed amp  or  %ed inten

Seven data files must be created and reside in the users library prior to program execution. This procedure will be explained further in section A-3 of this chapter.

If the program has been used to display data, causing these data files to have the correct information stored in them, it is unnecessary to re-fill them with the same information again upon subsequent executions of the program that uses the same input data tape.

Comment indicators '/*' are recognized and cause the compiler to cease compiling and continue reading, until a '*/' is sensed, when compiling continues again. The statements included betweem '/* and */' are recognized as unexecutable comment statements and have no effect upon program execution.

The six routines readdata(), amptomag(), phtophas(), amplevel(), phlevel(), and intlevel() are used to fill the appropriate data files.

Since only one test data tape was avaliable, with its information being stored in data file "tpone", it was unnecessary to execute these routines each and every use of the program.

Comment indicators were placed prior to and following each routine call from main(), in order to preclude execution of six time consuming routines. Use of these comment indicators significantly reduces the compilation, and execution time of the program. If it becomes necessary to change a data file, because of additional or different data tapes, the simple removal of the appropriate comment indicators will easily falicitate the use of the specified routine.

The main() routine sets three variables (n, q, and show) to initial values for use by subsequent called routines. The program terminates upon the execution of an exit(), which is executed in the finish() routine.

c. fileopen()

The purpose of fileopen(), is to open seven data files, which must already exist in the users library, for reading and writing of information. If these files do not exist in the user's library, refer to Ref.[12] for further edification.

It should be kept in mind that tpone, amp, ph, mag, phase, inten, and mag2 are external files that have data stored in them, and that this data must be stored by a write instruction and must be accessed by a read instruction.

It is the user's responsibilty to ensure that all necessary files do exist and are accessible prior to program execution. Returned values are 1 if successful opens have been executed; otherwise the program will terminate upon an unsuccessful opening of any file.

d. readdata()

The purpose of readdata() is to separate the data from tpone into two files, amp and ph.

The original values of data were recorded onto a papertape in alternating sequence, amplitude then phase, for in excess of 4096 data pairs. The data was recorded in ASCII format, thus it is also converted from ASCII to integer within the routine readdata().

The variable cc is a character array four characters long. When the array is filled with three characters, and a ' ' (space), as the fourth character, the library routine atoi() is called which converts the contents of the cc array into an integer variable m. Depending upon whether the variable K is odd or even, m is then written into either the amp file or the ph file. This sequence is repeated 4096 times. Upon completion of readdata(), no further use of tpone is required , and two seperate data files, amp and ph will each have 4096 integer values of amplitude and phase stored in them respectively.

e. amptomag()

The purpose of amptomag(), is three fold: first it converts the relative values in amp (in db) into actual values in mag (in db); secondly, it finds the intensity (in db) of the values in mag and assigns them to mag2; and thirdly, amptomag determines maximum and minimum values for magnitude (mag) and magnitude squared (mag2).

One value of amp is read, mag is assigned a specific value according to the equation:

$$mag = -20.18 - .1734 * amp$$

This is necessary because the data acquisition system records relative values. For display purposes actual values are required necessitating a system calibration to produce the above equation. If the calculated value of mag is greater than the current value of maximum amplitude, then max amplitude is assigned this value. If this value of mag is less than the current value of minimum amp, then min amp is assigned this value. The magnitude squared value is then tested in the same manner, and finally mag value is written into mag file, and the mag2 value is written into the mag2 file.

f. phtophas()

The routine phtophas() is similar to amptomag() in purpose namely to convert from relative values of phase to actual values, and to find maximum and minimum values.

One value of ph is read and depending upon whether it is less than or equal to or greater than 127, it is assigned a specfic value of phase (in degrees) according to:

```
if ph < 127
    phase = 1.139 + 1.4946 * ph
else
    phase = -380.28 + 1.4917 + ph
```

As with amp this equation results from calibrations performed on the acquisition system.

Maximum and minimum values are tested and saved accordingly. Finally the value of phase is written into the phase file.

g. amplevel()

The purpose of amplevel() is to divide the range of amplitudes into 16 equal levels, (because each color table has a possibility of sixteen color entries), to test each individual value and to assign it a new level number according to its value. If maxa has a value of 600 and mina has a value of 200, sixteen equal levels would be 25 units wide. The general equation used is:

difference = maximum - minimum;

For each value:

```
                        (maximum - value) * 16
level value = 16 - ------------------------
                            difference
```

Thus a value of 317 in the above example would be given a level value of 4.

```
             ( 600 - 317 * 16 )
    16   -   ------------------   =   4
                    400
```

upon completion of amplevel(). The file amp will then contain integers from zero to fifteen representing the various ampltitude levels. These values will be used as color table indices in subequent functions, for displaying the image.

h. phlevel()

The purpose of phlevel() is indentical to the purpose of amplevel(). Upon completion of phlevel(), the file ph will contain integers from zero to fifteen representing the various phase levels.

i. intlevel()

As with amplevel() and phlevel(), the purpose of intlevel() is to generate a file with values from zero to fifteen representing various levels of intensity. Upon completion of intlevel() the file inten will contain integer values from 0 to 15 representing the various intensity levels.

Each of the six preceeding routines are called from main(), and are only called one time. If the program has been previously compiled and excuted, such that current data is stored in amp, ph, and inten files; deletion of the comment indicators should be seriously be considered prior to compiling and execution of the program.

j. ctable()

The purpose of ctable() is to define additional color tables that can be used for display.

The five arrays have been declared globally and by the use of the triple() routine, various color values have been assigned to each element of the five arrays.

Remember that there are four color tables that are system color tables, that are also availiable for use, namely greys, blues, and greens. The arrays are assigned specific color table numbers and names by use of the clrtbl (n, name) routine.

k. display()

The purpose of this routine is to create the image on the display screen according to values stored in the desired display file. Once the image has been drawn, the routine calls the routine draw() to display the color table that is in use, sets the desired selection indicator to zero and calls the routine change(). Control is never returned to main() and the program can only terminate in the routine finish() which is called by change().

The selected color table is the variable n and the brightest color entry is assigned for use as writing text on the screen. All colors are available for use in image displaying however one of the colors is used to interact with the user, namely entry 15.

The COP starts at the upper left, at cartesian point (30.,69.375). For 64 values of i, X is incremented by 0.625, the color level value is obtain from the selected display file, used as the variable in color() and a block 0.625 by 0.625 units is drawn by the block() instruction. After 64 blocks have been drawn, Y is decremented by 0.625. X is assigned the value of 69.375 and for 64 values is decremented by 0.625. The color level values from the display TYPE file, are obtained and the block drawn in the appropriate color. Figure 12 is a pictorial representation of this procedure.

The seek() instructions are system functions that move a pointer to the specefied position of the specified file.

seek (fd2,0,0);

orders the pointer to point to the file described by "file description two" (fd2), point to the zero position ( the first zero) with a zero offset (the second zero).

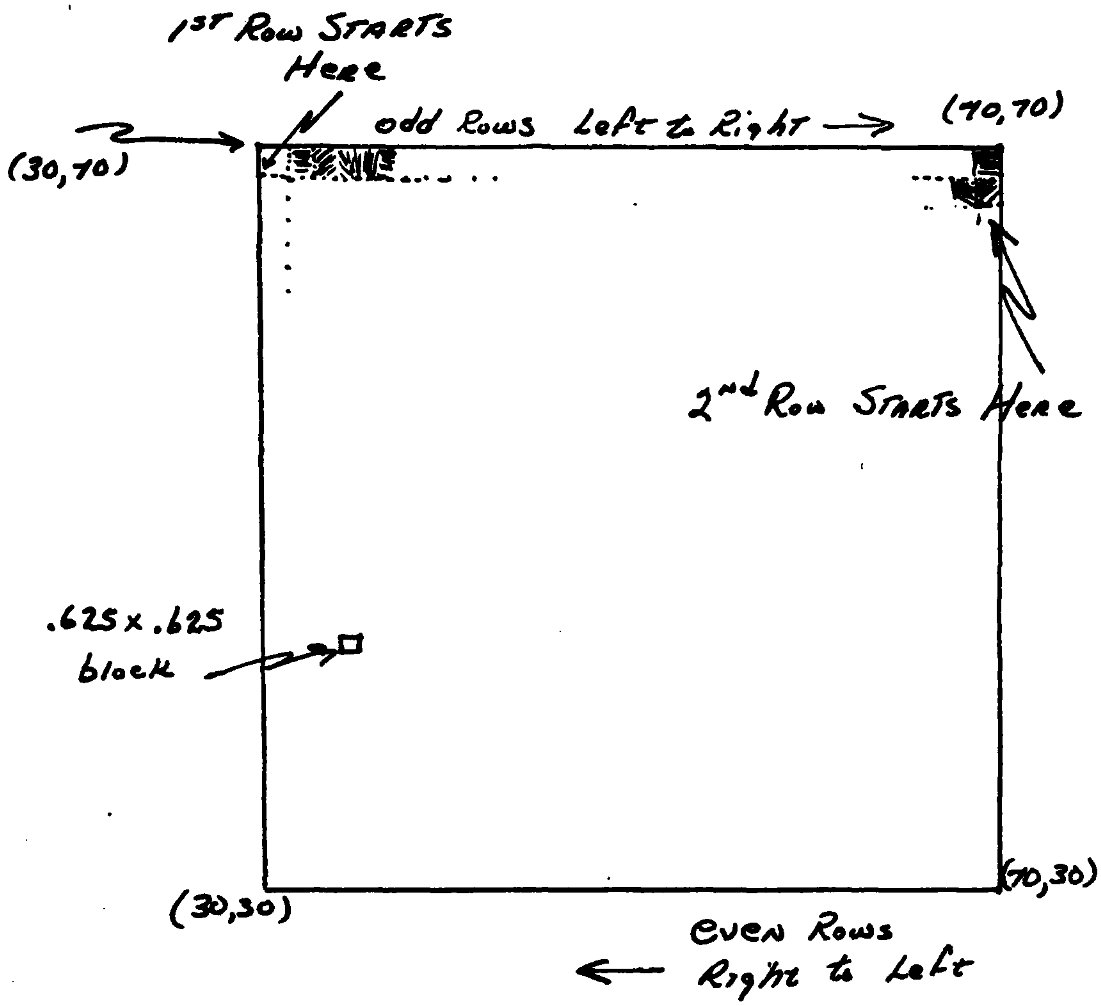
The variable show is used to simply indicate which display has been selected to be plotted.

Show= 1 ==> Amplitude display

Show = 2 ==> Phase display

Show = 3 ==> Intensity display

When display() is completed the image can remain on the screen indefinitely, until an erase() is excuted by completing the action indicated at the top of the screen, which is typing a 'c' for change or a 'q' for quit.




Display Representation

Figure 11

l. draw(p)

The routine draw() is called from two different routines, from showclr() and from display(). The purpose of draw() is to display the current color table on the RAMTEK screen with corresponding color levels indicated. There are sixteen colors in each table and the size of each color block is 3x4 units. The variable Z is a character variable. By adding 060 to the integer variable p, the statement out(Z) can be excuted. Out() displays a single character on the RAMTEK screen at the current operating point. Upon completion of drawing the color table execution is returned to the calling routine.

m. page1()

The purpose of page1() is simply to output text to the RAMTEK screen for introductory comments. When the letter 'c' is typed on the RAMTEK keyboard the routine terminates, and returns to main().

n. page2()

The purpose of page2() is to allow the user to select the display he desires to see. The integer variable show is set to zero and the program will wait until a number is typed on the RAMTEK keyboard, followed by a carriage return using the CR key.

o. page3()

The purpose of page3() is to allow the user to view the avaible color tables and then select which color table he wishes to be used to generate the image. There are nine different color tables: four system reserved color tables, and five program generated. The integer variable n is used to indicate the selected color table.

The viewing of color tables is accomplished by calling the routine showclr(). The user may already know what tables are avialable and the corresponding table number, thereby not wishing to view the nine tables. This option is also available, simply by typing the letter 'n' indicating that no - the user does not wish to see the color tables.

p. showclr()

The purpose of the routine showclr() is to call draw() for displaying the current color table and asking the user if he wishes to see the next color table.

q. change()

The purpose of the routine change() is to determine whether the user wishes to select another desired display , and/or change the color table currently in use. When the user indicates that a change in display is desired the routines, page2() and page3(), are called to obtain necessary information concerning the selected display and color to be displayed. Once this is accomplished, the routine display() is again called to produce the desired display.

The routine change() can only be exited by indicating that the user desires to quit, by typing the letter 'q' when asked to do so. The display can be changed as many times as the user desires to do so.

r. finish()

The purpose of the routine finish() is to verify that the user does in fact wish to quit. The user is given a second and final chance to indicate whether he does really want to stop, or does want to go back and look at another selected display.

3. Program Execution

It is assumed that the user of this program does have at least some introductory experience with the facilities in the Computer Laboratory, and with the PWB/UNIX operating system. In order for the user to sucessfully display the image on the RAMTEK system a number of actions must have been accomplished.

a. Program Loading

The program see3.c must be copied into the user's library so that he may make desired changes, and so that compilation may be accomplished.

As indicated in section III A-2, seven data files must reside in the user's library. Those files are :

tpone

amp

ph

phase

mag

int

mag2

They can be created by using the edit command "ed", then writing one blank line (carriage return), and then quiting the edit mode. For example to create the amp file:

```
%ed amp
0 ?
>a
 (return)
 .
>1w
>q
%
```

would accomplish the task.

b. Program compilation.

Once the user has copied and made any desired edits to his personal copy of the program, the program can be compiled in order to produce an executable file named a.out. The compilation is accomplished by typing the following system command :

```
% ramtek see3.c
```

This action will produce an executable file named a.out. It is highly recommened that the move command

```
mv a.out anotherfilename
```

be executed in order to reduce the possibility of erasing the file a.out by another comlipation prior to completion of its use.

Once the compilation has been completed, typing the filename a.out will cause execution to begin. Typing anotherfilename, if the move (mv) command was executed, will also cause execution to commence.

c. RAMTEK Execution

In order to use the RAMTEK system verification from the Computer Laboratory staff personnel should be obtained as to whether RAMTEK is operational or not. When RAMTEK is operational two power switches must be turned on. One switch is on the CRT housing cabinet; it is a knob located on the front lower left corner marked on - off. The other switch is on the keyboard, a green push type switch on the upper right corner. Both of these switches must be on before the program is executed in order to have meaningfull results.

d. Editing the Program

The user can edit his copy of the display program in the normal PWB/UNIX editing mode. Most of the changes will probably be the addition or change of color tables, or increasing the size of the display.

C. SYSTEM DISPLAYS

The photographs shown in the following pages are pictoral evidence that the routine appears to accomplish the desired task, although much of the visual impact is lost by being limited to black and white photos.

1. Program Development

The display program was developed in sequential steps. A specific routine was written and tested individually and then it was incorporated with the previously completed routines.

2. Display Photographs

The photographs on the following pages show how some improvements to the program affected the display output. The last sets of photographs show the display output of the completed program, as listed in appendix D. This program is re-entrant, allowing the user to select an alternate display and different color tables for use in subsequent display outputs at the current session.

Figure 12 shows one of the first images produced. The object is a circular aperture placed in front of the source transducer. The sound field is scanned immediately behind the object. Previous images (not shown) filled the screen and did not have any margins. In figure 12 the image is rectangular, controlled by the block() instruction, to be 40 wide and 70 high. The display used white as the background color and it used shades of red and blues as the primary colors of the table.

Figures 13 - 16 show the image to be blocked to 40X40. As can be seen in Fig.16, this square image allowed for information to be presented in the borders of the screen.

Figures 17 - 20 are presented to show the "interaction" that the user has with the system. The user must respond to the requests presented on the screen - or the system will remain in an idle state waiting for a response.

Figures 21 - 26 show the three possible kinds of display images, utilizing two different color tables. Figures 21,23 and 25 use the shades of red table, while Figs. 22,24 and 26 use a mixed color table. Figures 21 and 22 are of the amplitude plots; Figs. 23 and 24 are the phase plots; and Figs. 25 and 26 are the intensity plots.

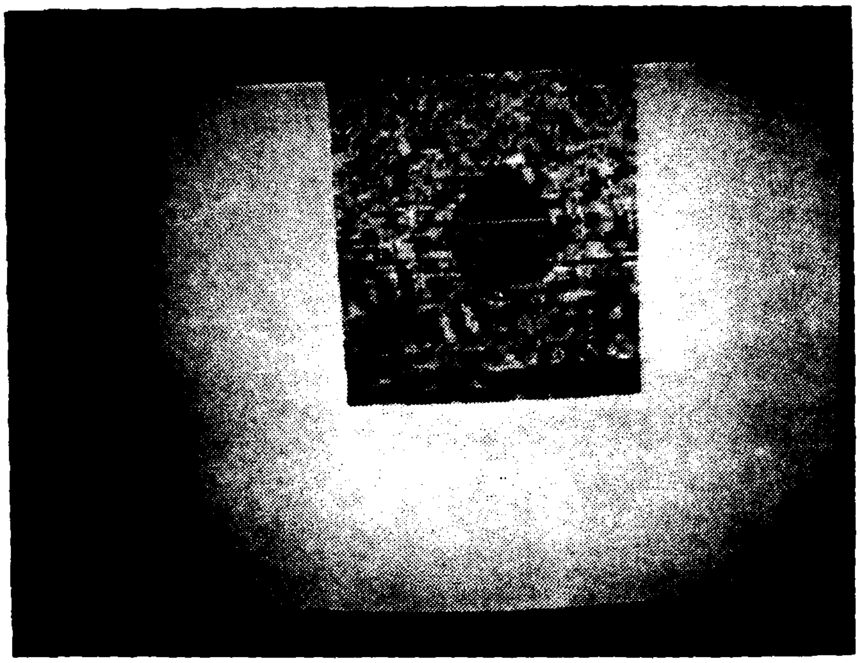


Figure 12. Early Developement Stage

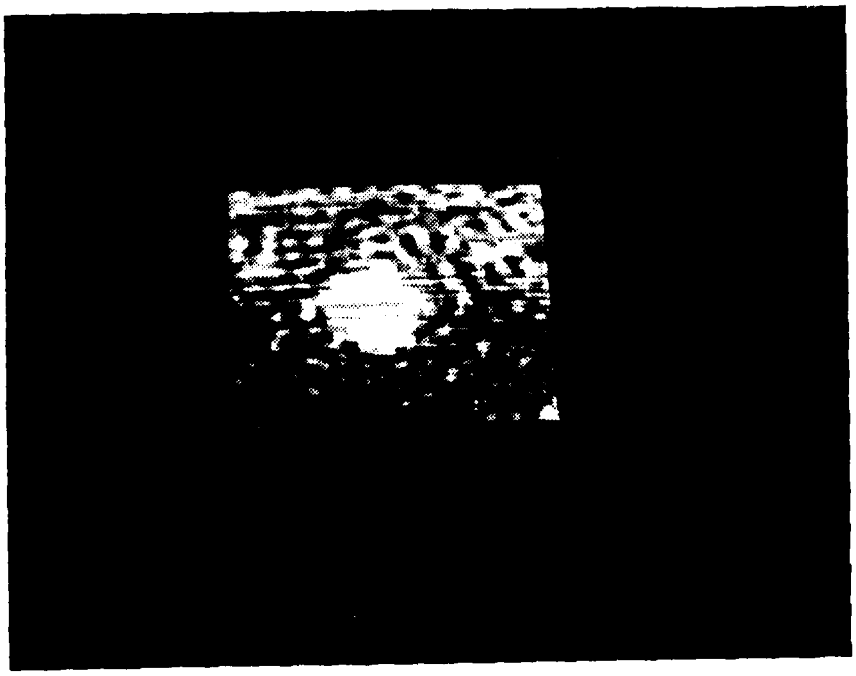


Figure 13.

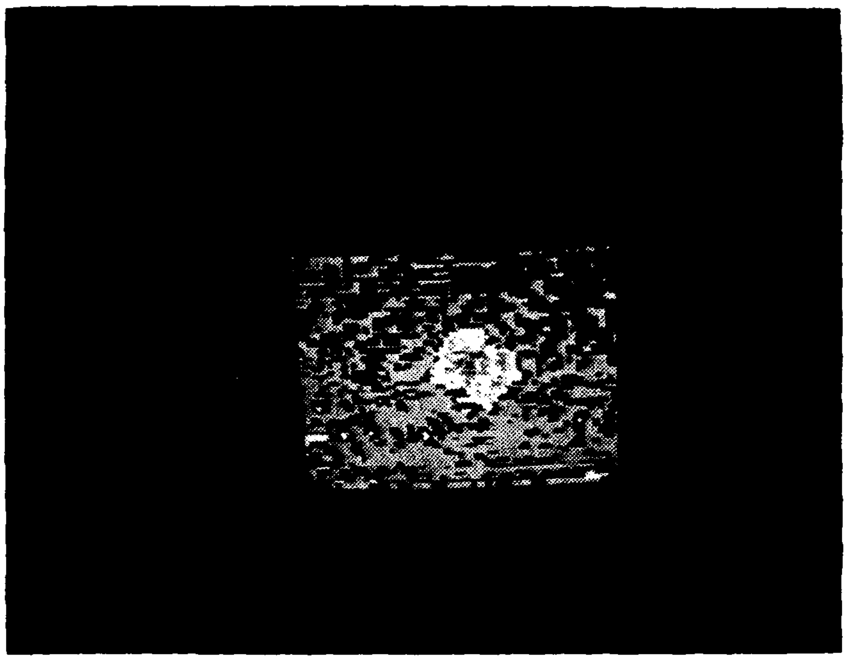


Figure 14.

Figure 15.

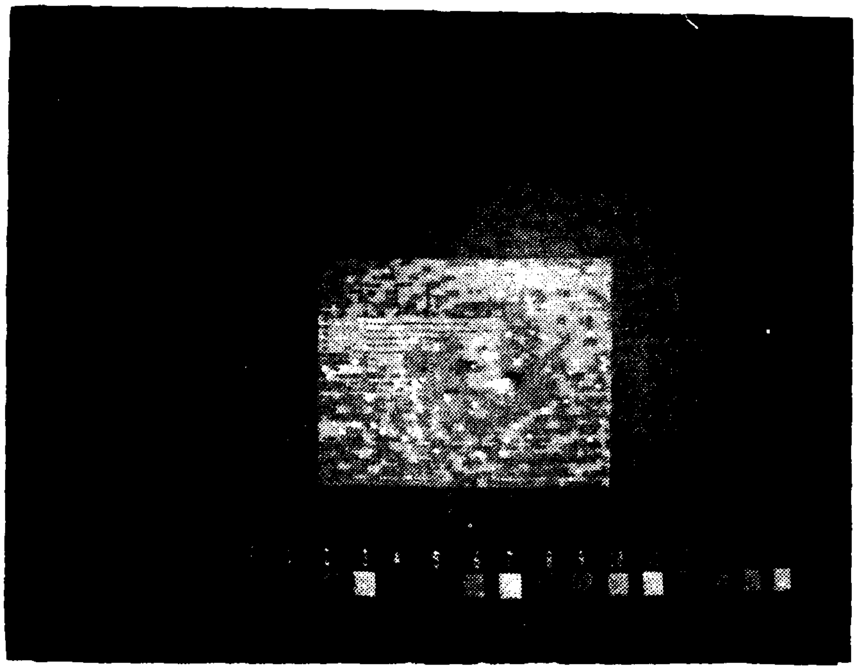


Figure 16.

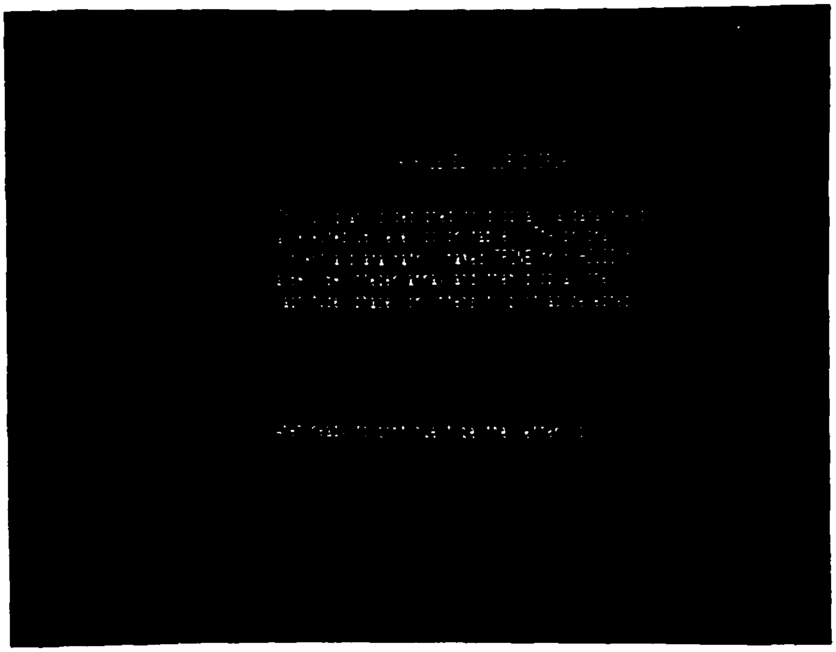


Figure 17.

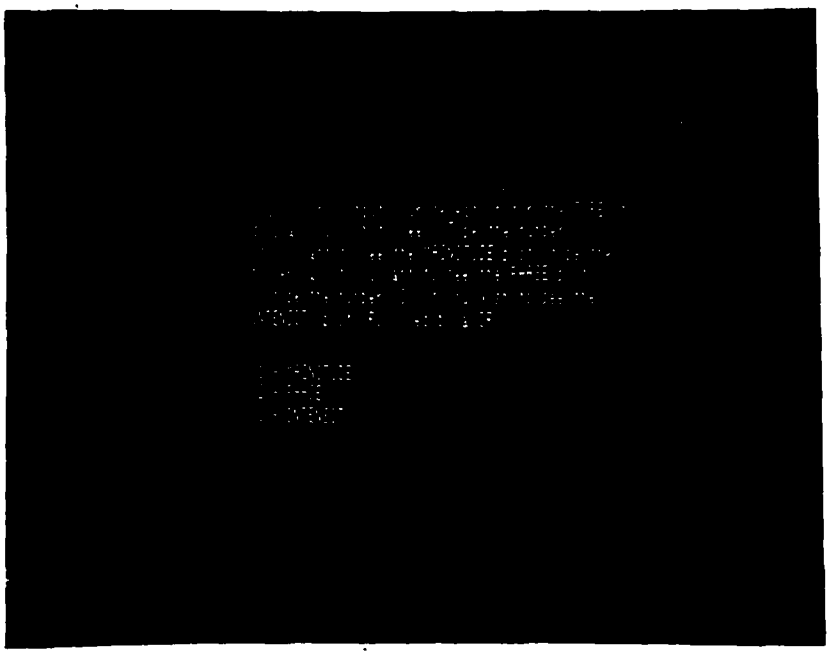


Figure 18.

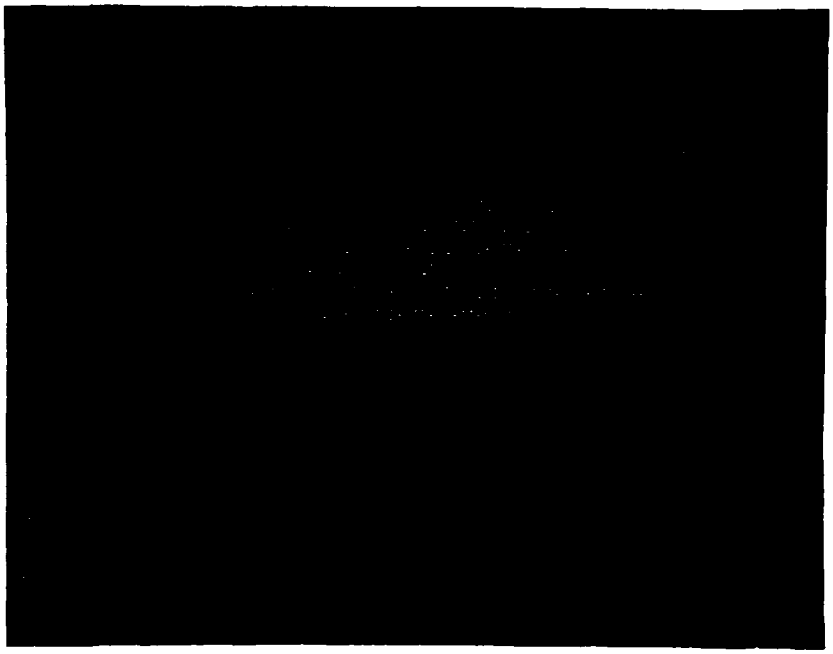


Figure 19.

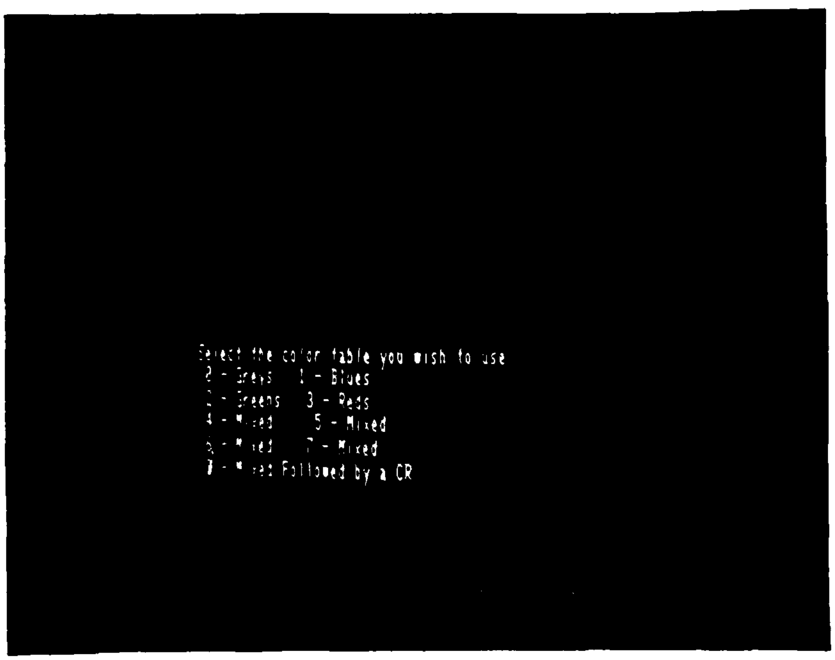



Figure 20.

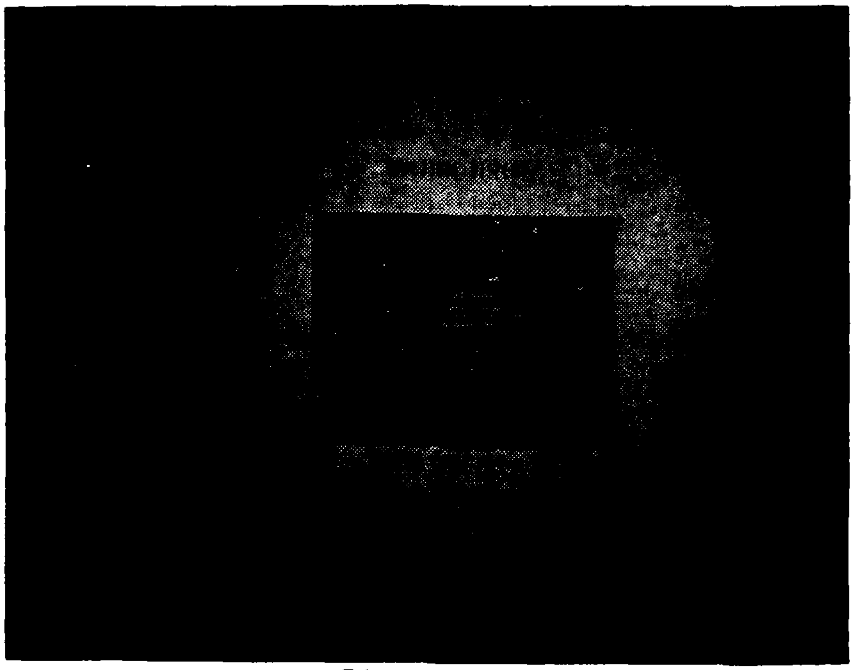


Figure 21.

Amplitude Display - Reds Color Table

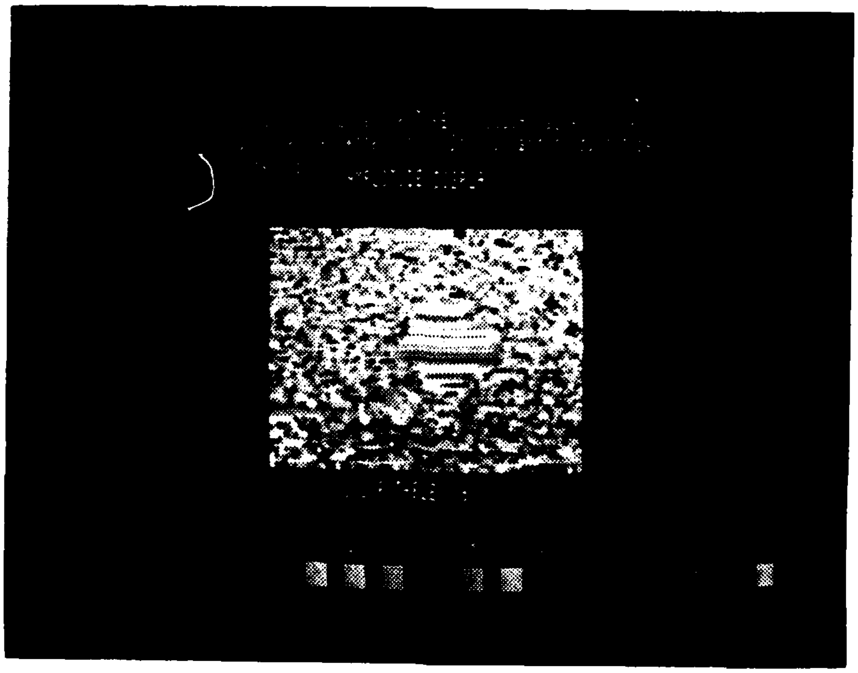


Figure 22.

Amplitude Display - Mixed Colors

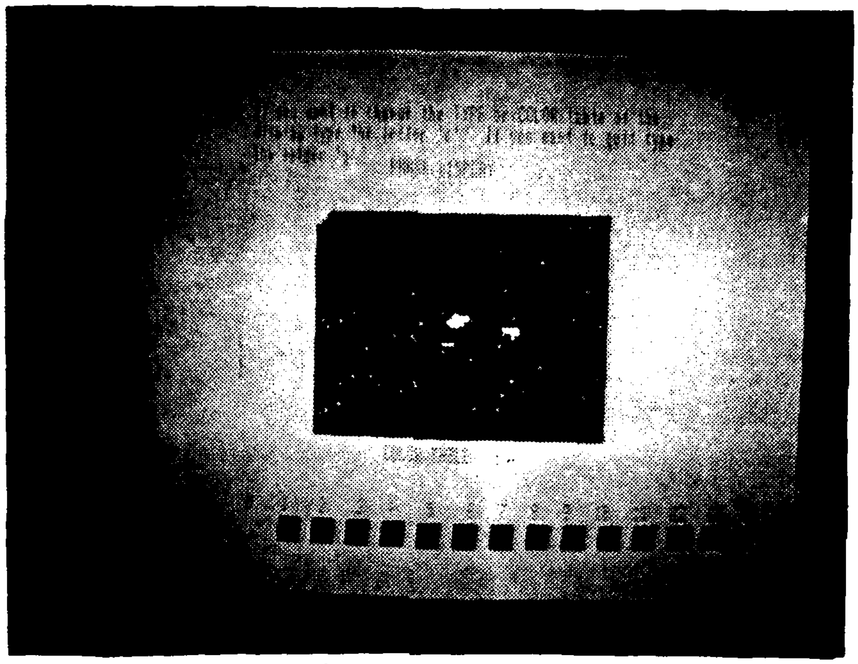


Figure 23.

Phase Disolay - Reds Color Table

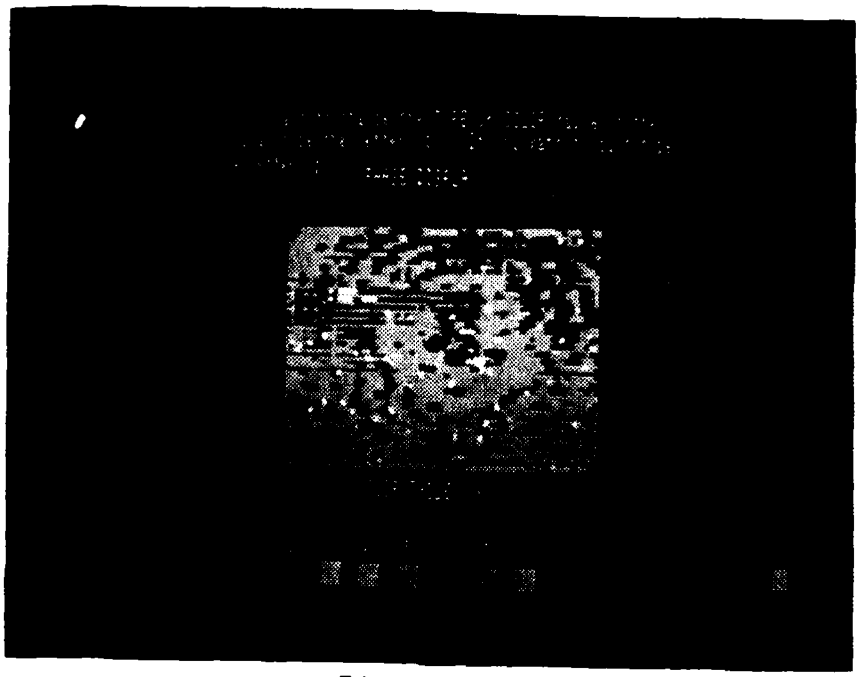


Figure 24.

Phase Display - Mixed Colors

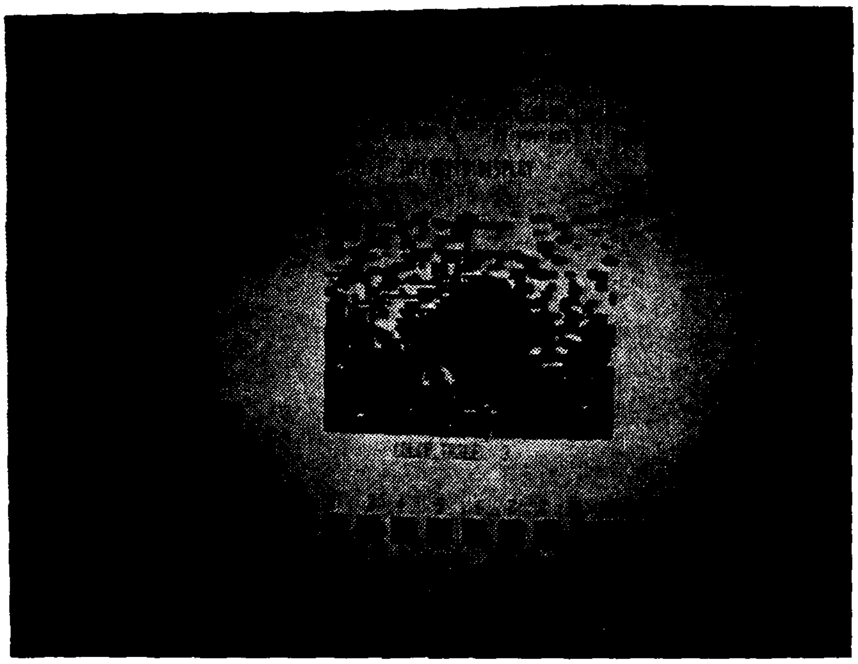


Figure 25.

Intensity Display - Reds Color Table

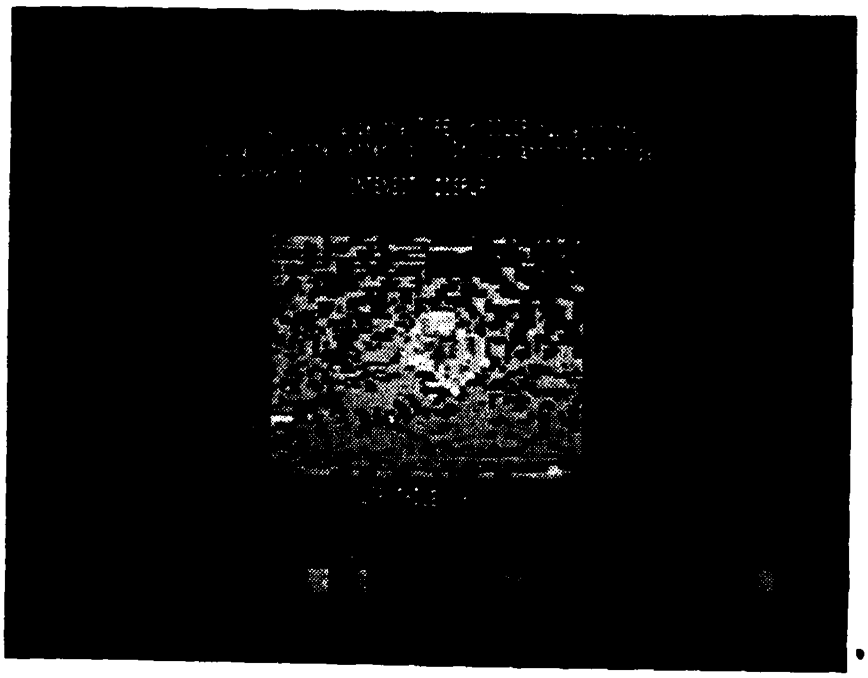


Figure 26.

Intensity Display - Mixed Colors

## CONCLUSIONS AND RECOMMENDATIONS

Although the developed system is not an optimum system in speed or in size, it does allow 16-level color code displays of two dimensional data arrays to be visually displayed. The system described here was designed for part of an ultrasonic imaging system. However this display portion of the overall system could be used to display two dimensional data from any source. The one requirement is that the data be stored in alternating values on the data tape. It must be recognized that software modifications could be almost unlimited, in order to achieve many possible improvements.

Included here are a few recommendations that would definitely improve the efficiency of the current program. By reducing the time required to construct the image the processing capabilities of the system would be increased. First, allow the program itself to create the necessary data files, rather than the user being required to do so. Secondly, allow the user to generate color tables during program execution. This would allow for additional investigations to be conducted at the terminal by increasing the color selection to the maximum possible . Thirdly, it is recommended that furture endeavours in this system utilize the magnetic tape capabilities of the PDP 11/50 computer.

That is the necessary data files and computer program should be kept stored on magnetic tape when not in use. This would reduce memory requirements when the program is not in use and allow for easier transportation of data.

APPENDIX A

RESERVED WORDS

a
ADOFF
adon
ALPHA
axis
b
BKON
BKOFF
bkrnd
blank
BLK
block
bracket
bt1
bt2
bt4
bt5
buff
bytenct
c
cft
change
clrhold
clrtbl
code
einst
epage1
epage2
epage3
erase
ERS
fixpt
flip
fname
fp
FPOFF
fpon
getf
getnum
GRAPHCRT
GRAPHELM
GRAPHVEC
head
headptr
heat
holdx
holdy
index
inptrs
instr
pause
pick
plotpt
plotln
ploth
point
pointr
proc1
proc2
proc3
proc4
proc5
ptrbuff
putup
qptr
qptr1
qt
quest
ql
ramtek
RASTERD
retchar
scissor
SCR
screen

codeit
coke
color
colort
colortbl
comb
COMMA
COMPD
conve
conv1
copy
CR
ct1
ct2
ct3
ct4
cursv
curso
cursh
d
data
datap
dblwid
disp
dump

inst1-inst80
inter
int53
int60
itoa
IXOFF
IXON
LCM
lcmhold
LER
Lex
LE1
LE2
LLR
LLX
LL1
LL2
LTA
LTD
lttr
LXD
moreinst
n0-n17
octbl
out

scroll
SDCO
setmode
setup
size
skip
SSCALL
strout
strtxy
systbl
tblwho
TRANSD
triple
upcnt
vector
wait
WDOFF
WDON
writon
xaxis
xmin
xmax
yaxis
yaxisf
ymin
ymax

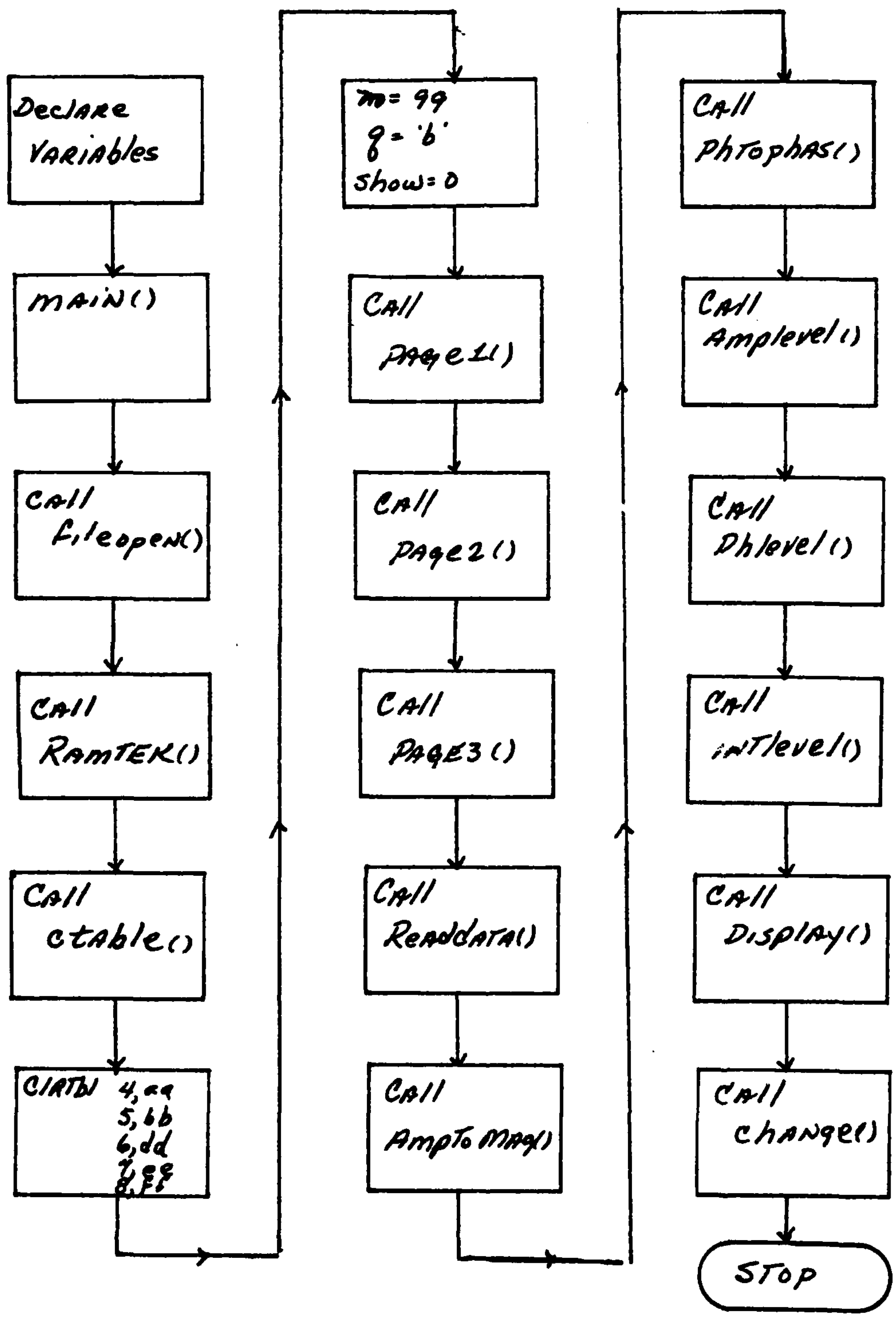
# APPENDIX B

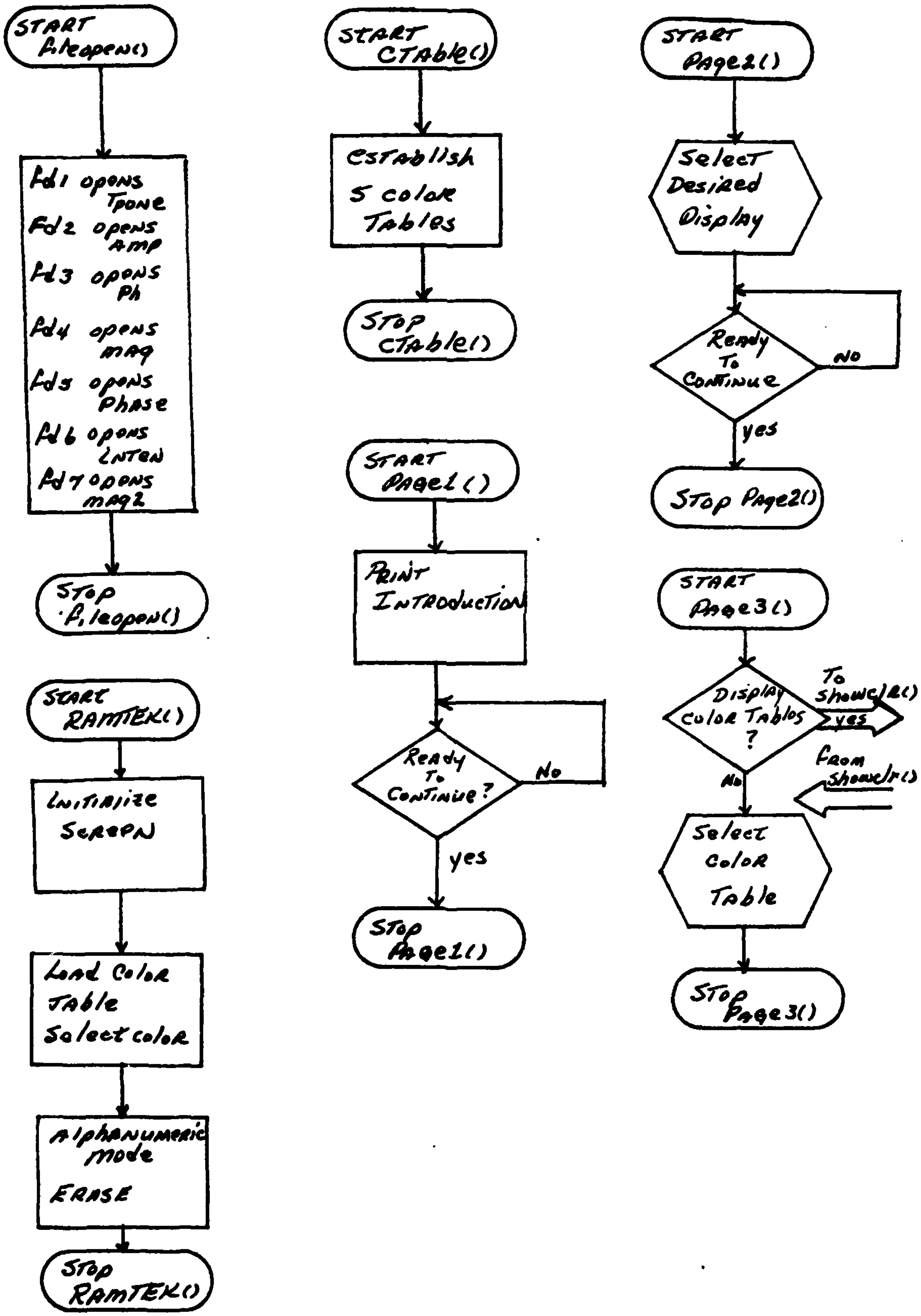
## C Language Reserved Words

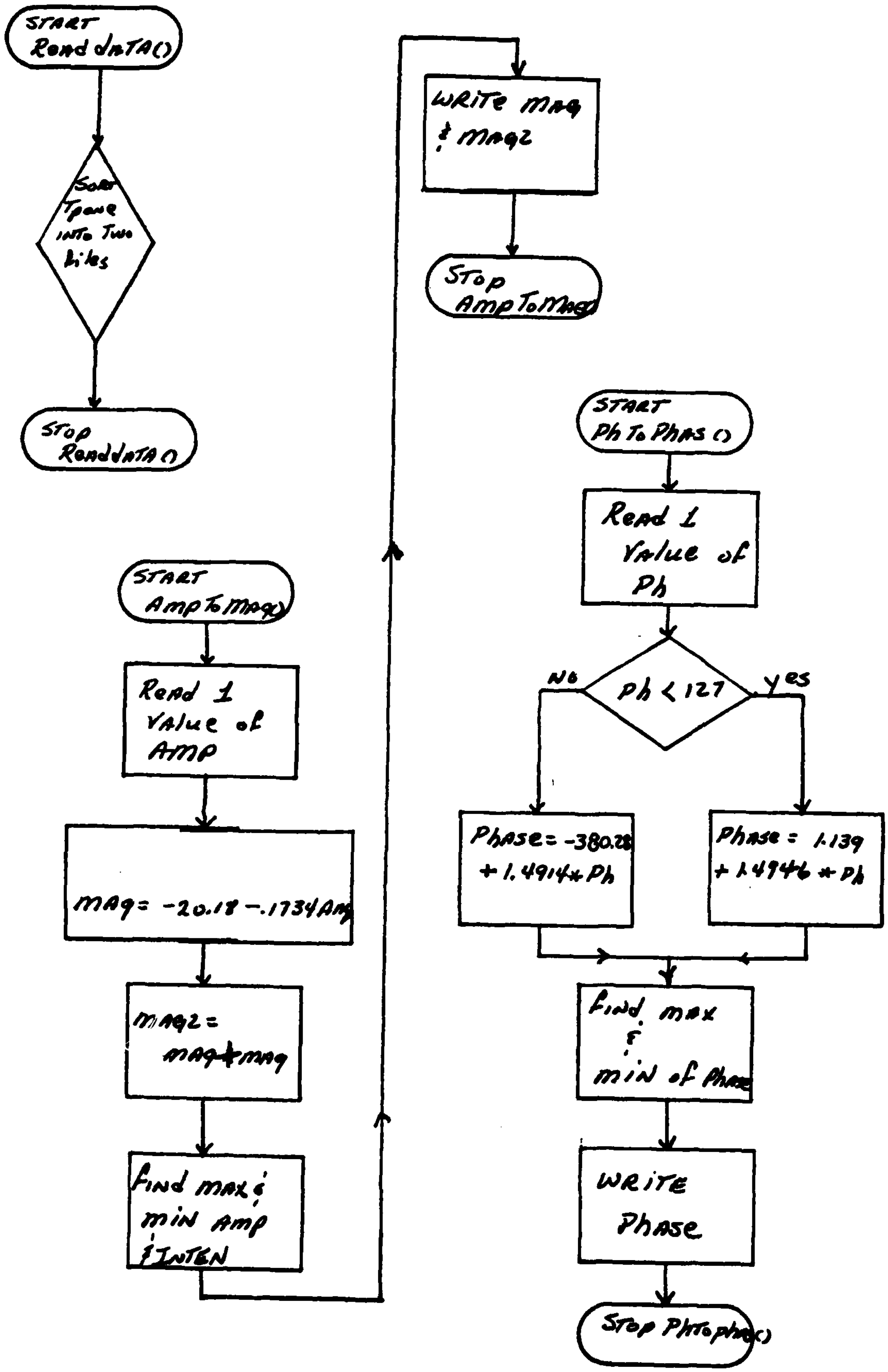
| | | |
|---|---|---|
| int | char | float |
| double | struct | auto |
| extern | register | static |
| goto | return | sizeof |
| break | continue | if |
| else | for | do |
| while | switch | case |
| default | entry | |

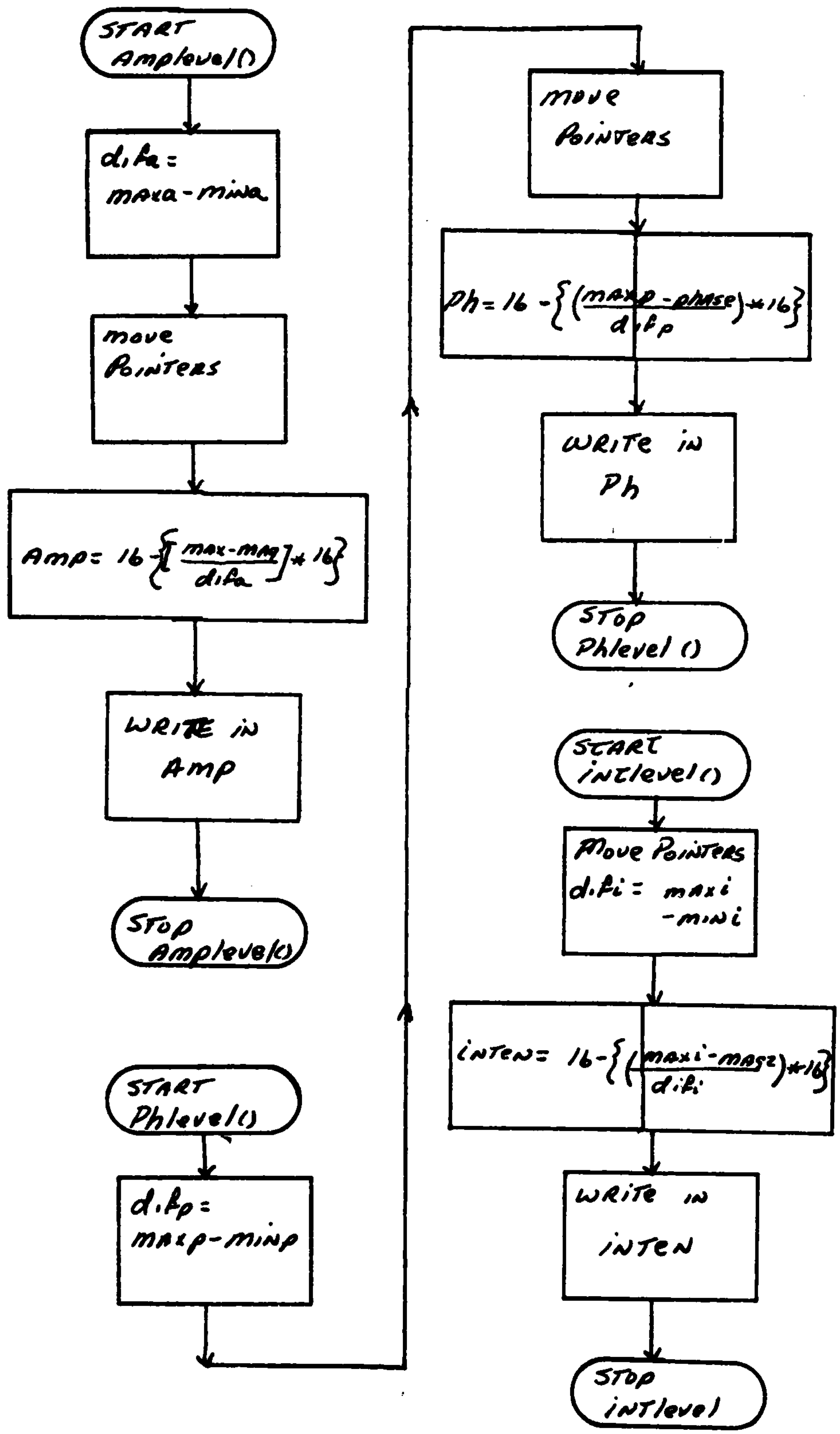
APPENDIX C

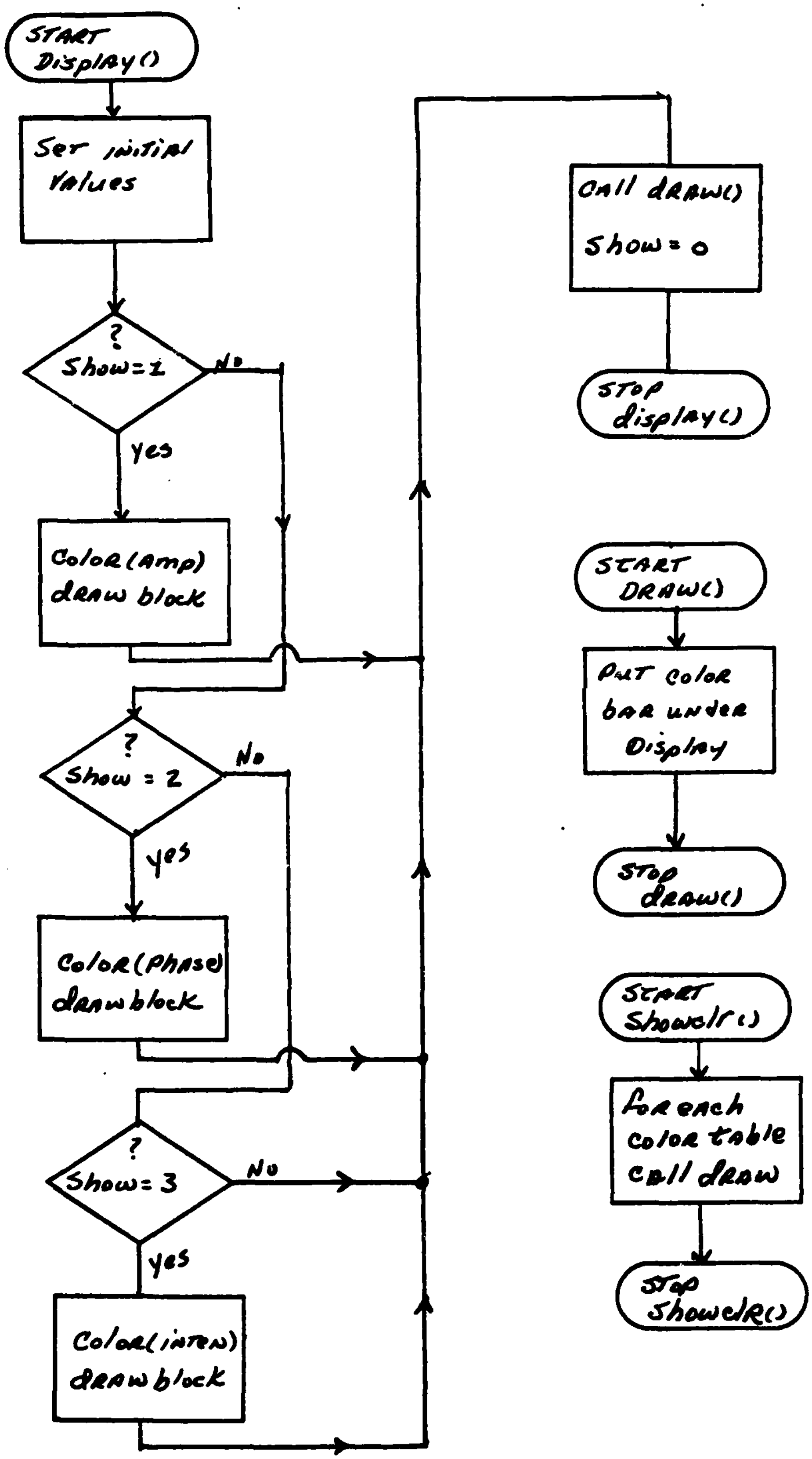
Detailed Flow Chart

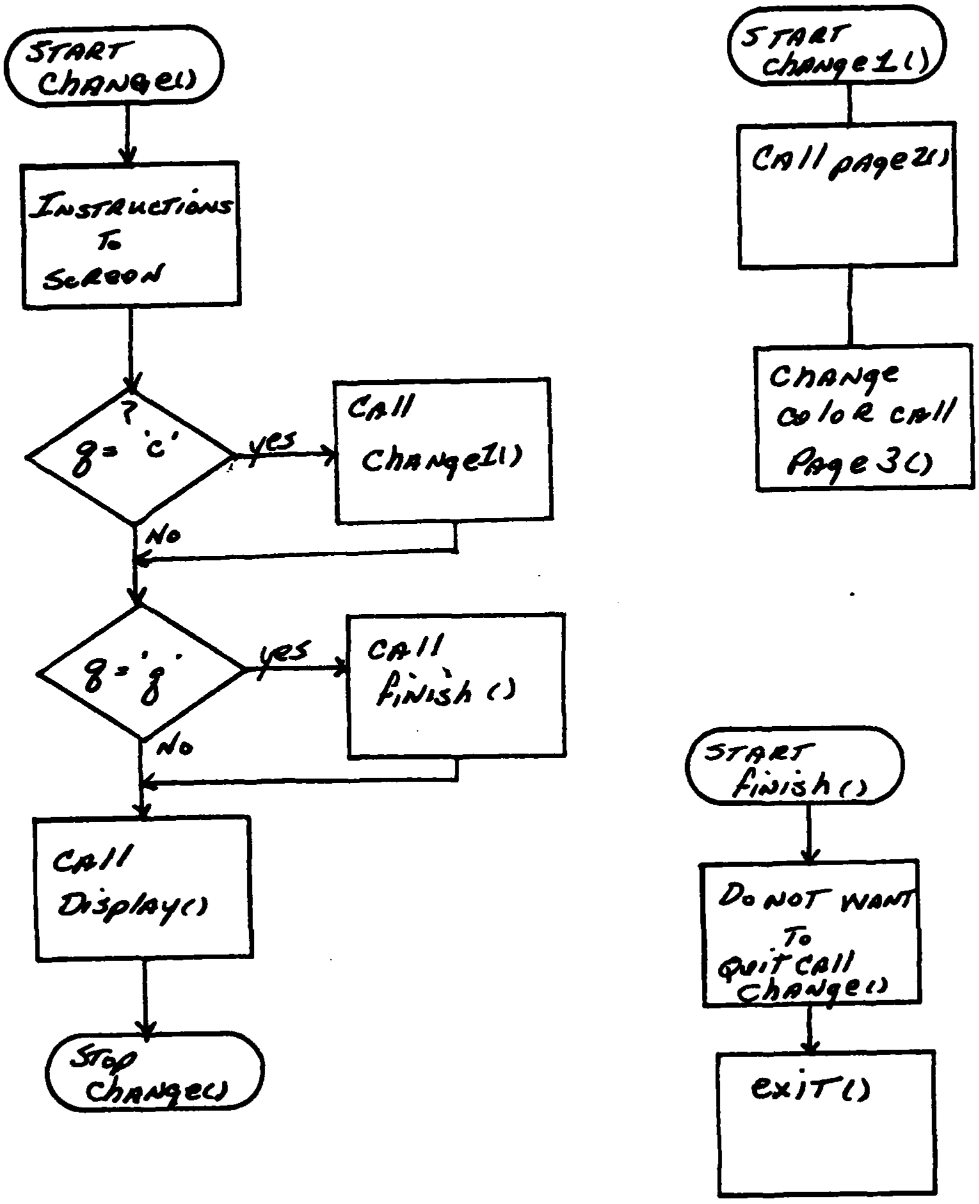



START fileopen()
fd1 opens Tpone
fd2 opens Amp
fd3 opens Ph
fd4 opens mag
fd5 opens Phase
fd6 opens Inten
fd7 opens mag2
Stop fileopen()
START RAMTEK()
Initialize Screen
Load Color Table
Select color
Alphanumeric mode
ERASE
Stop RAMTEK()
START CTable()
establish 5 color Tables
Stop CTable()
START Page1()
Print Introduction
Ready To Continue?
No
yes
Stop Page1()
START Page2()
Select Desired Display
Ready To Continue
No
yes
Stop Page2()
START Page3()
Display Color Tables ?
To Showclr()
yes
No
From Showclr()
Select Color Table
Stop Page3()


START
Read dATA()
Sort Tpane into Two files
Stop ReaddATA()
START
AmpToMag()
Read 1 Value of AMP
mag= -20.18 -.1734Amp
mag2= mag*mag
Find max & min Amp & INTEN
WRITE mag & mag2
Stop AmpToMag
START
PhToPhas()
Read 1 Value of Ph
Ph < 127
No
Yes
Phase= -380.28 + 1.4914*Ph
Phase = 1.139 + 1.4946 * Ph
Find max & min of Phase
WRITE Phase
Stop PhToPhas()


START
Amplevel()
difa =
maxa - mina
move
Pointers
Amp= 16 - {[max-mag / difa] * 16}
WRITE in
Amp
STOP
Amplevel()
START
Phlevel()
difp =
maxp - minp
move
Pointers
Ph = 16 - {(maxp - phase / difp) * 16}
WRITE in
Ph
STOP
Phlevel ()
START
intlevel()
Move Pointers
difi = maxi
- mini
inten = 16 - {(maxi - magz / difi) * 16}
WRITE in
inten
STOP
intlevel


START Display()
Set initial values
? Show=1
No
yes
Color(amp) draw block
? Show = 2
No
yes
Color(Phase) draw block
? Show= 3
No
yes
Color(inten) draw block
Call draw()
Show = 0
STOP display()
START DRAW()
Put color bar under Display
STOP draw()
START Showclr()
for each color table call draw
STOP ShowclR()


START change()
INSTRUCTIONS To SCREEN
q = 'c'
yes
Call Change1()
No
q = 'q'
yes
Call finish()
No
Call Display()
Stop change()
START change1()
Call page2()
Change color call Page3()
START finish()
Do not want To quit call change()
exit()

```
/*   Program for displaying data on the RAMTEK display unit */

/***** DECLARATIONS *****/

int amp, ph, inten, i, j, k, m, n, show, flag  ;
int fd1, fd2, fd3, fd4, fd5, fd6, fd7 ;
int aa[16], bb[16], dd[16], ee[16], ff[16] ;
char cc[4], *pc, q, z ;
float mag, phase, x, y ;
float maxa, mina, maxp, minp, difa, difp ;
float maxi, mini, difi, mag2 ;

/***** MAIN PROGRAM ( calls most of the subroutines ) *****/

main ()
{
     fileopen () ;
     ramtek () ;
     erase () ;
     ctable () ;
     page1 () ;
     page2 () ;
     page3 () ;
/*
     readdata () ;
*/
/*
     amptomag () ;
*/
/*
     phtophas () ;
*/
/*
     amplevel () ;
*/
/*
     phlevel () ;
*/
/*
     intlevel () ;
*/
     display () ;
}

/***** OPENS ALL NECESSARY FILES *****/

fileopen ()
{
     fd1 = open("tpone",0) ;      /* OPEN INPUT FILE FOR READ ONLY */
          if ( fd1 < 0 ) {
               printf(" CANNOT OPEN DATA FILE \n" ) ;
               printf(" PLEASE INSURE THAT TPONE IS IN LIBRARY \n" ) ;
               exit () ;
          }

     fd2 = open("amp",2) ;        /* OPEN AMP FILE FOR READ/WRITE */
          if ( fd2 < 0 ) {
               printf(" CANNOT OPEN AMP FILE \n" ) ;
               printf(" PLEASE INSURE THAT AMP IS IN LIBRARY \n" ) ;
               exit () ;
```

```
        }

    fd3 = open("ph",2) ;          /* OPEN PH FILE FOR READ/WRITE */
        if ( fd3 < 0 ) {
            printf(" CANNOT OPEN PH FILE \n" ) ;
            printf(" PLEASE INSURE THAT PH IS IN LIBRARY \n" ) ;
            exit () ;
        }

    fd4 = open("mag",2) ;         /* OPEN MAG FILE FOR READ/WRITE */
        if ( fd4 < 0 ) {
            printf(" CANNOT OPEN MAG FILE \n" ) ;
            printf(" PLEASE INSURE THAT MAG IS IN LIBRARY \n") ;
            exit () ;
        }

    fd5 = open("phase",2) ;       /* OPEN PHASE FILE FOR READ/WRITE
        if ( fd5 < 0 ) {
            printf(" CANNOT OPEN PHASE FILE \n" ) ;
            printf(" PLEASE INSURE THAT PHASE IS IN LIBRARY \n" ) ;
            exit () ;
        }

    fd6 = open("inten",2) ;       /* OPEN INTENSITY FILE FOR READ/WR
        if ( fd6 < 0 ) {
            printf(" CANNOT OPEN INTENSITY FILE \n" ) ;
            printf(" PLEASE SEE THAT INTENSITY IS IN LIBRARY \n" )
            exit () ;
        }

    fd7 = open("mag2",2) ;        /* OPEN MAG SQUARED FILE */
        if ( fd7 < 0 ) {
            printf(" CANNOT OPEN MAG SQUARED FILE \n" ) ;
            printf(" PLEASE CHECK LIBRARY FILES \n" ) ;
            exit () ;
        }

    return ;
    }

/***** READ DATA FROM TPONE INTO EITHER AMP OR PH FILE *****/

readdata ()
{
    for ( i = 0; i < 4096; i++ ) {
        for ( k = 0; k < 2; k++ ) {
            pc = cc ;
            do {
                read(fd1,pc,1) ;
                if ( *pc == '\n' )  pc = &cc[-1] ;
            }
            while ( *pc++ != ' ' ) ;
                m = atoi (cc) ;
                if ( k == 0 )
                    write(fd2,&m,2) ;
                else
                    write(fd3,&m,2) ;
        }
    }
    return ;
```

```
}

/***** CONVERT INTEGER AMP VALUES INTO dB FLOATING POINT MAG VALUE

amptomag ()
{
    seek(fd2,0,0) ;             /* POINT TO BEGINNING OF AMP FILE */
    seek(fd4,0,0) ;             /* POINT TO BEGINNING OF MAG FILE :*/
    seek(fd7,0,0) ;             /* POINT TO BEGINNING OF MAG2 FILE */
    maxa = -999.0 ;
    mina =  999.0 ;
    maxi =  000.0 ;
    mini =  999.0 ;
    for ( i = 0; i < 4096; i++ ) {
        read(fd2,&amp,2) ;  /* READ 2 BYTES INTO AMP */
        mag = -20.18 - .1734 * amp ;
        mag2 = mag + mag ;
        if ( maxa < mag ) maxa = mag ;
        if ( mina > mag ) mina = mag ;
        if ( maxi < mag2 ) maxi = mag2 ;
        if ( mini > mag2 ) mini = mag2 ;
        write(fd4,&mag,4) ;
        write(fd7,&mag2,4) ;
    }

    return ;
}

/***** CONVERT INTEGER PH VALUES TO dB FLOATING POINT PHASE VALUES

phtophas ()
{
    seek(fd3,0,0) ;             /* POINT TO BEGINNING OF PH FILE */
    seek(fd5,0,0) ;             /* POINT TO BEGINNING OF PHASE FILE */
    maxp = -999.0 ;
    minp =  999.0 ;
    for ( i = 0; i < 4096; i++ ) {
        read(fd3,&ph,2) ;   /* READ 2 BYTES OF PH */
        if ( ph < 127 )
            phase = 1.139 + 1.4946 * ph ;
        else
            phase = -380.28 + 1.4917 * ph ;
        write(fd5,&phase,4) ;   /* WRITE 4 BYTES INTO PHASE */
        if ( maxp < phase ) maxp = phase ;
        if ( minp > phase ) minp = phase ;
    }
    return ;
}

/***** CONVERT THE FLOATING POINT VALUES INTO INTEGER VALUES OF
       BETWEEN 0 AND 15 ( for color table use )            *****

amplevel ()
{
    difa = maxa - mina ;
    seek(fd2,0,0) ;
    seek(fd4,0,0) ;
    for ( i = 0; i < 4096; i++ ) {
        read(fd4,&mag,4) ;
        amp = 16 - ((( maxa - mag ) / difa ) * 16 ) ;
```

```
        write(fd2,&amp,2) ;
    }
    return ;
}

phlevel ()
{
    difp = maxp - minp ;
    seek(fd3,0,0) ;
    seek(fd5,0,0) ;
    for ( i = 0; i < 4096; i++ ) {
        read(fd5,&phase,4) ;
        ph = 16 - ((( maxp - phase ) / difp ) * 16. ) ;
        write(fd3,&ph,2) ;
    }
    return ;
}

intlevel ()
{
    seek(fd6,0,0) ;
    seek(fd7,0,0) ;
    difi = maxi - mini ;
    for ( i = 0; i < 4096; i++ ) {
        read(fd7,&mag2,4) ;
        inten = 16 - ((( maxi - mag2 ) / difi ) * 16 ) ;
        write(fd6,&inten,2) ;
    }
    return ;
}

/***** DECLERATION OF VARIOUS POSSIBLE COLOR TABLES *****/

ctable ()
{

    aa[0]=triple(15,15,15);
    aa[1]=triple(15,15,10);
    aa[2]=triple(15,15,05);
    aa[3]=triple(15,15,00);
    aa[4]=triple(15,10,15);
    aa[5]=triple(15,05,15);
    aa[6]=triple(15,00,15);
    aa[7]=triple(15,10,10);
    aa[8]=triple(15,10,05);
    aa[9]=triple(15,10,00);
    aa[10]=triple(15,05,10);
    aa[11]=triple(15,05,05);
    aa[12]=triple(15,05,00);
    aa[13]=triple(10,10,10);
    aa[14]=triple(10,10,05);
    aa[15]=triple(05,05,05);

    bb[0]=triple(00,00,00);     /* black */
    bb[1]=triple(01,01,01);     /*       */
    bb[2]=triple(02,02,02);     /*        */
    bb[3]=triple(03,03,03);     /*     */
    bb[4]=triple(04,04,04);     /*       */
    bb[5]=triple(05,05,05);     /*       */
    bb[6]=triple(06,06,06);     /*        */
```

```
	bb[7]=triple(07,07,07);		/*		*/
	bb[8]=triple(08,08,08);		/*		*/
	bb[9]=triple(09,09,09);		/*		*/
	bb[10]=triple(10,10,10);	/*		*/
	bb[11]=triple(11,11,11);	/*		  */
	bb[12]=triple(12,12,12);	/*			*/
	bb[13]=triple(13,13,13);	/*			*/
	bb[14]=triple(14,14,14);	/*			*/
	bb[15]=triple(15,15,15);	/* white */

	dd[0]=triple(00,00,00);		/* black */
	dd[1]=triple(15,00,08);		/* violet */
	dd[2]=triple(04,00,12);		/* purple */
	dd[3]=triple(15,00,15);		/* magenta */
	dd[4]=triple(12,04,15);		/* puce */
	dd[5]=triple(08,08,15);		/* pink */
	dd[6]=triple(00,00,15);		/* red */
	dd[7]=triple(00,12,15);		/* orange */
	dd[8]=triple(00,15,15);		/* yellow */
	dd[9]=triple(00,15,08);		/* yellow - green */
	dd[10]=triple(00,15,00);	/* green */
	dd[11]=triple(15,08,00);	/* green - blue */
	dd[12]=triple(08,15,00);	/* blue - green */
	dd[13]=triple(15,15,00);	/* cyan */
	dd[14]=triple(15,00,00);	/* blue */
	dd[15]=triple(15,15,15);	/* white */

	ee[0]=triple(00,00,00);		/* black */
	ee[1]=triple(00,00,05);		/*  */
	ee[2]=triple(00,00,10);		/*  */
	ee[3]=triple(00,00,15);		/* red */
	ee[4]=triple(00,05,00);		/*  */
	ee[5]=triple(00,05,05);		/*  */
	ee[6]=triple(00,05,10);		/*  */
	ee[7]=triple(00,05,15);		/*  */
	ee[8]=triple(00,10,00);		/*  */
	ee[9]=triple(00,10,05);		/*  */
	ee[10]=triple(00,10,10);	/*  */
	ee[11]=triple(00,10,15);	/*  */
	ee[12]=triple(00,15,00);	/* green */
	ee[13]=triple(00,15,05);	/*  */
	ee[14]=triple(00,15,10);	/*  */
	ee[15]=triple(00,15,15);	/* yellow */

	ff[0]=triple(15,00,00);		/* blue */
	ff[1]=triple(15,00,05);		/*  */
	ff[2]=triple(15,00,10);		/*  */
	ff[3]=triple(15,00,15);		/* magenta */
	ff[4]=triple(15,05,00);		/*  */
	ff[5]=triple(15,05,05);		/*  */
	ff[6]=triple(15,05,10);		/*  */
	ff[7]=triple(15,05,15);		/*  */
	ff[8]=triple(15,10,00);		/*  */
	ff[9]=triple(15,10,05);		/*  */
	ff[10]=triple(15,10,10);	/*  */
	ff[11]=triple(15,10,15);	/*  */
	ff[12]=triple(15,15,00);	/* cyan */
	ff[13]=triple(15,15,05);	/*  */
	ff[14]=triple(15,15,10);	/*  */
	ff[15]=triple(15,15,15);	/* white */
```

```

	clrtbl (4,aa) ;
	clrtbl (5,bb) ;
	clrtbl (6,dd) ;
	clrtbl (7,ee) ;
	clrtbl (8,ff) ;

	return ;

}

/***** ROUTINE FOR DISPLAY PORTION OF PROGRAM *****/

display ()
{
	flag = 0 ;
	erase () ;
	screen (0.0, 0.0, 100.0, 110.0 ) ;
	colort(n) ;
	color(15) ;
	x = 30.0 ;  y = 69.375 ;

/***** AMPLITUDE DISPLAY ROUTINE *****/

	if ( show == 1 ) {
	strtxy(40.,80.) ;
	strout ("AMPLITUDE DISPLAY") ;
	strtxy(x,y) ;
	seek(fd2,0,0) ;

	for ( i = 0; i < 64; i++ ) {
		y = 69.375 - ( i * .625 ) ;
		for ( j = 0; j < 64; j++ ) {
			x = 30. + ( .625 * j ) ;
			read(fd2,&amp,2) ;
			color(amp) ;
			block( x, y, x+.625, y+.625 ) ;
		}
		i++ ;
		y = y - .625 ;
		for ( j = 0; j < 64; j++ ) {
			x = 69.375 - ( .625 * j ) ;
			read(fd2,&amp,2) ;
			color(amp) ;
			block( x, y, x+.625, y+.625 ) ;
		}
	}
	draw (n) ;
	show = 0 ;
	change () ;
	return ;
	}

/***** ROUTINE FOR PHASE DISPLAY PORTION OF PROGRAM *****/

	else
	if ( show == 2 ) {
	strtxy(40.,80.) ;
	strout("PHASE DISPLAY") ;
	strtxy(x,y) ;
```

```
     seek(fd3,0,0) ;
     for ( i = 0; i < 64; i++ ) {
          y = 69.375 - ( i * .625 ) ;
          for ( j = 0; j < 64; j++ ) {
               x = 30. + ( .625 * j ) ;
               read(fd3,&ph,2) ;
               color(ph) ;
               block( x, y, x+.625, y+.625 ) ;
          }
          i++ ;
          y = y - .625 ;
          for ( j = 0; j < 64; j++ ) {
               x = 69.375 - ( .625 * j ) ;
               read(fd3,&ph,2) ;
               color(ph) ;
               block( x, y, x+.625, y+.625 ) ;
          }
     }
     draw (n) ;
     show = 0 ;
     change () ;
     return ;
     }

/***** ROUTINE FOR INTENSITY DISPLAY *****/

     else
     if ( show == 3 ) {
     strtxy(40.,80.) ;
     strout ("INTENSITY DISPLAY") ;
     strtxy(x,y) ;
     seek(fd6,0,0) ;
     for ( i = 0; i < 64; i++ ) {
          y = 69.375 - ( i * .625 ) ;
          for ( j = 0; j < 64; j++ ) {
               x = 30. + ( .625 * j ) ;
               read(fd6,&inten,2) ;
               color(inten) ;
               block( x, y, x+.625, y+.625 ) ;
          }
          i++ ;
          y = y - .625 ;
          for ( j = 0; j < 64; j++ ) {
               x = 69.375 - ( .625 * j ) ;
               read(fd6,&inten,2) ;
               color(inten) ;
               block( x, y, x+.625, y+.625 ) ;
          }
     }
     draw (n) ;
     show = 0 ;
     change () ;
     return ;
     }
     return ;
}

/***** PROGRAM TO DRAW AND LABLE COLOR LEVELS *****/

draw (p)
```

```
int p ;
{
    if ( flag == 1 ) erase () ;
    colort(p) ;
    color(15) ;
    strtxy(9.,18.) ;
    strout("LEVEL      0     1     2     3    4     5     6    7    8    9
    strtxy(20.,10.) ;
    y = 10. ;
    for ( j = 0; j < 16; j++ ) {
        x = 20. + ( 5. * j ) ;
        color(j) ;
        block(x,y,x+3.,y+4.) ;
        strtxy (40.,28.) ;
        strout ("COLOR TABLE ");
        z = p + 060 ;
        strtxy (55.,28.) ;
        out (z) ;
    }
    return ;
}

/***** INSTRUCTIONS AND INTRODUCTION *****/

page1 ()
{
    strtxy (40.,90.) ;
    strout ("16 - LEVEL COLOR DISPLAY") ;
    strtxy (23.,80.) ;
    strout ("This program is designed to display a data field with
    strtxy (23.,76.) ;
    strout ("a selected 16 level color table.  The program will ")
    strtxy (23.,72.) ;
    strout ("convert a data matrix named TPONE from ASCII to") ;
    strtxy (23.,68.) ;
    strout ("a 64 x 64 integer array and then display the") ;
    strtxy (23.,64.) ;
    strout ("magnitude, phase, or intensity plot as selected") ;
    strtxy (23.,40.) ;
    strout ("When ready to continue type the letter 'c' ") ;
    while ( q != 'c' )
        q = getchar () ;
    q = 'b' ;
    erase () ;
    return ;
}

page2 ()
{
    show = 0 ;
    q = 'b' ;
    strtxy (20.,80.) ;
    strout ("You will now input your selection for the TYPE of");
    strtxy (20.,76.) ;
    strout ("display you wish to see.  Type the number '1' ") ;
    strtxy (20.,72.) ;
    strout ("if you want to see the MAGNITUDE plot, type the") ;
    strtxy (20.,68.) ;
    strout ("number '2' if you want to see the PHASE plot") ;
    strtxy (20.,64.) ;
```

AD-A083 854 NAVAL POSTGRADUATE SCHOOL MONTEREY CA F/G 14/5
A 16 - LEVEL COLOR DISPLAY SYSTEM FOR TWO DIMENSIONAL ULTRASONI--ETC(U)
DEC 79 E L MOON

UNCLASSIFIED NL

2 OF 2
AD A
083854

END
DATE
FILMED
6-80
DTIC

```
     strout ("or type the number '3' if you wish to see the") ;
     strtxy (20.,60.) ;
     strout ("INTENSITY plot. Followed by a CR. ") ;
     strtxy (20.,50.) ;
     strout (" 1 - MAGNITUDE ") ;
     strtxy (20.,46.) ;
     strout (" 2 - PHASE ") ;
     strtxy (20.,42.) ;
     strout (" 3 - INTENSITY ") ;
     while ( show == 0 )
         show = getnum (10) ;
     strtxy (20.,30.) ;
     strout ("When ready to continue type the letter 'c' ") ;
     while ( a != 'c' )
         a = getchar () ;
     a = 'b' ;
     erase () ;
     return ;
}

page3 ()
{
     a = 'b' ;
     n = 99 ;
     strtxy (20.,80.) ;
     strout ("If you desire to see the color tables that") ;
     strtxy (20.,76.) ;
     strout ("are loaded in this program that you may choose") ;
     strtxy (20.,72.) ;
     strout ("from for displaying the data type the letter 'y'") ;
     strtxy (20.,68.) ;
     strout("  If you already know what color") ;
     strtxy (20.,64.) ;
     strout ("table you wish to use  or if you do not desire to see
     strtxy (20.,60.) ;
     strout ("the color tables, type the letter 'n' ") ;
     while ( a == 'b' )
         a = getchar () ;
     if ( a == 'y' ) showclr () ;
     erase () ;
     colort (0) ;
     color (15) ;
     strtxy (20.,50.) ;
     strout ("Select the color table you wish to use ") ;
     strtxy (20.,46.) ;
     strout (" 0 - Greys          1 - Blues") ;
     strtxy (20.,42.) ;
     strout (" 2 - Greens              3 - Reds") ;
     strtxy (20.,38.) ;
     strout (" 4 - Mixed                   5 - Mixed") ;
     strtxy (20.,34.) ;
     strout (" 6 - Mixed               7 - Mixed") ;
     strtxy (20.,30.) ;
     strout (" 8 - Mixed Followed by a CR ") ;
     while ( n == 99 )
         n = getnum (10) ;
     a = 'b' ;
     erase () ;
     return ;
}
```

```

/***** PROGRAM TO DISPLAY COLOR TABLES *****/

showclr ()
{
     flag = 1 ;
     erase () ;
     for ( i = 0; i < 9; i++ ) {
          a = 'b' ;
          draw (i) ;
          setmode (0,0) ;
          strtxy (50.,50.) ;
          strout (" Next color table ? y = yes ; n = no ") ;
          while ( a == 'b' )
               a = retchar () ;
          if ( a == 'n' ) i = 10 ;
          else continue ;
     }
     colort (0) ;
     color  (15) ;
     erase () ;
     return ;
}

/***** ROUTINE TO CHANGE THE TYPE OR COLOR OF THE DISPLAY *****/

change ()
{

     a = 'b' ;
     strtxy (20.,90.) ;
     strout ("If you want to change the TYPE or COLOR table of the")
     strtxy (20.,86.) ;
     strout ("display type the letter 'c'.  If you want to quit type
     strtxy (20.,82.) ;
     strout ("the letter 'q'.") ;
     while ( a == 'b' )
          a = retchar () ;
     if ( a == 'q' )
          finish () ;
     else
     {
          erase () ;
          page2 () ;
          a = 'b' ;
          strtxy (20.,80.) ;
          strout ("If you want to change the color table selected typ
          while ( a == 'b' )
               a = retchar () ;
          if ( a == 'y' ) {
               erase () ;
               page3 () ;
               }
     }
     display () ;
     return ;
}

finish ()
{
```

```
    a = 'b' ;
    colort (0) ;
    color (15) ;
    erase () ;
    strtxy (20.,80.) ;
    strout ("You have indicated that you wish to terminate this se
    strtxy (20.,76.) ;
    strout ("if you do wish to stop type a capital 'S' .  Otherwis
    strtxy (20.,72.) ;
    strout ("type a 'c' and you can continue.");
    while ( a == 'b' )
        a = retchar () ;
    if ( a == 'S' )
        exit () ;
    else
    {
        erase() ;
        change () ;
    }
}
```

# REFERENCES


[1] Patton, J.W., A Hardware Design for a Computer Aided Acoustic Imaging System, Masters Thesis, Naval Postgraduate School, Monterey, 1977.

[2] Powers, J.P., DeBlors, J.R.V., O'Bryon, R.T. and Patton J.W., 'A Computer Aided Ultrasonic Imaging System', Acoustical Holography vol 8, AF Metherell, Ed., pp 233-248, Plenum Press, 1979

[3] See for example the special Issue an Acoustic Imaging, Proc IEEE April 1979

[4] Wade, G. Acoustic Imaging with Holography and Lenses, IEEE Trans. on Sonics and Ultrasonics, Vol SU - 22 No 6, pp 385 - 394, 1975

[5] Thurstone, F.L. Ultrasound Holography and Visual Reconstruction Proc. Symp. Biomed Eng. 1,12,1968.

[6] Powers J.P. "Computer simulation of linear acoustic diffraction", Acoustical Holography, Vol. 7, (L.W. Kessler, Ed.) Plenum Press. pp 193-205,1977.

[7] Kernigham, B.W., Programing in C A tutorial, Bell Laboratories, Murry HIll, N.J.

[8] "Ramtek GX-100 A Programming Manual", Ramtek Corp,1974.

[9] "Ramtek GX-100B Programming Manual," Ramtek Corp.,1975.

[10] Mc Neil C.E. and Hanston D.P. The Development of A User Oriented Interface for a Computer Driven Graphics Device, Masters Thesis, Naval Postgraduate School, Monterey, June 1977.

[11] Ritchie D.M. C Reference Manual Bell Laboratories, Murray Hill,N.J.

[12] Kernigham B.W. Unix for Beginners, BEll Laboratories Murray Hill, N.J. (As updated by C.E. Irvine, NPS Computer Laboratory), May 1978.

[13] Ritchie D.M. and Thompson K. The Unix Time Sharing System Bell Laboratories, Murry Hill, N.J.

## INITIAL DISTRIBUTION LIST

| | | |
|---|---|---|
| 1. | Defense Document Center<br>Cameron Station<br>Alexandria, Virginia 22314 | 2 |
| 2. | Library, Code 0142<br>Naval Postgraduate School<br>Monterey, California 93940 | 2 |
| 3. | Department Chairman, Code 62<br>Department of Electrical Engineering<br>Naval Postgraduate School<br>Monterey, California 93940 | 1 |
| 4. | Professor J. P. Powers, Code 62PO<br>Department of Electrical Engineering<br>Naval Postgraduate School<br>Monterey, California 93940 | 5 |
| 5. | Dr. Newell Booth, Code 6513<br>Naval Ocean Systems Center<br>San Diego, California 92152 | 2 |
| 6. | Mr. Norman Caplan<br>Automation, Bioengineering and<br>Science Systems Program<br>Engineering Division<br>National Science Foundation<br>1800 G St.<br>Washington D.C. 20550 | 1 |
| 7. | Professor G. A. Rahe, Code 52RA<br>Department of Computer Science<br>Naval Postgraduate School<br>Monterey, California 93940 | 1 |
| 8. | LT Eugene L. Moon USN<br>5544 Charlotte Way<br>Livermore, California 94550 | 1 |