(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) Publication number: **0 130 301**
A2

(12) EUROPEAN PATENT APPLICATION

(21) Application number: **84104627.9**

(22) Date of filing: **25.04.84**

(51) Int. Cl.⁴: **G 09 G 1/00**, G 09 G 1/16

(30) Priority: **31.05.83 US 499451**
31.05.83 US 499458
31.05.83 US 499452
31.05.83 US 499453

(43) Date of publication of application: **09.01.85**
Bulletin 85/2

(84) Designated Contracting States: **DE FR GB IT**

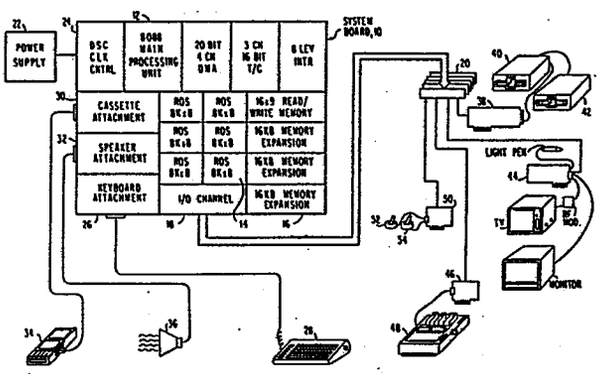
(71) Applicant: **International Business Machines Corporation, Old Orchard Road, Armonk, N.Y. 10504 (US)**

(72) Inventor: **Stephens, Lawrence Keith, 2303 Harbinger Dr., Dallas, Tx. 75252 (US)**

(74) Representative: **Bonin, Jean-Jacques, Compagnie IBM France Département de Propriété Industrielle, F-06610 La Gaude (FR)**

(54) Device for displaying alphanumeric and graphics characters.

(57) The invention relates to a computer having an all points addressable display terminal (44) and a cursor positioning device (52). It comprises means for storing one or several tables of selectable characters, means for selecting a cursor character from said table, said character being movable by said cursor positioning device on the display, and means for fixing the selected character at a desired point on the display. The cursor characters may be alphanumeric characters, graphics characters, alphanumeric character strings or even blank characters.



EP 0 130 301 A2

0130301

DEVICE FOR DISPLAYING ALPHANUMERIC AND GRAPHICS CHARACTERS

The present invention generally relates to Computer Assisted Design (CAD) systems, and more particularly to an inexpensive and easy to use CAD application for personal computers.

Computer Assisted Design (CAD) and Computer Assisted Manufacturing (CAM) systems have been used for some time in the aircraft and automotive industries to design and manufacture aerodynamic and mechanical components. Such systems typically comprise a main frame computer, large bulk memory systems including tape units, rigid disk units and removable disk pack units, high resolution All Points Addressable (APA) Cathode Ray Tube (CRT) displays, a large Random Access Memory (RAM) of sufficient capacity to store the graphics application and address each pixel of the high resolution displays, and Input/Output (I/O) devices such as digitizer pads with cursors and plotters. These systems are very expensive, but their cost could be justified because of the large sums of money invested in the design and manufacture of an aircraft or a new automobile model. The price of CAD systems has come down significantly over the past decade due to economies of computer and memory system manufacture, and because of that, CAD systems are being applied to many new uses among which are architectural design and the layout of photoresist patterns for integrated circuits. Nevertheless, CAD systems are still quite expensive, and their use is generally limited to correspondingly expensive applications.

At the other end of the spectrum are the so-called personal computers based on the microprocessors which have been developed over the past decade. These typically comprise a mother board containing the microprocessor, a Read Only Memory (ROM) encoded with the Basic Input/Output System (BIOS) for controlling the microprocessor, a limited amount of RAM, and a number of adapters for interfacing with various I/O devices. These I/O devices may include a keyboard, a medium or high

resolution CRT display, one or more floppy disk drives, and a printer such as one of the more popular dot matrix printers. Although personal computers are small and compact, they are capable of some fairly sophisticated applications. They are especially well suited to business applications such as accounting, data base management and business analysis. Recently, a number of business applications have been developed which include graphics support. These applications take the input or calculated numerical data and produce line graphs, bar charts and pie charts which are much easier to interpret than the raw numerical data. Prints of these graphical displays are made by reading out the data in the APA display RAM to a dot matrix printer provided with a graphical capability or to an inexpensive pen plotter. The latter device is also capable of generating transparencies for use in overhead projectors. The acceptance of business applications with graphics support has been immediate and substantial with the result that there is a considerable demand for graphics applications which are not necessarily limited to business graphs. The ability to generate schematic diagrams, flow charts, floor plans and similar graphic displays would be highly desirable in the production of technical manuals, advertising layouts and the like.

It is therefore an object of the invention to provide an inexpensive Computer Assisted Design (CAD) application for personal computers.

It is another object of the invention to provide a CAD system for personal computers which is easy to use and facilitates the generation of schematic diagrams, flow charts and other free form graphics displays.

It is a further object of the invention to provide a user friendly CAD application for personal computers which is operated by an inexpensive joy stick or similar device and supports a dot matrix printer or inexpensive plotter.

The objects of the invention are accomplished by making the cursor symbol a graphics character or an A/N string which may be moved about the display screen by means of a joystick or similar input device. Once the graphics character or A/N string is positioned at the desired location, the operator presses a command button, and the graphics symbol or A/N string is fixed in position on the display screen by reading the symbol data into the display buffer at that position. The cursor symbol can be moved again to another location on the display and another character fixed in position on the display by pressing the command button. Different cursor symbols can be selected from symbol tables so that a variety of symbols can be used to generate the graphics display. The current cursor symbol is demarked from other graphics characters fixed in the display by continuously exclusive ORing the cursor symbol with the background graphics data. In addition, an erase function is provided to allow the correction of mistakes and modification of standard symbols.

The foregoing and other objects, advantages and aspects of the invention will be better understood from the following detailed description of the invention making reference to the accompanying drawings, in which:

Figure 1 is a system block diagram of a typical personal computer on which the application according to the invention is operated;

Figure 2 is a block diagram of a color/graphics monitor adapter of the type which is required to support the application according to the invention;

Figure 3 is a block diagram of a game control adapter of the type which provides a joystick input to the personal computer shown in Figure 1;

Figure 4 is a flow diagram illustrating the procedure for loading a cursor symbol table;

Figure 5 is a flow diagram illustrating the procedure for displaying a loaded cursor symbol table;

Figure 6 is a flow diagram illustrating the procedure for selecting a new cursor symbol for a loaded symbol table;

Figure 7 is a flow diagram illustrating the procedure for demarking the current cursor symbol from other graphics symbols which may be displayed on the screen;

Figure 8 is a flow diagram illustrating the procedure for generating a circle of any radius at any desired position on the display screen;

Figure 9 is a flow diagram illustrating the procedure for entering and positioning A/N strings in the display;

Figure 10 is a flow diagram illustrating the procedure for erasing previously entered graphics data on the screen; and

Figure 11 illustrates a sample display generated using the application according to the invention.

In order to better understand the invention, a typical personal computer will be first described with reference to Figure 1 of the drawings. The system or mother board 10 includes the microprocessor 12, ROM 14, RAM 16, and an I/O channel 18 which includes a number of I/O expansion slots 20 for the attachment of various options. A power supply 22 provides power to the mother board 10 and the attached options. The mother board 10 in addition includes a crystal oscillator, clock and control circuits 24 and a keyboard attachment 26 to which a keyboard 28 is attached. In addition, the mother board may also include other attachments such as a cassette attachment 30 and a speaker attachment 32 to which are connected a cassette recorder 34 and a speaker 36, respectively. The expansion slots 20 are designed to receive

any of the various adapter printed circuit cards shown in the figure. More specifically, a diskette drive adapter 38 may be plugged into one of the slots 20. This adaptor 38 is required to support one or more diskette drives 40 and 42. A color/graphics monitor adapter 44 may also be plugged into one of the slots 20, and this adpter supports either a home color TV or an RGB monitor and a light pen. A parallel printer adapter 46 may be plugged into another one of slots 20 to support, for example, a dot matrix printer 48. Finally, a game control adapter 50 can be plugged into a remaining one of the slots 20 to support one or more joy sticks 52 and 54. Other adapters may be plugged into the slots 20, but only those necessary to support the present invention are illustrated.

The color/graphics adapter 44 has two basic modes of operation; alphanumeric (A/N) and APA. In both modes, A/N characters are defined in a character box and formed from a ROM characters. Figure 2 is a block diagram of the adapter 44 which contains a display buffer 56 and a CRT controller device 58 such as a Motorola 6845 IC. The controller device 58 provides the necessary interface to drive a raster scan CRT. The display buffer 56 can be addressed by both the CPU and the controller device 58 through address latches 60 and 62, respectively. Data is read out of the display buffer to data latches 64 and 66 which provide outputs to a graphics serializer 68 and a character generator comprising ROM 70 and an alpha serializer 72. The outputs of the serializers 68 and 72 are provided to the color encoder 74 which either drives an RGB monitor directly or provides an output to the composite color generator 76 that drives a home color TV. The color encoder 74 also receives the output of the palette/overscan circuits 78 which provides intensity information. The composite color generator 76 receives horizontal and vertical synchro signals from the CRT controller device 58 and timing control signals from the timing generator and control circuits 80. The timing generator and control circuits also generate the timing signals used by the CRT controller device 58 and

the display buffer 56 and resolves the CPU and controller contentions for accessing the display buffer.

Figure 3 is a block diagram of the game control adapter 50. The adapter comprises instruction decode circuits 80 which may be composed of 74LS138 IC's. The data bus is buffered by a 74LS244 buffer/driver 82. The digital inputs to this buffer/driver are provided by trigger buttons on the joy sticks. The joystick positions are indicated by a potentionmeter for each coordinate which must be converted to digital pulses by resistance to pulse converter 84. This is accomplished by providing a one-shot for each potentiometer so that the potentiometer varies the time constant of its associated one-shot. A select output from decoder 80 causes the one shots to be fired to provide pulse outputs to the buffer/driver 82.

Although the invention is described as using a joy stick to position a cursor symbol on the display screen, it will be understood by those skilled in the art that other input devices can be used including cursor keys on a keyboard. The cursor keys are, however, inherently slow to operate, and so it is prefereable to use a joy stick or similar type input device. Besides a joy stick, a "mouse"might just as well be used. These devices have a ball on the bottom of a palm size controller, and the ball is rolled on a flat surface to control the cursor position.

Typically, the ball actuates potentiometers in a manner quite similar to a joy stick. Thus, everywhere a joy stick is mentioned in the description of the invention, those skilled in the are will recognize that a "mouse" or other similar input device could be substituted.

According to the invention, the cursor on the CRT display is replaced by a graphics symbol or an A/N character string and moved by means of a joy stick or similar device. When the graphics symbol or A/N character string are positioned on the

display at the desired position, the operator presses a trigger button on the joy stick and the graphics symbol or A/N character string remain fixed at that location by reading the symbol data into the display buffer. A new graphics character or A/N character string can then be selected for the cursor symbol and the process repeated so that a schematic diagram, flow chart or similar graphics display can be built. Previously positioned graphics characters or A/N character strings can be erased totally or partially by means of a box cursor and the operation of the trigger button on the joy stick. This allows not only for the correction of errors but also the generation of modified characters giving more flexibility to the defined character tables. In addition, since the selected cursor symbol will remain the cursor symbol until changed even after a graphics character or A/N character string has been positioned on the display screen, the cursor symbol is at all times exclusive ORed with display data of all coincident pixels as it is moved about the display screen to provide a clear and visible demarkation of the cursor symbol from other symbols previously placed at various locations on the display screen. Thus, the invention allows a fully interactive positioning of graphics characters and/or A/N character strings at any addressable point on the display screen. Since the screen information is contained in the APA display buffer, the screen can be printed in the usual way to provide a hard copy output thereby facilitating the production of technical illustrations, manuals or the like.

The underlying feature of the invention is the use of a graphics character as the cursor symbol. Therefore, one and preferably more symbol tables are provided. For example, a table could be provided for electrical symbols, another for architectural symbols, and another for industrial process symbols. Each symbol in each table is identified by number so that the code for a symbol includes both the table to which it belongs as well as its number within the table. In order to select a cursor symbol, a symbol table must first be loaded into RAM. This process is illustrated by the flow diagram

shown in Figure 4. When the operator requests a new symbol table, s/he is first prompted for the name of the symbol table as indicated by block 86. The name input by the operator is checked to determine if it is a valid name, that is it identifies a table that exists in the current library of tables. This is indicated by the decision block 88. If the name is not a valid name, an error message is displayed to the operator at block 90 and the operator is again prompted for the name of the symbol table desired. When a valid name is input by the operator, the old cursor symbol is exclusive ORed with itself to delete the symbol from the display screen as indicated by block 92. Then in block 94 the new symbol table is loaded into RAM, and in block 96, the first graphics character is exclusive ORed with the current cursor symbol to cause the first graphics character to be displayed as the cursor symbol. In other words, the first graphics character is the default cursor symbol.

The default cursor symbol may not be the symbol desired by the operator, so it may be desirable to display the selected symbol table to permit selection of the desired symbol. This process is illustrated by the flow diagram shown in Figure 5. When the operator requests that the symbol table be displayed, the title of the currently selected symbol table is first displayed as indicated by block 98. The title will always be displayed during this process no matter how the field of the display may change. In other words, any given symbol table may be too large to fit on a single screen and it may be necessary to scroll the display in order for the operator to view all the symbols in the table. While the field of the display may be scrolled, the title placed on the screen by block 98 will remain. Once the title of the table has been put up, the numbers for the various graphics characters are next put up as indicated by block 100, and then the actual graphics characters are put up adjacent their corresponding number as indicated by block 102. Three function keys identified as F3, F2 and F1 are monitored to detect if they have been pressed by the operator as indicated by the decision blocks 104, 106 and

108. If for example key F3 has been pressed, then the graphics screen is redrawn as indicated by block 110. When this is done, the operator is presented a display of the graphics screen as s/he had generated it to that point in time. If F2 is pressed, then the symbol table is scrolled down a predetermined amount as indicated by block 112, but if F1 is pressed, the symbol table is scrolled up a predetermined amount as indicated by block 114. In other words, the function keys F3, F2 and F1 give the operator control of the display screen after the symbol table is displayed. F3 allows the operator to exit the display, and F2 and F1 allow the operator to scroll the display.

It is not necessary to display the symbol table each time it is desired to change the cursor symbol. The operator may already know the numbers of the symbols s/he wants to use in generating a graphics display, or more likely, the operator will have printed copies of the symbol tables to refer to. In any event, once a symbol table has been loaded according to the process illustrated in Figure 4 and the first symbol of the table is displayed as the default cursor symbol, the operator may wish to change the cursor symbol. This is accomplished with the selection of a new symbol according to the procedure illustrated in the flow diagram of Figure 6. The operator selects a cursor symbol by number within the currently loaded table. The first thing that is done when a cursor symbol selection has been made is to retrieve the table entry as indicated in block 116 and then in block 118 validate the entry. It will be understood that the various cursor symbol tables will not necessarily be the same size and that a symbol number that is valid for one table may be valid for another. Should the operator enter an invalid symbol number, an error message is displayed as indicated by block 120 and the operator is returned to the selection menu. Assuming that a valid symbol number is selected, the old cursor symbol is deleted from the screen by exclusive ORing the symbol with itself as indicated in block 122. Then using the table entry, the offset into the symbol table is determined in block 124.

This provides the access to the desired symbol code for the character generator in block 126, and in block 128, the new cursor symbol is displayed by exclusive ORing the symbol with the background data on the screen.

This latter process is the basis for demarking the current cursor symbol from other graphics symbols already placed in the graphics display. It will be appreciated that since the cursor of the subject invention is not a conventional cursor mark but rather a graphics symbol that is moved like a cursor to a desired position on the display screen and then fixed by command, there is the possibility that the operator might lose track of where and which of several currently displayed symbols is the cursor. This is accomplished in part by making the cursor symbol a flashing symbol as is conventional, but in addition, the current cursor symbol is exclusive ORed with the backround display data to clearly demark the symbol where ever it may be on the screen from other graphics data already in place on the screen. This procedure is illustrated by the flow diagram shown in Figure 7. In block 130, the current X, Y position as commanded by the joy stick control and the cursor symbol data are input and exclusive ORed. Then in blocks 132 and 134, the X and Y positions are temporarily stored as X_{old} and Y_{old} . The current X and Y coordinates are obtained from the joy stick or cursor key input in block 136. Then in blocks 138 and 140 X_{old}, Y_{old} and the cursor symbol data are exclusive ORed and current X, Y and the cursor symbol data are exclusive ORed. This removes the cursor symbol from the display screen and then redisplays it at its new location. The process is then repeated.

Besides the several symbol tables from which the operator can select a variety of cursor symbols, lines can be drawn between positioned symbols by indicating the coordinates of the end points of the line in the conventional manner. In addition, many graphics displays will make use of circles or arcs. Rather than provide a table of circles, a process for displaying a circle of any desired radius is provided. This

process is illustrated in the flow diagram shown in Figure 8. First the operator presses a function key F5 to indicate that s/he desires to draw a circle. This causes a conventional cursor symbol to appear on the screen and represents the center of the circle. The operator can then position this center on the screen using the joy stick. Then by pressing the trigger button on the joy stick, the circle center is fixed as indicated by block 142. Once this is done, the X, Y coordinates of the center are obtained in block 144, and in block 146, a circle of five units is drawn. This is the smallest diameter circle that is displayed. The trigger button is then monitored as indicated by the decision block 146, and if it is pressed, the circle is expanded by one unit in block 148. In this way the operator can increase the size of the circle, and when the desired size has been reached, the operator releases the trigger button.

As previously mentionned, the cursor symbol may be an A/N string as well as a graphics character. The operator enters the A/N mode by making the appropriate menu selection. The process is illustrated by the flow diagram shown in Figure 9. The number of A/N characters entered are counted and so in block 150, the counter is set to 1. The operator is prompted in block 152 to input text from the keyboard, and as each character is keyed, the character equivalent of the operator's entry is put on the screen in block 154. Assuming that the "ENTER" key has not been pressed in decision block 156, the counter is incremented by one in block 158 and then checked in decision block 160 to determine if the maximum allowed number of characters has been entered. In the case illustrated, the maximum number of characters is sixty, but any number of characters can be arbitrarily set. The process continues until either the operator presses the "ENTER" key or the maximum number of characters has been entered at which time a buffer is loaded with all the A/N characters keyed by the operator as indicated by block 162. This buffer is treated as the cursor symbol data which can be positioned anywhere on the display screen by use of the joy stick. Thus, the A/N string is

continually exclusive ORed on the screen as the current cursor until the trigger button is pressed by the operator as indicated by block 164. In other words, when the operator enters the text mode from the menu, s/he first keys in the desired text, presses "ENTER" and then moves the A/N string around the screen as the current cursor symbol. When the text string is in the desired position, the trigger button on the joystick is pressed and the A/N string is fixed in the display data field.

All good designers need an eraser to correct mistakes and modify standard symbols. The erase mode is entered by making the appropriate selection from a menu, and upon entry into this mode, the cursor symbol is changed to a rectangular box of predetermined dimensions. The process is illustrated by the flow diagram shown in Figure 10. After the menu selection, the current X, Y position of the "eraser" rectangle is obtained as indicated in block 166. In decision block 168, the trigger button on the joy stick is monitored to determine if it has been pressed. If it has not been pressed, the position of the "eraser" rectangle is checked again and so on while the operator moves the "eraser" rectangle around the display screen. When the "eraser" rectangle is positioned over that area of the display screen which is desired to be erased, the operator presses the trigger button which causes all the display data within the "eraser" rectangle to be set to "zeros" to blank that area of the display screen as indicated by block 170. It is also possible to move the "eraser" rectangle with the joy stick while pressing the trigger button which will result in all display data within the path of the "eraser" rectangle to be reset to "zeros". The procedure allows graphics data to be removed from the display screen easily and accurately.

Figure 11 is an illustration of a graphics display constructed using the invention. Only a few graphics symbols were used plus circles, lines and A/N strings. Each symbol was selected from a table of symbols and then positioned at a desired

location on the display screen using the joy stick and trigger button. At the bottom of the display is a menu from which the operator may make selections of operating modes.

Attached hereto as an appendix is the code listing of the application according to the invention. This code listing was prepared using the IBM Personal Computer BASIC Compiler.

From the foregoing, it will be appreciated that the invention provides an inexpensive CAD application for personal computers which is easy to use and facilitates the generation of many graphics displays that heretofore could be generated using only much more expensive equipement.

0130301

PAGE 1
03-16-84
14:01:33

```
Offset  Data    Source Line          IBM Personal Computer BASIC Compiler V1.00

 001A   0002    on error goto 31300
 0035   0002    '
 0035   0002    defint a-z
 0036   0002    gosub 30000  ' turn off all function keys
 003B   0002    key off
 0041   0002    chrh = 36
 0048   0004    chrw = 48
 004F   0006    deadband = 20
 0056   0008    xmax = 319
 005D   000A    ymax = 199
 0064   000C    x = 160
 006B   000E    y = 100
 0072   0010    nofilefl = 0
 0079   0012    numbsymb = 128
 0080   0014    speed = 1
 0087   0016    stateflag = 1
 008E   0018    dim boxsel(904)
 008F   072A    rotatesym = 0
 0096   072C    '
 0096   072C    boxoff = 456
 009D   072E    bigboxoff = 496
 00A4   0730    smalloff = 980
 00AB   0732    blueoff = 1554
 00B2   0734    greenoff = 1594
 00B9   0736    redoff   = 1634
 00C0   0738    whiteoff = 1674
 00C7   073A    symbtoff = 1714
 00CE   073C    newsymboff = 8000
 00D5   073E    '
 00D5   073E    oldoff = 20
 00DC   0740    ochrh = chrh
 00E3   0742    ochrw = chrw
 00EA   0744    startnbr = 0
 00F1   0746    osoff = 20
 00F8   0748    selflag = 0
 00FF   074A    button = 0
 0106   074C    entryflg = 0
 010D   074E    '
 010D   074E    '     DDT record's arrays
 010D   074E    '
 010D   074E    dim entry(30)
 010E   078C    dim typeddt(30)
 010F   07CA    dim ulxchar(30)
 0110   0808    dim ulychar(30)
 0111   0846    dim alarm(30)
 0112   0884    dim ulxvalue(30)
 0113   08C2    dim ulyvalue(30)
 0114   0900    dim boxsizex(30)
 0115   093E    dim boxsizey(30)
 0116   097C    ddtrecno = 0
 011D   097E    dim colr(3000)
 011E   20F0    dim xnew(3000)
 011F   3862    dim ynew(3000)
 0120   4FD4    '
```

```
Offset  Data    Source Line          IBM Personal Computer BASIC Compiler V1.00

 0120   4FD4    initflag = 0
 0127   4FD6    colorflg = 0
 012E   4FD8    joyx = 0
 0135   4FDA    joyy = 0
 013C   4FDC    '
 013C   4FDC    '    Variables for the symbol generator
 013C   4FDC    '
 013C   4FDC    dim box0(8)
 013D   4FEE    dim box1(8)
 013E   5000    dim box2(8)
 013F   5012    dim box3(8)
 0140   5024    xbox = 1
 0147   5026    ybox = 1
 014E   5028    xboxmin = 1
 0155   502A    yboxmin = 1
 015C   502C    xboxmax = 196
 0163   502E    yboxmax = 146
 016A   5030    xreal = 278
 0171   5032    yreal = 1
 0178   5034    '
 0178   5034    '
 0178   5034    '    To enable the user to change colors quickly,
 0178   5034    '    it became necessary to read in some data points
 0178   5034    '    These represent bit masks for identifying
 0178   5034    '    colors.
 0178   5034    '    Here is where they are read from a data statement.
 0178   5034    '
 0178   5034    dim newcolor(15)
 0179   5054    '
 0179   5054    dim chcolor(31)
 017A   5094    '
 017A   5094    dim coloro(3)
 017B   509C    '
 017B   509C    '    newcolor = the array of new color masks
 017B   509C    '
 017B   509C    for i = 0 to 15
 0181   509C         read newcolor(i)
 018F   509E    next i
 019E   509E    '
 019E   509E    '    chcolor = the array of original masks to be recognized
 019E   509E    '              and changed.
 019E   509E    '
 019E   509E    for i = 0 to 31
 01A4   509E         read chcolor(i)
 01B2   509E    next i
 01C1   509E    '
 01C1   509E    '    coloro = offset values for the individual colors
 01C1   509E    '             into the 2 mask arrays.
 01C1   509E    '
 01C1   509E    for i = 0 to 3
 01C7   509E         coloro(i) = i * 8
 01DC   509E    next i
 01EB   509E    '
 01EB   509E    '
```

```
Offset  Data    Source Line          IBM Personal Computer BASIC Compiler V1.00

 01EB   509E    '
 01EB   509E    '  offset2 is the offset past the offsets and number of
 01EB   509E    '  symbols into the actual symbols in the symbol table.
 01EB   509E    '  initially it is set to reflect the default symbol table
 01EB   509E    '
 01EB   509E    offset2 = 258
 01F2   50A0    '
 01F2   50A0    '     get segment register from dos offset
 01F2   50A0    '
 01F2   50A0    def seg = 0
 01F8   50A0    storage = peek(&h3fe) + 256 * peek(&h3ff)
 022B   50A2    def seg = storage
 0233   50A2    '
 0233   50A2    ' initial cursor will be the duck
 0233   50A2    '
 0233   50A2    bload "worksym.sym",20
 023D   50A2    '
 023D   50A2    '     set up color blocks
 023D   50A2    '
 023D   50A2    for i = 0 to 3
 0243   50A2    '
 0243   50A2         off1 = i * 40
 024E   50A4         poke blueoff + off1, 20
 0262   50A4         poke blueoff + off1 + 1, 0
 0276   50A4         poke blueoff + off1 + 2, 12
 028C   50A4         poke blueoff + off1 + 3, 0
 02A2   50A4    '
 02A2   50A4         if i = 0 then ij = 0
 02B4   50A6         if i = 1 then ij = 85
 02C6   50A6         if i = 2 then ij = 170
 02D8   50A6         if i = 3 then ij = 255
 02EA   50A6         for j = 4 to 39
 02F1   50A6             poke blueoff + j + off1, ij
 030B   50A8         next j
 031A   50A8    '
 031A   50A8    next i
 032C   50A8    '
 032C   50A8    '  initial symbol table will be defaulted
 032C   50A8    '
 032C   50A8         bload "DEFAULT.SYM",symbtoff
 0337   50A8    '
 0337   50A8         def seg
 033B   50A8    '
 033B   50A8    '     get the joystick settings
 033B   50A8    '
 033B   50A8         gosub 22000
 0340   50A8    '
 0340   50A8         gosub 15000      ' switch to the graphics tube
 0345   50A8    '
 0345   50A8         gosub 502 ' display prompts
 034A   50A8    '
 034A   50A8         soff = 20
 0351   50AA         call overlay( x, y, storage, soff)
 036A   50AA    '
```

```
Offset  Data    Source Line
 036A   50AA    '
 036A   50AA    '     Turn on the joystick button
 036A   50AA    '
 036A   50AA    strig(0) on
 0370   50AA    on strig(0) gosub 70
 0379   50AA    '
 0379   50AA    '     Set up the joystick qualifiers
 0379   50AA    '
 0379   50AA          nxvar = xvar - deadband
 0384   50AE          xvar = xvar + deadband
 038F   50AE          xvar1 = xvar + 20
 0399   50B0          nxvar1 = nxvar - 20
 03A3   50B2          xvar2 = xvar1 + 40
 03AD   50B4          nxvar2 = nxvar1 - 40
 03B7   50B6    '
 03B7   50B6          nyvar = yvar - deadband
 03C2   50BA          yvar = yvar + deadband
 03CD   50BA          yvar1 = yvar + 20
 03D7   50BC          nyvar1 = nyvar - 20
 03E1   50BE          yvar2 = yvar1 + 40
 03EB   50C0          nyvar2 = nyvar1 - 40
 03F5   50C2    '
 03F5   50C2          def seg = storage
 03FD   50C2    '
 03FD   50C2    '     If monochrome is attached put up the help screen
 03FD   50C2    '
 03FD   50C2          if peek(14) = 1 then gosub 31400
 0418   50C2    '
 0418   50C2    switch = 1
 041F   50C4    while switch
 042B   50C4    '
 042B   50C4          xold = x
 0432   50C6          yold = y
 0439   50C8          osoff = soff
 0440   50C8          if key2 = 71 then x = x - speed: y = y - speed
 0461   50CA          if key2 = 72 then y = y - speed
 0477   50CA          if key2 = 73 then x = x + speed: y = y - speed
 0498   50CA          if key2 = 75 then x = x - speed
 04AE   50CA          if key2 = 77 then x = x + speed
 04C4   50CA          if key2 = 79 then x = x - speed: y = y + speed
 04E5   50CA          if key2 = 80 then y = y + speed
 04FB   50CA          if key2 = 81 then x = x + speed: y = y + speed
 051C   50CA          key2 = 0
 0523   50CA          if joystick <> 0 then gosub 5
 0533   50CC    '
 0533   50CC    '     Check cursor positioning
 0533   50CC    '
 0533   50CC          gosub 2
 0538   50CC    '
 0538   50CC    '
 0538   50CC    '     Continually xor the cursor onto the screen.
 0538   50CC    '
 0538   50CC          if stateflag = 5 or stateflag = 8 then gosub 3010
 055D   50CC          if stateflag = 7 then gosub 41 else if stateflag <> 5 and
```

```
Offset  Data    Source Line          IBM Personal Computer BASIC Compiler V1.00

                stateflag <> 8 then gosub 10
 0595   50CC    '
 0595   50CC    '    Handle below the display line functions
 0595   50CC    '
 0595   50CC         if colorflg = 0 then if y + chrh > 176 then entryflg = 1:
                gosub 3
 05BC   50CC         if colorflg = 0 then if entryflg > 0 then if y + chrh < 17
                2 then initflag = 0: stateflag = 1: gosub 14800: gosub 9: gosub
                 502
 05FF   50CC    '
 05FF   50CC    '    get key input from the user.
 05FF   50CC    '
 05FF   50CC         if button = 1 then gosub 1000 ' switch checker
 060F   50CC         button = 0
 0616   50CC    '
 0616   50CC         k$ = inkey$
 061F   50D0         if k$ <> "" then if len(k$) < 2 then gosub 55 else gosub 6
                0
 0648   50D0    '
 0648   50D0    '
 0648   50D0    wend
 064C   50D0    '
 064C   50D0    end
 0650   50D0    '
 0650   50D0    1 '
 0651   50D0    '
 0651   50D0    '    Display prompt lines
 0651   50D0    '
 0651   50D0         locate 24,1
 065E   50D0         print "Symbol  Display  Load  Overlay    Erase";
 0666   50D0         locate 25,1
 0673   50D0         print "Color   Reset    Text  File   Associate";
 067B   50D0    '
 067B   50D0         return
 067E   50D0    '
 067E   50D0    2 '
 067F   50D0    '
 067F   50D0    '    Check x and y positioning and adjust accordingly
 067F   50D0    '
 067F   50D0         if x + chrw > xmax  then x = xmax - chrw else if x < 0 the
                n x = 0
 06B0   50D0         if y + chrh > ymax  then y = ymax - chrh else if y < 0 the
                n y = 0
 06E1   50D0    '
 06E1   50D0         return
 06E4   50D0    '
 06E4   50D0    3 '
 06E5   50D0    '
 06E5   50D0    '    Handle positioning of the cursor
 06E5   50D0    '    And branch accordingly.
 06E5   50D0    '
 06E5   50D0    '    reset state flag to reflect the proper state
 06E5   50D0    '    Default = Place (1)
 06E5   50D0    '
```

```
Offset  Data    Source Line
 06E5   50D0        if stateflag = 2 then: stateflag = 1: gosub 14800: gosub 5
                02: return
 0704   50D0    '
 0704   50D0    '     Change cursor, set state for selection, and prompt user.
 0704   50D0    '
 0704   50D0        if selflag = 0 then gosub 4: stateflag = 0: gosub 502: cal
                l overlay(x, y, storage, soff): selflag = 1
 0740   50D0    '
 0740   50D0    '
 0740   50D0    '     Note: no changing of the true character to anything else
 0740   50D0    '           should go above here.
 0740   50D0    '
 0740   50D0        xch = x + 1
 0748   50D2        ych = y + 1
 0750   50D4    '
 0750   50D4        if ych < 192 then gosub 31000 else if ych > 191 then gosub
                 31100
 0775   50D4    '
 0775   50D4        gosub 8
 077A   50D4        button = 0
 0781   50D4    '
 0781   50D4        return
 0784   50D4    4 '
 0785   50D4    '
 0785   50D4    '     Change to the degrees
 0785   50D4    '
 0785   50D4        quadnew = 0
 078C   50D6        quadold = 0
 0793   50D8        call overlay(x, y, storage, soff)
 07AC   50D8        befselx = x
 07B3   50DA        befsely = y
 07BA   50DC        oldoff = soff
 07C1   50DC        soff = smalloff  ' initially a degrees sign
 07C8   50DC        ochrh = chrh
 07CF   50DC        ochrw = chrw
 07D6   50DC        chrh = 3
 07DD   50DC        chrw = 3
 07E4   50DC        y = 187
 07EB   50DC    '
 07EB   50DC        return
 07EE   50DC    '
 07EE   50DC    5 '
 07EF   50DC    '
 07EF   50DC    '     Joystick subroutine
 07EF   50DC    '
 07EF   50DC            x2 = stick(0)
 07F9   50DE            y2 = stick(1)
 0804   50E0    '
 0804   50E0            if x2 > xvar then joyx = joyx + 1 else if x2 < nxvar t
hen jo
                yx = joyx - 1
 0831   50E0            if y2 > yvar then joyy = joyy + 1 else if y2 < nyvar t
hen jo
                yy = joyy - 1
 085E   50E0    '
 085E   50E0            if joyx > 6 then x = x + speed: joyx = 0
```

```
Offset  Data    Source Line          IBM Personal Computer BASIC Compiler V1.00

 087B   50E0             if joyy > 6 then y = y + speed: joyy = 0
 0898   50E0             if joyx < -6 then x = x - speed: joyx = 0
 08B5   50E0             if joyy < -6 then y = y - speed: joyy = 0
 08D2   50E0     '
 08D2   50E0             if x2 > xvar1 then x = x + SPEED else if x2 < nxvar1 t
hen x
                 = x - SPEED
 0905   50E0             if y2 > yvar1 then y = y + SPEED else if y2 < nyvar1 t
hen y
                 = y - SPEED
 0938   50E0     '
 0938   50E0             if x2 > xvar2 then x = x + SPEED * 3 else if x2 < nxva
r2 the
                 n x = x - SPEED * 3
 0975   50E0             if y2 > yvar2 then y = y + SPEED * 3 else if y2 < nyva
r2 the
                 n y = y - SPEED * 3
 09B2   50E0     '
 09B2   50E0         return
 09B5   50E0     '
 09B5   50E0     8 '
 09B6   50E0     '
 09B6   50E0     '    If first row of choices is selected then distinguish the
 09B6   50E0     '    function.
 09B6   50E0     '
 09B6   50E0     '    locate 2,1
 09B6   50E0     '    print "HS:  x = y = ";x;y;
 09B6   50E0     '
 09B6   50E0         if quadnew = 1 and button = 1 then gosub 230:  return ' Se
                 lect new cursor character
 09DE   50E0         if quadnew = 2 and button = 1 then gosub 250: return  ' di
                 splay symbol table
 0A06   50E0         if quadnew = 3 and button = 1 then gosub 240: return' load
                  new symbol table
 0A2E   50E0         if quadnew = 4 and button = 1 then stateflag = 3: gosub 40
                 0: return  ' overlay
 0A5D   50E0         if quadnew = 5 and button = 1 then stateflag = 2: gosub 40
                 0: return ' Erase mode
 0A8C   50E0     '
 0A8C   50E0     '    If second row of choices is selected then determine the
 0A8C   50E0     '    function.
 0A8C   50E0     '
 0A8C   50E0     '    locate 3,1
 0A8C   50E0     '    print "HS:  x = y = ";x;y;
 0A8C   50E0     '
 0A8C   50E0         if quadnew = 6 and button = 1 then stateflag = 10 : messag
                 e$ = "Point to color to change and press RED"
 0ABC   50E4         if quadnew = 6 and button = 1 then gosub 800: return  ' ch
                 ange color
 0AE4   50E4         if quadnew = 7 and button = 1 then gosub 700: return  ' re
                 set
 0B0C   50E4         if quadnew = 8 and button = 1 then gosub 950: return ' tex
                 t
 0B34   50E4         if quadnew = 9  and button = 1 then gosub 1800: return ' f
                 ile
 0B5C   50E4         if quadnew = 10 and button = 1 then stateflag = 4: gosub 4
                 00  ' associate
 0B88   50E4     '
```

```
Offset  Data    Source Line
 0B88   50E4          return
 0B8B   50E4    '
 0B8B   50E4    9 '
 0B8C   50E4    '
 0B8C   50E4    '    Clear extra graphics area
 0B8C   50E4    '
 0B8C   50E4          yput = 178
 0B93   50E6          xput = 310
 0B9A   50E8          call place(xput, yput, storage, blueoff)
 0BB3   50E8          yput = 188
 0BBA   50E8          call place(xput, yput, storage, blueoff)
 0BD3   50E8    '
 0BD3   50E8          return
 0BD6   50E8    '
 0BD6   50E8    10 '
 0BD7   50E8    '
 0BD7   50E8    '    xor the screen  with the cursor character.
 0BD7   50E8    '
 0BD7   50E8          call overlay( xold, yold, storage, osoff)
 0BF0   50E8          call overlay( x, y, storage, soff)
 0C09   50E8    '
 0C09   50E8          return
 0C0C   50E8    '
 0C0C   50E8    20 '
 0C0D   50E8    '
 0C0D   50E8    '    pset the current cursor character
 0C0D   50E8    '
 0C0D   50E8          call place (x, y, storage, soff)
 0C26   50E8          call overlay(x, y, storage, soff)
 0C3F   50E8    '
 0C3F   50E8          return
 0C42   50E8    30 '
 0C43   50E8    '
 0C43   50E8    '    erase the current cursor area
 0C43   50E8    '
 0C43   50E8          locate 24,1
 0C50   50E8          call place(x, y, storage, blueoff)
 0C69   50E8          call overlay(x, y, storage, soff)
 0C82   50E8    '
 0C82   50E8          return
 0C85   50E8    '
 0C85   50E8    40 '
 0C86   50E8    '
 0C86   50E8    '          xor the symbol
 0C86   50E8    '
 0C86   50E8          call overlay(x, y, storage, soff)
 0C9F   50E8          for i = 0 to 1000: j = 0: next i
 0CBC   50E8    '
 0CBC   50E8          return
 0CBF   50E8    '
 0CBF   50E8    41 '
 0CC0   50E8    '
 0CC0   50E8    '    text handler
 0CC0   50E8    '
```

```
Offset  Data    Source Line          IBM Personal Computer BASIC Compiler V1.00

 0CC0   50E8         put (xold,yold), boxsel, xor
 0CD5   50E8         put (x, y), boxsel, xor
 0CEA   50E8    '
 0CEA   50E8         return
 0CED   50E8    '
 0CED   50E8    42 '
 0CEE   50E8    '
 0CEE   50E8    '    text putter
 0CEE   50E8    '
 0CEE   50E8         put (xold, yold), boxsel, pset
 0D02   50E8         put (xold, yold), boxsel, xor
 0D17   50E8    '
 0D17   50E8         return
 0D1A   50E8    '
 0D1A   50E8    55 '
 0D1B   50E8    '
 0D1B   50E8    '    Handle the few normal keys we handle
 0D1B   50E8    '
 0D1B   50E8         if k$ = "" then return
 0D2C   50E8         key2 = asc(k$)
 0D36   50E8         if key2 = 43 then button = 1 else beep
 0D4F   50E8         for q = 0 to 100: d = 0:next
 0D6B   50EC         return
 0D6E   50EC    '
 0D6E   50EC    60 '
 0D6F   50EC    '
 0D6F   50EC    '    handle extended keys
 0D6F   50EC    '
 0D6F   50EC    '
 0D6F   50EC         k$ = right$(k$,1)
 0D7F   50EC         key2 = asc(k$)
 0D89   50EC         if key2 > 58 and key2 < 69 then key2 = key2 - 58 else retu
                rn
 0DB9   50EC    '
 0DB9   50EC    '    do not allow a switch of states during a function.
 0DB9   50EC    '
 0DB9   50EC         if stateflag <> 1 and stateflag <> 3 then return
 0DDC   50EC    '
 0DDC   50EC         on key2 gosub  300, 300, 15900, 1100,   1900 ,2000,  1700,
                 16000,            17000,  29000
 0DF8   50EC    '
 0DF8   50EC    '                       +     -   end  line  circle  box   screen
                 gen symbol rotate  confirm
 0DF8   50EC    '                     cursor speed
 0DF8   50EC    '
 0DF8   50EC         return
 0DFB   50EC    '
 0DFB   50EC    70 '
 0DFC   50EC    '
 0DFC   50EC    '    Joystick has been pushed
 0DFC   50EC    '
 0DFC   50EC         button = 1
 0E03   50EC         for q = 0 to 100: d = 0:next
 0E1F   50EC         return
```

```
                                                                PAGE  10
                                                                03-16-84
                                                                14:01:33
Offset  Data    Source Line          IBM Personal Computer BASIC Compiler V1.00

 0E22   50EC    '
 0E22   50EC    100 '
 0E23   50EC    '
 0E23   50EC    '    F1 will speed up the cursor each time it is pressed.
 0E23   50EC    '
 0E23   50EC         speed = speed + 1
 0E2B   50EC         return
 0E2E   50EC    '
 0E2E   50EC    200 '
 0E2F   50EC    '
 0E2F   50EC    '    F2 will slow down the cursor each time it is pressed.
 0E2F   50EC    '
 0E2F   50EC         speed = speed - 1
 0E37   50EC         if speed < 1 then speed = 1
 0E49   50EC    '
 0E49   50EC         return
 0E4C   50EC    '
 0E4C   50EC    230 '
 0E4D   50EC    '
 0E4D   50EC    '    Prompt for entry number
 0E4D   50EC    '
 0E4D   50EC         gosub 14800
 0E52   50EC    '
 0E52   50EC    233 '
 0E53   50EC    '
 0E53   50EC    '    This will allow entry from Pf key 2
 0E53   50EC    '
 0E53   50EC         switch1 = 1
 0E5A   50EE         while switch1
 0E66   50EE              numdisp$ = str$(numbsymb - 1)
 0E74   50F2              a = 4 - len(numdisp$)
 0E82   50F4              numdisp$ = string$(a, " ") + numdisp$
 0E97   50F4              message$ = "Enter table entry 0 -" + numdisp$  + " [
]"
 0EAB   50F4              inputcol = 28
 0EB2   50F6              gosub 29010
 0EB7   50F6              input;"", num$
 0EC7   50FA              if val(num$) > numbsymb - 1 or val(num$) < 0 then gosu
b 231
                    else gosub 232
 0F0A   50FA         wend
 0F0E   50FA    '
 0F0E   50FA    '    Clean-up and return
 0F0E   50FA    '
 0F0E   50FA         stateflag = 1
 0F15   50FA         gosub 502
 0F1A   50FA         return
 0F1D   50FA    '
 0F1D   50FA    231 '
 0F1E   50FA    '
 0F1E   50FA    '    bad entry of symbol number
 0F1E   50FA    '
 0F1E   50FA         numdisp$ = num$
 0F27   50FA         a = 4 - len(numdisp$)
 0F35   50FA         numdisp$ = string$(a, " ") + numdisp$
 0F4A   50FA         message$ = "Entry:" + numdisp$ + "  was invalid.  Retry? Y
```

```
Offset  Data    Source Line          IBM Personal Computer BASIC Compiler V1.00

                /N [ ]"
 0F5E   50FA         inputcol = 38
 0F65   50FA         gosub 29010
 0F6A   50FA         input;"",y$
 0F7A   50FE         if y$ <> "Y" and y$ <> "y" then switch1 = 0
 0FA4   50FE         return
 0FA7   50FE    '
 0FA7   50FE    232'
 0FA8   50FE    '
 0FA8   50FE    '    Good symbol number entered, so  get the symbol number.
 0FA8   50FE    '
 0FA8   50FE         call overlay( x, y, storage ,soff)    ' clean up the old c
                ursor
 0FC1   50FE         symnum =  val(num$)
 0FCE   5100         gosub 14000    ' get the new symbol
 0FD3   5100    '
 0FD3   5100    '    reset the cursor
 0FD3   5100    '
 0FD3   5100         gosub 2
 0FD8   5100         call overlay( x, y, storage, soff)
 0FF1   5100    '
 0FF1   5100    '    Re-initialize the old offsets to reflect the new character
                .
 0FF1   5100    '
 0FF1   5100         ochrw = chrw
 0FF8   5100         ochrh = chrh
 0FFF   5100    '
 0FFF   5100         oldoff = soff
 1006   5100    '
 1006   5100         switch1 = 0
 100D   5100         return
 1010   5100    '
 1010   5100    240 '
 1011   5100    '
 1011   5100    '  This will be the driver for the change symbol table code
 1011   5100    '
 1011   5100    gosub 14800 ' cleanup the arrow
 1016   5100    '
 1016   5100    switch1 = 1   ' initialize the loop
 101D   5100    '
 101D   5100    while switch1
 1029   5100         nofilef1 = 0
 1030   5100         message$ = "Symbol table? [          ]"
 1039   5100         inputcol = 16
 1040   5100         gosub 29010   ' set up for prompt input
 1045   5100         input;"",symbolt$
 1055   5104         if len(symbolt$) > 10 then gosub 241 else gosub 242
 1070   5104    wend
 1074   5104    '
 1074   5104    '    Clean-up and head home
 1074   5104    '
 1074   5104    stateflag = 1
 107B   5104    gosub 502
 1080   5104    return
```

```
Offset  Data    Source Line          IBM Personal Computer BASIC Compiler V1.00

 1083   5104    '
 1083   5104    241  '
 1084   5104    '
 1084   5104    '     This code re-prompts if table name too long.
 1084   5104    '
 1084   5104         nofilefl = 1
 108B   5104         message$ = "Name too long, try again? Y/N [ ]"
 1094   5104         inputcol = 32
 109B   5104         gosub 29010 ' prompt
 10A0   5104         input;"",y$
 10B0   5104         if y$ <> "Y" and y$ <> "y" then switch1 = 0
 10DA   5104         return
 10DD   5104    '
 10DD   5104    '  This subroutine will check for existence of the table.
 10DD   5104    '
 10DD   5104    '
 10DD   5104    242 '
 10DE   5104         on error goto 243
 10E5   5104         file$ = symbolt$ + ".SYM"
 10F3   5108         open file$ for input as #3
 1104   5108         on error goto 31300
 110B   5108         if (switch1 = 1) and (nofilefl = 0) then gosub 244
 1130   5108         return
 1133   5108    '
 1133   5108    243   '
 1134   5108    '
 1134   5108    '    This code handles disk errors for symbol table
 1134   5108    '
 1134   5108         nofilefl = 1
 113B   5108         message$ = "Table not found, try again? Y/N [ ]"
 1144   5108         inputcol = 34
 114B   5108         gosub 29010 ' handle prompt
 1150   5108         input;"",y$
 1160   5108         if y$ <> "Y" and y$ <> "y" then switch1 = 0
 118A   5108         resume next
 118E   5108    '
 118E   5108    244  '
 118F   5108    '
 118F   5108    '     This code loads the symbol table
 118F   5108    '
 118F   5108         close #3
 1196   5108    '
 1196   5108    '     Clean up the old cursor and load the first item initially
 1196   5108    '
 1196   5108         call overlay( x, y, storage, soff)
 11AF   5108         def seg = storage
 11B7   5108         bload file$,symbtoff
 11C2   5108         switch1 = 0
 11C9   5108    '
 11C9   5108    '     numbsymb = number of entries in the symbol table
 11C9   5108    '
 11C9   5108         numbsymb = peek(symbtoff) + 256 * peek(symbtoff + 1)
 11FF   5108         def seg
 1203   5108    '
```

```
Offset  Data    Source Line          IBM Personal Computer BASIC Compiler V1.00

 1203   5108    '    offset2 must be re-calculated
 1203   5108    '
 1203   5108         offset2 = numbsymb * 2 + 2
 1210   5108    '
 1210   5108         symnum = 0
 1217   5108         gosub 14000 'get new character
 121C   5108    '
 121C   5108    '    Check the positioning and put up the new symbol
 121C   5108    '
 121C   5108         gosub 2
 1221   5108         stateflag = 1
 1228   5108         gosub 502
 122D   5108    '
 122D   5108         call overlay( x, y, storage, soff)
 1246   5108    '
 1246   5108         return
 1249   5108    '
 1249   5108    250 '
 124A   5108    '
 124A   5108    '    Display symbol table function
 124A   5108    '
 124A   5108    '
 124A   5108    '    Clean-up the degrees.
 124A   5108    '
 124A   5108         gosub 14800
 124F   5108    '
 124F   5108    '    Push the offset
 124F   5108    '
 124F   5108         toff = soff
 1256   510A         ochrh = chrh
 125D   510A         ochrw = chrw
 1264   510A    '
 1264   510A    '    save screen away
 1264   510A    '
 1264   510A         def seg = &hb800
 126B   510A         bsave "screen.scr",0,&h4000
 1277   510A         def seg
 127B   510A         startnbr = 0
 1282   510A         gosub 255
 1287   510A    '
 1287   510A         switch1 = 1
 128E   510A         while switch1
 129A   510A    251       k$ = inkey$
 12A3   510A              if k$ <> "" then k$ = right$(k$,1) else 251
 12C7   510A              key2 = asc(k$)
 12D1   510A              if key2 > 58 and key2 < 68 then gosub 252
 12F6   510A         wend
 12FA   510A    '
 12FA   510A    '    re-display screen, cleanup pointers and return
 12FA   510A    '
 12FA   510A         cls
 12FE   510A         def seg = &hb800
 1305   510A         bload "screen.scr",0
 130E   510A         def seg
```

```
Offset  Data    Source Line          IBM Personal Computer BASIC Compiler V1.00

 1312   510A    '
 1312   510A    '    Pop stack to get the good offsets
 1312   510A    '    re-initialize the prompt and switches
 1312   510A    '
 1312   510A         soff = toff
 1319   510A         oldoff = soff
 1320   510A         chrh = ochrh
 1327   510A         chrw = ochrw
 132E   510A    '
 132E   510A         stateflag = 1
 1335   510A         gosub 502
 133A   510A    '
 133A   510A         return
 133D   510A    '
 133D   510A    252 '
 133E   510A    '              f7                  f8          f2
 133E   510A    '    If key2 = 65 then previous, 66 = next, 60 = quit
 133E   510A    '
 133E   510A         if key2 = 60 then switch1 = 0: return
 1353   510A         locate 24,1
 1360   510A         if key2 = 66 then startnbr = startnbr + 15
 1375   510A         if key2 = 65 then startnbr = startnbr - 15
 138A   510A         if startnbr > numbsymb - 1 then startnbr = numbsymb - 1: p
                rint "Press F2 to quit";: return
 13AC   510A         if startnbr < 0 then startnbr = 0
 13BE   510A         key2 = 0
 13C5   510A         gosub 255
 13CA   510A         return
 13CD   510A    '
 13CD   510A    255 '
 13CE   510A    '
 13CE   510A    '    Display 14 members of a symbol table
 13CE   510A    '
 13CE   510A    '
 13CE   510A         if startnbr < 0 then return
 13DC   510A    '
 13DC   510A         cls
 13E0   510A    '
 13E0   510A         locate 1,12
 13ED   510A         print "Symbol Table Display";
 13F5   510A         endnbr = startnbr + 14
 13FF   510C         if endnbr > numbsymb - 1 then endnbr = numbsymb - 1
 1416   510C         hor = 1
 141D   510E         ver = 5
 1424   5110         for symnum = startnbr to endnbr
 1431   5112    '
 1431   5112              gosub 14000 ' get symbol
 1436   5112    '
 1436   5112              if symnum < 100 then locate ver, hor + 3 else locate v
er, ho
                r + 2
 1467   5112              print symnum;
 1470   5112              dispx = 8 * hor + 32 - chrw/2  - 1
 1499   5114              dispy = 8 * ver + 24 - chrh/2  - 1
 14C2   5116              call overlay (dispx, dispy, storage, soff)
```

```
Offset  Data    Source Line
 14DB   5116             hor = hor + 8
 14E5   5116             if hor > 33 then hor = 1: ver = ver + 7
 1501   5116    '
 1501   5116         next symnum
 1515   5116    '
 1515   5116    return
 1518   5116    '
 1518   5116    300 '
 1519   5116    '
 1519   5116    '  This will be the end function.
 1519   5116    '
 1519   5116         gosub 32000
 151E   5116         return
 1521   5116    '
 1521   5116    400 '
 1522   5116    '
 1522   5116    '    draw/erase
 1522   5116    '
 1522   5116         call overlay (x, y, storage, soff)
 153B   5116    '
 153B   5116    '    Set up for the box for erasure or old character if draw
 153B   5116    '
 153B   5116         if stateflag = 4 then soff = smalloff: chrh = 3: chrw = 30
                : colorflg = 1: ymax = 176
 1569   5116         if stateflag = 2 then soff = boxoff: chrh = 12: chrw = 9
 1589   5116         if stateflag = 1 or stateflag = 3 then soff = oldoff: chrh
                 = ochrh: chrw = ochrw: ymax = 199: colorflg = 0
 15CC   5116    '
 15CC   5116         x = 160 - chrw / 2
 15E3   5116         y = 88 - chrh / 2
 15FA   5116    '
 15FA   5116         call overlay (x, y, storage, soff)      ' reset the cursor
 1613   5116         gosub 9                                 ' clear the extra graph
ics w/o LF
 1618   5116    '
 1618   5116         gosub 502       ' Prompt user with state
 161D   5116    '
 161D   5116         return
 1620   5116    '
 1620   5116    502 '
 1621   5116    '
 1621   5116    '    Prompt user with current state.
 1621   5116    '
 1621   5116         locate 23,1
 162E   5116    '
 162E   5116         print string$(39," ");
 163C   5116    '
 163C   5116         locate 23,1
 1649   5116    '
 1649   5116         if stateflag = 1 then print "Position symbol and press RED
                 to place";
 165C   5116    '
 165C   5116         if stateflag = 2 then print "Position the box and press RE
                D to erase";
 166F   5116    '
```

```
Offset  Data    Source Line
 166F   5116         if stateflag = 3 then print "Position symbol and press RED
                     to Xor";
 1682   5116    '
 1682   5116         if stateflag = 4 then print "Point to upper left of pictur
                    e and RED";
 1695   5116    '
 1695   5116         if stateflag = 5 then print "Enclose picture and press RED
                    ";
 16A8   5116    '
 16A8   5116         if stateflag = 6 then print "Center the value area and pre
                    ss RED";
 16BB   5116    '
 16BB   5116         if stateflag = 7 then print "Position the text and press R
                    ED";
 16CE   5116    '
 16CE   5116         if stateflag = 8 then print "Expand the box to the right a
                    nd down";
 16E1   5116    '
 16E1   5116         if stateflag = 13 then print "Point to center of the circl
                    e & RED";
 16F4   5116    '
 16F4   5116         if stateflag = 14 then print "Red = expand, F9 = chg col,
                    F10 = Halt";
 1707   5116    '
 1707   5116         if stateflag = 16 then print "Point to the first corner th
                    en RED";
 171A   5116    '
 171A   5116         if stateflag = 17 then print "Point to the opposite corner
                     then RED";
 172D   5116    '
 172D   5116         if stateflag = 20 then print "Point to the start and press
                     RED";
 1740   5116    '
 1740   5116         if stateflag = 21 then print "Point to the end and press R
                    ED";
 1753   5116    '
 1753   5116         gosub 1        ' display prompts
 1758   5116         entryflg = 0
 175F   5116         selflag = 0
 1766   5116         initflag = 0
 176D   5116         return
 1770   5116    '
 1770   5116    700 '
 1771   5116    '
 1771   5116    '    reset the screen
 1771   5116    '
 1771   5116         cls
 1775   5116         call overlay(x, y, storage, soff)
 178E   5116    '
 178E   5116    '    set default to be place
 178E   5116    '
 178E   5116         stateflag = 1
 1795   5116    '
 1795   5116    '    Reset the associate DOT file.
```

```
Offset  Data    Source Line          IBM Personal Computer BASIC Compiler V1.00

 1795   5116    '
 1795   5116         ddtrecno = 0
 179C   5116    '
 179C   5116         gosub 14800
 17A1   5116         gosub 502
 17A6   5116    '
 17A6   5116         return
 17A9   5116    '
 17A9   5116    800 '
 17AA   5116    '
 17AA   5116    '    This will be the change cursor character color routine
 17AA   5116    '    Prompt for color to be changed
 17AA   5116    '
 17AA   5116         gosub 14800
 17AF   5116    '
 17AF   5116    810 '
 17B0   5116    '
 17B0   5116    '    Some entries into color donot need the cursor cleaned up
 17B0   5116    '
 17B0   5116         if stateflag < 10 then stateflag = 10: message$ = "Point t
                o color to change and press RED"
 17CB   5116    '
 17CB   5116         colorflg = 1
 17D2   5116         gosub 29000 ' clean up prompt area
 17D7   5116         xput = 0
 17DE   5116         yput = 185
 17E5   5116         for i = 0 to 3
 17EB   5116              xput = i * 80 + 20
 17F9   5116              if i = 0 then call place(xput, yput, storage, blueoff)
 181D   5116              if i = 1 then call place(xput, yput, storage, greenoff
)
 1841   5116              if i = 2 then call place(xput, yput, storage, redoff)
 1865   5116              if i = 3 then call place(xput, yput, storage, whiteoff
)
 1889   5116              call overlay(xput, yput, storage, boxoff)
 18A2   5116         next i
 18B4   5116    '
 18B4   5116         locate 23,1
 18C1   5116         print message$
 18C9   5116    '
 18C9   5116         x = 1
 18D0   5116         y = 188
 18D7   5116         chrh = 5
 18DE   5116         chrw = 5
 18E5   5116         soff = smalloff
 18EC   5116         call overlay(x, y, storage, soff)
 1905   5116    '
 1905   5116         return
 1908   5116    '
 1908   5116    801 '
 1909   5116    '
 1909   5116    '    get first color
 1909   5116    '
 1909   5116         original = int(abs(x - 1) / 80)
 1928   5118         stateflag = 11
 192F   5118         locate 23,1
```

```
Offset  Data    Source Line          IBM Personal Computer BASIC Compiler V1.00

 193C   5118         print "Point to change to color and press RED";
 1944   5118    '
 1944   5118         return
 1947   5118    '
 1947   5118    802 '
 1948   5118    '
 1948   5118    '    get second color
 1948   5118    '
 1948   5118         newcol = int(abs(x - 1) / 80)
 1967   511A         gosub 14800
 196C   511A         call overlay(x, y, storage, soff)
 1985   511A         gosub 803
 198A   511A    '
 198A   511A    '    set draw to be the default mode
 198A   511A    '
 198A   511A         stateflag = 1
 1991   511A         gosub 502   ' check for draw, erase
 1996   511A         colorflg = 0
 199D   511A         return
 19A0   511A    '
 19A0   511A    803 '
 19A1   511A    '
 19A1   511A    '    Change color
 19A1   511A    '
 19A1   511A         origo = coloro(original)
 19AF   511C         newo = coloro(newcol) / 2   ' offset into the new color mas
                k table.
 19C7   511E    '
 19C7   511E    '    Clean-up the old cursor
 19C7   511E    '
 19C7   511E         call overlay(x, y, storage,soff)
 19E0   511E    '
 19E0   511E         gosub 804 'change the bits in storage
 19E5   511E    '
 19E5   511E    '    Put up new cursor
 19E5   511E    '
 19E5   511E         call overlay(x, y, storage, soff)
 19FE   511E    '
 19FE   511E         return
 1A01   511E    '
 1A01   511E    804 '
 1A02   511E    '
 1A02   511E    '    Change color in storage and set switch1 = 1 to end
 1A02   511E    '    the main loop.
 1A02   511E    '
 1A02   511E         def seg = storage   ' point to symbol table area
 1A0A   511E    '
 1A0A   511E         xvalch! = peek(soff) + 256 * peek(soff+1)
 1A43   5122         yvalch! = peek(soff+2) + 256 * peek(soff+3)
 1A80   5126         xvalch! = xvalch! / 2
 1A8D   5126    '
 1A8D   5126    '    Get the number of bytes of date in the cursor
 1A8D   5126    '
 1A8D   5126         bytes =  int(( xvalch! * 2.0  + 7.0 ) / 8.0 ) * yvalch!
```

```
Offset  Data    Source Line          IBM Personal Computer BASIC Compiler V1.00

 1AAC   5128    '
 1AAC   5128    '    add 4 to bytes to look past the x and y sizes
 1AAC   5128    '    3 bytes to loops because of 0th element
 1AAC   5128    '
 1AAC   5128         init = soff + 4
 1AB7   512A         loops = bytes + soff + 3
 1AC5   512C    '
 1AC5   512C         for j = init to loops
 1AD2   512E    '
 1AD2   512E    '          a is my stepper through the masks
 1AD2   512E    '
 1AD2   512E              a = origo      ' initialize stepper to first offset in
 mask
 1AD9   512E              cvalue = peek(j)
 1AEC   5130    '
 1AEC   5130    '         Step through the 4 bit pairs / byte
 1AEC   5130    '
 1AEC   5130              for i = 0 to 3
 1AF2   5130    '
 1AF2   5130                   one = chcolor(a)       ' mask to identify change c
haract
                er
 1B00   5132                   two = chcolor(a + 1)   'test mask
 1B0E   5134                   three = 255 - one      ' cleanup mask for insertin
g char
                acter
 1B19   5136                   four = newcolor( newo + i)
 1B2B   5138    '
 1B2B   5138    '    I have half a mind not to document this, but......
 1B2B   5138    '    Compare the 2 bit value to the highest value it could have
                .
 1B2B   5138    '    If it equals the test mask then it is the correct color.
 1B2B   5138    '    Then Clean out the found bit values and or in the new
 1B2B   5138    '    mask.
 1B2B   5138    '
 1B2B   5138              if (cvalue and one) = two then cvalue = cvalue an
d thre
                e or four
 1B4B   5138    '
 1B4B   5138              a = a + 2        ' increment the stepper to the ne
xt two
                 bit pair
 1B54   5138    '
 1B54   5138         next i
 1B63   5138    '
 1B63   5138         poke j,cvalue         ' replace the updated value
 1B73   5138    '
 1B73   5138       next j
 1B87   5138    '
 1B87   5138       def seg
 1B8B   5138    '
 1B8B   5138    '    End main loop by setting good switch
 1B8B   5138    '
 1B8B   5138       switch1 = 0
 1B92   5138       return
 1B95   5138    '
 1B95   5138    900 '
 1B96   5138    '
 1B96   5138    '    Associate a symbol with a process variable.
```

```
Offset  Data    Source Line          IBM Personal Computer BASIC Compiler V1.00

 1B96   5138    '
 1B96   5138    '
 1B96   5138    '    First clean-up the box!!!!
 1B96   5138    '    Save the box size
 1B96   5138    '
 1B96   5138         line (xfirst, yfirst) - (x, y), 0, b
 1BAF   513C         put (xfirst,yfirst), boxsel, pset
 1BC3   513C         sizex = x - xfirst + 1
 1BD1   513E         sizey = y - yfirst + 1
 1BDF   5140    '
 1BDF   5140         if ddtrecno > 30 then locate 25,1: print "Variable file fu
                    ll";: return
 1C02   5140         switch1 = 1
 1C09   5140         while switch1
 1C15   5140              message$ = "Name of the variable:  [        ]"
 1C1E   5140              inputcol = 25
 1C25   5140              gosub 29010
 1C2A   5140              input; "",var$
 1C3A   5144              a = len (var$): var$ = var$ + string$(8 - a, " ")
 1C5A   5144              gosub 901
 1C5F   5144              if nofilefl = 0 then gosub 906
 1C6F   5144              if nofilefl = 1 then gosub 903 else gosub 904
 1C87   5144    '
 1C87   5144         wend
 1C8B   5144    '
 1C8B   5144    '    Variable name was found then get pointing for value
 1C8B   5144    '    else return.
 1C8B   5144    '
 1C8B   5144         if nofilefl = 0 then chrh = ochrh: chrw = ochrw: soff = ol
                    doff: x = 160 - chrh / 2: y = 88 - chrh / 2: call overlay(x, y,
                     storage, soff)
 1CF2   5144         if nofilefl = 0 then ymax = 199: stateflag = 1: gosub 502:
                     colorflg = 0
 1D17   5144         if nofilefl = 1 then soff = smalloff: call overlay(x, y, s
                    torage, soff): gosub 502
 1D47   5144         if nofilefl = 1 then chrh = 5: chrw = 20
 1D60   5144    '
 1D60   5144         return
 1D63   5144    '
 1D63   5144    901 '
 1D64   5144    '
 1D64   5144    '    Check for existence of variable name
 1D64   5144    '
 1D64   5144         nofilefl = 0
 1D6B   5144         on error goto 930
 1D72   5144         open "varfile.tab" as #2 len = 64
 1D85   5144         on error goto 31300
 1D8C   5144    '
 1D8C   5144    '    If no file then return
 1D8C   5144    '
 1D8C   5144         if nofilefl <> 0 then nofilefl = 0: return
 1DA1   5144    '
 1DA1   5144         field #2, 2 as typeup$, 8 as varup$, 2 as f$, 2 as entry$,
                     50 as fill$
```

```
Offset  Data    Source Line          IBM Personal Computer BASIC Compiler V1.00

 1DCF   5158    '
 1DCF   5158         get #2, 1
 1DD9   5158         num = cvi(typeup$)
 1DE4   515A         for i = 2 to num + 1
 1DF2   515C              get #2, i
 1DFD   515C              if var$ = varup$ then goto 902
 1E0F   515C         next i
 1E20   515C    '
 1E20   515C         nofilef1 = 0
 1E27   515C    '
 1E27   515C         close #2
 1E2E   515C         return
 1E31   515C    '
 1E31   515C    902  '
 1E32   515C    '
 1E32   515C    '    Variable was found, so set the flag.
 1E32   515C    '
 1E32   515C         entrynum = cvi(entry$)
 1E3D   515E         typeentry = 1
 1E44   5160         nofilef1 = 1
 1E4B   5160         close #2
 1E52   5160    '
 1E52   5160         return
 1E55   5160    '
 1E55   5160    903 '
 1E56   5160    '
 1E56   5160    '    good variable name entered
 1E56   5160    '
 1E56   5160         ddtrecno = ddtrecno + 1
 1E5E   5160         entry(ddtrecno) = entrynum
 1E6D   5160         typeddt(ddtrecno) = typeentry
 1E7C   5160    '
 1E7C   5160         ulxchar(ddtrecno) = xfirst
 1E8B   5160         ulychar(ddtrecno) = yfirst
 1E9A   5160         boxsizex(ddtrecno) = sizex * 2
 1EAB   5160         boxsizey(ddtrecno) = sizey
 1EBA   5160    '
 1EBA   5160    '    Alarm is a variable for future different alarms.
 1EBA   5160    '
 1EBA   5160         alarm(ddtrecno) = 1
 1EC7   5160    '
 1EC7   5160         switch1 = 0
 1ECE   5160         stateflag = 6
 1ED5   5160         return
 1ED8   5160    '
 1ED8   5160    904 '
 1ED9   5160    '
 1ED9   5160    '   This code handles invalid variable name
 1ED9   5160    '
 1ED9   5160         message$ = "Variable not found try again? Y/N [ ]"
 1EE2   5160         inputcol = 36
 1EE9   5160         gosub 29010 ' handle prompt
 1EEE   5160         input;"",y$
 1EFE   5160         if y$ <> "Y" and y$ <> "y" then switch1 = 0
```

```
Offset  Data    Source Line          IBM Personal Computer BASIC Compiler V1.00

 1F28   5160    '
 1F28   5160         return
 1F2B   5160    '
 1F2B   5160    906 '
 1F2C   5160    '
 1F2C   5160    '    Check screen file.
 1F2C   5160    '
 1F2C   5160         on error goto 940
 1F33   5160    '
 1F33   5160         open "scrfile.tab" as #2 len = 72
 1F46   5160    '
 1F46   5160         field #2, 2 as typeup$, 70 as entry$
 1F5C   5160         on error goto 31300
 1F63   5160    '
 1F63   5160    '    If no file then return
 1F63   5160    '
 1F63   5160         if nofilef1 <> 0 then nofilef1 = 0: return
 1F78   5160         get #2, 1
 1F82   5160         num = cvi(typeup$)
 1F8D   5160         field #2, 2 as entry$, 8 as varup$, 62 as fill$
 1FAB   5160    '
 1FAB   5160         for i = 2 to num + 1
 1FB9   5162              get #2, i
 1FC4   5162              if var$ = varup$ then goto 908
 1FD6   5162         next i
 1FE7   5162    '
 1FE7   5162         nofilef1 = 0
 1FEE   5162    '
 1FEE   5162         close #2
 1FF5   5162    '
 1FF5   5162         return
 1FF8   5162    '
 1FF8   5162    908 '
 1FF9   5162    '
 1FF9   5162    '    Screen was found, so set the flag.
 1FF9   5162    '
 1FF9   5162         entrynum = cvi(entry$)
 2004   5162         typeentry = 2
 200B   5162         nofilef1 = 1
 2012   5162         close #2
 2019   5162    '
 2019   5162         return
 201C   5162    '
 201C   5162    910 '
 201D   5162    '
 201D   5162    '    This code will store away the value location.
 201D   5162    '
 201D   5162         xput = x
 2024   5162         yput = y - 3
 202E   5162         if xput < 0 then xput = 0
 2040   5162         if yput < 0 then yput = 0
 2052   5162    '
 2052   5162         ulxvalue(ddtrecno) = xput
 2061   5162         ulyvalue(ddtrecno) = yput
```

```
Offset  Data    Source Line          IBM Personal Computer BASIC Compiler V1.00

 2070   5162    '
 2070   5162    '     Clean-up the dot
 2070   5162    '
 2070   5162          call overlay(x, y, storage, soff)
 2089   5162          chrh = ochrh
 2090   5162          chrw = ochrw
 2097   5162          x = 160 - chrh / 2
 20AE   5162          y = 88 - chrw / 2
 20C5   5162          colorflg = 0
 20CC   5162          soff = oldoff
 20D3   5162          ymax = 199
 20DA   5162    '
 20DA   5162    '     Put the cursor back up.
 20DA   5162    '
 20DA   5162          call overlay(x, y, storage, soff)
 20F3   5162          stateflag = 1
 20FA   5162          gosub 502
 20FF   5162    '
 20FF   5162          return
 2102   5162    '
 2102   5162    930 '
 2103   5162    '
 2103   5162    '     No variable table found
 2103   5162    '
 2103   5162          gosub 29000
 2108   5162          close #2
 210F   5162          locate 23,1
 211C   5162          print "Variable table file was not found.";
 2124   5162          locate 25,1
 2131   5162          print "Run <Variable Table Generator>";
 2139   5162    '
 2139   5162          nofilefl = 1
 2140   5162          for i = 0 to 1000: j = 0: next i
 215D   5162    '
 215D   5162          resume next
 2161   5162    '
 2161   5162    940 '
 2162   5162    '
 2162   5162    '     No Screen table found
 2162   5162    '
 2162   5162          gosub 29000
 2167   5162          close #2
 216E   5162          locate 23,1
 217B   5162          print "Alarm / Action table file was not found";
 2183   5162          locate 25,1
 2190   5162          print "Run <Alarm / Action Definition>";
 2198   5162    '
 2198   5162          nofilefl = 1
 219F   5162          for i = 0 to 1000: j = 0: next i
 21BC   5162    '
 21BC   5162          resume next
 21C0   5162    '
 21C0   5162    950 '
 21C1   5162    '
```

```
                                                                    PAGE  24
                                                                    03-16-84
                                                                    14:01:33
Offset  Data    Source Line          IBM Personal Computer BASIC Compiler V1.00

 21C1   5162    '     Text to the screen in small format
 21C1   5162    '
 21C1   5162          gosub 14800 ' clean up the arrow
 21C6   5162    '
 21C6   5162          oldoff = soff
 21CD   5162    '
 21CD   5162          gosub 29000 ' clear prompt area
 21D2   5162          switch1 = 1
 21D9   5162          m = 1
 21E0   5164          textchrw = 0
 21E7   5166          locate 23,1
 21F4   5166          print "Enter alphanumerics, then  ---";
 21FC   5166          locate 24,1
 2209   5166          print "["; string$(37," "); "]";
 2221   5166          while switch1
 222D   5166               tkey2 = 0
 2234   5168               key2 = 0
 223B   5168     953       k$ = inkey$
 2244   5168               if k$ = "" then 953
 224F   5168               if len(k$) > 1 then key2 = 0
 2264   5168               if asc(k$) = 8 then key2 = -8
 2279   5168               if asc(k$) = 32 then key2 = 40
 228E   5168               if asc(k$) < 123 and asc(k$) > 96 then tkey2 = asc(k$)
 : key
                     2 = asc(k$) - 86
 22CC   5168               if asc(k$) < 91 and asc(k$) > 64 then tkey2 = asc(k$)
: key2
                      = asc(k$) - 54
 230A   5168               if asc(k$) < 58 and asc(k$) > 47 then tkey2 = asc(k$)
: key2
                      = asc(k$) - 47
 2348   5168               if asc(k$) = 13 then switch1 = 0: goto 952
 2361   5168               locate 25,1
 236E   5168               print string$(39," ");
 237C   5168               if key2 = 0 then locate 25,1: print "Please use letter
s or n
                     umbers only"; : goto 952
 23A0   5168    '
 23A0   5168    '
 23A0   5168               if key2 = 40 then m = m + 1
 23B3   5168               if key2 = -8 then if m > 1 then m = m - 1: key2 = 40
 23D8   5168               xi = m * 5
 23E3   516A               yi = 186
 23EA   516C               soff = smalloff + key2 * 14
 23F9   516C               if m <> 0 then call place(xi, yi, storage, soff)
 241D   516C               if tkey2 > 0 then m = m + 1
 2430   516C               if m > 59 then switch1 = 0
 2442   516C    '
 2442   516C    '     Branch around for error
 2442   516C    '
 2442   516C    952  '
 2443   516C    '
 2443   516C          wend
 2447   516C    '
 2447   516C    '     Clean up the cursor
 2447   516C    '
 2447   516C          soff = oldoff
 244E   516C          call overlay(x, y, storage, soff)
```

```
Offset  Data    Source Line          IBM Personal Computer BASIC Compiler V1.00

 2467   516C    '
 2467   516C
 2467   516C    '
 2467   516C    '    Make the text the current character
 2467   516C    '
 2467   516C         xi = xi + 5
 2471   516C         get (5,185) - (xi, 191), boxsel
 248E   516C    '
 248E   516C    '    Put up the new cursor
 248E   516C    '
 248E   516C            x = 0
 2495   516C    955     put (x,y),boxsel, xor
 24AA   516C    '
 24AA   516C    '    Put in the chrh and width
 24AA   516C    '
 24AA   516C         chrw = xi - 5
 24B4   516C         chrh = 6
 24BB   516C         ymax = 176
 24C2   516C         colorflg = 1
 24C9   516C    '
 24C9   516C         stateflag = 7
 24D0   516C    '
 24D0   516C         gosub 502
 24D5   516C    '
 24D5   516C         return
 24D8   516C    '
 24D8   516C    960 '
 24D9   516C    '
 24D9   516C    '    Clean-up text
 24D9   516C    '
 24D9   516C         ymax = 199
 24E0   516C         colorflg = 0
 24E7   516C         put (x, y), boxsel, xor
 24FC   516C    '
 24FC   516C         chrh = ochrh
 2503   516C         chrw = ochrw
 250A   516C         x = 160 - chrw / 2
 2521   516C         y = 88 - chrh / 2
 2538   516C         call overlay(x, y, storage, soff)
 2551   516C    '
 2551   516C         stateflag = 1
 2558   516C    '
 2558   516C         gosub 502
 255D   516C    '
 255D   516C         return
 2560   516C    '
 2560   516C    1000 '
 2561   516C    '
 2561   516C    '    Switch selector to allow different functions to be invoked
                 .
 2561   516C    '
 2561   516C         if stateflag = 1 then gosub 20: goto 1001
 2575   516C    '
 2575   516C         if stateflag = 2 then gosub 30: goto 1001
```

```
Offset  Data    Source Line          IBM Personal Computer BASIC Compiler V1.00

 2589   516C    '
 2589   516C         if stateflag = 3 then gosub 40: goto 1001
 259D   516C    '
 259D   516C         if stateflag = 4 then gosub 3000: goto 1001
 25B1   516C    '
 25B1   516C         if stateflag = 5 or stateflag = 8 then gosub 900: goto 100
                1
 25DA   516C    '
 25DA   516C         if stateflag = 6 then gosub 910: goto 1001
 25EE   516C    '
 25EE   516C         if stateflag = 7 then gosub 42 : gosub 960: goto 1001
 2607   516C    '
 2607   516C         if stateflag = 10 then gosub 801: goto 1001
 261B   516C    '
 261B   516C         if stateflag = 11 then gosub 802: goto 1001
 262F   516C    '
 262F   516C         if stateflag = 12 then gosub 1121: goto 1001
 2643   516C    '
 2643   516C         if stateflag = 13 then gosub 1910: goto 1001
 2657   516C    '
 2657   516C         if stateflag = 15 then gosub 1950: goto 1001
 266B   516C    '
 266B   516C         if stateflag = 16 then gosub 2010: goto 1001
 267F   516C    '
 267F   516C         if stateflag = 17 then gosub 2020: goto 1001
 2693   516C    '
 2693   516C         if stateflag = 18 then gosub 2030: goto 1001
 26A7   516C    '
 26A7   516C         if stateflag = 20 then gosub 1110: goto 1001
 26BB   516C    '
 26BB   516C         if stateflag = 21 then gosub 1120: goto 1001
 26CF   516C    '
 26CF   516C    '    if stateflag = 22 then gosub 3000: goto 1001
 26CF   516C         gosub 55
 26D4   516C    '
 26D4   516C    1001 '
 26D5   516C    '
 26D5   516C         return
 26D8   516C    '
 26D8   516C    1100 '
 26D9   516C    '
 26D9   516C    '    Draw a line between two pointings
 26D9   516C    '
 26D9   516C         stateflag = 20
 26E0   516C         gosub 14800
 26E5   516C         if firstx = 0 then firstx = x: firsty = y
 26FE   5170         call overlay(x, y, storage, soff)
 2717   5170    '
 2717   5170         soff = smalloff
 271E   5170         chrh = 3: chrw = 3: ymax = 176: colorflg = 1
 273A   5170         x = firstx
 2741   5170         y = firsty
 2748   5170         call overlay(x, y, storage, soff)
 2761   5170    '
```

```
Offset  Data    Source Line
 2761   5170         gosub 502
 2766   5170    '
 2766   5170         return
 2769   5170    '
 2769   5170    1110 '
 276A   5170    '
 276A   5170    '    First pointing for the line
 276A   5170    '
 276A   5170         stateflag = 21
 2771   5170         firstx = x + 2
 277A   5170         firsty = y + 1
 2782   5170         gosub 502
 2787   5170    '
 2787   5170         return
 278A   5170    '
 278A   5170    1120 '
 278B   5170    '
 278B   5170    '    Second point for a line
 278B   5170    '
 278B   5170         secondx = x + 2
 2794   5172         secondy = y + 1
 279C   5174         message$ = "Choose a color and press RED"
 27A5   5174         stateflag = 12
 27AC   5174         ymax = 199
 27B3   5174         colorflg = 0
 27BA   5174    '
 27BA   5174         gosub 800
 27BF   5174    '
 27BF   5174         return
 27C2   5174    '
 27C2   5174    1121 '
 27C3   5174    '
 27C3   5174    '    get the color and draw the line
 27C3   5174    '
 27C3   5174         number = int(abs(x - 1) / 80)
 27E2   5176    '
 27E2   5176         gosub 14800
 27E7   5176    '
 27E7   5176         line (firstx, firsty) - (secondx, secondy), number
 2802   5176    '
 2802   5176    '    Keep the last line pointings.
 2802   5176    '
 2802   5176         firstx = secondx - 1
 280A   5176         firsty = secondy - 1
 2812   5176    '
 2812   5176         stateflag = 1
 2819   5176         colorflg = 0
 2820   5176    '
 2820   5176         call overlay(x, y, storage, soff)
 2839   5176    '
 2839   5176         gosub 502
 283E   5176    '
 283E   5176         return
 2841   5176    '
```

```
Offset  Data    Source Line          IBM Personal Computer BASIC Compiler V1.00

 2841   5176    1700 '
 2842   5176    '
 2842   5176    '     Load an existing screen
 2842   5176    '
 2842   5176          nofilefl = 0
 2849   5176          inputcol = 22
 2850   5176          message$ = "Enter screen name:  [           ]"
 2859   5176          gosub 29010
 285E   5176          input; "", file$
 286E   5176          file$ = file$ + ".scr"
 287A   5176          gosub 1710
 287F   5176          on error goto 31300
 2886   5176    '
 2886   5176    '     Error on opening the screen file.
 2886   5176    '
 2886   5176          if nofilefl <> 0 then: gosub 502: return
 2899   5176          call overlay(x, y, storage, soff)
 28B2   5176          def seg = &hb800
 28B9   5176          bload file$,0
 28C2   5176          soff = oldoff
 28C9   5176          chrh = ochrh
 28D0   5176          chrw = ochrw
 28D7   5176          call overlay(x, y, storage, soff)
 28F0   5176          gosub 502
 28F5   5176    '
 28F5   5176          return
 28F8   5176    '
 28F8   5176    1710 '
 28F9   5176    '
 28F9   5176    '     Check for existence of the screen
 28F9   5176    '
 28F9   5176          on error goto 1711
 2900   5176          open file$ for input as #3
 2911   5176          close #3
 2918   5176          return
 291B   5176    '
 291B   5176    1711 '
 291C   5176    '
 291C   5176    '     No file found
 291C   5176    '
 291C   5176          gosub 29000
 2921   5176          locate 23,1
 292E   5176          print "File "; file$; " was not found.";
 2940   5176          for i = 0 to 10000: j = 0: next i
 295D   5176    '
 295D   5176          nofilefl = 1
 2964   5176    '
 2964   5176          resume next
 2968   5176    '
 2968   5176    1800 '
 2969   5176    '
 2969   5176    '     File the dynamic display table away on disk
 2969   5176    '
 2969   5176          gosub 14800 ' reset the cursor
```

```
Offset  Data    Source Line          IBM Personal Computer BASIC Compiler V1.00

 296E   5176         gosub 29000 ' clear prompt area
 2973   5176         if ddtrecno = 0 then gosub 1850: return
 2986   5176         call overlay(x, y, storage, soff) ' clean up the cursor
 299F   5176         switch1 = 1
 29A6   5176         while switch1
 29B2   5176              message$ = "Please enter the display name[        ]"
 29BB   5176              inputcol = 31
 29C2   5176              gosub 29010
 29C7   5176              input; "",file$
 29D7   5176              if file$ <> "" then gosub 1810
 29EA   5176    '
 29EA   5176         wend
 29EE   5176    '
 29EE   5176         call overlay(x, y, storage, soff) ' put up the old cursor
 2A07   5176    '
 2A07   5176         stateflag = 1
 2A0E   5176         gosub 502
 2A13   5176    '
 2A13   5176         return
 2A16   5176    '
 2A16   5176    1810 '
 2A17   5176    '
 2A17   5176    '    File the ddt and the screen
 2A17   5176    '
 2A17   5176         switch1 = 0
 2A1E   5176         nofilefl = 0
 2A25   5176    '
 2A25   5176    '    GET the description info. for the display directory
 2A25   5176    '
 2A25   5176         gosub 1815
 2A2A   5176    '
 2A2A   5176         gosub 29000 ' clear the prompt area
 2A2F   5176         locate 23,1
 2A3C   5176         print "Please check storage medium";
 2A44   5176    '
 2A44   5176         locate 25,1
 2A51   5176         print "Then press any key.";
 2A59   5176    1811 if inkey$ = "" then 1811
 2A65   5176    '
 2A65   5176         ddt$ = "test"
 2A6E   517A         on error goto 1820
 2A75   517A         open ddt$ for output as #3
 2A87   517A         on error goto 31300
 2A8E   517A         close #3
 2A95   517A         if nofilefl > 0 then return
 2AA3   517A         KILL DDT$
 2AAA   517A         if nofilefl > 0 then return
 2AB8   517A    '
 2AB8   517A         ddt$ = FILE$ + ".DDT"
 2AC6   517A         open ddt$ as #3 len = 20
 2AD9   517A         if nofilefl > 0 then return
 2AE7   517A         field #3, 2 as entry$, 2 as ulxchar$, 2 as ulychar$, 2 as
                  boxx$, 2 as boxy$, 2 as alarm$, 2 as ulxvalue$, 2 as ulyvalue$,
                   2 as typddt$, 2 as fill$
```

```
Offset  Data    Source Line          IBM Personal Computer BASIC Compiler V1.00

 2B3D   519A    '
 2B3D   519A    '    Put up the number of records in a header rec.
 2B3D   519A    '
 2B3D   519A         lset entry$ = mki$(ddtrecno)
 2B4A   519A    '
 2B4A   519A         put #3, 1
 2B54   519A         if nofilefl > 0 then return
 2B62   519A    '
 2B62   519A         for i = 1 to ddtrecno
 2B6F   519C              lset entry$ = mki$(entry(i))
 2B82   519C              lset typddt$ = mki$(typeddt(i))
 2B95   519C              lset ulxchar$ = mki$(ulxchar(i))
 2BA8   519C              lset ulychar$ = mki$(ulychar(i))
 2BBB   519C              lset boxx$ = mki$(boxsizex(i))
 2BCE   519C              lset boxy$ = mki$(boxsizey(i))
 2BE1   519C              lset alarm$ = mki$(alarm(i))
 2BF4   519C              lset ulxvalue$ = mki$(ulxvalue(i))
 2C07   519C              lset ulyvalue$ = mki$(ulyvalue(i))
 2C1A   519C    '
 2C1A   519C              j = i + 1
 2C22   519C              put #3, j
 2C2D   519C              if nofilefl > 0 then return
 2C3B   519C    '
 2C3B   519C         next i
 2C4F   519C
 2C4F   519C         close #3
 2C56   519C         if nofilefl > 0 then return
 2C64   519C    '
 2C64   519C    '    Save the display description table away
 2C64   519C    '
 2C64   519C         ddt$ = "display.tab"
 2C6D   519C         open ddt$ as #3 len = 80
 2C80   519C         if nofilefl > 0 then return
 2C8E   519C         field #3, 8 as entryname$, 70 as entry$, 2 as fill$
 2CAC   51A0         j = len(file$)
 2CB6   51A0    '
 2CB6   51A0         for k = 1 to 100
 2CBD   51A0              get #3, k
 2CC8   51A2              if file$ = left$(entryname$,j) then 1812
 2CDE   51A2              if left$(entryname$,5) = "@#$%!" then 1812
 2CF3   51A2         next
 2D02   51A2    '
 2D02   51A2         gosub 29000
 2D07   51A2         locate 23,1
 2D14   51A2         print "Display table is full";
 2D1C   51A2         for i = 0 to 9999: j = 1: next
 2D39   51A2    '
 2D39   51A2         return
 2D3C   51A2    '
 2D3C   51A2    1812 '
 2D3D   51A2         lset entry$ = message$
 2D46   51A2         lset entryname$ = file$
 2D4F   51A2         put  #3, k
 2D5A   51A2    '
```

```
Offset  Data    Source Line          IBM Personal Computer BASIC Compiler V1.00

 2D5A   51A2         close #3
 2D61   51A2    '
 2D61   51A2    '    Save the screen away
 2D61   51A2    '
 2D61   51A2         ddt$ =  FILE$ + ".scr"
 2D6F   51A2         def seg = &hb800
 2D76   51A2    '
 2D76   51A2    1814 bsave ddt$, 0, &h4000
 2D82   51A2    '
 2D82   51A2    '
 2D82   51A2         return
 2D85   51A2    '
 2D85   51A2    1815 '
 2D86   51A2    '
 2D86   51A2    '    Get the description data for the display
 2D86   51A2    '
 2D86   51A2         gosub 29000
 2D8B   51A2         locate 23,1
 2D98   51A2         print "Enter description on the next 2 lines."
 2DA0   51A2         i = 1
 2DA7   51A2         j = 24
 2DAE   51A2         flag = 0
 2DB5   51A4         message$ = ""
 2DBE   51A4         while j < 26
 2DC9   51A4              while i < 39
 2DD4   51A4                   locate j,i
 2DE3   51A4                   charflag = 0
 2DEA   51A6    1816           a$ = inkey$: if a$ = "" then 1816
 2DFE   51AA                   if len(a$) < 2 then gosub 1817: if charflag = 0 t
hen pr
                    int a$;: i = i + 1: message$ = message$ + a$
 2E38   51AA                   if len(a$) > 1 then beep
 2E4A   51AA                   if flag > 0 or (( i > 31 ) and ( j > 24 )) then j
 = 30:
                     i = 99
 2E85   51AA              wend
 2E89   51AA              j = j + 1
 2E91   51AA              i = 1
 2E98   51AA         wend
 2E9C   51AA    '
 2E9C   51AA         return
 2E9F   51AA    '
 2E9F   51AA    1817 '
 2EA0   51AA    '
 2EA0   51AA    '    Length of inkey = 1
 2EA0   51AA    '
 2EA0   51AA         if asc(a$) = 13 then flag = 1: charflag = 1: return
 2EBF   51AA         if asc(a$) = 8 and j = 24 and i = 1 then charflag = 1: bee
                p: return
 2EFD   51AA         if asc(a$) = 8 then if i = 1 then j = j - 1: i = 38 else i
                 = i - 1
 2F30   51AA         if asc(a$) = 8 then leng = len(message$) - 1: locate j,i:
                print " ";
 2F60   51AC         if asc(a$) = 8 then message$ = left$(message$, leng): char
                flag = 1: return
 2F89   51AC    '
```

```
Offset  Data    Source Line          IBM Personal Computer BASIC Compiler V1.00

 2F89   51AC         return
 2F8C   51AC    '
 2F8C   51AC    1820 '
 2F8D   51AC    '
 2F8D   51AC    '    Disk error
 2F8D   51AC    '
 2F8D   51AC         nofilefl = 1
 2F94   51AC         gosub 29000
 2F99   51AC         locate 23,1
 2FA6   51AC         print "Disk error check disk drive B";
 2FAE   51AC         for i = 0 to 10000: j = 0: next i
 2FCB   51AC    '
 2FCB   51AC         resume next
 2FCF   51AC    '
 2FCF   51AC    1850 '
 2FD0   51AC    '
 2FD0   51AC    '    Check for save of the screen
 2FD0   51AC    '
 2FD0   51AC         locate 23,1
 2FDD   51AC         stateflag =1
 2FE4   51AC         print "Only the screen will be saved.";
 2FEC   51AC         beep
 2FF0   51AC         for i = 0 to 9999: j = 1: next
 300D   51AC    '
 300D   51AC         message$ = "Name of the screen:  [         ]"
 3016   51AC         inputcol = 23
 301D   51AC         gosub 29010
 3022   51AC         input; "",file$
 3032   51AC         if file$ = "" then return
 3043   51AC         file$ = file$ + ".scr"
 304F   51AC         call overlay( x, y, storage, soff)
 3068   51AC         def seg = &hb800
 306F   51AC         bsave file$, 0, &h4000
 307B   51AC         call overlay( x, y, storage, soff)
 3094   51AC    '
 3094   51AC         return
 3097   51AC    '
 3097   51AC    1900 '
 3098   51AC    '
 3098   51AC    '    circle
 3098   51AC    '
 3098   51AC    '    Keep the user above the 23rd line
 3098   51AC    '
 3098   51AC         ymax = 176
 309F   51AC         colorflg = 1
 30A6   51AC         stateflag = 13 'prompt for a pointing
 30AD   51AC    '
 30AD   51AC         call overlay(x,y,storage,soff)
 30C6   51AC    '
 30C6   51AC         soff = smalloff
 30CD   51AC         ochrh = chrh
 30D4   51AC         ochrw = chrw
 30DB   51AC         chrh = 3
 30E2   51AC         chrw = 3
```

```
Offset  Data    Source Line          IBM Personal Computer BASIC Compiler V1.00

 30E9   51AC         call overlay(x,y,storage,soff)
 3102   51AC    '
 3102   51AC         gosub 502
 3107   51AC    '
 3107   51AC         return
 310A   51AC    '
 310A   51AC    1910 '
 310B   51AC    '
 310B   51AC    '    more circle
 310B   51AC    '
 310B   51AC    '    Set up recovery routine for circle's off the screen
 310B   51AC    '
 310B   51AC         stateflag = 14
 3112   51AC    '
 3112   51AC         call overlay(x, y, storage, soff)
 312B   51AC         gosub 502
 3130   51AC         count = 3
 3137   51AE         color1 = 2
 313E   51B0         switch1 = 1
 3145   51B0         x = x + 2
 314E   51B0         y = y + 1
 3156   51B0         if y + count > 176 then y = 176 - count
 3171   51B0         circle (x, y), count, color1
 3184   51B0         while switch1
 3190   51B0    '
 3190   51B0              key1 = 0
 3197   51B2              k$ = inkey$
 31A0   51B2              if k$ <> "" then if len(k$) > 1 then key1 = asc(right$
(k$,1)
                  ) else key1 = asc(k$)
 31D9   51B2              if key1 = 68 then switch1 = 0
 31EB   51B2              if key1 = 67 then color1 = color1 + 1: if color1 > 3 t
hen co
                  lor1 = 1
 3210   51B2              button = 0
 3217   51B2              if key1 = 43 then button = 1
 3229   51B2              circlex = x - count
 3234   51B4              circley = y - count
 323F   51B6              if circlex < 0 then circlex = 0
 3251   51B6              IF CIRCLEY < 0 THEN CIRCLEY = 0
 3263   51B6              x1 = x + count
 326E   51B8              y1 = y + count
 3279   51BA              IF X1 > 319 THEN x1 = 319
 328C   51BA              if Y1 > 176 then y1 = 176
 329F   51BA              get (circlex, circley) - (x1,y1), boxsel
 32BF   51BA              put (circlex, circley), boxsel, xor
 32D4   51BA              if button = 1 then count = count + 1
 32E7   51BA              if count >= 30 then count = 30
 32F9   51BA              if y + count > 176 then y = 176 - count
 3314   51BA              circle (x, y), count, color1
 3327   51BA    '
 3327   51BA         wend
 332B   51BA    '
 332B   51BA         button = 0
 3332   51BA    '
 3332   51BA         message$ = "Point to the color to fill the circle"
```

```
Offset  Data    Source Line          IBM Personal Computer BASIC Compiler V1.00

 333B   51BA         stateflag = 15
 3342   51BA         circlex = x + 1
 334A   51BA         circley = y + 1
 3352   51BA         IF CIRCLEX > 319 THEN CIRCLEX = 319
 3365   51BA         if CIRCLEY > 176 then circley = 176
 3378   51BA    '
 3378   51BA         gosub 502
 337D   51BA         ymax = 199
 3384   51BA         gosub 810
 3389   51BA    '
 3389   51BA         return
 338C   51BA    '
 338C   51BA    1950 '
 338D   51BA    '
 338D   51BA    '    put up the circle and fill it appropriately
 338D   51BA    '
 338D   51BA         newcol = int(abs(x - 1) / 80)
 33AC   51BA    '
 33AC   51BA    1951     paint (circlex, circley), newcol, color1
 33BF   51BA    '
 33BF   51BA         colorflg = 0
 33C6   51BA         stateflag = 1
 33CD   51BA         gosub 14800
 33D2   51BA    '
 33D2   51BA         gosub 502
 33D7   51BA    '
 33D7   51BA         return
 33DA   51BA    '
 33DA   51BA    2000 '
 33DB   51BA    '
 33DB   51BA    '    Draw a box
 33DB   51BA    '
 33DB   51BA         colorflg = 1
 33E2   51BA         ymax = 176
 33E9   51BA         call overlay(x, y, storage, soff)
 3402   51BA         soff = smalloff
 3409   51BA         ochrh = chrh
 3410   51BA         ochrw = chrw
 3417   51BA         chrw = 3
 341E   51BA         chrh = 3
 3425   51BA         stateflag = 16
 342C   51BA         gosub 502
 3431   51BA         call overlay(x, y, storage, soff)
 344A   51BA    '
 344A   51BA         return
 344D   51BA    '
 344D   51BA    2010 '
 344E   51BA    '
 344E   51BA    '    more box
 344E   51BA    '
 344E   51BA         stateflag = 17
 3455   51BA         fx = x + 2
 345E   51BC         fy = y + 1
 3466   51BE         gosub 502
```

```
Offset  Data    Source Line
 346B   51BE    '
 346B   51BE         return
 346E   51BE    '
 346E   51BE    2020 '
 346F   51BE    '
 346F   51BE    '    more box
 346F   51BE    '
 346F   51BE         sx = x + 2
 3478   51C0         sy = y + 1
 3480   51C2    '
 3480   51C2         call overlay(x, y, storage, soff)
 3499   51C2    '
 3499   51C2         stateflag = 18
 34A0   51C2         ymax = 199
 34A7   51C2         message$ = "Point to the color to fill the box with"
 34B0   51C2         gosub 810
 34B5   51C2    '
 34B5   51C2         return
 34B8   51C2    '
 34B8   51C2    2030 '
 34B9   51C2    '
 34B9   51C2    '    more box
 34B9   51C2    '
 34B9   51C2         stateflag = 1
 34C0   51C2    '
 34C0   51C2         newcol = int(abs(x - 1) / 80)
 34DF   51C2    '
 34DF   51C2         line (fx,fy) - (sx,sy), newcol, BF
 34FA   51C2    '
 34FA   51C2         colorflg = 0
 3501   51C2         gosub 14800
 3506   51C2         gosub 502
 350B   51C2    '
 350B   51C2         return
 350E   51C2    '
 350E   51C2    3000 '
 350F   51C2    '
 350F   51C2    '    Save the first point for the alarm display.
 350F   51C2    '
 350F   51C2         xfirst = x + 1
 3517   51C2         yfirst = y + 1
 351F   51C2         call overlay(x, y, storage, soff)
 3538   51C2         x = x + 2
 3541   51C2         y = y + 1
 3549   51C2         for i = 1 to 1000
 3550   51C2              j = 0
 3557   51C2         next
 3567   51C2    '
 3567   51C2    '    get the picture area for storage and draw the enclosed are
                 a box
 3567   51C2    '
 3567   51C2         get (xfirst,yfirst) - (x + 20,y + 20), boxsel
 358D   51C2         stateflag = 5
 3594   51C2         colorflag = 1
```

```
Offset  Data    Source Line          IBM Personal Computer BASIC Compiler V1.00

 359B   51C4         ymax = 176
 35A2   51C4         chrw = 1
 35A9   51C4         gosub 502
 35AE   51C4    '
 35AE   51C4         return
 35B1   51C4    '
 35B1   51C4    3010 '
 35B2   51C4    '
 35B2   51C4    '    Draw the expanding box
 35B2   51C4    '
 35B2   51C4         if x - xfirst >= 0 and xold >= xfirst and yold >= yfirst t
                hen line (xfirst,yfirst) - (xold,yold), 0, b
 3601   51C4         if x - xfirst >= 0 then put (xfirst,yfirst), boxsel, pset
 3625   51C4    '
 3625   51C4         if x - xfirst < 0 then beep: stateflag = 8: gosub 502: ret
                urn
 3648   51C4         if y - yfirst < 0 then beep: stateflag = 8: gosub 502: ret
                urn
 366B   51C4         if x - xfirst > 47 then x = xfirst + 47
 3686   51C4         if y - yfirst > 40 then y = yfirst + 39
 36A1   51C4    '
 36A1   51C4         if stateflag = 8 then stateflag = 5: gosub 502
 36B8   51C4    '
 36B8   51C4         get (xfirst,yfirst) - (x,y), boxsel
 36D8   51C4         line (xfirst, yfirst) - (x,y), 2, b
 36F2   51C4
 36F2   51C4    '
 36F2   51C4         return
 36F5   51C4    '
 36F5   51C4    14000 '
 36F6   51C4    '
 36F6   51C4    '    Get pointer from symbol table and initialize chrh & chrw
 36F6   51C4    '    Pass:  symnum = symbol table number
 36F6   51C4    '
 36F6   51C4              symbnum = ( symnum + 1) * 2 + symbtoff
 3707   51C6    '
 3707   51C6              def seg = storage
 370F   51C6              soff = peek(symbnum) + 256 * peek(symbnum + 1)
 3745   51C6              soff = soff + offset2 + symbtoff
 3754   51C6              oldoff = soff
 375B   51C6    '
 375B   51C6    '    set up new offset to symbol table proper
 375B   51C6    '
 375B   51C6    '    Get new symbol's height and width
 375B   51C6    '
 375B   51C6              chrw = peek(soff) + 256 * peek(soff + 1)
 3791   51C6              chrh = peek(soff+2) + 256 * peek(soff + 3)
 37CB   51C6              chrw = chrw / 2
 37DD   51C6              def seg
 37E1   51C6    '
 37E1   51C6         return
 37E4   51C6    '
 37E4   51C6    14800 '
 37E5   51C6    '
```

```
Offset  Data    Source Line          IBM Personal Computer BASIC Compiler V1.00

 37E5   51C6    '     clean-up DEGREES cursor
 37E5   51C6    '
 37E5   51C6    '
 37E5   51C6          call overlay(x, y, storage, soff)
 37FE   51C6    '
 37FE   51C6    '     Put up the old cursor.
 37FE   51C6    '
 37FE   51C6          chrh = ochrh
 3805   51C6          chrw = ochrw
 380C   51C6          soff = oldoff
 3813   51C6    '
 3813   51C6          x = 160 - chrw / 2
 382A   51C6          y = 88 - chrh / 2
 3841   51C6    '
 3841   51C6          call overlay(x, y, storage, soff)
 385A   51C6    '
 385A   51C6          return
 385D   51C6    '
 385D   51C6    14999 '
 385E   51C6    '
 385E   51C6    '     Data statements (initially only for change cursor)
 385E   51C6    '
 385E   51C6    '     Newcolor bit masks
 385E   51C6    '
 385E   51C6    data 0, 0, 0, 0, 64, 16, 4, 1, 128, 32, 8, 2, 192, 48, 12, 3
 3860   51C6    '
 3860   51C6    '     Original color to be changed bit masks
 3860   51C6    '
 3860   51C6    '     Blue:
 3860   51C6    '
 3860   51C6    data 192, 0, 48, 0, 12, 0, 3, 0
 3862   51C6    '
 3862   51C6    '     Green:
 3862   51C6    '
 3862   51C6    data 192, 64, 48, 16, 12, 4, 3, 1
 3864   51C6    '
 3864   51C6    '     Red:
 3864   51C6    '
 3864   51C6    data 192, 128, 48, 32, 12, 8, 3, 2
 3866   51C6    '
 3866   51C6    '     White:
 3866   51C6    '
 3866   51C6    data 192, 192, 48, 48, 12, 12, 3, 3
 3868   51C6    '
 3868   51C6    15000 '
 3869   51C6    '
 3869   51C6    '  This code enables a program to switch to the color monitor
 3869   51C6    '
 3869   51C6          def seg = 0
 386F   51C6          a = peek(&h410)
 3881   51C6          width 80
 3888   51C6          poke &h410, (a and &hcf) or &h20
 38A0   51C6          width 40
 38A7   51C6          screen 1
```

```
Offset  Data    Source Line          IBM Personal Computer BASIC Compiler V1.00

 38AE   51C6         screen 0
 38B4   51C6         locate ,,1,6,7
 38CD   51C6         screen 1
 38D4   51C6         color 16, 0
 38E0   51C6         cls
 38E4   51C6         key off
 38EA   51C6         def seg
 38EE   51C6    '
 38EE   51C6         return
 38F1   51C6    '
 38F1   51C6    15900 '
 38F2   51C6    '
 38F2   51C6    '    Rotate the current character
 38F2   51C6    '
 38F2   51C6         call overlay(x, y, storage, soff)
 390B   51C6         def seg = &hb800
 3912   51C6         bsave "screen.scr",0,&h4000
 391E   51C6         def seg
 3922   51C6         cls
 3926   51C6    '
 3926   51C6         locate 23,1
 3933   51C6         print "Please enter angle to rotate: [   ]";
 393B   51C6         locate 23,32
 3948   51C6         input ;"",a$
 3958   51C6    '
 3958   51C6         a = val(a$)
 3965   51C6         c! = a / 57.29578
 3976   51CA         s! = c!
 397F   51CE         c! = cos(c!)
 398A   51CE         s! = sin(s!)
 3995   51CE    '
 3995   51CE         i = 100
 399C   51CE         j = 100
 39A3   51CE         call place(i, j, storage, soff)
 39BC   51CE    '
 39BC   51CE         k = 0
 39C3   51CE         for i = 0 to chrw
 39CF   51D0              for j = 0 to chrh
 39DB   51D2                   colr(k) = point(100 + i, 100 + j)
 39F7   51D2                   newx! = i * c! - j * s!
 3A19   51D6                   newy! = i * s! + j * c!
 3A3B   51DA                   xnew(k) = int(newx!)
 3A4E   51DA                   ynew(k) = int(newy!)
 3A61   51DA    '
 3A61   51DA                   k = k + 1
 3A69   51DA    '
 3A69   51DA              next j
 3A7D   51DA    '
 3A7D   51DA         next i
 3A91   51DA         cls
 3A95   51DA    '
 3A95   51DA         k = 0
 3A9C   51DA         a = 320
 3AA3   51DA         b = 200
```

```
Offset  Data    Source Line          IBM Personal Computer BASIC Compiler V1.00

 3AAA   51DC         c = 0
 3AB1   51DE         d = 0
 3AB8   51DE    '
 3AB8   51DE         for i = 0 to chrw
 3AC4   51E0              for j = 0 to chrh
 3AD0   51E2                   t = xnew(k) + 100
 3AE1   51E4                   u = ynew(k) + 100
 3AF2   51E6                   if t < a then a = t
 3B06   51E6                   if u < b then b = u
 3B1A   51E6                   if t > c then c = t
 3B2E   51E6                   if u > d then d = u
 3B42   51E6    '
 3B42   51E6                   pset (t,u),colr(k)
 3B58   51E6    '
 3B58   51E6                   k = k + 1
 3B60   51E6    '
 3B60   51E6              next j
 3B74   51E6    '
 3B74   51E6         next i
 3B88   51E6    '
 3B88   51E6         get (a,b) - (c,d), boxsel
 3BA8   51E6         soff = newsymboff
 3BAF   51E6         chrh = abs(d - b) + 1
 3BC3   51E6         chrw = abs(c - a) + 1
 3BD7   51E6    '
 3BD7   51E6    '    Put character out.
 3BD7   51E6    '
 3BD7   51E6         i! = chrw
 3BE3   51EA         j! = chrh
 3BEF   51EE         i  = 4! + int(( i! * 2! + 7! ) / 8! ) * j!
 3C13   51EE    '
 3C13   51EE         def seg
 3C17   51EE         a = varptr(boxsel(0))
 3C1E   51EE    '
 3C1E   51EE         for k = 0 to i
 3C2A   51F0    '
 3C2A   51F0             j = a + k
 3C35   51F0             j = peek( j )
 3C48   51F0             l = newsymboff + k
 3C53   51F2             def seg = storage
 3C5B   51F2             poke l, j
 3C6B   51F2             def seg
 3C6F   51F2    '
 3C6F   51F2         next k
 3C80   51F2    '
 3C80   51F2         def seg = &hb800
 3C87   51F2         bload "screen.scr",0
 3C90   51F2         def seg
 3C94   51F2    '
 3C94   51F2         call overlay (x, y, storage, soff)
 3CAD   51F2    '
 3CAD   51F2         return
 3CB0   51F2    '
 3CB0   51F2    16000 '
```

```
Offset  Data    Source Line          IBM Personal Computer BASIC Compiler V1.00

 3CB1   51F2    '
 3CB1   51F2    '     Generate symbol
 3CB1   51F2    '
 3CB1   51F2          call overlay(x, y, storage, soff)
 3CCA   51F2    '
 3CCA   51F2          def seg = &hb800
 3CD1   51F2          bsave "screen.scr",0,&h4000
 3CDD   51F2          def seg
 3CE1   51F2    '
 3CE1   51F2          c1 = 1
 3CE8   51F4    '
 3CE8   51F4          get (0,0)-(3,3),box0
 3D01   51F4    '
 3D01   51F4          line (0,0)-(3,3),1,bf
 3D14   51F4          get (0,0)-(3,3),box1
 3D2D   51F4    '
 3D2D   51F4          line (0,0)-(3,3),2,bf
 3D40   51F4          get (0,0)-(3,3),box2
 3D59   51F4    '
 3D59   51F4          line (0,0)-(3,3),3,bf
 3D6C   51F4          get (0,0)-(3,3),box3
 3D85   51F4    '
 3D85   51F4          line (0,0)-(3,3),0,bf
 3D97   51F4    '
 3D97   51F4          gosub 16010     ' Put up the screen
 3D9C   51F4          gosub 16020     ' Put up the mode
 3DA1   51F4    '
 3DA1   51F4          switch1 = 1
 3DA8   51F4          while switch1
 3DB4   51F4    '
 3DB4   51F4               gosub 16030
 3DB9   51F4    '
 3DB9   51F4    16001 '
 3DBA   51F4    '
 3DBA   51F4               key1$ = inkey$
 3DC3   51F8               if key1$ = "" then 16001
 3DCE   51F8    '
 3DCE   51F8               if len(key1$) > 1 then key1$ = right$(key1$,1)
 3DEC   51F8               key1 = asc(key1$)
 3DF6   51F8               if key1 > 58 and key1 < 69 then gosub 16040 ' function
 keys
 3E1B   51F8               if key1 > 70 and key1 < 82 then gosub 16050 ' cursor k
eys
 3E40   51F8    '
 3E40   51F8          wend
 3E44   51F8    '
 3E44   51F8          return
 3E47   51F8    '
 3E47   51F8    16010 '
 3E48   51F8    '
 3E48   51F8    '     Initialize the screen
 3E48   51F8    '
 3E48   51F8          def seg = &hb800
 3E4F   51F8          bload "create.scr",0
 3E58   51F8          def seg
 3E5C   51F8    '
```

```
Offset  Data    Source Line          IBM Personal Computer BASIC Compiler V1.00

 3E5C   51F8    '     Draw the initial color box
 3E5C   51F8    '
 3E5C   51F8          line (233,11)-(238,21),c1,bf
 3E73   51F8    '
 3E73   51F8          return
 3E76   51F8    '
 3E76   51F8    16020 '
 3E77   51F8    '
 3E77   51F8    '     Switch modes from move to draw and vice-versa
 3E77   51F8    '
 3E77   51F8          locate 22,1
 3E84   51F8          print string$(6," ");
 3E92   51F8    '
 3E92   51F8          locate 22,1
 3E9F   51F8          if mode = 0 then mode = 1: print " MOVE "; else mode = 0:
                print " DRAW ";
 3ECB   51FA    '
 3ECB   51FA          return
 3ECE   51FA    '
 3ECE   51FA    16030 '
 3ECF   51FA    '
 3ECF   51FA    '     OVERLAY the box away
 3ECF   51FA    '
 3ECF   51FA          if c1 = 0 then put (xbox,ybox),box0,xor
 3EEF   51FA          if c1 = 1 then put (xbox,ybox),box1,xor
 3F0F   51FA          if c1 = 2 then put (xbox,ybox),box2,xor
 3F2F   51FA          if c1 = 3 then put (xbox,ybox),box3,xor
 3F4F   51FA    '
 3F4F   51FA          return
 3F52   51FA    '
 3F52   51FA    16040 '
 3F53   51FA    '
 3F53   51FA    '     Function keys
 3F53   51FA    '
 3F53   51FA          key1 = key1 - 58
 3F5D   51FA    '
 3F5D   51FA    '                      F1     F2      F3      F4      F5      F6      F
                7      F8      F9      F10
 3F5D   51FA    '
 3F5D   51FA          on key1 gosub 16100, 16100, 16300, 16010, 16100, 16100, 16
                100, 16100, 16700, 16020
 3F79   51FA    '
 3F79   51FA          return
 3F7C   51FA    '
 3F7C   51FA    16050 '
 3F7D   51FA    '
 3F7D   51FA    '     Cursor keys
 3F7D   51FA    '
 3F7D   51FA          if key1 = 71 then if ybox <> yboxmin and xbox <> xboxmin t
                hen gosub 16052: ybox = ybox - 5: xbox = xbox - 5: goto 16051
 3FC9   51FA          if key1 = 73 then if ybox <> yboxmin and xbox <> xboxmax t
                hen gosub 16052: ybox = ybox - 5: xbox = xbox + 5: goto 16051
 4015   51FA          if key1 = 79 then if ybox <> yboxmax and xbox <> xboxmin t
                hen gosub 16052: ybox = ybox + 5: xbox = xbox - 5: goto 16051
```

```
Offset  Data    Source Line          IBM Personal Computer BASIC Compiler V1.00

 4061   51FA         if key1 = 81 then if ybox <> yboxmax and xbox <> xboxmax t
                  hen gosub 16052: ybox = ybox + 5: xbox = xbox + 5: goto 16051
 40AD   51FA    '
 40AD   51FA         if key1 = 72 then if ybox <> yboxmin then gosub 16052: ybo
                  x = ybox - 5: goto 16051
 40D8   51FA         if key1 = 75 then if xbox <> xboxmin then gosub 16052: xbo
                  x = xbox - 5: goto 16051
 4103   51FA         if key1 = 77 then if xbox <> xboxmax then gosub 16052: xbo
                  x = xbox + 5: goto 16051
 412E   51FA         if key1 = 80 then if ybox <> yboxmax then gosub 16052: ybo
                  x = ybox + 5: goto 16051
 4159   51FA    '
 4159   51FA    16051 '
 415A   51FA    '
 415A   51FA         return
 415D   51FA    '
 415D   51FA    16052 '
 415E   51FA    '
 415E   51FA    '    Put the box on the grid and the picture.
 415E   51FA    '
 415E   51FA         if mode = 1 then gosub 16030: return
 4171   51FA         line (xbox,ybox)-(xbox+3,ybox+3),c1,bf
 4192   51FA    '
 4192   51FA         x1 = xreal + xbox / 5
 41B0   51FA         y1 = yreal + ybox / 5
 41CE   51FA         pset (x1,y1),c1
 41DE   51FA    '
 41DE   51FA         return
 41E1   51FA    '
 41E1   51FA    '
 41E1   51FA    16100 '
 41E2   51FA    '
 41E2   51FA    '    F2 = quit
 41E2   51FA    '
 41E2   51FA         switch1 = 0
 41E9   51FA         def seg = &hb800
 41F0   51FA         bload "screen.scr",0
 41F9   51FA         def seg
 41FD   51FA    '
 41FD   51FA         call overlay(x, y, storage, soff)
 4216   51FA    '
 4216   51FA         return
 4219   51FA    '
 4219   51FA    16300 '
 421A   51FA    '
 421A   51FA    '    Save symbol away
 421A   51FA    '
 421A   51FA         lastentry = numbsymb * 2 + symbtoff
 4229   51FC         def seg = storage
 4231   51FC         lastoff = peek(lastentry) + 256 * peek(lastentry + 1)
 4267   51FE         lastoff = lastoff + offset2 + symbtoff
 4276   51FE         x1 = peek(lastoff) + 256 * peek(lastoff + 1)
 42AC   51FE         x1! = x1 / 2
 42BD   5202         y1! = peek(lastoff + 2) + 256 * peek(lastoff + 3)
```

```
Offset  Data    Source Line          IBM Personal Computer BASIC Compiler V1.00

 42FA   5206          def seg
 42FE   5206          numbytes = 4! + int(( x1! * 2! + 7!) / 8!) * y1!
 4322   5208          endshift = lastoff + numbytes + 2
 432F   520A    '
 432F   520A          x1 = xreal+39
 4339   520A          y1 = yreal+29
 4343   520A          get (xreal,yreal)-(x1,y1),boxsel
 4363   520A    '
 4363   520A    '     First get the first two bytes
 4363   520A    '
 4363   520A          firstpic = lastentry + 2
 436C   520C    '
 436C   520C    '     Do the shifting
 436C   520C    '
 436C   520C          def seg = storage
 4374   520C    '
 4374   520C          oldchar1 = peek(firstpic)
 4387   520E          oldchar2 = peek(firstpic + 1)
 439B   5210          j = firstpic + 2
 43A4   5210          for i = j to endshift step 2
 43B1   5212    '
 43B1   5212               overlay1 = peek(i)
 43C4   5214               overlay2 = peek(i+1)
 43D8   5216    '
 43D8   5216               poke i, oldchar1
 43E8   5216               poke i + 1, oldchar2
 43F9   5216    '
 43F9   5216               oldchar1 = overlay1
 4400   5216               oldchar2 = overlay2
 4407   5216    '
 4407   5216          next i
 4419   5216    '
 4419   5216    '     Poke in the new number of symbols
 4419   5216    '
 4419   5216          def seg
 441D   5216          numbsymb = numbsymb + 1
 4425   5216          offset2 = numbsymb * 2 + 2
 4432   5216          a = varptr(numbsymb)
 4439   5216          i = peek(a)
 444C   5216          j = peek(a + 1)
 4460   5216          def seg = storage
 4468   5216          poke symbtoff, i
 4478   5216          poke symbtoff + 1, j
 4489   5216    '
 4489   5216    '     Poke in the new last entry
 4489   5216    '
 4489   5216          reallastoff = endshift - offset2 - symbtoff
 449A   5218          def seg
 449E   5218          a = varptr(reallastoff)
 44A5   5218          i = peek(a)
 44B8   5218          j = peek(a + 1)
 44CC   5218          def seg = storage
 44D4   5218          poke firstpic, i
 44E4   5218          poke firstpic + 1, j
```

```
Offset  Data    Source Line
 44F5   5218    '
 44F5   5218    '    Poke in the symbol
 44F5   5218    '
 44F5   5218         def seg
 44F9   5218         a = varptr(boxsel(0))
 4500   5218    '
 4500   5218         for k = 0 to 303
 4506   5218    '
 4506   5218             j = a + k
 4511   5218             j = peek( j )
 4524   5218             l = endshift + k
 452F   5218             def seg = storage
 4537   5218             poke l, j
 4547   5218             def seg
 454B   5218    '
 454B   5218         next k
 455B   5218    '
 455B   5218    '    update soff to point to the shifted buffer
 455B   5218    '
 455B   5218         soff = soff + 2
 4564   5218    '
 4564   5218         return
 4567   5218    '
 4567   5218   16700 '
 4568   5218    '
 4568   5218    '    F9 - Change color
 4568   5218    '
 4568   5218         c1 = c1 + 1
 4570   5218         if c1 = 4 then c1 = 0
 4582   5218    '
 4582   5218         line (233,11)-(238,21),c1,bf
 4599   5218    '
 4599   5218         return
 459C   5218    '
 459C   5218    '
 459C   5218   17000 '
 459D   5218    '
 459D   5218    '    Rotate through the symbol table
 459D   5218    '
 459D   5218         num$ = str$(rotatesym)
 45AA   5218         gosub 232
 45AF   5218         rotatesym = rotatesym + 1
 45B7   5218         if rotatesym => numbsymb then rotatesym = 0
 45CB   5218    '
 45CB   5218         return
 45CE   5218    '
 45CE   5218   22000 '
 45CF   5218    '
 45CF   5218    '  This will get the profile settings for the joystick
 45CF   5218    '
 45CF   5218         def seg = storage
 45D7   5218         xvar = peek(2) + 256 * peek(3)
 460A   5218         yvar = peek(4) + 256 * peek(5)
 463D   5218         if xvar = 0 then joystick = 0 else joystick = 1
```

```
Offset  Data    Source Line          IBM Personal Computer BASIC Compiler V1.00

 4659   5218    '
 4659   5218        def seg
 465D   5218    '
 465D   5218        return
 4660   5218    '
 4660   5218    29000 '
 4661   5218    '
 4661   5218    '    Clear the prompt area
 4661   5218    '
 4661   5218        locate 23,1
 466E   5218        print string$(39," ");
 467C   5218    '
 467C   5218        locate 24,1
 4689   5218        print string$(39," ");
 4697   5218    '
 4697   5218        locate 25,1
 46A4   5218        print string$(39," ");
 46B2   5218    '
 46B2   5218        return
 46B5   5218    '
 46B5   5218    29010 '
 46B6   5218    '
 46B6   5218    '    Subroutine to handle prompts on line 23
 46B6   5218    '
 46B6   5218    '    Clear the prompt area
 46B6   5218    '    prompt on line 23
 46B6   5218    '    column = 1
 46B6   5218    '    message$ = prompt
 46B6   5218    '    inputcol = column number of the input
 46B6   5218    '
 46B6   5218        gosub 29000
 46BB   5218    '
 46BB   5218        locate 23,1
 46C8   5218        print message$;
 46D0   5218        locate 23, inputcol
 46DE   5218    '
 46DE   5218        return
 46E1   5218    '
 46E1   5218    30000 '
 46E2   5218    '
 46E2   5218    '    turn off all function keys
 46E2   5218    '
 46E2   5218        for i = 1 to 10
 46E9   5218            key(i) off
 46F1   5218        next i
 4700   5218        for i = 1 to 10
 4707   5218            key i,""
 4712   5218        next i
 4721   5218        return
 4724   5218    '
 4724   5218    31000 '
 4725   5218    '
 4725   5218    '    Quadrant 1 - 6 checker
 4725   5218    '
```

```
Offset  Data    Source Line          IBM Personal Computer BASIC Compiler V1.00

 4725   5218         if x < 50 then quadnew = 1
 4737   5218         if x > 49 and x < 125 then quadnew = 2
 475E   5218         if x > 124 and x < 173 then quadnew = 3
 4786   5218         if x > 172 and x < 265 then quadnew = 4
 47AF   5218         if x > 264 then quadnew = 5
 47C2   5218    '
 47C2   5218         gosub 31200
 47C7   5218    '
 47C7   5218         return
 47CA   5218    '
 47CA   5218    31100 '
 47CB   5218    '
 47CB   5218    '    Quadrant 7 - 11 checker
 47CB   5218    '
 47CB   5218         if x < 50 then quadnew = 6
 47DD   5218         if x > 49 and x < 118 then quadnew = 7
 4804   5218         if x > 117 and x < 173 then quadnew = 8
 482C   5218         if x > 172 and x < 226 then quadnew = 9
 4855   5218         if x > 225 then quadnew = 10
 4868   5218    '
 4868   5218         gosub 31200
 486D   5218    '
 486D   5218         return
 4870   5218    '
 4870   5218    31200 '
 4871   5218    '
 4871   5218    '    invert the proper areas
 4871   5218    '
 4871   5218         if initflag > 0 and quadnew = quadold then return
 4896   5218    '
 4896   5218    '    If not the first then clean up the old one.
 4896   5218    '
 4896   5218         if initflag <> 0 then get (olda, oldb) - (oldc, oldd), box
                sel: put (olda, oldb), boxsel, preset
 48D6   5220    '
 48D6   5220         if quadnew > 0 and quadnew < 6 then b = 183: d = 191
 4904   5220         if quadnew > 5 then b = 192: d = 199
 491D   5220         if quadnew = 1 then a = 0: c = 49
 4936   5220         if quadnew = 2 then a = 50: c = 124
 494F   5220         if quadnew = 3 then a = 125: c = 172
 4968   5220         if quadnew = 4 then a = 173: c = 264
 4981   5220         if quadnew = 5 then a = 265: c = 319
 499A   5220         if quadnew = 6 then a = 0: c = 49
 49B3   5220         if quadnew = 7 then a = 50: c = 117
 49CC   5220         if quadnew = 8 then a = 118: c = 172
 49E5   5220         if quadnew = 9  then a = 173: c = 225
 49FE   5220         if quadnew = 10 then a = 226: c = 319
 4A17   5220    '
 4A17   5220    '    if first time through set the initial state.
 4A17   5220    '
 4A17   5220         get (a,b) - (c,d), boxsel: put (a,b), boxsel, preset
 4A4C   5220         olda = a: oldb = b: oldc = c: oldd = d
 4A68   5220         quadold = quadnew
 4A6F   5220         initflag = 1
```

```
Offset  Data    Source Line          IBM Personal Computer BASIC Compiler V1.00

 4A76   5220    '
 4A76   5220         return
 4A79   5220    '
 4A79   5220    31300 '
 4A7A   5220    '
 4A7A   5220    '    All error handler
 4A7A   5220    '
 4A7A   5220         message$ = "System Error, continue? Y/N [ ]"
 4A83   5220         inputcol = 30
 4A8A   5220         gosub 29010
 4A8F   5220         input;"",y$
 4A9F   5220         if y$ <> "Y" and y$ <> "y" then gosub 32000
 4AC7   5220         stateflag = 1
 4ACE   5220         gosub 502
 4AD3   5220         resume next
 4AD7   5220    '
 4AD7   5220    31400 '
 4AD8   5220    '
 4AD8   5220    '    Put up the Help screen
 4AD8   5220    '
 4AD8   5220         def seg = &hb000
 4ADF   5220         bload "help.hlp",0
 4AE8   5220         def seg
 4AEC   5220    '
 4AEC   5220         return
 4AEF   5220    '
 4AEF   5220    32000 '
 4AF0   5220    '
 4AF0   5220    '   Prompt for sureness....
 4AF0   5220    '
 4AF0   5220         message$ =  "Are you sure? Y/N [ ]"
 4AF9   5220         inputcol = 20
 4B00   5220         gosub 29010           ' prompt
 4B05   5220         input;"",y$
 4B15   5220         if y$ <> "Y" and y$ <> "y" then gosub 14800: stateflag = 1
                : gosub 502: return
 4B4C   5220         gosub 29000
 4B51   5220    '
 4B51   5220    '    return to menu
 4B51   5220    '
 4B51   5220         clear
 4B55   5220         run "engineer.exe"
 4B5C   5220         end
 4B60   5220
 6861   5220

22151 Bytes Available
 3207 Bytes Free

    0 Warning Error(s)
    0 Severe  Error(s)
```

0130301

CLAIMS

1. Device for displaying alphanumeric and graphics characters in a computer having an interactive all points addressable display terminal (44) and a cursor positioning device (52), characterized in that it comprises:

means for storing at least one table of selectable cursor characters,

means for selecting a cursor character from said table, the selected character being displayable on the all points addressable display and movable by said cursor positioning device, and

means associated with said cursor positioning device for fixing the selected character at a desired point on the all points addressable display, thereby faciliting the generation of a graphics display including any arbitrary selection of said cursor characters.

2. Device according to claim 1 wherein said display terminal includes a display buffer (56) for addressing said all points addressable display and said means for fixing the selected character at a desired point on the all points addressable display includes means for reading current cursor position and character data into said display buffer.

3. Device according to claims 1 or 2, comprising:

means for inputting an alphanumeric character string not exceeding a predetermined maximum length as said cursor character and

means for selecting said alphanumeric character string as the current cursor character.

4. Device according to claim 3, wherein said means for inputting an alphanumeric character string is a keyboard (28).

5. Device according to any one of the preceding claims, comprising means for exclusive ORing the selected cursor character data with the corresponding data in the all points addressable display at the current position of the selected cursor character whereby the currently selected cursor character is clearly demarked from characters previously fixed at various points on said all points addressable display.

6. Device according to claim 1, comprising:

means for displaying the outline of a geometric figure as defining a blank current cursor character, alphanumeric and graphics characters on the all points addressable display within said outline being clearly visible and said outline being movable by said cursor positioning device, and

means associated with said cursor positioning device for erasing all character data within said outline while said outline is positioned at a desired location on said all points addressable display.

7. Device according to any one of the preceding claims wherein said cursor positioning device includes a joy stick (52, 54) and said means for fixing the selected character at a desired point on the all points addressable display further includes a trigger button associated with said joy stick.

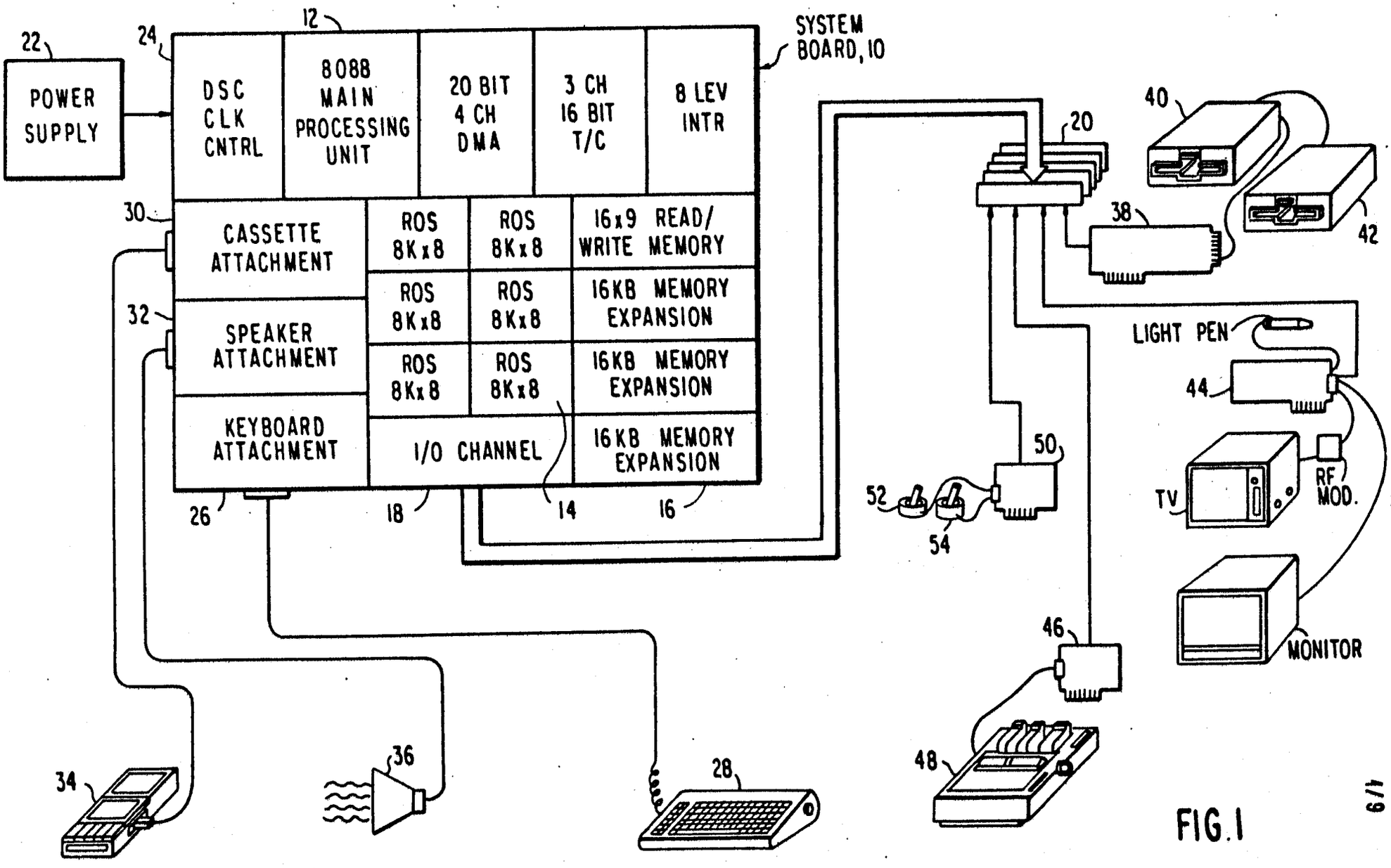


FIG. 1

C130301

1/9

FIG. 2

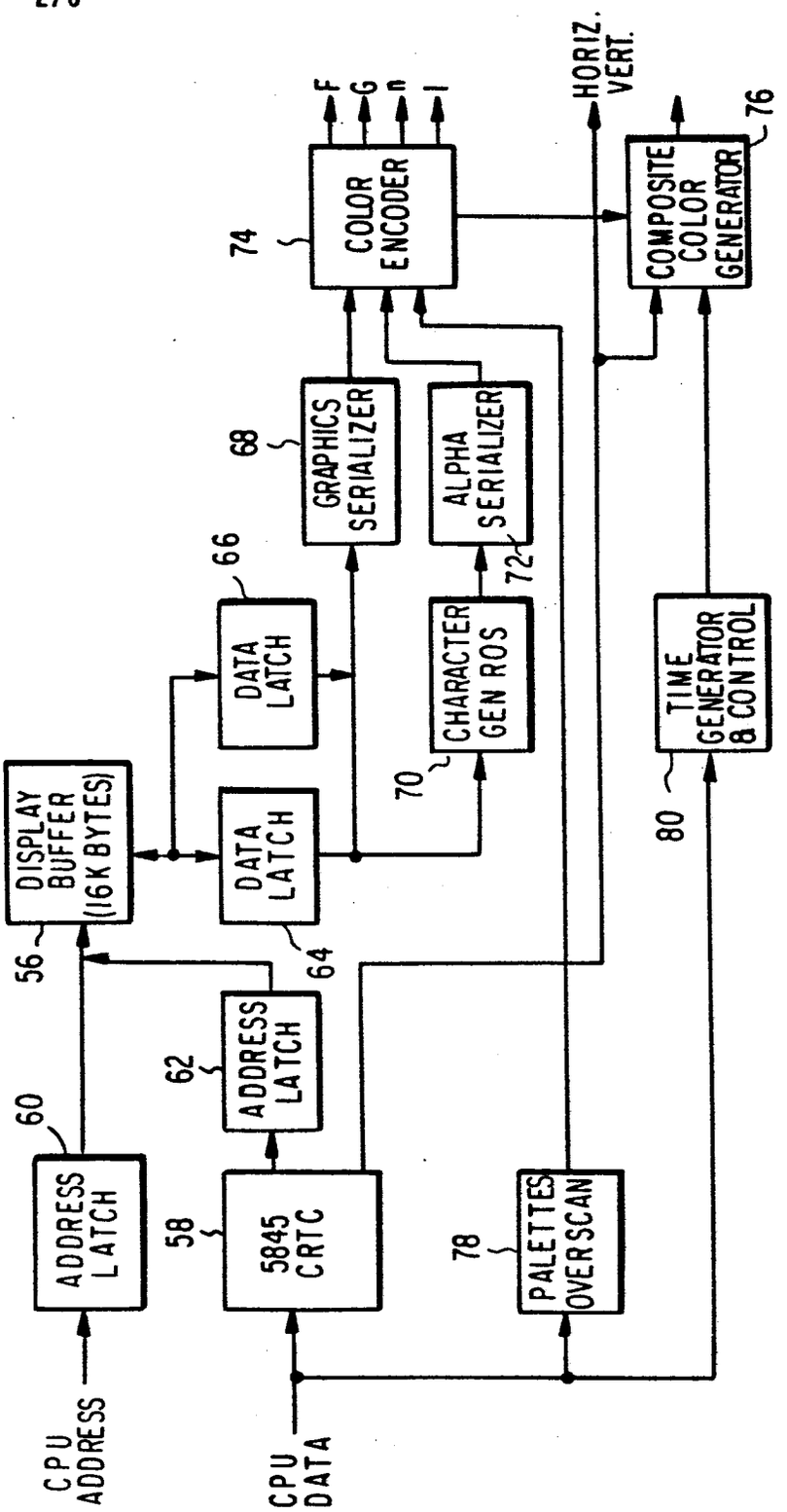

CPU ADDRESS
CPU DATA
ADDRESS LATCH
60
DISPLAY BUFFER (16K BYTES)
56
5845 CRTC
58
ADDRESS LATCH
62
DATA LATCH
64
DATA LATCH
66
CHARACTER GEN ROS
70
ALPHA SERIALIZER
72
GRAPHICS SERIALIZER
68
COLOR ENCODER
74
F
G
n
I
PALETTES OVERSCAN
78
TIME GENERATOR & CONTROL
80
COMPOSITE COLOR GENERATOR
76
HORIZ.
VERT.

FIG. 3

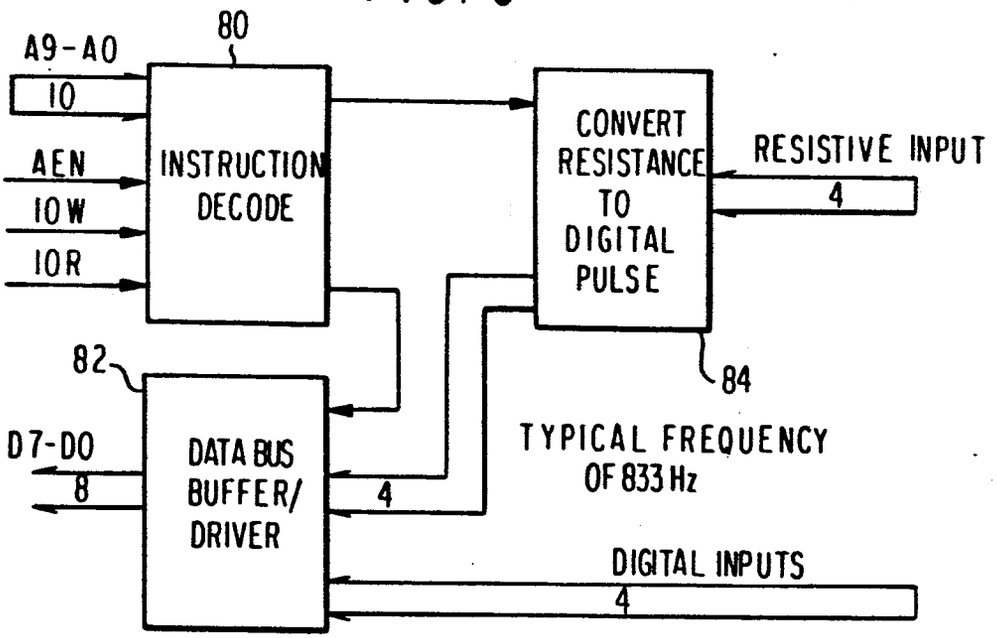

A9-A0
10
AEN
IOW
IOR
80
INSTRUCTION DECODE
CONVERT RESISTANCE TO DIGITAL PULSE
RESISTIVE INPUT
4
84
82
D7-D0
8
DATA BUS BUFFER/ DRIVER
4
TYPICAL FREQUENCY OF 833 Hz
DIGITAL INPUTS
4

FIG. 8

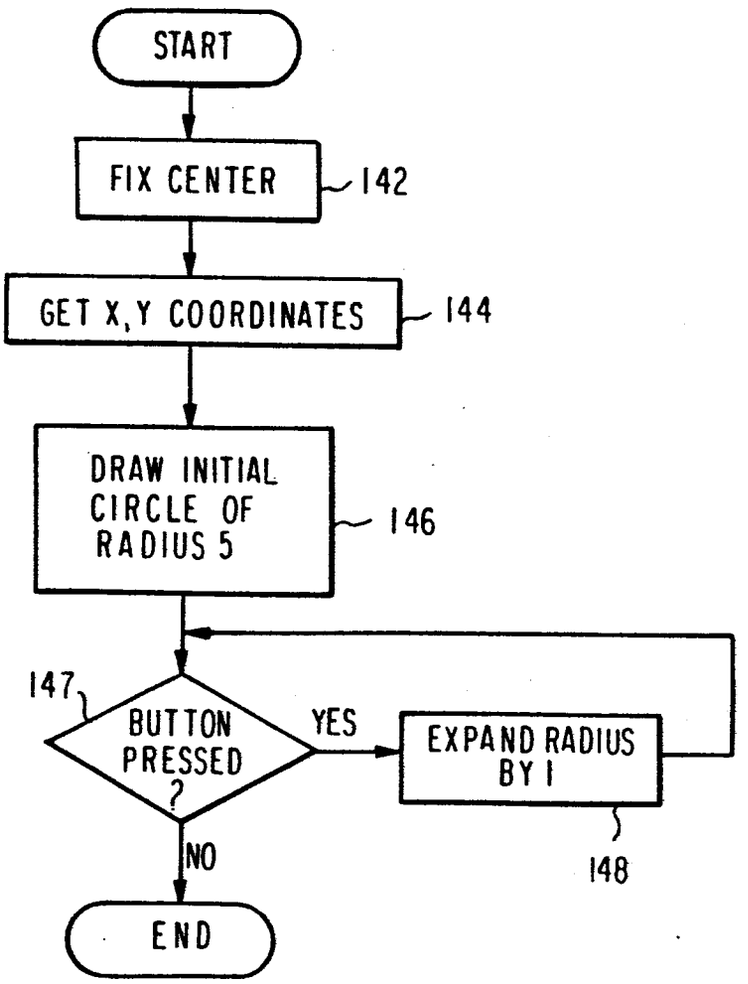

START
FIX CENTER
142
GET X,Y COORDINATES
144
DRAW INITIAL CIRCLE OF RADIUS 5
146
147
BUTTON PRESSED ?
YES
EXPAND RADIUS BY 1
148
NO
END

FIG.4

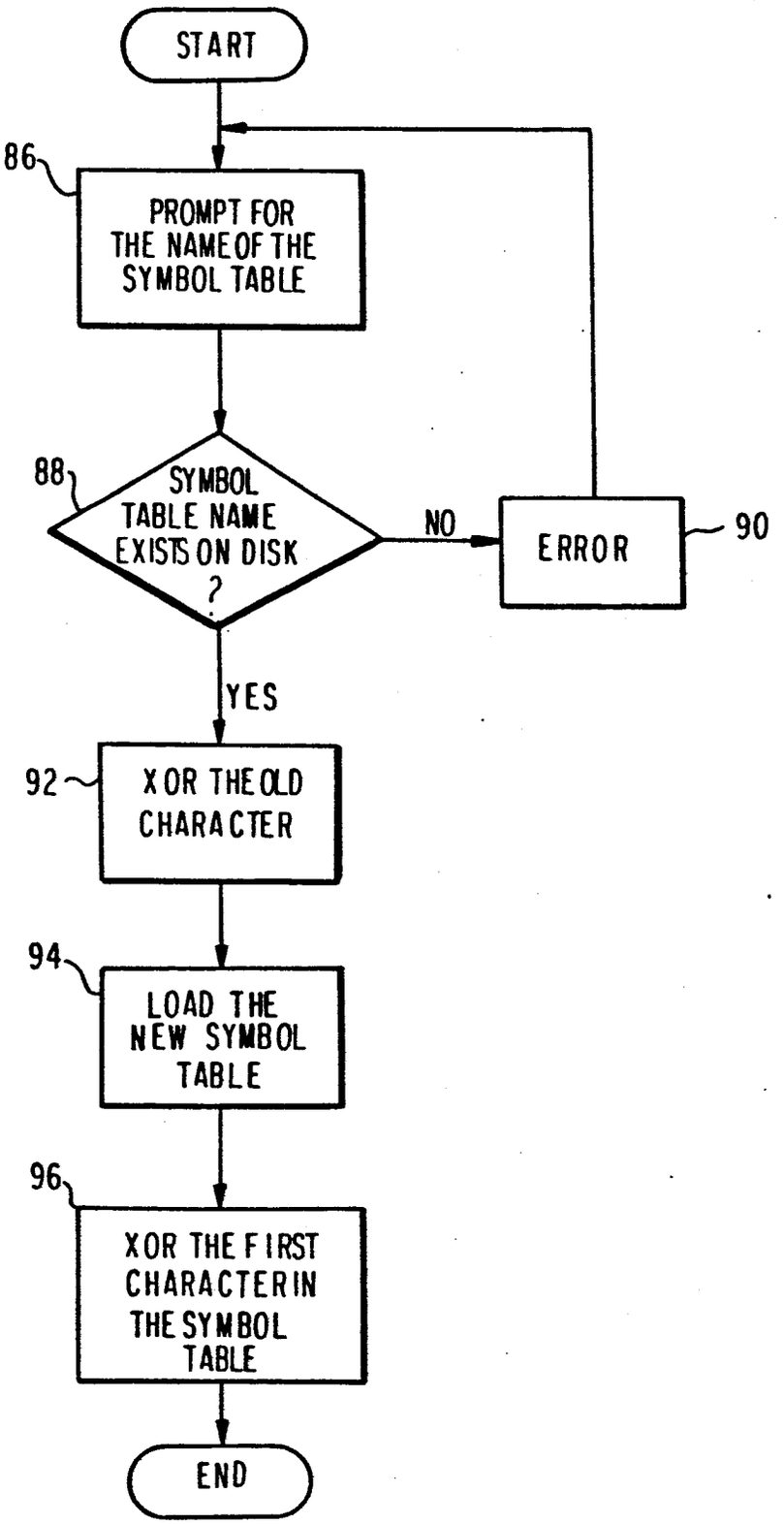

START
86
PROMPT FOR THE NAME OF THE SYMBOL TABLE
88
SYMBOL TABLE NAME EXISTS ON DISK ?
NO
ERROR
90
YES
92
XOR THE OLD CHARACTER
94
LOAD THE NEW SYMBOL TABLE
96
XOR THE FIRST CHARACTER IN THE SYMBOL TABLE
END

FIG. 5

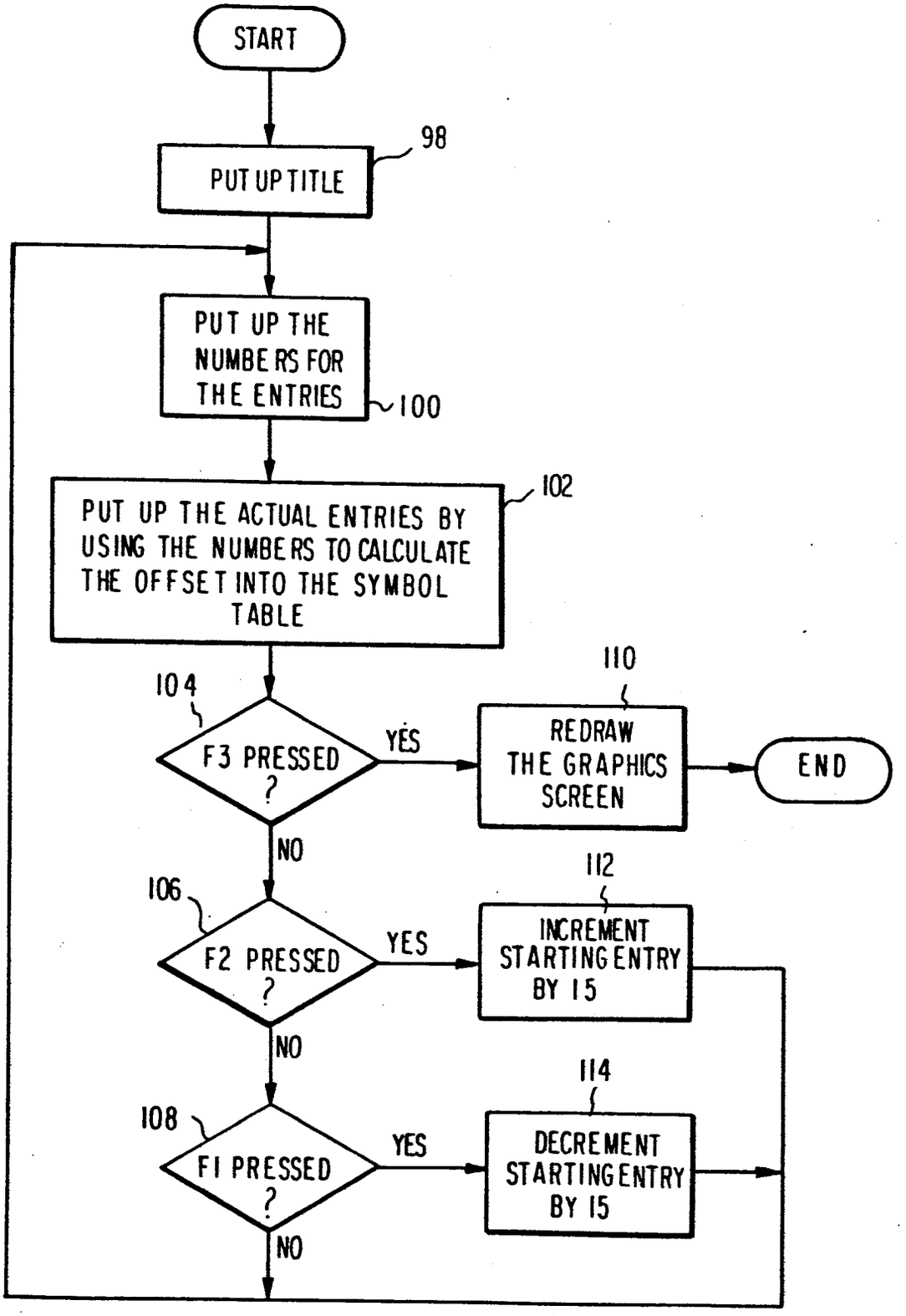

START
PUT UP TITLE
98
PUT UP THE NUMBERS FOR THE ENTRIES
100
PUT UP THE ACTUAL ENTRIES BY USING THE NUMBERS TO CALCULATE THE OFFSET INTO THE SYMBOL TABLE
102
104
F3 PRESSED ?
YES
110
REDRAW THE GRAPHICS SCREEN
END
NO
106
F2 PRESSED ?
YES
112
INCREMENT STARTING ENTRY BY 15
NO
108
F1 PRESSED ?
YES
114
DECREMENT STARTING ENTRY BY 15
NO

FIG. 6

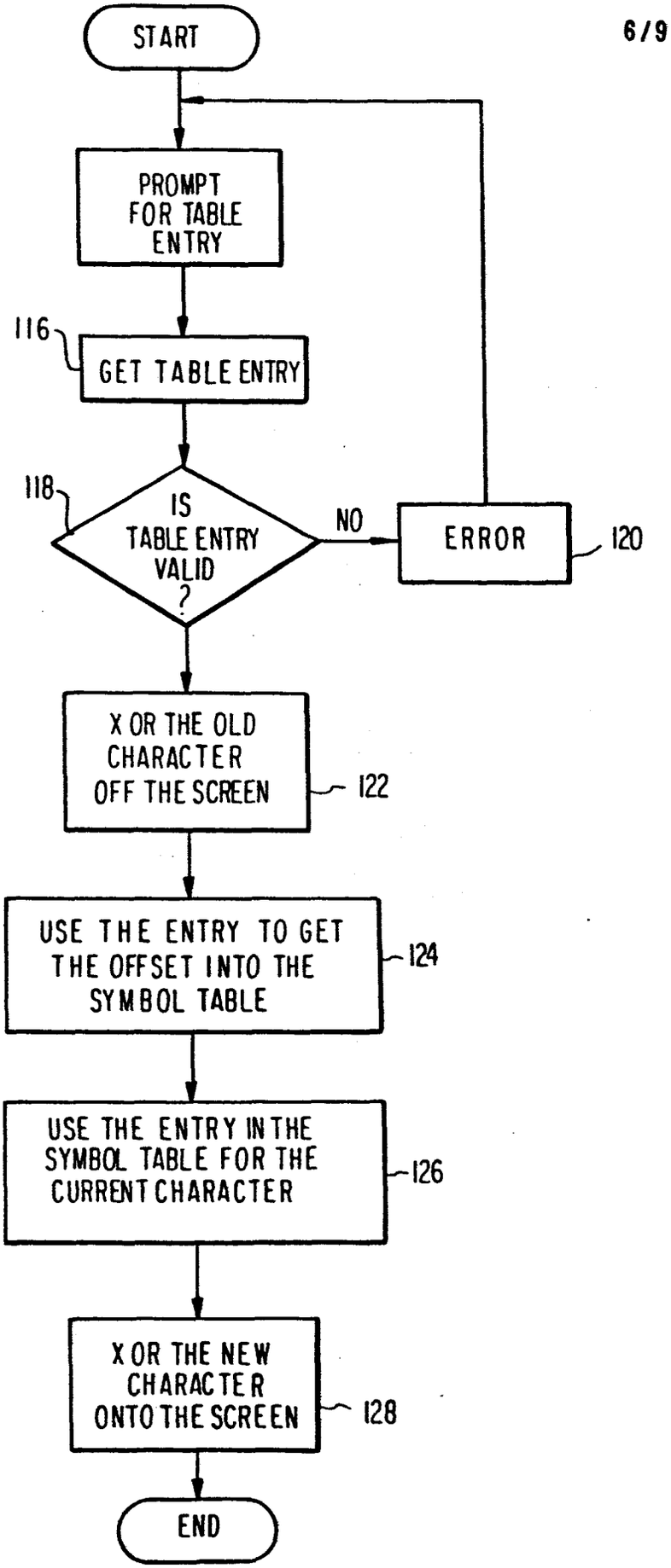

START
PROMPT FOR TABLE ENTRY
116
GET TABLE ENTRY
118
IS TABLE ENTRY VALID ?
NO
ERROR
120
X OR THE OLD CHARACTER OFF THE SCREEN
122
USE THE ENTRY TO GET THE OFFSET INTO THE SYMBOL TABLE
124
USE THE ENTRY IN THE SYMBOL TABLE FOR THE CURRENT CHARACTER
126
X OR THE NEW CHARACTER ONTO THE SCREEN
128
END

FIG. 10

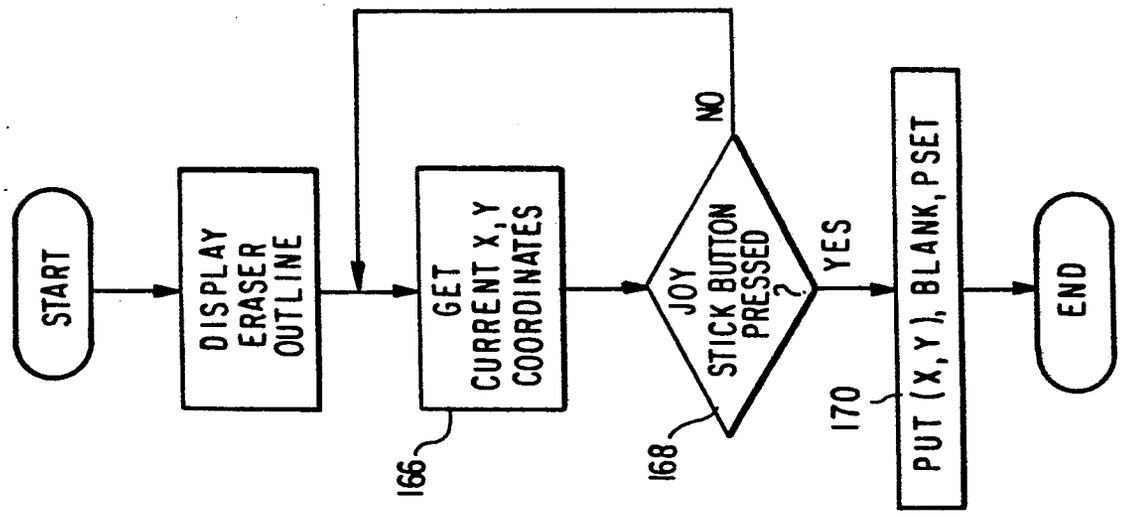

START
DISPLAY ERASER OUTLINE
GET CURRENT X, Y COORDINATES
166
JOY STICK BUTTON PRESSED ?
168
NO
YES
PUT (X, Y), BLANK, PSET
170
END

FIG. 7

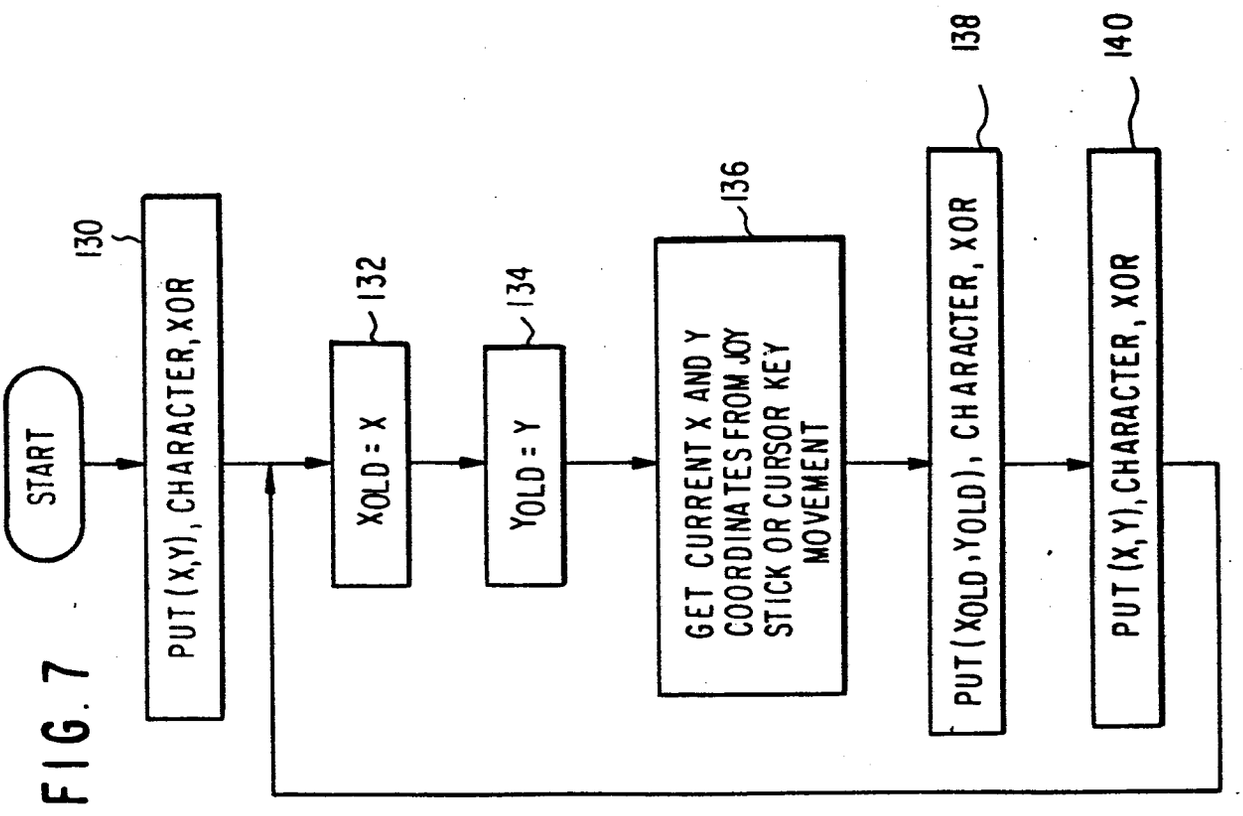
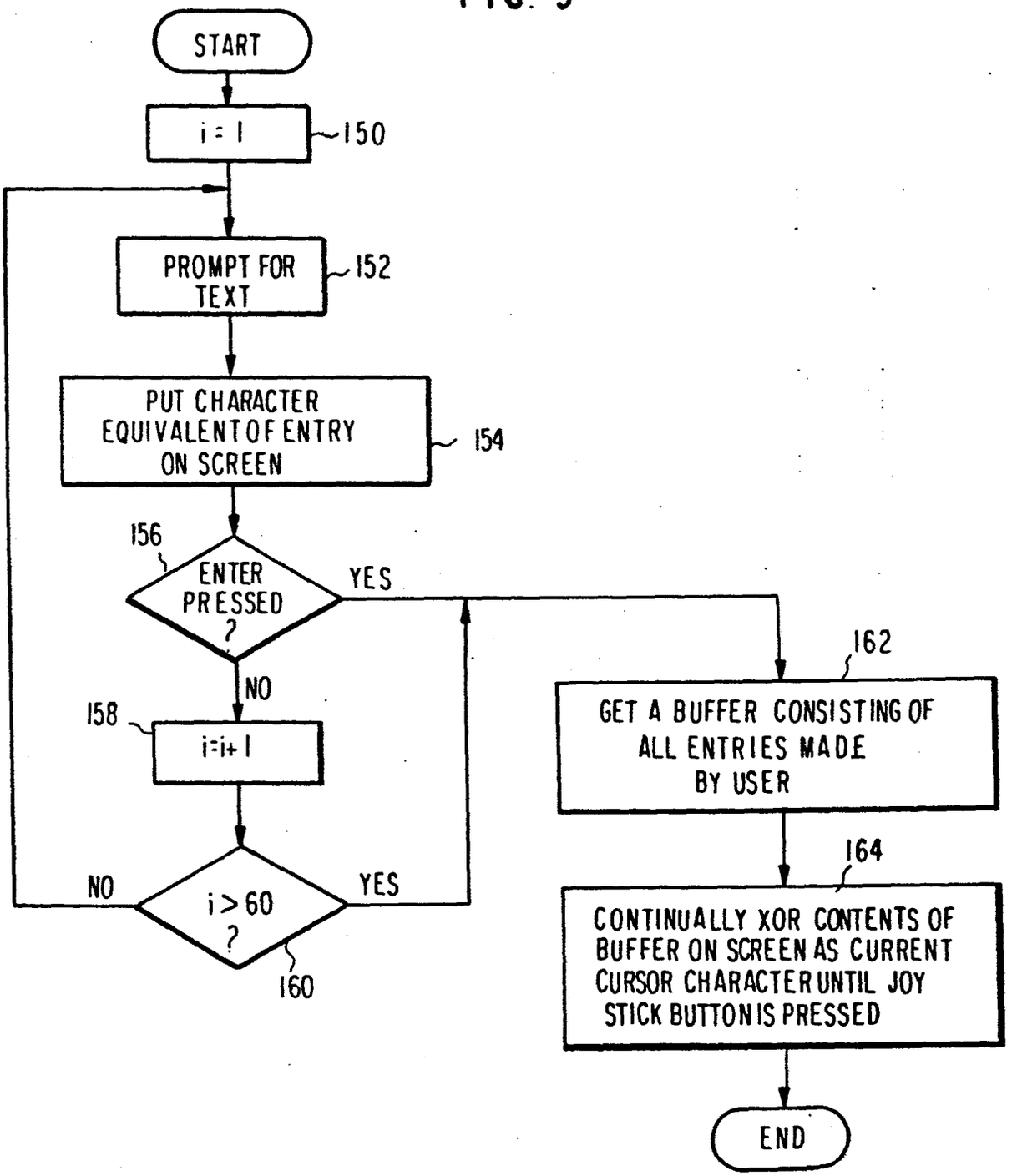
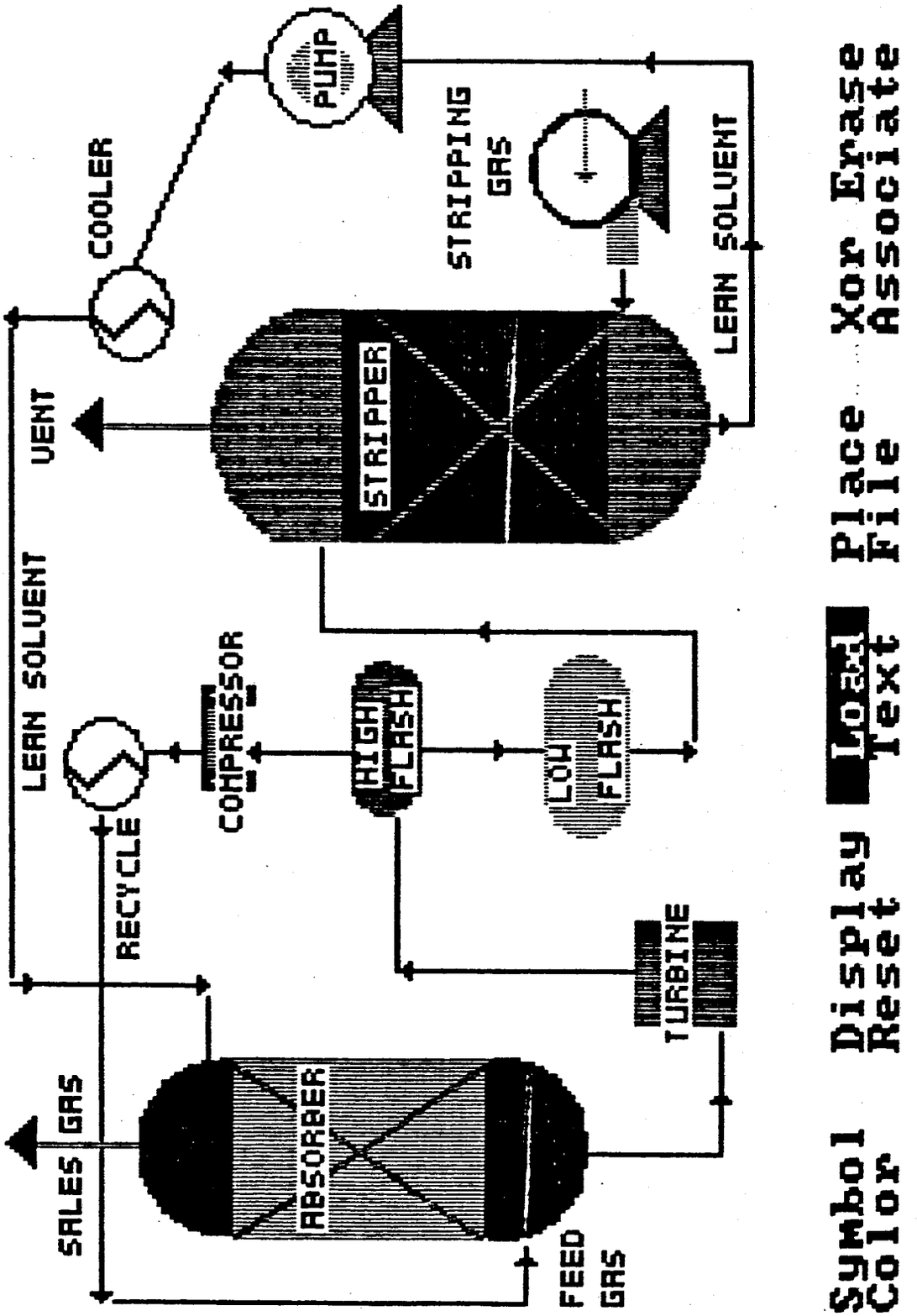

START
PUT (X,Y), CHARACTER, XOR
130
XOLD = X
132
YOLD = Y
134
GET CURRENT X AND Y COORDINATES FROM JOY STICK OR CURSOR KEY MOVEMENT
136
PUT (XOLD, YOLD), CHARACTER, XOR
138
PUT (X,Y), CHARACTER, XOR
140

FIG. 9


START
i = 1
150
PROMPT FOR TEXT
152
PUT CHARACTER EQUIVALENT OF ENTRY ON SCREEN
154
156
ENTER PRESSED ?
YES
NO
158
i=i+1
NO
i > 60 ?
YES
160
162
GET A BUFFER CONSISTING OF ALL ENTRIES MADE BY USER
164
CONTINUALLY XOR CONTENTS OF BUFFER ON SCREEN AS CURRENT CURSOR CHARACTER UNTIL JOY STICK BUTTON IS PRESSED
END



Symbol Display Load Place Xor Erase
Color Reset Text File Associate

FIG. 11