CONOGRAPHIC
CORPORATION

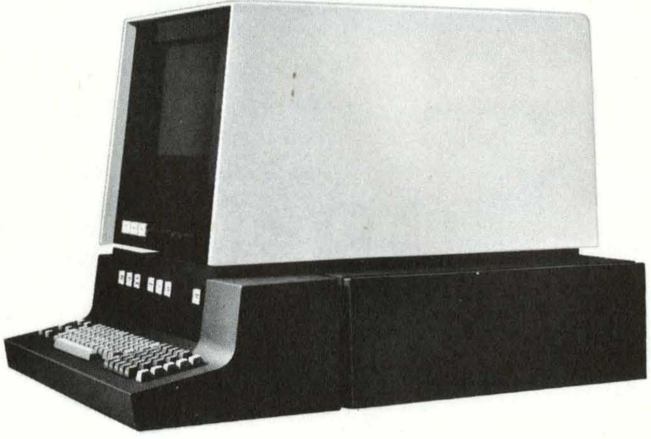
JOHN R. COFFMAN
MANAGER
FIELD SALES

6 GILL STREET • WOBURN, MASSACHUSETTS • (617) 935-7300

SYSTEM REFERENCE MANUAL

CONOGRAPHIC
CORPORATION

SYSTEM REFERENCE MANUAL



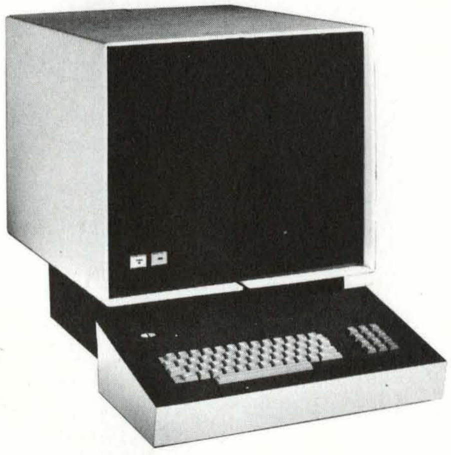
How to use the Conograph Display Systems

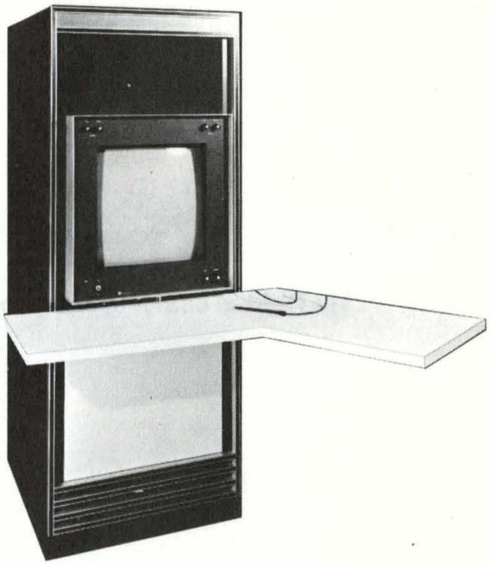
10

14

21

23





DIRECT COMMENTS CONCERNING THIS MANUAL TO

CONOGRAPHIC CORPORATION, 380 GREEN STREET, CAMBRIDGE, MASSACHUSETTS 02139 TEL. (617) 491-5820

May 1971

The Right to Change Specifications is Reserved

Written for Conographic Corporation by William English. Printed in the United States
SRM 1

Contents

1. INTRODUCTION 1-1
1.1 Logical Organization 1-3
1.2 Models and Options 1-5
1.3 Conographic Parameters 1-6

2. PROGRAMMING 2-1
2.1 Raster 2-1
2.2 Mode Commands 2-3
2.3 Long Graphic Mode 2-4
2.4 Short Graphic Mode 2-7
2.5 Symbol Modes 2-8
Italics 2-9
Symbol Scale 2-10
2.6 Function Commands 2-11
Receiver 2-12
CRT control 2-12
Position and computation commands 2-13
Scale factors 2-17
Cursors 2-20
Dash control 2-21
Intensity variation 2-22
Offset 2-22
Light pen 2-24
Keyboard bell 2-25
2.7 Symbol Subroutine 2-25
2.8 Buffer 2-27

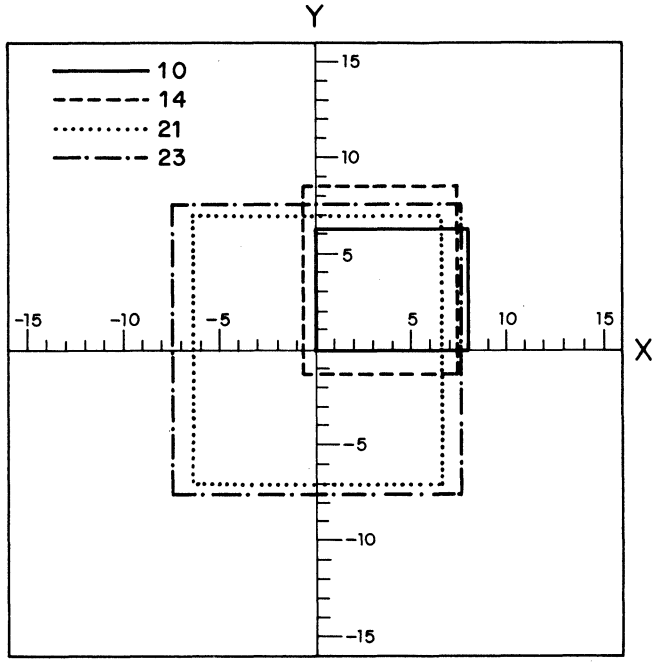
3. OPERATION 3-1
3.1 Keyboard and Console 3-1
3.2 Software 3-2

APPENDICES
A Installation A-1
B Command Codes B-1
ASCII character commands B-2
C Determination of Conographic Parameters C-1
Curves, type I C-1
Rotation C-3
Figures C-3

1 Introduction

The Conographic systems utilize the new technique of Conography to display general curves, figures, alphanumerics and other symbols by combining conic sections with the traditional straight lines. Four units are currently available, with model numbers that indicate the size of the screen diagonal in inches – 10, 14, 21 and 23. The Model 10 uses a storage tube and has relatively limited refresh capability; the others are nonstore displays and their much greater speed allows full refreshing. With the Model 10 is an acoustic pointer that can be used, either on the screen or with a separate tablet, for direct manual control of the CRT beam in generating arbitrary curves and figures. The larger models have a standard light pen, but a pointer with or without tablet may be used with them. Also available with the Model 10 is a hardcopy unit, which makes a permanent record on paper of the information displayed on the screen.

Physically every display consists of three parts connected by cables: CRT monitor, control unit and keyboard. These may simply be placed on a table as shown in the frontispiece, or mounted in a standard 19-inch rack with the keyboard protruding. Or the CRT monitor and keyboard may be placed in any arrangement on a table with the control unit mounted in a nearby rack. The CRT monitor contains the CRT and its associated deflection amplifiers, high voltage power supplies, and *z*-axis electronics; the control unit contains the display processor, interface, Conographic generator, and whatever read-only or core memory is included in the system.



DISPLAY RASTER

The data format used provides a 13-bit addressable raster that is 32 inches square in raster units of ¹⁄₂₅₆ inch. The actual visible window within the raster depends on the size of the display as illustrated here. The largest displays have three-dimensional capability, in which the addressable raster is a 32-inch cube; the visible depth is approximately equivalent to the other dimensions.

All data used in the system is in the form either of 8-bit bytes or 16-bit words. Basically the system has two plotting modes plus a control mode in which it executes only commands that change the mode (mode commands are always one word). The two plotting modes are graphic and symbol, but there are a number of variations of each. In the graphic modes the display processor operates on words. It can execute plotting and function commands, and can escape to control mode. Plotting commands are groups of words that actually cause the processor to plot vectors, curves, and figures. In long graphic mode, the format requires two words for a vector (relative or absolute), four for a curve, six for a figure (three-dimensional plotting requires half again the number of words for each case). In short graphic format, the parameters are packed two to a word and the processor is restricted to relative vectors and curves; the former requires one word, the latter two.

In the various symbol modes the processor handles a string of 8-bit bytes, each of which causes the symbol generator to perform some operation. This may be the displaying of a character or any other symbol, but it may also be the execution of some function like a tab or carriage return, entering or leaving italic mode, erasing the screen, or escaping to control mode. There is no fixed correspondence between the bytes (say as ASCII codes) and the operations of the symbol generator. Instead each byte simply addresses a location in a read-only memory in the generator, which location in turn refers to a subroutine consisting of short graphic plotting commands and function commands. Hence the actual result of giving a particular symbol byte depends only upon the subroutine associated with the ROM location corresponding to that byte. Moreover the operator can select among a number of different symbol sets or fonts simply by changing the set of locations to which the symbol codes refer. Any number of fixed fonts can be accommodated simply by increasing the size of the ROM, but through the addition of a subroutine controller and core memory, the user can generate any desired fonts and can change them at any time under program control.

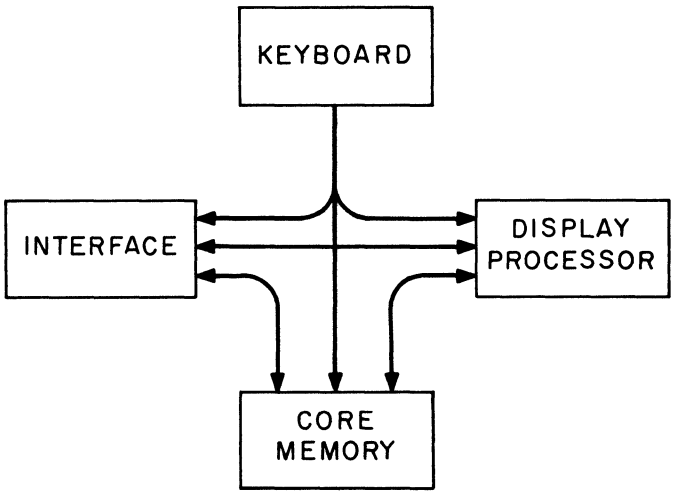
The function commands perform all of the operations necessary (other than mode changes) for overall control of the system, from selecting the CRT mode, setting and reading various operating parameters, or enabling the light pen, to repositioning the beam for a space or carriage return, or performing the computations necessary for horizontal and vertical tabbing. These commands can be executed in graphic mode as well as being executed from the subroutine memory in symbol mode. However not all function commands can be executed in all circumstances, as a given function may simply be meaningless in some particular situation.

Of the five symbol modes, four are alphanumeric. All symbols are plotted relative to a base beam position, but in alphanumeric mode the characters are plotted at eight times normal resolution: in other words the graphic subroutine commands position the beam in terms of raster units that are only $\frac{1}{2048}$ inch, allowing generation of much smoother figures in small areas. When displayed, the characters are expanded by a factor of four for better visibility (ten characters per inch). Besides the basic alphanumeric mode, there is also a protected alphanumeric mode in which the data strings supplied say by a computer in a time sharing facility cannot be modified from the keyboard. An option provides two modes that are equivalent to the basic and protected modes except that all displayed characters are rotated 90° counterclockwise.

The other symbol mode is template mode, which is identical to basic alphanumeric except that the graphic subroutine commands use normal raster units and are not subsequently expanded. This allows a string of data bytes to display items from a font or template of larger symbols, such as those used in drafting or design work (*eg* circuit and statistical symbols).

1.1 LOGICAL ORGANIZATION

The basic organization of a Conographic display is as illustrated here. The processor handles all necessary computations and governs the movement of internal information, in particular the supplying of plotting data to the Conographic generator, in response to incoming data. An interface connects the system to some external program source such as a computer or data communication line; the design of the interface depends of course on the type of external information setup. Through the interface, an external source can exercise immediate control over the display by means of direct communication with the processor, or can load data into a core memory for subsequent internal operation (the core memory is optional on the 10, standard on other models). Feedback from the processor through the interface is in the form of information supplied in response to interface signals or an interrupt produced by say the detection of a spot by the light pen. The operator at the keyboard can control the display directly and can transmit information through the interface or into core. With keyboard, pointer and light pen, the operator can generate graphic material, and through use of display software in a computer, be updating a computer-stored display file, which is in turn refreshing and modifying the material displayed on the screen. Or the computer can load a file into the internal core memory; then in a local keyboard editing mode, the operator can modify the core file, which both controls and responds to the processor; and finally, from the keyboard, the computer can be instructed to retrieve the edited file.

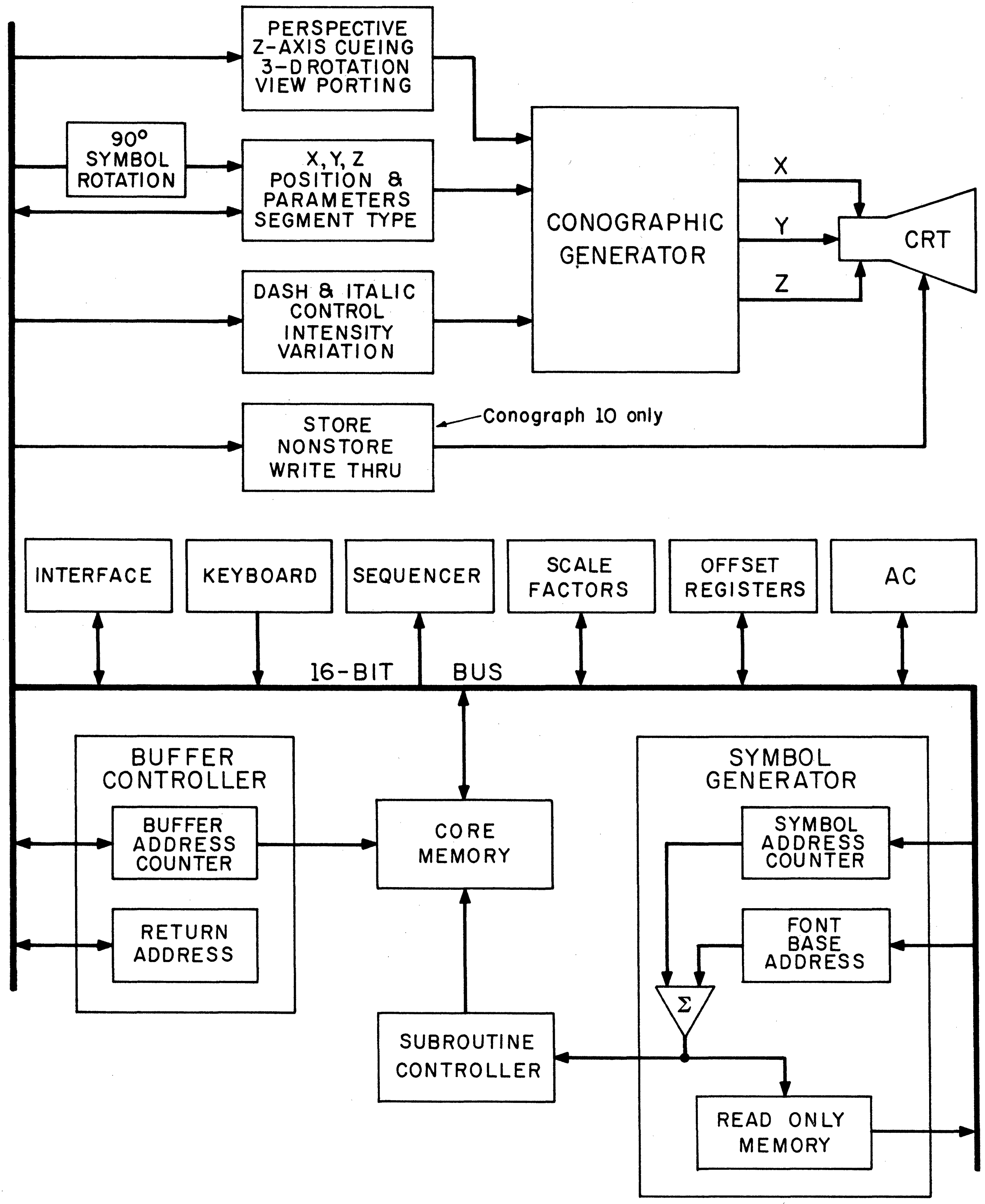


SIMPLIFIED DISPLAY ORGANIZATION

The illustration on the next page shows the detailed organization of the display. For operation from a data communication line the interface would include facilities to convert serial input into bytes (least significant bit first), and pairs of bytes into words (first byte on the left). Conversely, the transmitter would have to convert from parallel to serial. The interface may also contain a word counter for controlling the length of input and output, although it could be designed to terminate on some special character. For connection to a computer, the interface would probably handle full words through a 16-bit buffer, would have word and address counters for operation through a data channel directly to computer memory, and would contain such control circuits as busy and done flags, perhaps a light pen flag, and interrupt facilities.

If core memory and the buffer controller are included, the system can operate from a program in its own core of up to 65K words. The controller includes an address counter to govern the program sequence, a return address register for use in subroutine jumps, and facility for keeping a cursor to orient the operator. *Eg* in symbol mode, the cursor would ordinarily be an underscore that would mark the character position on the screen (and hence the buffer location) that would be affected by any action taken at the keyboard.

The symbol generator includes read-only memory, an address counter, and a base address register



DISPLAY ORGANIZATION

that selects a font. For each symbol the address counter receives an 8-bit byte that selects one location out of the first 256 in the font; and unless the symbol generation requires only a single command, the counter is then reloaded by a jump to control the subroutine sequence. The same 8-bit codes are used for any font, as the counter contents are added to the base address; thus all addressing is to a preselected font and the fonts are variable in length. The counter is twelve bits, the base register is sixteen. Hence any font is limited to 4K locations, and the total symbol memory is limited to 65K. If core memory is included, a subroutine controller can reference core instead of ROM for arbitrarily defined symbol fonts that can be changed at will. The buffer and subroutine controllers share a common core memory, so the programmer must keep track of the parts of core used for each. The total combined memory capacity can be greater than 65K, but subroutine capacity is limited to 65K – in other words any part of core can be used by the buffer, but the part of core that overlaps the ROM (*ie* that has the same addresses) can be used only by the buffer.

The display is organized around a 16-bit data bus in which the least significant bus line corresponds to the raster unit. The sequencer interprets input from whatever source, and controls the movement of information and selection of modes. At the end of each operation (an entire subroutine in the symbol generator is regarded as a single operation), the sequencer determines what input source to respond to next. The keyboard has priority over the interface (this cannot affect the interface adversely as the keyboard is usually much slower), and either can interrupt the buffer. Information is supplied to the Conographic generator through buffers for beam position, curve parameters, and segment type. The scale factors, offset registers, and dash control implement features described in the next section. The accumulator AC is used by the sequencer for computations that are invisible to the programmer, but it can also be used by function commands to control such operations as horizontal and vertical tabbing. Specification of the CRT mode – store, nonstore, write thru – is required on the Conograph 10 only.

1.2 MODELS AND OPTIONS

All displays include generators for points, vectors, curves, figures and symbols, and have italics for symbols and input scaling for short graphic commands. These short form commands have only six magnitude bits of plotting information for each parameter, and the scale factor determines the order of magnitude – the least significant bit can vary by powers of two from $\frac{1}{256}$ inch to $\frac{1}{4}$ inch, the most significant bit from $\frac{1}{8}$ inch to 8 inches.

The core memory with buffer and subroutine controllers has already been described. The display processor extension includes vertical and horizontal tabbing, 90° counterclockwise rotation of symbols, object scaling, and dot and dash control. With object scaling, all relative plotting commands are multiplied by a scale factor of $\frac{1}{8}$ to 8 in steps of $\frac{1}{8}$; hence the size of any object on the screen can be varied up or down without changing its position (separate factors for each dimension allow change in shape as well). The dot and dash control allows the program to plot dashed curves, wherein the lengths of both the dashes and the spaces between them are program selectable.

Other options are the following.

Intensity variation. Allows the program to select a different intensity, from among sixteen, for each curve.

Offset. The selection of a fixed position offset (which can be different for different dimensions) to move the entire display to a different part of the screen. In particular this allows the programmer to move the visible window around in the addressable raster by adding the offsets to the program parameters – in other words by moving any hidden part of the addressable raster into the window.

Picture scaling. Multiplication of all plotting commands by a scale factor of ⅛ to 8 in steps of ⅛, allowing variation in the size of the entire programmed picture, *ie* both object size and position. Different scale factors can be selected for different dimensions.

Three-dimensional rotation. The ability to rotate objects on the screen through any dimension.

Perspective. The ability to show three-dimensional objects on the two-dimensional screen in true perspective. (Error less than 2%.)

Z-axis cueing. The use of additional plotting information to produce realistic three-dimensional figures by using a variable intensity to represent depth (*ie* beam motion perpendicular to the screen).

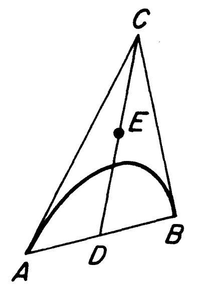
View porting. The ability to reduce and clip a number of full screen pictures and show them simultaneously in different parts of the screen, or even to reduce and clip a single picture and show it simultaneously in a number of views. *Eg* a three-dimensional object may be reduced to one-fourth size and, in the four screen quadrants, shown front view, top view, side view, and some angular view in perspective. (Error less than 2%.)

The only features standard to the Conograph 10 are those that are basic to all machines. As options the Model 10 can have additional read-only memory for extra symbol fonts, 4K core memory with buffer controller and/or subroutine controller, and the display processor extension. The Model 14 has a 4K core memory with buffer controller as standard equipment, and can have the subroutine controller, picture scaling, and the processor extension, which in this case includes offset and intensity variation. The 21 has as standard everything available to the 14, and as options can have perspective, three-dimensional rotation, *z*-axis cueing, and multiple view porting. On the 23 all the listed features are standard, except additional memory and peripherals, which are available optionally on all machines.

1.3 CONOGRAPHIC PARAMETERS

Drawing a vector requires the same information as with any ordinary display: upon receiving the coordinates of the endpoint or the components of the vector, the Conographic generator simply plots points from the present beam position along a straight line to the given end point or along the path defined by the components for the distance specified by them. The number of components or coordinates that must be given depends of course on whether the display is in two or three dimensions.

To draw Conographic curves and figures requires specification of certain parameters and the segment type. Consider the curved segment AB illustrated here. If the segment is such that it lies entirely on one side of the chord joining its end points, it has no interior points of inflection, the tangents to the segment at its end points intersect at C on the same side of the chord as the curve lies, and it does not extend further from the chord than the midpoint E of the chord bisector CD, then it is a segment of Type I. Either or both end points may be inflection points with respect to the extension of the curve, *eg* by plotting additional Type I segments. If the tangents at the end points coincide, *ie* the slopes are both equal to the slope of the chord, the curve is a degenerate case of a Type I segment and is plotted by the generator as a straight line (in other words it is equivalent to a vector).

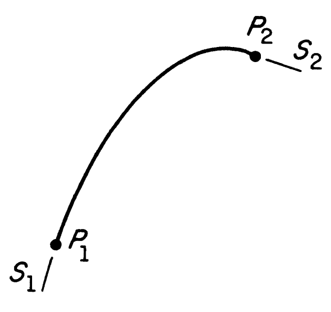


If the slopes at the end points are equal but do not equal the slope of the chord (*ie* the tangents are parallel but not coincident), the curve is a Type II segment. Such a segment must still lie entirely on one side of the chord connecting the end points and have no interior point of inflection, but it must have equal slopes at the end points. A Type II segment may be plotted as two consecutive Type I segments, but the consecutive plotting of two arbitrary Type I segments does not necessarily produce a Type II segment.

A Type IV segment is an ellipse, or in the special case, a circle. Although a Type IV segment can be constructed of four Type I segments, it is not simply such a construction as any numbers of Type I segments can be used to produce arbitrary closed curves. A Type III segment is the extension of a Type II segment by a Type I segment along the path of the ellipse defined by the Type II segment, which is half an ellipse.

For the Conographic generator the path of a segment of any type is defined by Conographic parameters, two for each dimension. Segments of all types are divided into two categories depending upon the plotting data given for them. A segment whose trajectory begins at the present beam position is referred to as a "curve" regardless of its type, *ie* even if it is of Type IV (a closed figure). On the other hand a trajectory generated by a plotting command that specifies a center point and then draws the trajectory about that point is referred to as a "figure" even if it is only a Type I segment. A curve is regarded as relative and requires specification only of parameters and segment type; in addition to that information, a figure requires specification of a center point, and is regarded as absolute or relative depending upon whether the center point is specified by absolute coordinates or by vector components from the present beam position (of course neither the center point nor a vector defining it is actually displayed). The center point of a figure is the actual center of the Type IV segment (*ie* the ellipse) defined by the given parameters; and a Type I, II or III segment with the same parameters lies on that ellipse. Generally Type I segments are drawn as curves and ellipses are drawn as figures, but this is not necessary. Thus a single curve plotting command can display the Type I segment defined by the parameters, or can extend it into a segment of any other type including a complete ellipse. Similarly a plotting command for a figure can be limited to displaying half of the ellipse or a Type I or Type III segment of it (in all cases starting from the same point).

Any two-dimensional Conographic segment is uniquely determined by specifying a segment type and four parameters, J, K, L and M, plus positioning information – the current beam position for a curve, the center point for a figure. For three dimensions an additional pair of parameters, N and P, are required. Although the parameters can be calculated for any arbitrary curve, the usual procedure is to calculate them from fixed sets of formulas for the different types of curve characteristics that usually are relevant to graphics being generated on a display screen; *eg* an arbitrary curve being produced by a sequence of Type I incremental segments. A typical situation is the plotting of a Type I segment to fit between two given end points with specified slopes at those end points as shown at the right. For convenience, define a and b as the components of the chord connecting the end points, *ie*



$$a = X_2 - X_1$$

$$b = Y_2 - Y_1$$

where X_n and Y_n are the coordinates of the point P_n. Using these quantities and the slopes S_1 and S_2

at P_1 and P_2 respectively, the parameters can be computed from these formulas:

$$\begin{aligned} J &= \frac{S_1 a - b}{S_2 - S_1} \\ K &= a + J \\ L &= S_2 J \\ M &= b + L \end{aligned}$$

Another typical requirement is a circle of radius R about some given center point. This would require the plotting command for a figure wherein the programmer specifies a Type IV segment, gives the center point, and specifies parameters defined as follows:

$$\begin{aligned} J &= R \\ K &= 0 \\ L &= 0 \\ M &= R \end{aligned}$$

For curve plotting accuracy equivalent to that used in initial beam positioning, the parameters must be given with the same resolution (*ie* number of bits) used to represent coordinates and components.

2 Programming

This chapter describes all of the mode, plotting and function commands for the Conographic displays. The format of each command word is shown in a box in which numbers represent fixed bits, and letters represent quantities that must be supplied by the programmer. Where detail is required, individual bits are shown; otherwise octal digits represent the contents. A word made up of a pair of 8-bit bytes is shown as two 3-digit octal codes, each representing eight bits; but the 6-digit octal code for the entire 16-bit word (where the left digit of 0 or 1 represents only the left bit) is shown at the right of the box. Stacked boxes represent commands requiring more than one word.

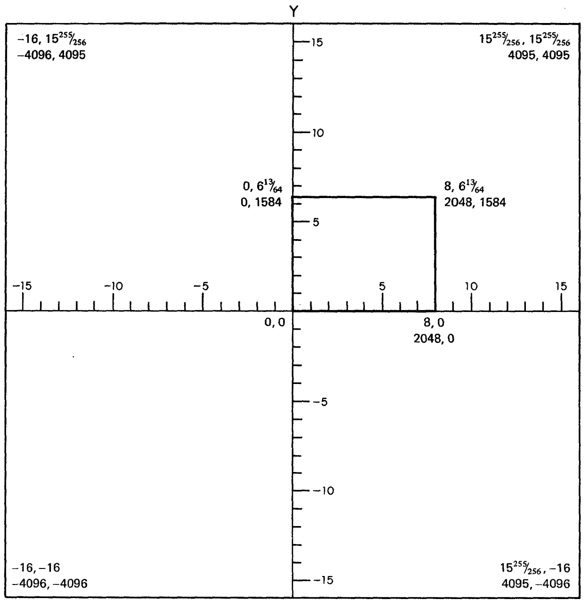
Every byte contains eight bits, but ASCII characters are uniquely determined by only seven – the eighth (left) bit is ordinarily used for parity in data communications. In the description of any command that has a fixed right byte, the 7-bit ASCII character for the seven low order bits of the byte follows the name of the command. When giving the command, the program must supply the correct 8-bit byte. At the keyboard the operator can do this simply by selecting the appropriate parity when typing the character.

Throughout this manual all numbers representing command words, register contents, codes and addresses are always octal. On the other hand the ordinary use of numbers in the text to specify raster units, word and byte lengths, bit values, etc. employs standard decimal notation.

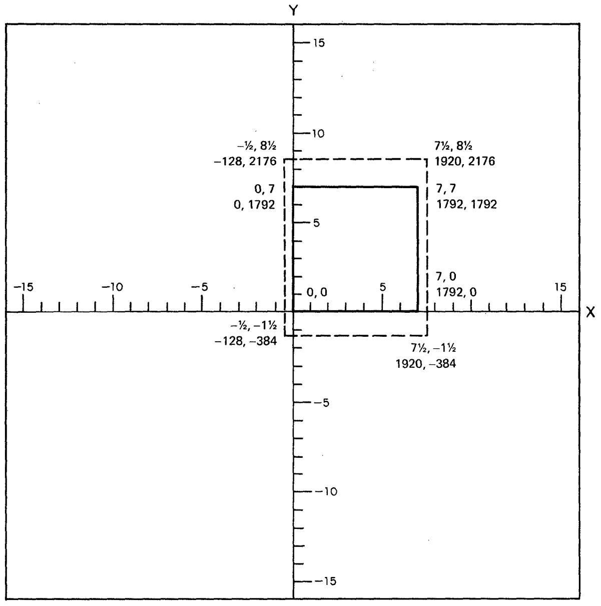
The system assumes the use of the twos complement convention for binary numbers used to specify plotting parameters, word counts and the like. With this convention positive numbers are represented as ordinary binary numbers with a 0 in the sign bit (if any). The negative of a number is represented by its twos complement, *ie* by taking the logical complement of the number including the sign, and adding 1 to the result. Note however that the twos complement applies only to that part of a word that represents the number for a parameter or component. *Eg* a 16-bit word may contain a 13-bit component and several control bits; for motion in the opposite direction, only the thirteen component bits are negated – the control bits must remain the same.

2.1 RASTER

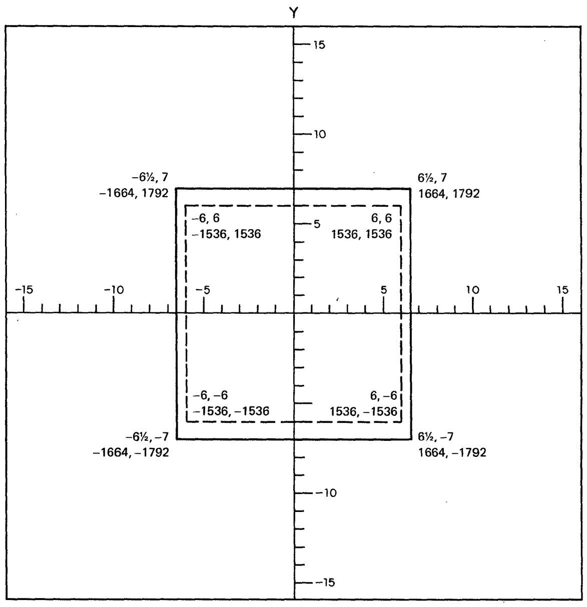
For every display the addressable raster is 32 inches square with 8192 beam positions along each edge. In each dimension there are thus 8191 raster units of $\frac{1}{256}$ inch each. The twos complement representation for negatives allows one more negative number than there are positive numbers, so the raster is actually 31 $\frac{255}{256}$ inches on an edge. The illustration on the next page shows the visible windows, divided into quality areas and fringe areas where applicable, in relation to the addressable raster for each of the displays.



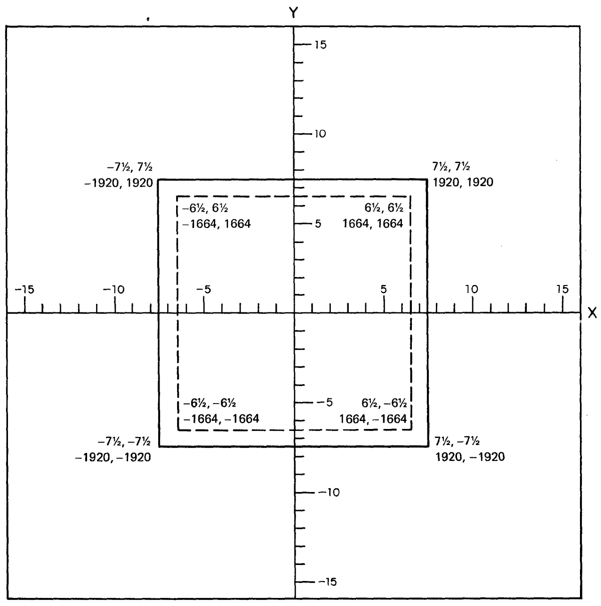
CONOGRAPH 10



CONOGRAPH 14



CONOGRAPH 21



CONOGRAPH 23

Coordinates given in inches and raster units. Solid line indicates quality area, dashed line shows fringe area.

RASTER CHARACTERISTICS

Although the program specifies the beam position with thirteen bits resolution, all computations in the display processor are kept in sixteen bits. Of the three extras, one is of higher order than the plotting data, the other two are of lower order. When objects are displayed using a scale factor less than unity, all detail below the raster unit is lost, but carrying the two extra low order bits in computations prevents the dropped bits of the commands from accumulating into a significant error. Thus in a string of relative commands, the actual beam position keeps catching up to the program position every time the extra low order bits accumulate into a raster unit.

The extra high order bit means that the actual beam positioning raster is 64 X 64. Hence if the program carries the beam off one edge of the addressable raster, the beam continues in the same direction even though the program may wrap around. The beam will wrap around only upon crossing the edge of the positioning raster, so the program must cross the addressable raster twice to wrap the beam around once.

Although addressing and beam position are limited only by the extent of the corresponding rasters, there are limitations on visible beam motion inherent in the analog circuitry. The beam can be moved any distance if only the end point is displayed or the entire motion is blank (no intensification). But for a continuous or dashed curve, the maximum component or parameter is 2047 raster units. In a single visible command, every component is actually interpreted modulo 2048 as far as beam intensification is concerned. *Eg* an attempt to draw a 9-inch horizontal vector would produce a visible line only one inch long despite the fact that the beam would move an additional eight inches (2048 raster units). This limitation does not depend on the shape of the screen; *eg* it is the same in both dimensions on the Model 10 even though that model has only 1584 raster units vertically. Note that this does not mean that figures can be no more than eight inches in diameter – the limitation is on components of motion. The maximum radius or semiaxis of a visible circle or ellipse is actually $8\sqrt{2}$.

2.2 MODE COMMANDS

Following an escape from one mode, the processor can place the system in any other. In graphic mode an escape occurs whenever the code 300 is encountered in the left half of a word. Such an escape can interrupt the command stream even in the middle of a multiword command, which is then ignored. Upon encountering an escape the processor places the system into the mode specified by the code in the right half of the same word that produced the escape. In other words a switch is made to a different mode by a single word containing 300 in the left byte, and this single word is both the escape and the mode command.

In symbol mode there is no fixed code for escape; the code that produces the escape is simply whichever one is set up to call the escape function command in the subroutine memory. When an escape occurs, the processor interprets the next byte in the symbol string as a right byte and enters the mode specified by it. However, any 300s inserted between the escape code and a legitimate mode code are ignored. Thus if the symbol bytes are being taken from words supplied through the interface, proper orientation can easily be achieved even when the escape code comes in a right byte; the escape is just followed by a graphic mode command, *ie* one with 300 in the left half.

Besides the modes described in Chapter 1, there are two other graphic modes and a no-op mode. There are extra codes for special forms of the long and short graphic modes; at present these have no function but can be implemented for any special circumstance that may be required (such as operation from 7-bit bytes). The no-op mode command causes the processor to ignore all incoming words until another mode command appears. Thus an extensive display file can include display routines, which are in their proper places in the file, but which can be performed or not entirely at the discretion of the programmer each time the file is run. Any such routine is effectively shut out by preceding it with a no-op command, but it can be displayed at any time simply by substituting an appropriate mode command for the no-op.

The 300 code is the ASCII character @ with even parity.

Enter Long Graphic	**FS**	300	234	140234
Enter Special Long Graphic	**GS**	300	235	140235
Enter Short Graphic	**RS**	300	236	140236
Enter Special Short Graphic	**US**	300	237	140237
Enter Alphanumeric	**DC2**	300	222	140222
Enter Alphanumeric Rotated	**FF**	300	214	140214
Enter Protected Alphanumeric	**DC3**	300	223	140223
Enter Protected Alphanumeric Rotated	**CR**	300	215	140215
Enter Template	**VT**	300	213	140213
Enter No-op	**SUB**	300	232	140232

2.3 LONG GRAPHIC MODE

In this mode, besides being able to escape and to execute all of the function commands described in later sections, the processor can execute plotting commands for all types of curves in all visual forms, with parameters that are the full thirteen bits resolution. All plotting commands use a multiword format wherein the first bits (left) of the first word specify the command type.

The left three bits of the second word are of the form *PDI* where a 1 in bit *I* selects the intensity level specified by the intensity register. If the intensity variation option is not included or bit *I* is 0, the beam is intensified at the normal level. The other two bits select the type of visual plotting as follows:

PDI	*Visual Characteristics*
00–	Continuous intensification
01–	Dashed line, as determined by dash control [§2.6]
10–	Point – only end point visible
111	Blank – nothing visible

In the final case bit *I* must be 1 so that 0s in the remaining bits cannot possibly cause the left byte to be interpreted as an escape. This does not really select an intensity level, since in the blank case there is no intensification anyway.

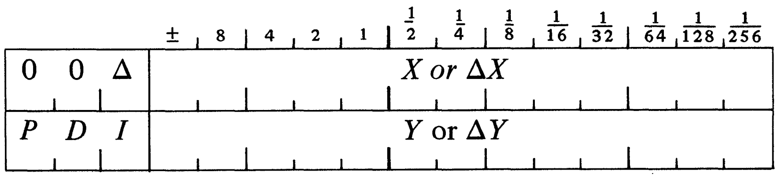
In the commands for curves and figures, the left two bits, *a* and *b*, of the final word (for two dimensions) specify the type of Conographic segment as follows:

abx	*Segment Type*
00–	I
01–	II
10–	III
111	IV

where again in the final case the third bit must be 1 to guard against an inadvertent escape.

Plot a Vector
(long format)

Bit value in inches



From the present beam position, plot a vector to the point *P* where *X* and *Y* are the absolute coordinates of *P* if Δ is 0, or Δ*X* and Δ*Y* are the components of *P* relative to the present position if Δ is 1. Regulate the visual characteristics of the vector as specified by *PDI*.

Plot a Curve
(long format)

Bit value in inches

			±	8	4	2	1	$\frac{1}{2}$	$\frac{1}{4}$	$\frac{1}{8}$	$\frac{1}{16}$	$\frac{1}{32}$	$\frac{1}{64}$	$\frac{1}{128}$	$\frac{1}{256}$
1	0	1							J						
P	D	I							K						
0	0	0							L						
a	b	x							M						

From the present beam position plot the curve defined by parameters J, K, L and M, to the extent indicated by ab and with the visual characteristics specified by PDI.

Plot a Figure
(long format)

Bit value in inches

			±	8	4	2	1	$\frac{1}{2}$	$\frac{1}{4}$	$\frac{1}{8}$	$\frac{1}{16}$	$\frac{1}{32}$	$\frac{1}{64}$	$\frac{1}{128}$	$\frac{1}{256}$
0	1	Δ							X_0 or ΔX_0						
P	D	I							Y_0 or ΔY_0						
0	0	0							J						
0	0	0							K						
0	0	0							L						
a	b	x							M						

Centered on the point P_0, plot the figure defined by J, K, L and M, to the extent indicated by ab and with the visual characteristics specified by PDI. If Δ is 0, X_0 and Y_0 are the absolute coordinates of P_0; if Δ is 1, ΔX_0 and ΔY_0 are the components of P_0 relative to the present beam position.

Remember that the largest component for any visible beam movement (continuous or dashed) is 2047 raster units ($8 - \frac{1}{256}$ inches). Thus in the specification of a component for a relative vector or the Conographic parameters for any curve or figure, the 8s bit must be null (*ie* equal to the sign). In the case of an absolute vector, a coordinate may have any value provided it does not result in moving the beam more than 2047 raster units from the present position. Of course, none of these restrictions applies to point or blank commands.

For a three-dimensional display, each long graphic command must contain additional information for the z dimension (the extra word or words follow the words for two dimensions as shown above and each contains thirteen bits of plotting information – there are no control bits). A vector command requires a third coordinate or component, a curve requires *N* and *P* parameters, and a figure requires three extra words for the third coordinate or component of the center point and the *N* and *P* parameters.

2.4 SHORT GRAPHIC MODE

This mode, like long graphic, allows escaping and the execution of all function commands, but plotting is limited to relative vectors and curves with only six magnitude bits for the specification of any component or parameter. The actual values of the magnitude bits in a short graphic command depend upon the graphic scale factor. Two of the function commands are used to specify and read this factor.

Set Graphic Scale Factor **6**

301	266			140666
000	0	*S*	0	0000*S*0

Set the scale factor for short graphic commands to *S* so that the six magnitude bits of such commands have the following values.

Bit Values in Inches

S	*MSB*					*LSB*	*Maximum*
0	$\frac{1}{8}$	$\frac{1}{16}$	$\frac{1}{32}$	$\frac{1}{64}$	$\frac{1}{128}$	$\frac{1}{256}$	$\frac{1}{4}-\frac{1}{256}$
1	$\frac{1}{4}$	$\frac{1}{8}$	$\frac{1}{16}$	$\frac{1}{32}$	$\frac{1}{64}$	$\frac{1}{128}$	$\frac{1}{2}-\frac{1}{128}$
2	$\frac{1}{2}$	$\frac{1}{4}$	$\frac{1}{8}$	$\frac{1}{16}$	$\frac{1}{32}$	$\frac{1}{64}$	$1-\frac{1}{64}$
3	1	$\frac{1}{2}$	$\frac{1}{4}$	$\frac{1}{8}$	$\frac{1}{16}$	$\frac{1}{32}$	$2-\frac{1}{32}$
4	2	1	$\frac{1}{2}$	$\frac{1}{4}$	$\frac{1}{8}$	$\frac{1}{16}$	$4-\frac{1}{16}$
5	4	2	1	$\frac{1}{2}$	$\frac{1}{4}$	$\frac{1}{8}$	$8-\frac{1}{8}$
6	8	4	2	1	$\frac{1}{2}$	$\frac{1}{4}$	$16-\frac{1}{4}$

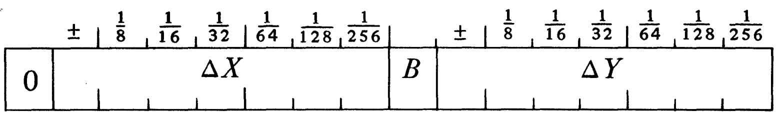
Stop, Read Graphic Scale Factor **&**

301	246	140646

Stop buffer and send the graphic scale factor to the interface. The scale factor S appears in a word as 0000S0, in a byte as 0S0.

Plot a Vector
(short format)

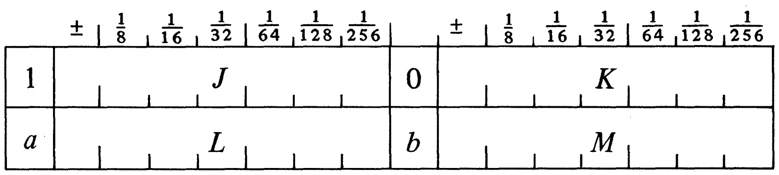
Unscaled bit value in inches



From the present beam position, plot a vector with components $2^S \times \Delta X$ and $2^S \times \Delta Y$, where S is the graphic scale factor. If B is 0 display the vector, otherwise do not (*ie* a 1 in B produces a blank line).

Plot a Curve
(short format)

Unscaled bit value in inches



From the present beam position, plot and display the curve defined by parameters $2^S J$, $2^S K$, $2^S L$ and $2^S M$, where S is the graphic scale factor, to the extent indicated by *ab* as follows:

ab	*Segment Type*
00	I
01	II
10	III
11	IV

Neither J nor L should be -64 (-77_8) as the word containing it would be interpreted as an escape (the twos complement form of -77 is 100).

As with long graphic format, short graphic components and parameters are limited to eight inches. This restriction affects the commands only at the largest scale factor: with 6 the MSB (the 8s bit) must be null.

2.5 SYMBOL MODES

In this mode each command is a single byte that refers to a subroutine in the symbol generator read-only memory or in the subroutine core memory. The processor stays in a given symbol mode until it encounters a command byte that refers to an escape subroutine, at which time it enters the mode specified by the next non-300 byte. Entry may be made to any mode including another symbol mode, but a graphic mode must be called if any function commands are to be executed.

The correspondence of command bytes to subroutines is entirely arbitrary; the subroutine for any given command may display an ASCII character or other figure, or it may do a special function like tab or carriage return, turn on the light pen, or ring the bell in the keyboard. However, fixed fonts prepared in ROM by Conographic Corporation use the following code assignments as standard, and it is recommended that they generally be used in order to avoid confusion. Fonts are usually set up to accept ASCII codes of either parity for a single set of 128 characters, but unique 8-bit codes can be assigned to allow a full complement of 256 including control characters.

Byte	*ASCII Character*	*Operation*
003,203	ETX	Disable receiver
007,207	BEL	Ring the bell in the keyboard
010,210	BS	Backspace – move beam one position left
011,211	HT	Horizontal tab – move beam right to next tab position (standard tabs are one-fifth screen width)
012,212	LF	Line feed – move beam down one line
013,213	VT	Vertical tab – move beam down to next tab position (standard tabs are one-fifth screen height)
014,214	FF	Form feed (home) – move beam to upper left corner
015,215	CR	Carriage return – move beam to left margin
016,216	SO	Enter italic mode
017,217	SI	Leave italic mode
030,230	CAN	Cancel – erase screen and enter store mode
033,233	ESC	Escape

The same set of symbol commands can refer to a font anywhere in memory because the symbol address is added to the font base address. To set up the system for symbol operation, the program must use this command to select the base address of the font before entering a symbol mode.

Set Font Base Address >

301	276
Address	

140676

Load the next word into the font base address register and clear the symbol address counter.

Italics. While the processor is in a symbol mode it can also be in italic mode, *ie* any symbol produced by a symbol mode command can be drawn in italic form; this applies to all symbols, even those in template mode, not just to alphanumeric characters. The following function commands control italic mode entry and exit.

Enter Italic Mode	**SO**	301	216	140616

Draw all symbols in italics until commanded to leave italic mode or there is an escape from symbol mode.

Leave Italic Mode	**SI**	301	217	140617

Draw subsequent symbols in standard form.

If an entire symbol string is to be drawn in italics, the program can enter italic mode along with setting the font base address prior to entering the symbol mode. Within a string, symbol commands can enter and leave italic mode by calling subroutines that are specifically for this purpose (the shift out and shift in characters respectively are normally used for this). Subroutines in symbol core memory can enter and leave italic mode, even for single symbols, by using the above function commands. In any event the processor automatically exits italic mode whenever there is an escape from symbol mode.

Symbol Scale. The plotting commands in symbol memory are in short graphic format. The graphic scale factor scales these commands only when the processor does not have the optional object scale factor. When object scaling is present, the graphic scale factor applies only to commands given in short graphic *mode* and the object scale factor applies to *all relative* commands including those in symbol memory.

In template mode, plotting commands operate in a normal manner and beam position is entirely arbitrary; at the end of a subroutine the position depends only on where the subroutine moved it. In all alphanumeric modes, the base beam position for each symbol is effectively fixed; the subroutine plotting commands move the beam relative to the base position by means of high resolution circuitry, and the exit from a subroutine automatically zeros the high resolution increment so that its effect (as determined by the short graphic commands) is entirely discounted when base position changes are being made, such as in spacing or tabbing.

With the use of high resolution, the short graphic subroutine commands move the beam in terms of raster units that are $\frac{1}{2048}$ inch, *ie* eight times as fine as the normal resolution of $\frac{1}{256}$ inch. The limits for this resolution are approximately a half inch in each direction from the base position, so that alphanumeric mode can draw symbols that are a maximum of a square inch in size (beam movement across an edge simply wraps around). Although subroutines in core can use any grid pattern, even a variable one, the standard alphanumeric fonts in ROM use a grid in which the sections are ten standard raster units high and six wide, with the base beam position in the lower left corner. At a given symbol position, a symbol command calls a subroutine, which draws the character within the rectangle and terminates with a function command that exits from the routine and spaces the beam to the next position, *ie* to the lower left corner of the next rectangle.

A 10 X 6 rectangle in normal raster units is equivalent to an 80 X 48 rectangle in the high resolution units. Within this the subroutine plotting commands are generally set up to produce alphanumeric characters that are 44 X 32. Without object scaling, the graphic scale factor can multiply the size and spacing of symbols by a power of 2 up to 2^6. With object scaling, the graphic scale factor does not affect symbol subroutine commands, and there is an automatic scaling upward by a factor of four, which for alphanumerics produces a display approximately like an ordinary typewriter – ten characters per inch and six lines per inch. By superimposing object scaling on the subroutine commands, character size and spacing can be varied from one-eighth the given size to eight times that size. At object scaling below one-fourth, the finest bit of resolution is lost. (Picture scaling can also be used; refer to the discussion of scaling in §2.7.)

The following table lists the characters per line, lines per page and character size in mils for various object scale factors based on subroutine commands using character spacing of 80 X 48 and character size of 44 X 32 (high resolution units) with the normal upward scaling by four. The left column gives the graphic scale factor that could be used to produce the same size characters if there were no object scaling.

Graphic Scale	*Object Scale*	*Characters per Line*	*Lines per Page*	*Character Size in Mils*
0	¼	341	158	21.5 X 15.6
1	½	170	79	43.0 X 31.3
	¾	113	52	64.5 X 46.9
2	1	85	39	85.9 X 62.5
	1¼	68	31	107.4 X 78.1
	1½	56	26	128.9 X 93.8
	1¾	48	22	150.4 X 109.4
3	2	42	19	171.9 X 125.0
	2¼	37	17	193.4 X 140.6
	2½	34	15	214.8 X 156.3
	2¾	31	14	236.3 X 171.9
	3	28	13	257.8 X 187.5
	3¼	26	12	279.3 X 203.1
	3½	24	11	300.8 X 218.8
	3¾	22	10	322.3 X 234.4
4	4	21	9	343.8 X 250.0

2.6 FUNCTION COMMANDS

A few function commands that are particularly relevant to short graphic and symbol modes have been described in previous sections, and a number that are principally for symbol subroutines and buffer operation are treated in sections that follow, but the bulk of the function commands are included here. All function commands use a full word and may require a second to supply an address, scale factor or other quantity. In the first word the left byte is a number from 301 to 377, and the right byte always contains a 1 in the left bit to distinguish the function commands from curve commands in short graphic format.

All function commands that read information (such as an offset, scale factor, position data) and send it out through the interface also stop the buffer (if it is running) so that the register read cannot change until some external action is taken.

Receiver

When information is sent over a data communication line, the text is preceded by information for synchronization and identification, and the beginning of the text is indicated by the character STX. If the display is connected to a communication line and the receiver in the interface is off, all incoming information is ignored until receipt of an STX (002 or 202), which turns on the receiver. All subsequent bytes are accepted including additional 002 and 202 bytes, which have no further affect on the receiver once it is on (the equivalent byte can easily appear in a plotting command). The STX cannot therefore be regarded as a command in the usual sense: only the first STX in a message can have any effect and the byte that precedes it is irrelevant; with the receiver on, the character is simply an ordinary byte and is not presently used in any function command.

The end of text character ETX may also appear arbitrarily without effect, but this character is used as the right byte in a 301 function command that turns off the receiver.

Disable Receiver	**ETX**	301	203	140603

Turn off the receiver so that no further information is accepted until an STX turns the receiver back on.

At power turnon, the display comes on in alphanumeric mode. Thus when first beginning transmission to a display over a communication line, it is recommended that following synchronizing and identification information, the sequence STX, 033, 300, 300 be sent. This turns on the receiver if it is off, supplies an escape for alphanumeric mode, and also provides an escape for graphic mode (one of the 300s in the pair is bound to be the left byte of a command word). The programmer can then place the display processor in whatever mode is desired.

If the user so desires, an interface to a computer can also be set up to turn on and off with STX and ETX.

CRT Control

Refresh tubes are used exclusively in Model 14 and larger displays, so there are no commands for controlling the way the tube operates. The Conograph 10 however has a storage tube so there are several commands for selecting the tube operating mode and erasing the screen.

Enter Store Mode	**ETB**	301	227	140627

Place the CRT in store mode so that all information displayed remains on the screen.

Enter Nonstore Mode	**DC4**	301	224	140624

Place the CRT in nonstore mode so that all information displayed on the screen is visible only transiently.

Enter Write Thru Mode	**DC1**	301	221	140621

Place the CRT in write thru mode so that all information subsequently displayed will be visible only transiently, but information already stored will not be affected.

Erase Screen and Enter Store Mode	**CAN**	301	230	140630

Erase all currently displayed information from the screen and then place the CRT in store mode.

Always erase the screen before entering nonstore mode if there is any information stored on the screen – otherwise garbage will result. In either nonstore or write thru mode the information displayed on the screen must be refreshed in order to remain visible (the recommended rate is fifty times per second). In write thru mode, information can be displayed, refreshed and altered without affecting data that has already been displayed in store mode. Hence the program can store some information, then write thru with data that may be modified from the keyboard. When the operator is satisfied with the correctness of the transient data, he can signal the program to store it and then go back to write thru mode for mode data to be presented for his consideration.

Timing. After entering write thru mode from store mode or after entering store mode from any other, the program must wait 20 ms before attempting to display further information. After giving the cancel command to erase the screen, the program must wait 500 ms before continuing. If storage is desired following erasure, it is not necessary to give a separate command to enter store mode – the cancel command leaves the display in that mode.

Position and Computation Commands

A number of commands are available for positioning the beam by loading or incrementing the position registers directly. The program can also read the position registers and can test their contents in order to make decisions dependent upon beam position. This last feature is useful for tabbing as the stopping position for a tab depends upon the present position as well as on where the tabs are located. Of course the beam position could be sent through the interface to a computer, and the computer program could carry out the necessary computations and subsequently position the beam by means of plotting commands, but the function commands described here allow such computations to be carried out by subroutines in the symbol memory.

Stop, Read X Position **SP**

301	240	140640

Stop the buffer and send the contents of the X position register to the interface.

Stop, Read Y Position !

301	241	140641

Stop the buffer and send the contents of the Y position register to the interface.

Stop, Read Z Position "

301	242	140642

Stop the buffer and send the contents of the Z position register to the interface.

Increment AC

310	1	*N*	1442(+) 1443(−)

Add *N* raster units to the present contents of the accumulator AC.

Increment X

311	1	*N*	1446(+) 1447(−)

Add *N* raster units to the present contents of the X position register. Positive *N* moves the beam right, negative *N* moves it left.

Increment Y

312	1	*N*	1452(+) 1453(−)

Add *N* raster units to the present contents of the Y position register. Positive *N* moves the beam up, negative *N* moves it down.

Increment Z

313	1	*N*	1456(+)
			1457(−)

Add *N* raster units to the present contents of the Z position register. Positive *N* brightens the beam, negative *N* dims it.

Increment AC and Exit

314	1	*N*	1462(+)
			1463(−)

Add *N* raster units to the present contents of the accumulator AC and terminate the symbol subroutine.

Increment X and Exit

315	1	*N*	1466(+)
			1467(−)

Add *N* raster units to the present contents of the X position register and terminate the symbol subroutine. Positive *N* moves the beam right, negative *N* moves it left.

Increment Y and Exit

316	1	*N*	1472(+)
			1473(−)

Add *N* raster units to the present contents of the Y position register and terminate the symbol subroutine. Positive *N* moves the beam up, negative *N* moves it down.

Increment Z and Exit

317	1	*N*	1476(+)
			1477(−)

Add *N* raster units to the present contents of the Z position register and terminate the symbol subroutine. Positive *N* brightens the beam, negative *N* dims it.

Load

320	20	0	*R*	15020*R*
Coordinate				

Replace the contents of the register specified by *R* with the next word.

R	*Register*
0	AC
1	X
2	Y
3	Z

Skip

324	2	*C*	0	*R*	1522*CR*

Compare the contents of AC with the contents of the position register specified by *R* and skip the next word if the condition specified by *C* is satisfied.

R	*Register*	*C*	*Condition*
0	AC	0	Never skip
1	X	1	Skip if register $R <$ AC
2	Y	2	Skip if register $R =$ AC
3	Z	3	Skip if register $R \leqslant$ AC
		4	Always skip
		5	Skip if register $R \geqslant$ AC
		6	Skip if register $R \neq$ AC
		7	Skip if register $R >$ AC

Transfer to AC

324	30	0	*R*	15230*R*

Replace the contents of AC with the contents of the register specified by *R*.

R	*Register*
0	AC
1	X
2	Y
3	Z

Transfer from AC

324	31	0	*R*	15231*R*

Replace the contents of the register specified by *R* with the contents of AC.

R	*Register*
0	AC
1	X
2	Y
3	Z

Ordinarily the graphic and object scale factors apply only to relative commands. But to avoid confusion in arranging text and the display of other symbol material, all positioning commands executed in symbol memory are scaled. This means that the scale factor applies not only to all relative commands, including increment, but also to load, which is absolute (of course there is no scaling in the mere transfer of data back and forth between *AC* and the position registers). In alphanumeric mode with object scaling, the automatic multiplication by four also applies to load as well as to the relative commands. Thus with an object scale factor of unity, alphanumeric load and increment commands (which have no connection with the high resolution positioning) move the beam in terms of $\frac{1}{32}$-inch raster units, *ie* four times the normal unit size. For standard alphanumerics at ten characters per inch, the standard character spacing is therefore produced by incrementing X by six.

The entire subroutine for a space or backspace can simply be a single Increment X and Exit in the location referenced by the symbol. This command can also be used to terminate any character subroutine at the same time that it moves the beam to the next character position. A carriage return is effected by Load X followed by the coordinate of the left margin. A line feed requires only one word, Increment Y and Exit. A symbol defined as "new line" can call a subroutine that is simply Load X, followed by the coordinate of the left margin, followed by Increment Y and Exit. The skip and transfer commands are especially useful for such operations as tabbing: the routine would first load AC and then alternately increment it and compare it against X or Y (for horizontal or vertical tabbing respectively) to determine when to stop the beam. The position registers can be read through the interface so a computer or other remote facility can determine the present beam position.

Scale Factors

The processor has three types of scaling: graphic, object and picture. The graphic scale factor [§2.4] applies only to relative plotting commands in short graphic mode except in symbol subroutines, where it applies to all positioning commands including load, provided there is no object scaling. If object scaling is present, graphic scaling does not apply to any commands executed from symbol memory; but object scaling applies to all of them as well as to all *relative* commands regardless of mode or origin. Picture scaling on the other hand applies to *all* beam positioning relative and absolute, in all modes and circumstances.

There is only one graphic scale factor for all dimensions, but the processor can have a single object or picture factor for all dimensions or separate factors for each. The following commands are used to set and read the object and picture scale factors. In each set command, the factor given can vary from 1 to 64, corresponding to scaling from ⅛ to 8 in steps of ⅛.

Set X Object Scale Factor **8** 140670

301	270	
000	0	*N*

Set the X object scale factor to *N* so that in subsequent relative commands and symbol subroutine commands the x component is multiplied by *N*. If there is only one object scale factor for all three dimensions, this command supplies it.

Set Y Object Scale Factor **9** 140671

301	271	
000	0	*N*

Set the Y object scale factor to *N* so that in subsequent relative commands and symbol subroutine commands the y component is multiplied by *N*.

Set Z Object Scale Factor **:** 140672

301	272	
000	0	*N*

Set the Z object scale factor to *N* so that in subsequent long graphic relative commands the z component is multiplied by *N*.

Set X Picture Scale Factor **;** 140673

301	273	
000	0	*N*

Set the X picture scale factor to *N* so that in all subsequent positioning commands the x component is multiplied by *N*. If there is only one picture scale factor for all three dimensions, this command supplies it.

Set Y Picture Scale Factor **<** 140674

301	274	
000	0	*N*

Set the Y picture scale factor to N so that in all subsequent positioning commands the y component is multiplied by N.

Set Z Picture Scale Factor =

301	275	
000	0	N

140675

Set the Z picture scale factor to N so that in all subsequent long graphic commands the z component is multiplied by N.

Stop, Read X Object Scale Factor (

301	250

140650

Stop the buffer and send the X object scale factor to the interface.

Stop, Read Y Object Scale Factor)

301	251

140651

Stop the buffer and send the Y object scale factor to the interface.

Stop, Read Z Object Scale Factor *

301	252

140652

Stop the buffer and send the Z object scale factor to the interface.

Stop, Read X Picture Scale Factor +

301	253

140653

Stop the buffer and send the X picture scale factor to the interface.

Stop, Read Y Picture Scale Factor ,

301	254

140654

Stop the buffer and send the Y picture scale factor to the interface.

Stop, Read Z Picture Scale Factor	-	301	255	140655

Stop the buffer and send the Z picture scale factor to the interface.

Graphic scaling is primarily to allow use of the short format, *ie* to be able with fewer bits to draw small figures with as fine a precision as the long format commands but still be able to produce large beam movements. Object scaling is for changing the relative sizes of objects in a picture without changing the size of the picture itself; in other words different parts of a picture can be varied in size while still remaining in the same position relative to one another. Picture scaling allows changes in the size of the entire display. *Eg* if a display file plots a picture that fills the addressable raster, it can be reduced so that the entire picture can be displayed in the visible area or expanded so that smaller sections can be seen in greater detail.

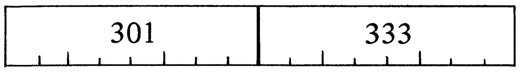
The order in which the scale factors are applied to any given instruction is graphic, object, picture. Of course all three can be applied only to plotting commands in short graphic mode. Object and picture scaling together can be applied to symbol subroutines and relative commands in long graphic mode (a figure command is regarded as relative or absolute as the positioning of the center point is relative or absolute).

Cursors

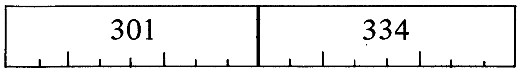
A cursor is a visual signal that tells the operator what place on the screen will be affected by the next action he takes at the keyboard. In graphic mode the form of the cursor is defined by the hardware and it consists of five dots: a center dot with another dot a quarter inch away in each direction. In store mode the center dot actually represents stationary beam position, but when the display is being refreshed, the beam moves too rapidly for its position to be indicated. In either case however the center dot of the cursor indicates the point at which the next plotting command will begin.

In symbol mode the form of the cursor is not defined, but if it is to be displayed, the processor assumes that a subroutine for it begins at location 400 in the presently selected font. Typically this subroutine marks the present symbol position with an underscore.

No cursor is displayed at all unless the display file issues a command to turn it on. Given that it is on, its method of presentation depends upon the source of display operation and the CRT mode. (Note: Method and form are independent. Regardless of method, to display the cursor the processor plots the five-point form for graphic mode and executes the subroutine at font location 400 for symbol mode.) When the display is operating from the buffer, it is assumed that the data is being refreshed, and the cursor position is indicated by a flag on some buffer location; every time the buffer address counter reaches the flagged location, the cursor appears. For operation through the interface, the cursor should be left on only in store mode; then the processor simply displays it automatically (in write thru mode) fifty times per second. During refresh operation through the interface, the beam position with time depends entirely on the action of the program, so it is up to the program to trigger the cursor each time it is desired. This is implemented by turning the cursor on and off with a pair of consecutive commands. Function commands that control the cursor are as follows.

Turn On Cursor [


140733

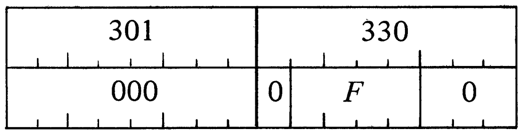
Turn Off Cursor \


140734

Dash Control

The generation of dashed lines is governed by a clock and a shift register. The clock shifts the register to the right, with bits coming out of the right end going into the left (in other words, a rotation), and the beam is intensified only while there is a 1 in the right bit. Hence the basic dash frequency is determined by the clock, and the relative lengths of the on and off parts of the line are determined by the bit configuration in the register. Suppose the dash frequency is set for a half inch; this means that the time between shifts is such that the beam moves a half inch. Then a pair of consecutive 1s in the shift register will intensify the beam for an inch, and three 0s will make it invisible for an inch and a half of the beam trajectory.

The shift register is twelve bits but can be expanded to twenty-four with the optional addition of another twelve on the low order end. Once the program selects the dash frequency and sets up the shift register by means of the commands which follow, then each long graphic command can choose whether the line it generates is to be dashed or not.

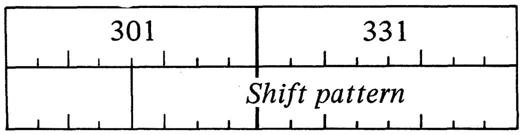
Set Dash Frequency X


140730

Set the dash shift frequency to *F* to select the period between shifts in terms of beam motion in inches as follows:

F	*Shift Separation*	*F*	*Shift Separation*
0	1/32	10	9/32
1	1/16	11	5/16
2	3/32	12	11/32
3	1/8	13	3/8
4	5/32	14	13/32
5	3/16	15	7/16
6	7/32	16	15/32
7	1/4	17	1/2

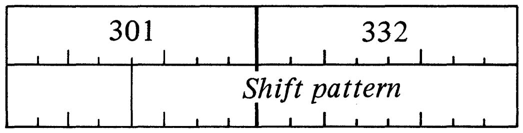
Set Dash Register Y



140731

Load the low order twelve bits of the next word into the dash shift register.

Set Dash Register Extension Z



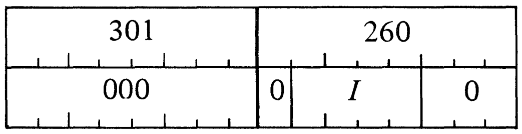
140732

Load the low order twelve bits of the next word into the low order half of the expanded dash shift register.

Intensity Variation

With this option the program can select one from among sixteen levels of beam intensification, and each long graphic plotting command can choose whether the plot shall be at normal intensity or at the optional program-selected intensity. The program uses this command to select the level.

Set Variable Intensity O



140660

Set the optional intensity level for long graphic commands to *I*, where variation from 0 to 17 corresponds to beam variation from dim to bright.

Offset

Optional offset registers allow the programmer to change the position of a picture in the addressable raster without affecting its size or shape and without tampering with the plotting commands in the display file. Every time an absolute positioning command is given, the contents of the offset registers are added to the corresponding coordinates prior to picture scaling; the offset is therefore applied to the coordinates of an absolute vector, the coordinates of the center point of an absolutely positioned figure, and the coordinate supplied to a position register by the load command. By adjusting the offsets, the program can move data from an invisible section of the raster onto the screen, producing the effect of moving the visible window around in the addressable raster. Suppose it is desired to view the rectangular section of the raster running from −13 to −5 inches in x and from −4 to +2 inches in y. Offsets of 3328 in x and 1024 in y would move this rectangle onto the Model 10 screen. A negative offset moves the picture down or to the left. A three-dimensional display can also be offset in depth, *ie* the brightness of the entire picture can be changed by a constant factor.

Caution

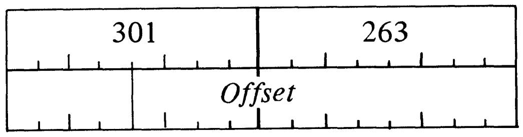
Remember [§2.1] that the beam positioning raster is actually 64 X 64 even though the addressable raster is only 32 X 32. Hence to offset negatively (down or left) requires a negative number of fourteen bits. This can be regarded as a large positive number that offsets negatively by wrapping the beam around the position raster in the positive direction.

In supplying an offset of fourteen bits in a 16-bit word, it is not necessary to mask out the left two bits; the programmer can simply negate the appropriate positive number of thirteen bits magnitude, so that the extra two bits on the left duplicate the sign of the 14-bit offset.

Do *not* give a negative offset that would wrap the beam around exactly once, *ie* that would really leave the display in its original position. Such an offset would be interpreted as an escape.

The program uses these instructions to load and read the offset registers.

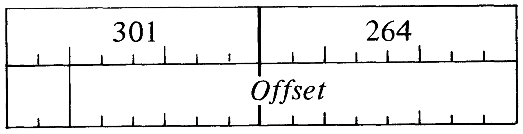
Load X Offset **3**



140663

Replace the contents of the X offset register with the low order fourteen bits of the next word.

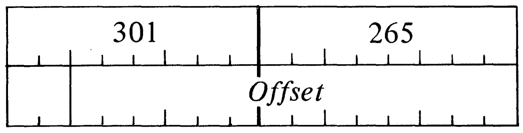
Load Y Offset **4**



140664

Replace the contents of the Y offset register with the low order fourteen bits of the next word.

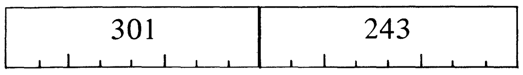
Load Z Offset **5**



140665

Replace the contents of the Z offset register with the low order fourteen bits of the next word.

Stop, Read X Offset #



140643

Stop the buffer and send the contents of the X offset register to the interface.

Stop, Read Y Offset	%	301	244	140644

Stop the buffer and send the contents of the Y offset register to the interface.

Stop, Read Z Offset	&	301	245	140645

Stop the buffer and send the contents of the Z offset register to the interface.

Light Pen

In communicating with a display file, the operator often wishes to specify some particular beam position, or perhaps a sequence of beam positions that make up some arbitrary curve. This can be done easily by use of a light pen, a device that signals the display processor whenever it detects a spot being displayed. The light pen is available only on the Model 14 and larger displays.

The light pen can be turned on and off by the operator and can also be enabled or disabled by the program. The display processor has a light pen flag which is set by detection of a spot if the pen is enabled; when the pen is disabled, spot detection is simply ignored. The light pen flag can be used for internal display operations and can also be used as a signal through the interface, *eg* to interrupt a computer. Commands that allow a display file in the buffer to respond to the light pen are described with the other buffer commands [§2.8]. The following two commands enable and disable the pen.

Enable Light Pen	P	301	320	140720

Enable the light pen so that detection of a spot sets the light pen flag.

Disable Light Pen	Q	301	321	140721

Disable the light pen so that the display processor ignores any detection signals from it.

Keyboard Bell

The display file can attract the attention of the operator by ringing a bell located in the keyboard. This command can be given through the interface, from the buffer, or from a symbol subroutine.

Ring Keyboard Bell	**BEL**	301	207	140607

2.7 SYMBOL SUBROUTINES

Many fixed fonts are available from Conographic Corporation in read-only memory, and fonts designed by the user can also be supplied in ROM. But if the display includes a subroutine controller and core memory, the user can design his own fonts and can change them at will under program control.

To design a font, the programmer simply writes a subroutine for each symbol command using short graphic plotting commands and the various function commands described in the preceding section. The font can begin at any location in memory; to select a particular font, the display file need only give the command Set Font Base Address to load the address of that location into the base address register. The first 256 locations beginning at the base address are referenced by the symbol commands. For most control symbols and all displayed symbols, the referenced location contains a jump to a subroutine elsewhere in the font. In template mode a subroutine can generate a figure of any size, even one that fills the entire screen. In alphanumeric mode the picture that can be drawn about a given base beam position is limited to one square inch (a half inch in every direction) by the high resolution circuitry, but the base position can be changed within the subroutine.

Every subroutine must end with a function command that terminates it, *ie* that exits from it returning control to the buffer or the interface. The only commands that terminate a subroutine are the escape command and the increment commands that also exit. The two sets of increment commands, with and without termination, provide considerably greater flexibility in subroutining. An increment without exit changes the alphanumeric base position without terminating the subroutine; this means that larger symbols can be drawn by combining plots about several base positions. Incrementing with exit saves memory space, as the subroutine can terminate with the same command that spaces to the next character position. In many cases this eliminates the subroutine altogether, for a control function such as space, backspace or line feed can be done by a single terminating command in the location referenced by the symbol command.

The jump command automatically saves the contents of the symbol address counter. This counter is twelve bits, so the address saved is twelve bits and points to the location following the jump in the presently selected font, *ie* in relation to the base address. This feature is not used in the initial subroutine call, but the subroutines themselves can use it to call other subroutines for elements common to a number of symbols. *Eg* in template mode, large circuit network symbols may be drawn partly by using short graphic commands and partly by calling for common elements such as resistors and capacitors. The return from any subroutine is made by a command that jumps to the location whose address was saved when the original subroutine jump was made. Note that only one level of subroutining is allowed – there is no nesting. Each jump saves a new address at the expense of losing the address previously saved.

Besides the function commands already described, the following three are needed to implement the subroutine procedures discussed above.

Jump to Symbol Subroutine

1 1 1	*Address*	1	*Address*

Jump within the currently selected font to the location whose address is specified by this command, and save the address of the next consecutive location following this command. The 12-bit jump address is made up of the right five bits in the left byte of the command and the right seven bits of the right byte.

Return from Symbol Subroutine ?

301	277	140677

Jump to the location whose address was saved by the last Jump to Symbol Subroutine.

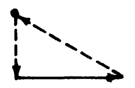
Escape @

300	300	140300

Terminate the subroutine and interpret the next non-300 byte as a mode command.

If the cursor is used, the processor displays it by executing the subroutine beginning at location 400 of the currently selected font. The following subroutine uses three short graphic vector commands, the first and third of which are blank, to display an underscore.

00000000	11010000	*Position:* $\Delta X = 0 \quad \Delta Y = -24$
00100000	00000000	*Vector:* $\Delta X = 32 \quad \Delta Y = 0$
01100000	10011000	*Position:* $\Delta X = -32 \quad \Delta Y = 24$
11001101	10000000	*Increment zero and exit*



A procedure that can be used to save space in symbol memory is to overlap subroutines for symbols that are very similar. *Eg* the letter 'F' is included within the letter 'E'. Thus if the subroutine for 'E' began by drawing the bottom line, the same subroutine could be used for 'F' simply by entering it at a later point. In the same way, the subroutine for 'F' could simply be part of that for 'R', the routine for 'O' part of that for 'Q'.

Subroutines are loaded into the core memory by the same commands that are used to load it for buffer operation [§2.8]. If a Model 10 has a subroutine controller but no buffer controller, then a font can be loaded by setting the starting address into the font base address and giving the same load command (with word count) that would be used for the buffer.

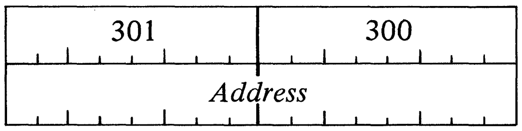
2.8 BUFFER

The buffer is useful primarily for allowing operator interaction with a display file that is run entirely inside the display instead of requiring continuous access to a computer. The buffer must be initialized from some external source, but once running, it can continually refresh the screen and can respond to action by the operator at the keyboard or using the light pen. The Model 14 and larger displays have a buffer as standard equipment. It is optional with the Model 10 and can be used effectively provided the operator works with a small part of the display at a time. *Eg* for editing text, the operator must work with a few lines being refreshed in write thru mode, inserting and deleting material or modifying it as he chooses. When the text is corrected to his satisfaction, he can call upon the computer to store it on the screen and load the next few lines into the buffer to contine the text.

To place a file in the buffer, the computer must specify the address for the beginning of the file, and then start the loading procedure with a command that supplies a word count. In a typical situation the interface operates through a data channel directly to computer memory to bring the specified number of words into buffer memory. Another command supplies a starting address and places the buffer in operation. The file in the buffer should start with a regeneration command that provides proper timing for refreshing the display; this command prevents the file from starting more often than every 20 ms, to give a refresh rate of fifty times per second. The material being refreshed is followed by a jump back to the starting point. Other commands available for use in the file allow subroutining and a means of responding to the light pen. The file can also terminate buffer operation and request an interrupt through the interface.

A cursor can be used to indicate the buffer location, and therefore raster position, that will be affected by action by the operator. When the computer turns on the cursor, the location presently addressed by the buffer address counter is flagged. Then as the operator inserts data into the file or deletes data from it, the flag moves from one location to the next, up or down, always indicating the location that will be affected next. The operator can also move the cursor from the keyboard to select the position (location) he wishes to modify. Every time the buffer controller runs through the file, it displays the cursor when it reaches the flagged location.

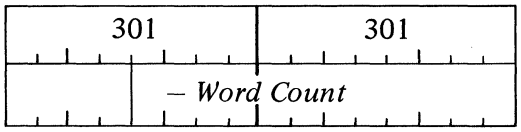
Stop, Set Buffer Address @



140700

Stop the buffer and load the next word into the buffer address counter.

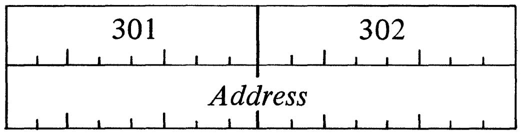
Load Core Memory A



140701

Load the low order twelve bits of the next word into the word counter and then load subsequent words that come in through the interface into core memory at the locations specified by the buffer address counter. As each word is loaded, both counters are incremented by one; the process stops when the word counter overflows.

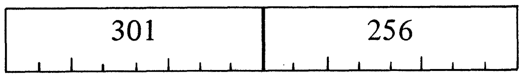
Start Buffer **B**



140702

Load the next word into the buffer address counter and start buffer operation at the location specified by it.

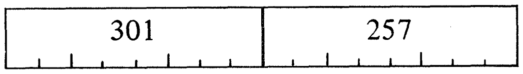
Stop, Read Buffer Address **.**



140656

Stop the buffer and send the current contents of the buffer address counter to the interface.

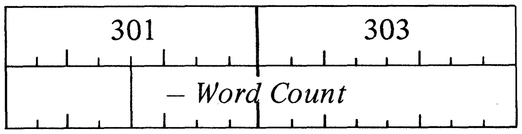
Stop, Read Cursor Address **/**



140657

Stop the buffer and send the address of the location that is flagged for the cursor to the interface.

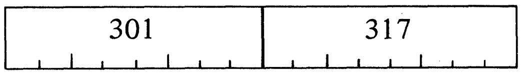
Read Core Memory **C**



140703

Load the low order twelve bits of the next word into the word counter and then send words from the locations specified by the buffer address counter to the interface. As each word is sent, both counters are incremented by one; the process stops when the word counter overflows.

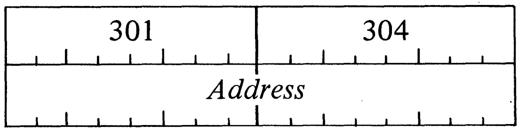
Start Regeneration Timer **O**



140717

Enter long graphic mode and restart the buffer when 20 ms have elapsed since the last time it started.

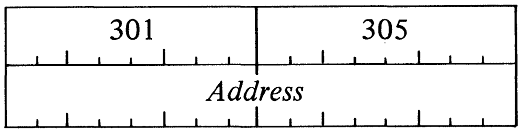
Jump **D**



140704

Load the next word into the buffer address counter and take the next command from the location then addressed by it.

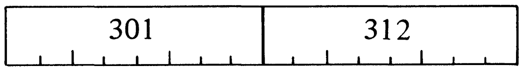
Jump to Buffer Subroutine **E**



140705

Save the current contents of the buffer address counter (which points to the *second* location following this command). Load the next word into the counter and take the next command from the location then addressed by it.

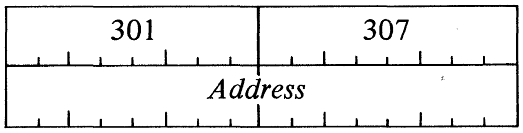
Return from Buffer Subroutine **J**



140712

Load the address saved by the last Jump to Buffer Subroutine into the buffer address counter and take the next command from the location then addressed by it.

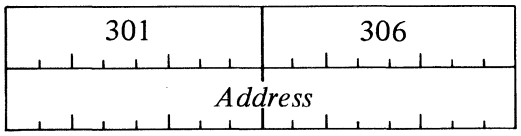
Jump if Spot Detected **G**



140707

If the light pen has detected a spot, load the next word into the buffer address counter and take the next command from the location then addressed by it.

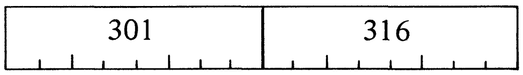
Jump if Spot Not Detected **F**



140706

If the light pen has not detected a spot, load the next word into the buffer address counter and take the next command from the location then addressed by it.

End Buffer Sequence **N**



140716

Stop the buffer and request an interrupt through the interface.

3 Operation

To enable a user to communicate with the display, the system includes a keyboard, various controls and indicators, a sonic pointer, and on the larger models, a light pen. The pointer emits a sonic signal and can be used either on the screen or on a separate tablet connected to the display; in either case its position within a rectangular area is determined by the relative strength of the signal along two strings of microphones located on a pair of adjacent sides.

All of these devices may be used for various methods of interaction between the operator and display software either in a local computer, at a remote station connected by communication lines, or in the display buffer.

3.1 KEYBOARD AND CONSOLE

The system ordinarily has a full ASCII keyboard with control and shift keys, although a Model 33 type keyboard may be used. In the upper half of the control unit front panel are a row of switches, a row of lights, and a key-operated power switch. The last has three positions: off, on, and a third lock position in which the key can be removed though power remains on.

At the right is an illuminated switch, ASCII/TTY, for specifying whether the keyboard is full ASCII or a Model 33; turning the switch to TTY allows the operator to generate capital letter codes without using the shift key, regardless of the type of keyboard. Also on the right, the WAIT light indicates that in a half duplex connection the interface is presently receiving information so the operator should not touch the keys. At the left end are a reset button that enables the receiver and places the system in alphanumeric mode, and a clear buffer button that allows the operator to place pairs of null codes in all of the buffer locations in core memory.

The remaining controls are for keyboard operation of the system. EVEN/ODD allows the operator to select the type of parity generated by the keyboard so he can produce any 8-bit code configuration. With the LINE/LOCAL switch in LINE, the keyboard communicates through the interface; with the switch set to LOCAL the keyboard operates directly on a display file that is running in the buffer. The last light at the left indicates when the buffer area is full, *ie* the operator can insert no more data. Pressing SEND transmits the contents of the buffer through the interface.

Local interaction between keyboard and buffer is ordinarily in symbol mode, as graphic commands are inconvenient to generate at the keyboard. If a text string is being displayed and refreshed on the screen, a cursor in the form of an underscore usually indicates the symbol position that will be affected by any action the operator takes. The cursor actually indicates a specific byte location in the buffer, but in symbol mode each byte is a command, so the symbol position on the screen is equivalent to the byte location. To delete a symbol from the screen, the operator simply presses the delete key; this removes the symbol by removing the symbol command from the buffer. When a character is deleted, all the symbol commands beyond it in the buffer are dropped down one position to fill the gap, so no blank space is left on the screen. The operator inserts a character in the text string simply by striking a key; the symbol previously in that location and all symbols beyond it are moved up one. To change a character, the operator must first delete it and then insert a new one. When a character is deleted the cursor remains in the same position, but insertion of a character moves the cursor to the next position.

The operator can also move the cursor independently of any text changes by means of buttons labeled with left and right arrows at the right end of the keyboard. Pressing the left arrow backspaces the cursor, *ie* moves it down one location in the buffer. This is not the usual backspace in the typewriter sense, as the button can backspace completely through the text string including carriage returns and line feeds; in other words backspacing out the left end of one line takes the cursor into the right end of the previous line (up). Similarly the right arrow spaces the cursor forward. The HOME button moves the cursor back to the initial buffer location, equivalent to the upper left corner of the screen.

There are actually four arrow buttons: left, right, up, down. All four can be used in graphic mode with a picture stored on the screen. Pressing a button moves the cursor in the indicated direction an amount equal to the minimum position change as specified by the graphic scale factor.

3.2 SOFTWARE

Conographic Corporation provides a variety of software to allow exceptionally effective use of the display by the user. This software is available on cassette tapes, and the package supplied to a user includes program listings, flow diagrams and program descriptions. All programs are written in Fortran so they can easily be adapted and modified by the user for his own purposes, regardless of the configuration or application of his particular system. Many programs and symbol fonts are available through various time sharing vendors for those using the display as a terminal connected to a time sharing facility.

The basic program package, Conopac-1, computes plotting commands with Conographic parameters from information supplied at the keyboard. Conopac-1 provides great flexibility in the form in which the user can supply the needed information.

Type of Command	*Data Supplied from Keyboard*
Curve	Two points and two slopes
Curve	Three points and the slope at any of the three
Curve	Three points, where the middle point is the relative maximum
Circle	Radius and position
Ellipse	Major and minor axes and angle of axes to coordinate system
Rotation	Reference point and angle of rotation about the point

The display file can be viewed at any time by keyboard command.

General purpose programs include an interactive symbol design package that allows the operator to design characters, symbols fonts and templates using the sonic pointer and tablet; and a curve fitting package through which the user can parameterize and transform curves defined by arrays of points, increments or vectors, or specified as mathematical functions. A text composition program, operating out of the display buffer, makes it possible for the user to design and display textual information with mixed fonts, vertical and horizontal lines, italics, scaling, tabulation and justification.

Among the time sharing programs are two that implement numerical control and circuit design. The former program, based on the APT/REMAPT language, makes it possible for the user to check the accuracy of a numerical control tape by actually drawing a picture of the part that would result from the specified tool motions. The other program provides automatic analysis of electronic circuit design; it displays block schematics and results, and offers easy substitution of values and tolerances.

APPENDIX A

INSTALLATION

All Conographic Corporation display systems mount in a standard 19-inch rack. The CRT monitor and control unit are each complete with power supplies and a cooling fan, and each requires only four bolts for rack mounting. At least two inches should be left open at the back of the rack for cabling and air venting; the air intakes are at the sides. The console protrudes one inch at the front of the rack. An expansion chassis for additional memory and other options can be mounted below the control unit. The illustration on the next page shows the external layout and dimensions of the Conograph 10 and 14.

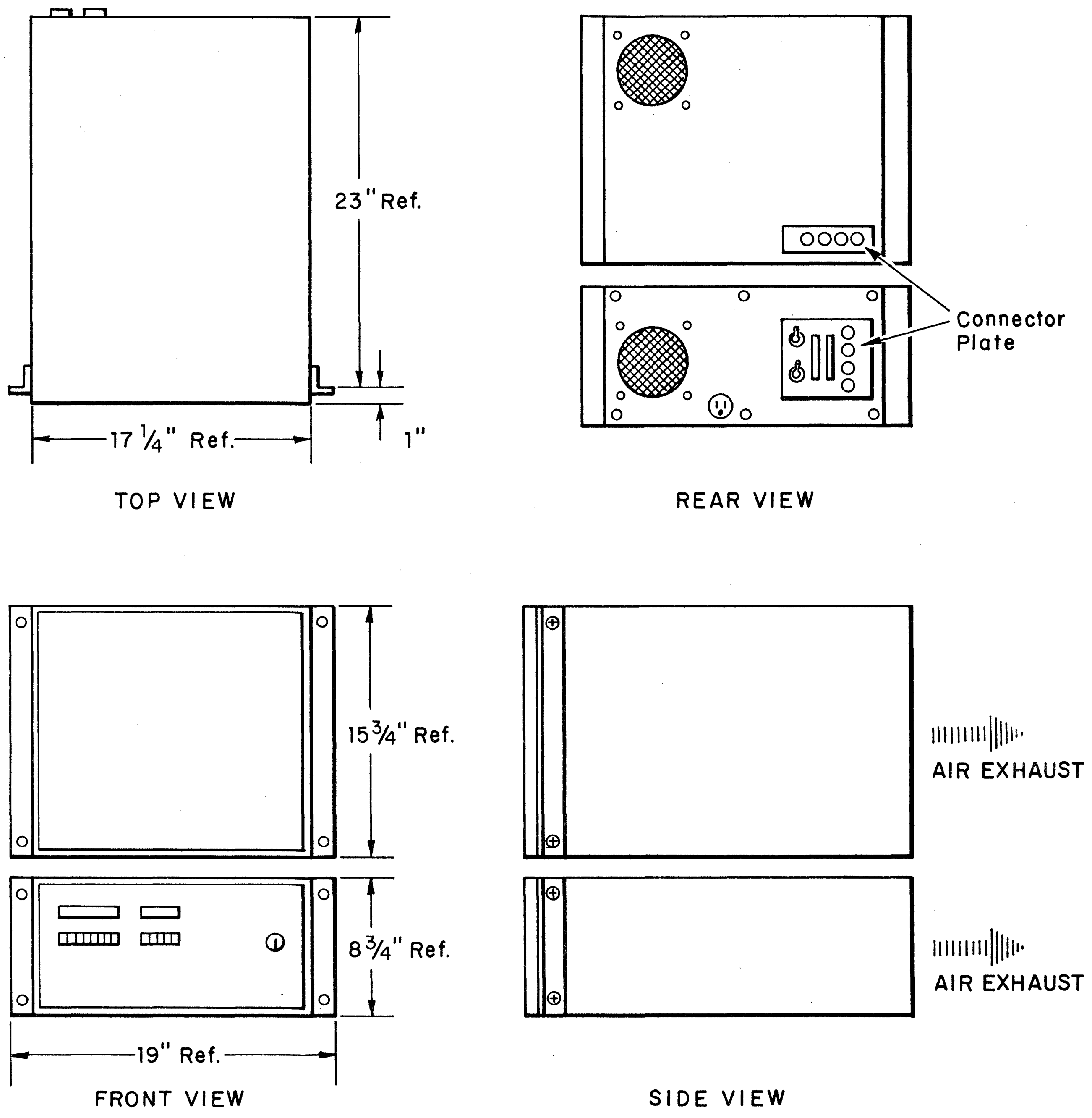
	Height	*Width (inches)*		*Depth*	*Weight*
	(inches)	*Table*	*Rack*	*(inches)*	*(pounds)*
Conograph 10/14	24½	17¼	19	24	120
Monitor	15¾				
Control unit	8¾				
Keyboard	3½	17¼		7½	8¾
Expansion chassis	8¾	17¼	19	24	32

It is recommended that the ambient temperature at the installation be maintained between 20° and 30°C, but the temperature can vary from 0° to 55° without adverse effect (the equipment can be stored in temperatures as high as 70°). The relative humidity can be as high as 90% noncondensating. (Although all exposed surfaces are treated to prevent corrosion, exposure to extreme humidity for long periods of time should be avoided.)

The display uses 47 to 63 Hz single phase line power, generally either 115 or 230 vac with a tolerance of ±10% (other frequencies and voltages are available on special order). At 115 vac the display requires less than 10 amperes and dissipates under 1200 watts. The power source should be capable of supplying 15 amperes; the power cable has a standard 3-wire plug and should be plugged into a receptacle rated at 15 amperes.

The display can be placed on a table, but if the monitor and control unit are in a rack, the keyboard must still be placed on a nearby table or on a shelf mounted on the front of the rack. At the back of the control unit are the main power cable for connection to the line source, and signal and power cables for connection to the monitor. If the hardcopy unit is used, the monitor cables from the control unit are connected to the hardcopy unit, which may be placed on top of the monitor, and additional cables connect it to the monitor. Other connectors and switches at the back of the control unit generally depend upon the type of interface used. In any event there must be a connection to a communication line or a cable to connect the interface to a computer or other external data source.

The cable from the keyboard connects directly into the front of the control unit below the switches. Excess cable can be rolled up and tucked inside the keyboard enclosure.



INSTALLATION: CONOGRAPH 10/14 – RACK MOUNTING

APPENDIX B

COMMAND CODES

The table beginning on the next page lists in hexadecimal and octal form the 8-bit codes that can be used as the right byte in a function command and lists the mode and function commands for those codes that are used. Since the left bit must be 1, there are only 128 possible 8-bit codes. A dagger (†) by a character means it must be transmitted using odd parity (to place a 1 in the left bit); the other characters provide the correct configuration using even parity.

ASCII CHARACTER COMMANDS

Hex	Octal	Character	Command
80	200	†NUL	
81	201	SOH	
02,82	002,202	STX	Enable receiver – only first STX (8 bits: 022,202) in incoming byte stream provided receiver is off; not useable in commands – prior bytes not received
83	203	†ETX	Disable receiver – no further reception
84	204	EOT	
85	205	†ENQ	
86	206	†ACK	
87	207	BEL	Ring bell in keyboard
88	210	BS	
89	211	†HT	
8A	212	†LF	
8B	213	VT	Enter template mode
8C	214	†FF	Enter alphanumeric mode with rotation
8D	215	CR	Enter protected alphanumeric mode with rotation
8E	216	SO	Enter italics
8F	217	†SI	Leave italics
90	220	DLE	
91	221	†DC1	Enter write thru mode (wait 20 ms)
92	222	†DC2	Enter alphanumeric mode
93	223	DC3	Enter protected alphanumeric mode
94	224	†DC4	Enter nonstore mode
95	225	NAK	
96	226	SYB	
97	227	†ETB	Enter store mode (wait 20 ms)
98	230	†CAN	Erase screen and enter store mode (wait 500 ms)
99	231	EM	
9A	232	SUB	Do nothing until 300 command
9B	233	†ESC	
9C	234	FS	Enter long graphic mode
9D	235	†GS	Enter special long graphic mode
9E	236	†RS	Enter short graphic mode
9F	237	US	Enter special short graphic mode
A0	240	SP	Stop, read X position
A1	241	†!	Stop, read Y position
A2	242	†"	Stop, read Z position
A3	243	#	Stop, read X offset
A4	244	†$	Stop, read Y offset
A5	245	%	Stop, read Z offset
A6	246	&	Stop, read graphic scale factor

Hex	Octal	Character	Command
A7	247	†'	
A8	250	†(	Stop, read X object scale factor‡
A9	251	)	Stop, read Y object scale factor
AA	252	*	Stop, read Z object scale factor
AB	253	†+	Stop, read X picture scale factor‡
AC	254	,	Stop, read Y picture scale factor
AD	255	†-	Stop, read Z picture scale factor
AE	256	†.	Stop, read buffer address
AF	257	/	Stop, read cursor address
B0	260	†0	Set variable intensity [0–170 *by 10s*]
B1	261	1	
B2	262	2	
B3	263	†3	Set X offset [ΔX]
B4	264	4	Set Y offset [ΔY]
B5	265	†5	Set Z offset [ΔZ]
B6	266	†6	Set graphic scale factor [0–60 *by 10s*]
B7	267	7	
B8	270	8	Set X object scale factor [1–100] = $\frac{1}{8}$ – 8‡
B9	271	†9	Set Y object scale factor [1–100] = $\frac{1}{8}$ – 8
BA	272	†:	Set Z object scale factor [1–100] = $\frac{1}{8}$ – 8
BB	273	;	Set X picture scale factor [1–100] = $\frac{1}{8}$ – 8‡
BC	274	†<	Set Y picture scale factor [1–100] = $\frac{1}{8}$ – 8
BD	275	=	Set Z picture scale factor [1–100] = $\frac{1}{8}$ – 8
BE	276	>	Set font base address [address]
BF	277	†?	Return from symbol subroutine
C0	300	@	Stop, set core address [address]
C1	301	†A	Load core memory $\|[-\text{WC}]\| \leqslant 4096$ (12 bits)
C2	302	†B	Start buffer [start address]
C3	303	C	Read core memory $\|[-\text{WC}]\| \leqslant 4096$ (12 bits)
C4	304	†D	Jump [address]
C5	305	E	Jump to subroutine [address]
C6	306	F	Jump if light pen not detected [address]
C7	307	†G	Jump if light pen detected [address]
C8	310	†H	
C9	311	I	
CA	312	J	Return from buffer subroutine
CB	313	†K	
CC	314	L	
CD	315	†M	
CE	316	†N	End buffer sequence and request interrupt
CF	317	O	Start regeneration timer – enter graphic mode and restart buffer when 20 ms elapsed since last start

‡If there is only one scale factor for all dimensions, use this code.

Hex	Octal	Character	Command
D0	320	†P	Enable light pen
D1	321	Q	Disable light pen
D2	322	R	
D3	323	†S	
D4	324	T	
D5	325	†U	
D6	326	†V	
D7	327	W	
D8	330	X	Set dash frequency [0–170 *by 10s*]
D9	331	†Y	Set dash register [12 shift bits]
DA	332	†Z	Set dash register extension [12 shift bits, low order]
DB	333	[	Turn on cursor
DC	334	†\	Turn off cursor
DD	335	]	
DE	336	↑^	
DF	337	†←_	
E0	340	†`	
E1	341	a	
E2	342	b	
E3	343	†c	
E4	344	d	
E5	345	†e	
E6	346	†f	
E7	347	g	
E8	350	h	
E9	351	†i	
EA	352	†j	
EB	353	k	
EC	354	†l	
ED	355	m	
EE	356	n	
EF	357	†o	
F0	360	p	
F1	361	†q	
F2	362	†r	
F3	363	s	
F4	364	†t	
F5	365	u	
F6	366	v	
F7	367	†w	
F8	370	†x	
F9	371	y	
FA	372	z	

Hex	Octal	Character	Command
FB	373	†{	
FC	374	|	
FD	375	†}	
FE	376	~	
FF	377	DEL	

APPENDIX C

DETERMINATION OF CONOGRAPHIC PARAMETERS

This appendix lists the formulas for calculating the Conographic parameters for various curves and figures. Included are a number of typical Type I segments defined by points and slopes, transformation of parameters for a curve under rotation, and circles and ellipses with beam motion in either direction.

A point labeled P_n in a diagram shall be assumed throughout to have coordinates X_n, Y_n and the slope at that point shall be S_n. In plotting commands, parameters are always given in the order *J, K, L, M,* but here they are listed in each case in the order that is most convenient for computation.

Curves, Type I

All formulas are given for curves that are assumed to be segments of Type I, starting at P_1 and ending at P_2. In any example, the curve shown can be extended into a segment of higher type simply by specifying the desired type in the plotting command. The coordinates of the end point in a curve segment of any type are as follows.

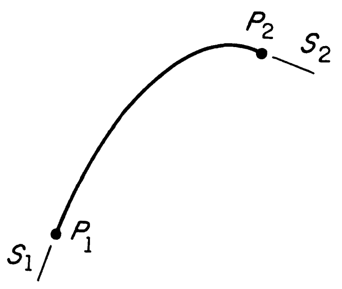
	End Point Coordinates	
Segment Type	*X*	*Y*
I	$X_1 - J + K$	$Y_1 - L + M$
II	$X_1 - 2J$	$Y_1 - 2L$
III	$X_1 - J - K$	$Y_1 - L - M$
IV	X_1	Y_1

Two points with slopes.

$$a = X_2 - X_1 \qquad b = Y_2 - Y_1$$

$$J = \frac{S_1 a - b}{S_2 - S_1} \qquad K = a + J$$

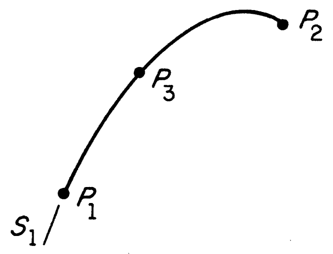
$$L = S_2 J \qquad M = b + L$$



If the slopes are equal, the segment is really Type II. However there is a special case in which the tangents are not only parallel but coincident, and the segment is degenerate, *ie* it is really a straight line. As such it can be specified by a vector command requiring only half as many words in the display file, but sometimes the programmer may prefer to treat it as a general curve. The parameters are

$$J = L = 0 \qquad K = a \qquad M = b$$

Three points and starting slope.



$a = X_2 - X_1$ $\qquad$ $b = Y_2 - Y_1$

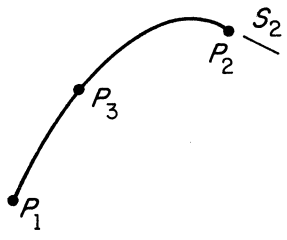
$c = X_2 - X_3$ $\qquad$ $d = Y_2 - Y_3$

$e = \dfrac{S_1 c - d}{S_1 a - b}$ $\qquad$ $f = \sqrt{1 - e^2}$

$L = \dfrac{(Y_3 - Y_1) - bf}{e + f - 1}$ $\qquad$ $M = L + b$

$K = \dfrac{M}{S_1}$ $\qquad$ $J = K - a$

Three points and ending slope.



$a = X_2 - X_1$ $\qquad$ $b = Y_2 - Y_1$

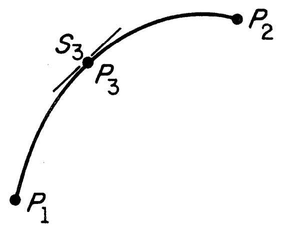
$c = X_3 - X_1$ $\qquad$ $d = Y_3 - Y_1$

$e = \dfrac{S_2 c - d}{S_2 a - b}$ $\qquad$ $f = \sqrt{1 - e^2}$

$K = \dfrac{(X_3 - X_2) + af}{e + f - 1}$ $\qquad$ $J = K - a$

$L = S_2 J$ $\qquad$ $M = L + b$

Three points and intermediate slope.



$a = X_2 - X_1$ $\qquad$ $b = Y_2 - Y_1$

$c = X_3 - X_1$ $\qquad$ $d = Y_3 - Y_1$

$c' = X_3 - X_2$ $\qquad$ $d' = Y_3 - Y_2$

$$q = \frac{S_3 c - d}{S_3 c' - d'}$$

$e = \dfrac{1 - q(1 \pm \sqrt{2q})}{q^2 + 1}$ $\qquad$ $f = \sqrt{1 - e^2}$

$J = \dfrac{c - af}{e + f - 1}$ $\qquad$ $K = J + a$

$L = \dfrac{d - bf}{e + f - 1}$ $\qquad$ $M = L + b$

In the formula for e, the sign must be chosen so that $e \geqslant 0$.

Three points with relative maximum. The intermediate point being given as the relative maximum of the segment is a special case of the preceding example. It is equivalent to the slope at P_3 equalling the slope of the chord P_1P_2. The formulas for the parameters remain the same, but in the equation for q, substitute b/a for S_3.

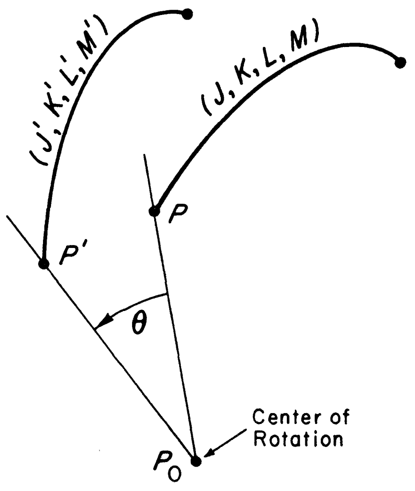
Rotation

A segment with starting point P and parameters J, K, L, M is rotated through an angle θ about a point P_0.

Starting point.

$$a = X - X_0 \qquad b = Y - Y_0$$

$$X' = X_0 + a \cos\theta - b \sin\theta$$
$$Y' = Y_0 + b \cos\theta + a \sin\theta$$



A starting point need be rotated only if it is positioned absolutely. In other words, in a sequence of relative commands the parameters for all must be rotated but the transformations need be applied to the starting point of only the first segment. In each subsequent segment the starting point is the end point of the preceding (rotated) segment.

Parameters. Transformation formulas for the parameters of the rotated curve as functions of θ and the original parameters are as follows.

$$e = \frac{L \tan\theta}{J}$$

$$f = \frac{\tan\theta + M/K}{1-e} \qquad g = \frac{\tan\theta + L/J}{1-e}$$

$$h = (M-L)\sin\theta - (K-J)\cos\theta$$
$$i = (K-J)\sin\theta + (M-L)\cos\theta$$

$$J' = \frac{fh+i}{f-g} \qquad K' = J' - h$$

$$L' = gJ' \qquad M' = fK'$$

Figures

Complete circles and ellipses are Type IV Conographic segments. Although they can be plotted as curves, the parameters are given here for drawing them as figures positioned about a specified center point P_0. The coordinates of the end point in a figure segment of any type are as follows.

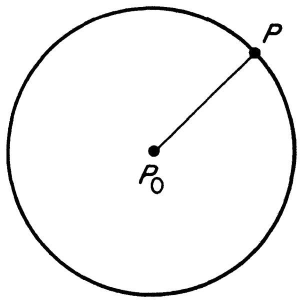
Segment Type	*End Point Coordinates* X	Y
I	$X_0 + K$	$Y_0 + M$
II	$X_0 - J$	$Y_0 - L$
III	$X_0 - K$	$Y_0 - M$
IV	$X_0 + J$	$Y_0 + L$

The endpoint of the Type IV segment is of course the starting point of the figure as well.

Circles. Parameters for drawing a circle clockwise or counterclockwise beginning at P are as follows.

$a = X - X_0$ $\qquad$ $b = Y - Y_0$

Clockwise	*Counterclockwise*
$J = a$	$J = a$
$K = b$	$K = -b$
$L = b$	$L = b$
$M = -a$	$M = a$

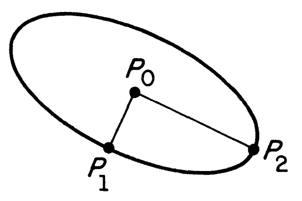


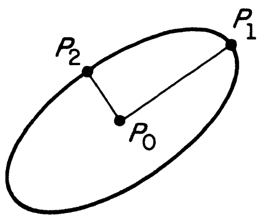
If a circle of radius R is to be drawn starting at the right of the center on the line $Y = Y_0$, the equations reduce to

Clockwise	*Counterclockwise*
$J = R$	$J = M = R$
$K = L = 0$	$K = L = 0$
$M = -R$	

Ellipses. Consider an ellipse in any orientation, where P_1 is an end point of one axis and P_2 is an end point of the other. Then the parameters

$$
\begin{aligned}
J &= X_1 - X_0 \\
K &= X_2 - X_0 \\
L &= Y_1 - Y_0 \\
M &= Y_2 - Y_0
\end{aligned}
$$





define an ellipse whose trajectory begins at P_1 and moves toward P_2. Hence simply by judicious choice of points, the programmer can draw an ellipse starting at either end of either axis and going in either direction. Ellipses can be drawn starting at any point, but the equations are somewhat more complicated (refer to the appropriate Conographic Application Notes).