Implementation of GROATS

This Algol Paper is not for general distribution. It describes the internal workings of the GROATS I/O package and is mainly produced as documentation of the system.

1. INTRODUCTION

This paper assumes that the reader is familiar with the GROATS system, has read the GROATS manual carefully and has a listing of the GROATS package. To avoid having to enclose identifiers in quotes, these will all be written as upper case letters in this paper. In fact all identifiers are lower case letters in the GROATS system.

The GROATS package is basically a set of Algol procedures and declarations which can augment any of the standard sets of permanent procedures (called PERM's). The original definition of PERM's in the Atlas Algol system did not allow any declarations other than procedure declarations. This was found to be extremely tiresome when defining the KDF9 and ELLIOTT PERM's. Any communication between routines had to be through special locations that the writer knew could not be overwritten. Several tricks were employed in passing such information from routine to routine. This often caused problems at a later date. For example the KDF9 magnetic tape procedures have an unusual way of defining whether the last record read was the last defined on the tape. Looking back on it, the method chosen was unfortunate and has caused considerable trouble. Unfortunately, removing this restriction on declaration types is sufficiently difficult that it is not worth the effort at this point in time. Basically what happens is that the property list information concerned with identifier names declared in the PERM is not retained. Therefore, although space is allocated in the stack for them, the user program, when it is being compiled, has no knowledge of their existance and so resets the stack pointer and reallocates this space to variables defined in the outer block of the program. It was found that a simple modification to the compiler could, however, allow the stack pointer for the PERM and program outer block to be the same. Consequently, if the same set of declarations are defined in the PERM and in the outer block of the program, then these variables are available to both the PERM and the outer block of the program. It is as though the declarations in the outer block of the program were moved to the outer block of the PERM. The set of declarations required in the PERM are defined as SC1 on R502 ALGOLIB. Every program using GROATS is therefore forced to have this declaration in the outer block of the program and it must be the only declaration in the outer block. If the set of declarations SC1 is not declared, then the stack in the program and PERM will not be compatible. This will mean variables may be overwritten and the control for the program could go awry.

Once this restriction has been put on the user program, the definition of an I/O package is much simpler. A large part can be written in Algol rather than machine code. Without this facility, global parameters have to be passed using absolute addresses and this, of course, forces the programmer to drop down into code procedures. This facility also means that, as well as the PERM procedures using these global variables, it is possible for the user program to access them. In

general an average user would not need to access these variables. However library procedures can make use of them and new parts to the system can be debugged in a program before adding to the PERM.

The GROATS package is defined as several library items on the disc area R011 SC4020LIB. A copy is also kept on the tape N1130 CC3 as a back-up facility. A deck of cards is also kept and these should always be identical with R011 SC4020LIB. The tape N1130 CC3 may be slightly out of date. A copy of the disc area is made onto tape every time a new supervisor version of the Algol system is installed. The layout of the PERM on R011 SC4020LIB is:-

Library Item	Contents
SC1	GROATS scalar and array declarations
SC2	TOBUFFER to PRINTINTEGER
SC3	IDENTIFICATION to FILEMARKS
SC4	TYPESTRING to TYPENEWLINES
SC5	CHARACTERFACTOR to MOVETYPETOTOP
SC6	VECTOR to SCALESAT
SC7	AXESAT to EXPANDANDROTATE
SC8	POINTER to USESAVED
SC9	LINEPRINTERON to LINEPRINTERSIM
SC15	SHADOW to RASTERLENGTH
SC16	SETTINGCHAR to SETPLOTCHAR
SC37	VECTOR {Algol version}
SC38	SUPVECTOR {Algol version}
SC39	SUBVECTOR {Algol version}
SC40	SEEVECTOR {Algol version}

The order of procedure declarations in the PERM is given by the GROATS block map in the Appendix 6 in the GROATS manual.

The global variables will normally be initialised to some value. This is done by the call of IDENTIFICATION. This procedure and its subroutines therefore serve two purposes. They define the identification frames to be added to the output at the front of a job and also correctly initialise all the global variables. A GROATS program will not, therefore, work at all without calling IDENTIFICATION first. If any graphic output is attempted without calling IDENTIFICATION, the program usually blows up on a division overflow caused by regions having upper and lower bounds the same. A complete list of global variables is given in Appendix 1 with their meanings and initial values.

2. THE OUTPUT BUFFER FOR THE SC4020

Communication with the SC4020 is by 7-track IBM tape. Records are less than or equal to 170 36-bit SC4020 words. (That is 127.4 Atlas 48-bit words). All output to the 7-track tape is buffered, a record at a time in the array SCBUFFER. This buffer is 170 words in length and, as it is being filled, contains one 36-bit SC4020 instruction per 48-bit word. The contents of each word consists of an unstandardised floating point number which is equivalent to the numeric value of the SC4020 instruction. That is the word contains:-

*030ABCDE/*FGHIJKLO where

A → L are the 12 octal digits of the SC4020 instruction. The variable

SCBFPOINTER will normally point to the next free location in the output buffer.

All entries in the buffer are added by the procedure TOBUFFER. This provides a rigorous interface between the output side of GROATS and the rest of the package. To change the format in the buffer, all that would be required is TOBUFFER being changed. The procedure TOBUFFER will add a single SC4020 instruction to the buffer. If the buffer is already full, this will be passed to the 7-track tape by the procedure OUTBUFFER and reinitialised before the instruction is added. The only other function performed by TOBUFFER is to add the instruction to a save area if it is required.

The procedure OUTBUFFER will normally output a full buffer when a call of TOBUFFER is made with the buffer already full. However, it can also be called from the procedure IGNORERESTOFRECORD. In either case the buffer is output up to SCBFPOINTER. OUTBUFFER will first print the contents of the buffer if TOBEPRINTED is true. This was a debug aid originally but can still be useful when turnround is poor. The procedure PRINTBUFFER is used to print the buffer. It outputs each instruction as 12 octal digits using the standard output character routine entered via location 0.4. Once the buffer is printed (if required), the SCBFPOINTER-1 words to be passed to the 7-track tape are packed up (by the procedure PACKBUFFER) so that, instead of having one 36 bit instruction per 48 bit word, no space is left between instructions and four 36-bit words are packed into three 48 bit Atlas words. Once this has been accomplished the buffer is passed to the 7-track tape. Care must be taken that a second attempt to output this buffer is not made when the buffer is in this packed up form. Otherwise the packed buffer will be packed with disasterous results. It can only occur if an error trap is initiated while the buffer is being output. This could be caused, for example, by a tape fault, computer time exceeded etc. To stop this causing trouble, the procedure OUTBUFFER has a local variable SCBPOINTER which is used by PACKBUFFER. This is set equal to SCBFPOINTER and SCBFPOINTER reset to 1 before PACKBUFFER is entered. Any error will then assume that this buffer has been output and will not attempt to output this buffer again. The variable SCRECORDS is incremented each time OUTBUFFER is called. The buffer is, of course, output only if TOIBMTAPE is true. Whenever TOIBMTAPE is false the buffer will not be passed to 7-track tape. However SCRECORDS will still have been incremented so that this gives a count of the number of records that might have been output.

The procedure IGNORERESTOFRECORD will cause partially full buffers to be output. The SC4020 instruction 07 is first added to the buffer before output takes place. The adding of this instruction is not essential. It is really a belt and braces technique. The FORTRAN package only outputs complete 170 word records. It uses 07 to output partial records by outputting a full record and ignoring those orders after the 07 instruction. This does mean that the tapes produced by the two systems do differ. Deck faults involving short records would be picked up by GROATS alone. On the other hand the FORTRAN package requires correct execution of the 07 order (This may not be completely true as the FORTRAN package fills the rest of the buffer up with NOOP's).

The reason for storing SC4020 words in the buffer initially as integers was so that all loading of the buffer could be done by normal Algol procedures and the buffer itself could be a standard Algol array. Little has been gained by this approach and a rewrite might well load the buffer in a packed form initially. However the integer format is ideal for saving SC4020 words in save areas. Manipulation of these is then by standard Algol instructions. There may well be a case for storing the words in a floating point form. (See Section on saving).

3. CONTROL PROCEDURES

Many basic control procedures just consist of a call of TOBUFFER with the relevant SC4020 instruction. For example:-

PROCEDURE	SC4020 OP CODE
STOPTYPING	12
PROJECTFORM	50
STOPSC4020	37
EXPANDIMAGE	44
REDUCEIMAGE	45
EXPOSELIGHT	04
EXPOSEHEAVY	02

Other procedures do other minor manipulations as well. For example SELECTCAMERA first checks that the camera to be selected is 1, 2 or 3 (indicating both) and resets CAMERA to the new value before adding the SELECT CAMERA instruction to the buffer. Calling the procedure with an argument greater than 3 is equivalent to 3 and less than 1 is equivalent to 1.

The procedures ADVANCEFILM and ADVANCEREPEAT output the SC4020 instruction and follow it by a call of the procedure ADVANC. This subroutine increments the counts MFFRAMES and HCFRAMES of frames generated and outputs the cutmark and frame count if they are required. The frame count is output as hardware characters and assumes that the region -1 is available and in the same form as its initialised state. In addition the typing position is reset to the top left hand corner.

The procedure ADVANCEREPEAT is similar to ADVANCEFILM in as far as it outputs the SC4020 instruction, ignores the rest of the record and calls ADVANC. In addition it must ensure that the counts of output are correctly updated. The procedure, when called with argument 0, will store the current values of SCRECORDS, HCFRAMES and MFFRAMES. When used with a non-zero value as argument, the number of extra frames being output due to the repeat can be calculated using the old and new values of the identifiers given above. The ADVANCEREPEAT facility will only work if the information being moved over is less than 512 records in length. A check is therefore made that this has not been exceeded. If it has, then a diagnostic is printed. Similarly a correct estimate of hardcopy and microfilm frames cannot be given if the camera selection has changed during the set of instructions involved in the ADVANCEREPEAT. This also causes a diagnostic to be issued.

Several control procedures just generate a STOPSC4020 order and issue a diagnostic. When the 7-track tape is being run, the SC4020 will stop and the operator will take the action defined in the diagnostic. So few people have attempted this that whether it will work in practice is still debatable. For example INSERTUSERFORMSLIDE will stop ready for the operator to mount a form slide. Similarly MOUNT35MMCAMERA will stop to allow the 35mm camera to be selected. The assumption is made that the 16mm camera is loaded initially. There is a problem here as no output can be made until IDENTIFICATION is called to initialise the buffer. However we wish to load the 35mm camera before calling IDENTIFICATION. The routine at the moment just does not work as STOPSC4020 is called before initialisation.

4. REGIONS

All plotting in GROATS is defined relative to some region. The user has a number of regions which he may use. These can be positioned anywhere on the plane that includes the plotting area. They may have any size or rectangular shape that the user wishes to specify. At any instance one particular region is the selected region and all plotting is defined in terms of this region. The region has its own unit system independent of the other regions. If an (X,Y) value is used in a line drawing or character plotting order then this (X,Y) is defined in terms of the units of the selected region. The position in the region is therefore defined and, as the position of the region relative to the plotting area is known, the position of (X,Y) on the plotting area is also known. Initially all the regions -2 to 10 are defined to have units equivalent to the basic SC4020 units and each region is coincident with the plotting area of the SC4020.

Each region has a number of parameters which are local to the region. These are stored in a set of arrays having dimensions [-2 : 10] and having a name starting with an initial letter S. These variables could have been accessed from this array all the time. However most references will be made to variables in the selected region. Therefore, for efficiency, a copy of the parameters for the selected region are stored in a set of scalar variables having the same name as the corresponding array variable except that the initial letter S is removed. In this section we shall write in terms of the scalar variables although it must be remembered that the corresponding arrays exist. The act of selecting a region must therefore store away the scalar variables for the previously selected region (this is necessary as they may have been altered) and replace these with the values for the newly selected region. The assumption is made that region selection is less frequent than actual plotting so that this overhead is made up for by the faster accessing of the selected region parameters.

The units and positions of a region are defined by the variables XMIN, YMIN, XMAX, YMAX, XPMIN, YPMIN, XPMAX and YPMAX. The first four variables define the units for the region and are set by a call of:-

```
limits (XMIN, YMIN, XMAX, YMAX);
```

The second set of four variables define the position of this region on the SC4020 plotting area. Thus (XMIN, YMIN) in the units of the selected region corresponds to (XPMIN, YPMIN) on the plotting area (These units are the SC4020 raster positions and correspond to X increasing from left to

right and Y increasing from top to bottom. X and Y both go from 0 to 1023). A call of the procedure:-

```
region (X1, Y1, X2, Y2, I);
```

will define

```
SXPMIN[I], SYPMIN[I], SXPMAX[I], SYPMAX[I]
```

in terms of the 8 parameters listed above for the selected region. Notice that regions are not really defined relative to each other. Although the position of the selected region is used in defining the new position of the region I, the result is that I is defined in terms of the plotting area. A later change in the position of the selected region will not change the position of region I. The complete set of variables associated with each region is as follows:-

Size Parameters

XMIN, YMIN, XMAX, YMAX, XPMIN, YPMIN, XPMAX, YPMAX, SIGNX, SIGNY, WNDOW, CONTAN, ROTORIGX, ROTORIGY, EXPANSIONFACTOR, SINROTANGLE, COSROTANGLE

Line drawing parameters

DOTTED, DARKVAR, THICKVAR, DRAWAXES, INVISIBLE, SHADOWVAR

Character printing parameters

XTYPE, YTYPE, FONTVAR, HARDWARE, HARDWARESPACE, FACTORDEFINED, HEIGHTCHAR, WIDTHCHAR, SPACECHAR, FACTORCHAR, DISPLACEMENT

The variables SIGNX and SIGNY are not independent variables. If XMIN < XMAX then SIGNX = +1 else -1. Similarly for SIGNY. This value is frequently required in the increment operation where you wish to move from left to right on the plotting area even though XMAX < XMIN. To save recomputing it each time, the value is only defined when LIMITS is called.

The two operations, rotation and expansion, applied to all plotting in a particular region are frequently required. Consequently three procedures ROTATE, EXPAND, and EXPANDANDROTATE are provided which define an origin, (ROTORIGX, ROTORIGY) and an EXPENSIONFACTOR and a rotation defined by (SINROTANGLE, COSROTANGLE). Notice that the sin and cos of the angle to be rotated are stored rather than the angle itself. This, of course, saves computing these trigonometric functions unnecessarily. Instead of the (X,Y) position given by the user, a new position (X′,Y′) is calculated. The action is as though the user specified (X′,Y′) with no rotation or expension rather than (X,Y).

```
X′ = ROTORIGX + EXPANSIONFACTOR* (DX*COSROTANGLE - DY*SINROTANGLE)
Y′ = ROTORIGY + EXPANSIONFACTOR* (DX*SINROTANGLE + DY*COSROTANGLE)
```

where

DX = X-ROTORIGX and DY = Y-ROTORIGY

Notice that any scissoring required is done with respect to the new values (X′,Y′). This means that scissoring will be with respect to the selected region in its unrotated form. Also rotation is in terms of the units defined for the region rather than its position on the plotting area. This can give unexpected results when the units in X and Y are different. To avoid distortion, a good rule would be to apply rotation only in regions having the same units in both directions.

5. LINE DRAWING

All line drawing routines eventually call:-

```
VECTOR (X1, Y1, X2, Y2);
```

where S is the selected region, P is the passive region and A is the active region. The values of S, P and A are stored in the variables SELECTEDREGION, PASSVE and ACTVE respectively. The call of VECTOR produces line or lines depending on the following variables:-

Selected Region

(1) Rotation parameters to convert (X,Y) to (X′,Y′). See Section 4.

(2) CONTAN, WNDOW

(3) XMIN, YMIN, XMAX, YMAX

(4) XPMIN, YPMIN, XPMAX, YPMAX

(5) DOTTED, THICKVAR, DARKVAR, DRAWAXES, INVISIBLE, SHADOWVAR

Passive Region

(1) SXPMIN[P], SYPMIN[P], SXPMAX[P], SYPMAX[P]

Active Region

(1) SXPMIN[A], SYPMIN[A], SXPMAX[A], SYPMAX[A]

(2) SWNDOW[A]

The procedure VECTOR has had several forms since GROATS was issued. It is the major routine of the system and most time is spent in this routine. Consequently every effort is made to improve the running speed of this procedure. The current form was originally coded as four standard Algol procedures VECTOR, SUPVECTOR, SUBVECTOR and SEEVECTOR. These are stored in library items SC37, SC38, SC39 and SC40 of R011 SC4020LIB. The actual form of these routines in the PERM is a hand coded version which should be very close in form to the Algol version. The SEEVECTOR procedure is an internal subroutine of SUPVECTOR in the hand coded version. It is hoped to keep the original Algol versions updated so that they can act as documentation and also aid the transfer of the package to another machine.

The procedure VECTOR may generate a number of lines depending on such parameters as DARKVAR, THICKVAR etc. The procedure VECTOR determines the number of lines that need to be drawn and calls SUPVECTOR to draw each one in turn. Thus SUPVECTOR is called to draw a single line between two points in the selected region. Due to scissoring by the selected region,

(also the passive and active regions), not all of the line will necessarily be visible. The procedure SUPVECTOR calculates the parts of the line that are visible and calls SUBVECTOR to output each one in turn. The procedure SUBVECTOR is requested to draw a line between two positions on the SC4020 plotting area. SUBVECTOR may have a choice of SC4020 instructions to choose if the line is horizontal or vertical and, if the line is longer than 64 raster positions in either coordinate, the line must be broken down into setments equivalent to SC4020 orders.

The procedure VECTOR in machine code is practically identical with the Algol version. It uses the stack for local variables and the following correspondence occurs between the temporary cells used in the machine code version and the local variables in the Algol version:-

Location	Variable in Algol version
B90 + 0	RX
+ 1	RY
+ 2	D
+ 3	DX
+ 4	DY
+ 5	ALPHA
+ 6	T2
+ 7	STH
+ 8	CTH
+ 9	XB1
+ 10	XB2
+ 11	YB1
+ 12	YB2
+ 13	M
+ 14	I
+ 15	DRK
+ 16	CC
+ 17	CMIN
+ 18	
+ 19	$T2^2$
+ 20	
+ 21	
+ 22	DOTTED
+ 23	M (floating point standardised)

In addition B82 is used to contain THICKVAR and B80 to contain DRK.

A summary of the action of this routine is as follows:-

(1) If ABSOLUTEVEC is false and SAVEADDR > 0 then the line drawing commands are being saved in a non-absolute form. The values of SELECTEDREGION - 100, X1, Y1, X2 and Y2 are all stored in the save area and SAVEADDR is updated.

(2) If SHADOWVAR ≠ 0 then the appropriate shading is added.

(3) (LASTX, LASTY) is set equal to (X2,Y2).

(4) If DOTTED > 0 then the line is to be drawn broken. The value of dotted gives the length of the line in raster units. The number of intervals between (X1,Y1) and (X2,Y2) is estimated and this rounded up to the next odd integer. This is to ensure that a visible segment of the broken line appears at each end. The line is then divided up into this number of segments and each one plotted in turn.

(5) If THICKVAR > 1 then extra THICKVAR - 1 lines are drawn on either side of the original line. These lines are drawn parallel to the original line and a distance about one raster unit away. First the distance away of a line parallel to the original line but passing through (X1 + RX, Y1 + RY) is calculated where (RX,RY) are the number of units in the X and Y directions equivalent to one raster unit. Using this value, a line is drawn parallel to the original line and the same distance away as this line. Other lines are multiples of this distance away.

(6) If DARKVAR > 1 then the line is output DARKVAR times. The first line is drawn from (X1,Y1) to (X2,Y2) while the second line is drawn from (X2,Y2) to (X1,Y1) and so on. Switching direction each time will normally ensure that the breakdown of the complete line into segments equivalent to SC4020 instructions will be different in each direction. Any gaps or superimposition caused by bad setting up of the SC4020 will now be less visible.

The procedure SUPVECTOR is similar again to the Algol version. The following equivalences exist between stack entries and local variables in the Algol version.

Location		Variable in Algol version	
B90 + 0		SEE /X1IN	
+ 1		Y1IN/X2IN	
+ 2		Y2IN/INSQ1	
+ 3		INSQ2/	
+ 4			
+ 5		XX1	
+ 6		YY1	
+ 7		XX2	
+ 8		YY2	
+ 9		I	
+ 10		C	
+ 11			
+ 12			
+ 13		SIGNX	
+ 14		SIGNY	
+ 15	)	XMIN	
+ 16	)	YMIN	Arguments to
+ 17	)	XMAX	SEEVECTOR subroutine
+ 18	)	YMAX	at label 60

In addition B92 frequently points at the arguments of the procedure so that

B92 + 0	≡	X1
+ 1	≡	Y1
+ 2	≡	X2
+ 3	≡	Y2

In addition the following variables will normally reside in B registers as follows:-

B96	X1IN	then	INSQ1
B97	Y1IN		
B98	X2IN	then	INSQ2
B99	Y2IN		

The only part of the machine code procedure SUPVECTOR which differs considerably from the Algol version is the subroutine at label 60 which replaces the SEEVECTOR procedure. Care has been taken to ensure that no unnecessary testing occurs in this routine. Appendix 2 gives a flow diagram of this procedure. Labels used in the machine code routine are marked. The assumption is made that, in general, a line is unlikely to be scissored by more than one line edge of the region. Consequently what looks like recalculation of expressions like (X2-X1), in practice, will not take place.

A summary of SUPVECTOR is as follows:-

(1) Initially convert (X1,Y1) and (X2,Y2) to raster units, first applying any rotation or expansion defined for the selected region. At this stage the absolute positions of the two end points on the plotting area of the SC4020 have been obtained. They may of course be outside the 1024 x 1024 raster.

(2) The line between these two points is scissored by the edges of the passive region so that only the part of the line interior to the passive region is considered. This is done by the internal subroutine SEEVECTOR.

(3) The line may or may not be scissored by the selected region. This depends on whether the region is defined as contained or extended and whether it is a window or a shield. The possibilities are:-

CONTAN	WNDOW	
true	true	The part of the line inside the region is drawn
true	false	None of the line is drawn
false	true	All of the line is drawn
false	false	The part of the line outside the region is drawn.

(4) If the active region is not the selected region and it is a shield then the parts of any lines that would have been drawn inside the active region are made invisible.

The procedure SUBVECTOR in machine code closely resembles the Algol version. The following equivalences exist between the stack entries and local variables in the Algol version.

Location	Variable in Algol version
B90 + 0	YS / XS
+ 1	TOP / TMP
+ 2	DX / SIGNDX
+ 3	DY / SIGNDY
+ 4	VECTORLENGTH / XORIG
+ 5	YORIG / XP1
+ 6	YP1 /
+ 7	DELY
+ 8	DELX
+ 9	XST
+ 10	YST
+ 11	
+ 12	
+ 13	
+ 14	X2
+ 15	Y2

In addition:-

B93	DX
B94	DY
B95	SIGNX
B96	SIGNY

The basic form of the procedure SUBVECTOR is:-

(1) Initially decide whether an axes rather than vector drawing order can be used. The line must be horizontal or vertical, long enough to be an axis and DRAWAXES must be true.

(2) If the line is of zero length and INVISIBLE is true then no line is drawn.

(3) The slope of the line is determined and the maximum length line segment is found where neither its X or Y increment exceeds 61 raster units. The maximum allowed is 63. The value 61 is chosen to allow for possible rounding errors.

(4) The line is divided up into segments of this length starting from (X1,Y1) and drawn.

The subroutine SEEVECTOR, given X1, Y1, X2, Y2 and the limits XMIN, YMIN, XMAX, YMAX defining a region, will return the values RPARAM1, RPARAM2, RPARAM3, RPARAM4 defining the end points of the part of the line that is interior to the region. In addition SEE is set to true if some part of the line is interior to the region. The procedure has undergone considerable modification since GROATS was started. In the following description it will be assumed that XMIN < XMAX and YMIN < YMAX. The alternative cases are catered for by suitable multiplications by +1 or -1 depending on whether the low bound is smaller than the upper bound or not. Instead of attempting to find in one step how much of a line is inside the region, the process is broken down into finding how much of the line is inside each of the four lines that border the region.

The process can be likened to cutting along each edge of the region in turn (going right to the edge of the 'paper', not stopping at the end of the region). More than one cut may scissor the line (all four can). Eventually the user is left with just the rectangular region and the part of the line drawn interior to the region. This procedure is reasonably elegant in that there are no special cases and the routine separates into four distinct parts:-

(1) Scissor with respect to x = XMIN
(2) " " " " x = XMAX
(3) " " " " y = YMIN
(4) " " " " y = YMAX

In (1), X1 and X2 are compared with XMIN. If both are less than XMIN then the line is completely outside the region. If X1 and X2 are both greater than XMIN then the line is not scissored at all by x = XMIN. If one is greater than XMIN and the other less than XMIN then the intersection of the line with x = XMIN is calculated and the end point outside the region is replaced by this intersection point.

6. CHARACTER PLOTTING

GROATS has a set of characters numbered from 0 to 255 and these may be output by the user in any size and at any position on the plotting area. In addition the form of the character may vary depending on which font is defined for the character to be output. The process of outputting a character is therefore:-

(1) Find character number
(2) Find which font is required
(3) Locate the template for this character in this font
(4) Find the size of character required and scale up the template until it fits this size
(5) Output the lines making up the character at the required position.

The array CHARACTER [0:3, 0:255] points to the templates for the characters. The font being used by the currently selected region is kept in FONTVAR. Therefore to output character C it would be necessary to examine CHARACTER [FONTVAR, C]. Information concerned with the character fonts 0, 1, 2 and 3 is stored in the areas *2660, *2664, *2670, *2674. Four blocks are allocated for each font definition. Let us define B as the base address for each font definition. For example the base address B for font 1 is *2664. If CHARACTER [FONTVAR, C] = 2P then the template for character C of font FONTVAR starts at location B + P + 133. (The reason for the 133 displacement will become apparent later). Each character definition consists of a set of commands which cause lines to be drawn between certain positions on the template. The template can be thought of as a GROATS region. It is obviously necessary to know the bounds of the template region and, in addition, the position of the character's centre. Each font has a template region whose units are defined for the whole font. This is equivalent to:-

```
limits (0, 2H, W, 0)
```

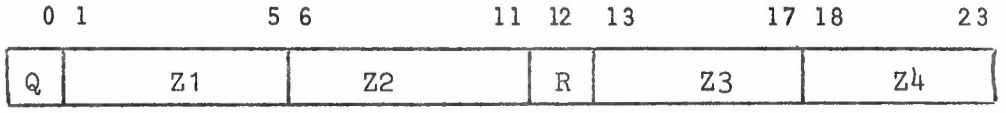
where H = FONTHEIGHT [FONTVAR]
W = FONTWIDTH [FONTVAR]

and the character's centre is assumed to be at:-

(CX,CY)

where CX = FONTCENTREX FONTVAR
CY = FONTCENTREY FONTVAR

In order to keep the storage requirements for a font definition to a minimum, it was decided to pack up the template commands for a particular character to one or two lines per half word. To do this it is necessary to make some restrictions on the template size. The limit imposed is that both H and W must be integer and less than or equal to 31 in value. Also all line commands must refer to integer positions in the template region. With these restrictions it is easily seen that an X position can be defined in 5 bits and a Y in 6 bits. Therefore two X and two Y values can be stored in a half word with two additional bits that can be set to define what these X and Y values mean. The template for a character therefore consists of a halfword giving the number of halfwords following that define the character. This is followed by half words having the general format



The count of half words is in fact stored in the character position so that, for example, a count of 0.3 indicates three half words to follow. The fields Z1 and Z3 normally represent X coordinates while Z2 and Z4 represent Y coordinates. Using the GROATS notation for line drawing, such a template command has the following meaning:-

Q = 0, R = 0	VECTOR (Z1, Z2, Z3, Z4)
Q = 0, R = 1	TOPOINT (Z1, Z2); TOPOINT (Z3, Z4)
Q = 1, R = 0	TOPOINT (Z1, Z2); LASTX = Z3; LASTY = Z4
Q = 1, R = 1	TYPEORPLOTCHAR (Z3, Z4, Z1*64 + Z2)

The compactness of this notation is particularly good for characters made up of a number of connected segments. The second category above means that these lines are defined at two per half word once a VECTOR command initiates the sequence.

Each font is assumed to have a basic size defined on the SC4020 raster. This is W raster positions wide and 2H raster positions in height. This is equivalent to the size defined by a call of CHARACTERFACTOR (1).

Once the template for a particular character has been decoded, its actual position and size in the currently selected region are defined by the parameters

HEIGHTCHAR = half height of character
WIDTHCHAR = width of character

and either the centre of the character is defined by the plotting routine or alternatively uses the assumed typing position of (XTYPE, YTYPE). A point (X,Y) in the template definition corresponds to a point (X',Y') when plotted where:-

X' = XTYPE + (X - CX)*WIDTHCHAR/W
Y' = YTYPE - (Y - CY)*HEIGHTCHAR/H

assuming (XTYPE, YTYPE) is the centre of the character to be plotted.

So far we have made the assumption that only software characters are available. The SC4020 does have a set of hardware characters and the facility is introduced to allow the user to mix hardware and software characters. Of course this is only sensible if the size of the software characters is defined as the same as the hardware characters. A special procedure is provided for declaring the size of software characters to be the same as hardware. Apart from setting the size correctly, it also sets the Boolean variable HARDWARE to true.

If a character has a possible hardware character that could be used in its place then CHARACTER [FONTVAR, C] instead of being set to 2P is set equal to $-(D*2^{14} + 2P)$ where D is the equivalent hardware character and P is the pointer in the font table as before. D is the SC4020 number of the character.

The basic routine for typing or plotting characters is TYPEORPLOTCHAR. It has a Boolean argument to define whether typing or plotting is required. This does not make any difference if a software character is being output. However with hardware characters, different SC4020 instructions will be used in the two cases. Special orders exist for packing together hardware characters being typed. The procedure first picks up the value of CHARACTER [FONTVAR, C] and, if it is negative and HARDWARE is true, the hardware character will be output. Before this is done the procedure INVIEW checks that the character is visible. This procedure checks the position with respect to the passive region and also the selected region (At the moment no check is made to see if it is shielded by the ACTIVE region. The ACTIVE region is probably only used in cine work whereas the hardware characters are mostly used in conventional plotting so it may not be too important).

To type a complete string, the procedure TYPESTRING is defined. However we have one important problem. The GROATS package has a 256 character set and yet in the data and program not more than 128 internal code characters can be written. This is solved by having the parameter DISPLACEMENT specify an offset from the character that is requested. For example, if the procedure TYPE(N) is called then, instead of outputting N, the character output is DISPLACEMENT + N.

The procedure TYPESTRING is designed to type a string that has been read from the input or defined in the program. It is frequently convenient to include format information in the string to be typed. This is done by the DEFINECHARASSPECIAL procedure which redefines a character as a setting procedure. This is implemented by having a third possibility for the contents of CHARACTER [FONTVAR, C]. If it contains SOFTMAX + 32*I + J then it is equivalent to the setting procedure defined by:-

```
DEFINECHARASSPECIAL (C, J, I);
```

The standard value of SOFTMAX is currently 4096. It could be changed to another value without too much trouble. The procedure STRINGCHARACTER is used to pass the characters of the string one at a time to the procedure TYPESTRING. The procedure will output the value 500 when it has passed on all the characters in the string. This is used therefore to denote the end of the string.

For each character of the string, the contents of CHARACTER [FONTVAR, C] are examined. If the value is greater than SOFTMAX then it is a setting procedure and the equivalent procedure is called. If the string being output is not hardware size then characters are drawn one at a time by TYPEORPLOTCHAR. If the string is being output hardware size then special action must be taken. The first character, assuming it has a hardware equivalent, can be output using the SC4020 TYPESPECIFIED POINT command. Assuming standard spacing between characters, following characters with hardware equivalents can be output 6 to an SC4020 word. The major part of TYPESTRING is therefore con cerned with this. It is essential that the SC4020 is returned to the non-typing mode by finishing the set of hardware characters with a STOPTYPING character. If non-standard spacing is specified between characters then the hardware characters have to be output one at a time using the TYPE SPECIFIED POINT command. Non-standard spacing is achieved by the CHARACTERSPACE procedure.

The procedure TYPENUMBER is designed to output a number in a similar format to the standard print routine in the I/O package accessed via location 0 (see Algol Paper No 2). The procedure modifies the output character routine by replacing its entry point in location 0.4 by a pointer internal to TYPENUMBER. This internal routine collects the characters of the number in the output format and stores them in a string internal to the procedure. TYPESTRING is then called to type this string.

The standard character fonts are stored on a disc area R067 CH in blocks 1 to 16 (4 blocks for each font). In the disc area for a font definition are stored the character templates and also the array CHARACTER entries and the values of FONTHEIGHT, FONTWIDTH, FONTCENTREX and FONTCENTREY. When the procedure FONT is called, the base address for the font is set up in B81, B80 is set to the font number and B82 is set to the first block where the font will be stored on a disc area. If the contents of (B81 + 0.4) is zero then the assumption is made that the font has not been used before. If no tape 123 is defined in the Job Description then the disc area R067 CH is loaded and blocks B82 to B82 + 3 on the disc area are read into B81 to B81 + *0003. If a tape or disc 123 is defined then the blocks are read from this device. This allows non-standard fonts to be provided by the user. The contents of the four blocks relative to the base address of the area is set as follows:-

0	FONTWIDTH
1	FONTHEIGHT
2	FONTCENTREX
3	FONTCENTREY
4	SOFTPOINTER
5	CHARACTER [I,0]
5.4	
132.4	CHARACTER [I,255]
133	Font templates

The CHARACTER entries either contain N or N.1 where N is an integer. When the font is read down, the variables in locations 0 to 132.4 are loaded into their respective variable positions. The array CHARACTER is loaded with -N if N.1 is set and N otherwise. The variable SOFTPOINTER points to the end of the font templates. It is only used if additional character definitions are made at a later time.

Given the font definition, it is usually a simple matter to define the size of the characters which will fit with the hardware characters. The size of hardware characters is basically 8 x 8 raster units and so it is just a matter of defining the size in terms of the region's units. However font 0 was defined initially on a 10 x 8 grid (W = 10, H = 8). Reducing the width of the character from 10 to 8 tended to distort the characters due to rounding. Consequently it was found that, if the character size was defined with a height of 8 raster positions and 10 in width, then the correct hardware equivalent would be obtained. To keep the size of each character to 8 raster units in width it is therefore necessary to define a character space of -2 raster positions. It would, of course, have been more sensible to define the font 0 on an 8 x 8 raster initially. Unfortunately, this was not done.

7. IDENTIFICATION

The output for a particular user is identified by the procedures IDENTIFICATION and ENDPLOTTING which should be called before and after the plotting required by the program. The definition of output generated by these routines is given in the GROATS manual while the initialised values of variables are given in Appendix 1.

The procedure ENDPLOTTING, in addition to producing graphical output to complete the job, will put 5 filemarks on the 7-track tape to ensure that the SD4020 operator realises that the end of information plotted has been reached. The operator will override a single filemark. If the job ends abnormally then it would be best if the procedure ENDPLOTTING could still be called. This is difficult in Algol as the basic trap routine will be entered and it will not be possible to call ENDPLOTTING from this routine as the correct calling sequence will not be known for the level of nesting. However if the trap routine is allowed to produce a retroactive trace then, at the end of this, the stack is left in the configuration that it would have on the top level of the program. A call of another procedure on this top level is then easily achieved. The procedures TRAPTO, ERRORTRAP do this. The procedure TRAPTO first finds the entry position of ERRORTRAP in the directory and plants a call to this procedure in the locations from label 4 onwards. It must then find the final 1117, 0, 0, 0 order in the trace routine and change it to jump to the calling sequence at label 4. The standard I/O package has this 1117 order a fixed distance away from the entry point to the trace procedure. However modifications are often made to the standard trace routine to allow special trap action to occur before the retroactive trace. Consequently the trace routine pointed at by location 11 may be a blister added on top of this basic routine. In this case it is necessary to move down the standard set of entry points until the basic trap routine is found. The most likely listers are those added by the KDF9 and ELLIOTT I/O packages. The standard I/O trace starts off

```
        121, 127, 127, -
B:      113,   0, 127, -
```

while blisters are set up to look like

```
        121, 127, 0, A
C:      121, 127, 0, B
```

The location A is the blister routine itself which will be entered for special diagnostics. Machine traps would normally go to B but will be blistered to go to C instead. It is, therefore, a simple matter to see if the order is a 113, 0, 127 or a 121, 127, 0 and therefore decide whether or not a lister is being looked at. If it is then the address of B can be found and the process repeated.

It is possible that an error trap can occur when the ENDPLOTTING routine has already been called from an error trap. For example, if the 7-track tape is already full then this situation could occur. The ERRORTRAP routine will only attempt to call ENDPLOTTING if it is the first time it has been entered.

Procedures IDENTIFICATION and ENDPLOTTING both output information to streams 12 and 13. These use the routines PRINTTX and PRINTINTEGER which are independent of the I/O package in use. Both routines ignore the standard Algol buffering and output the information to be printed directly using extra codes. They reselect the stream previously selected before exiting.

8. SAVING GRAPHICAL DATA

The procedures STARTSAVING, FINISHSAVING, USESAVED and ABSOLUTE between them allow users to save graphical information in one of two forms and later output it. The procedures require action to be taken either in the procedure TOBUFFER or VECTOR. Two variables SAVEADDR and ABSOLUTEVEC define whether the graphical information is to be saved. In addition SAVESTART and MAXSAVEADDR define the limits of the area into which the graphical information is being put.

If ABSOLUTE is set to true and SAVEADDR > 0 then as each SC4020 instruction is added to the buffer, it is also put in the SAVE area using the procedure STOREWORD. The instruction is put into the location SAVEADDR and SAVEADDR incremented by one unless SAVEADDR > MAXSAVEADDR.

If ABSOLUTE is set to false and SAVEADDR > 0 then, at each call of VECTOR, the quantities SELECTEDREGION - 100, X1, Y1, X2, Y2 are added to the save area. As the arguments to VECTOR will normally be non-integer, the quantities are stored in the save area as floating point standardised numbers. The procedure USESAVED will then pick up information from the save area and, depending on whether the first item it picks up is positive or negative, it will either add an SC4020 instruction to the buffer or call VECTOR. Unfortunately the arrays used for saving information were defined as INTEGER initially. This was because only the ABSOLUTE form of saving was allowed at that time. However these INTEGER arrays can contain REAL information and consequently care must be taken in how this information is accessed. The current version of USESAVED is careful to load information from a save array into a REAL variable initially thus ensuring that the rest of the routine will realise that the information coming from the array is in a standardised form. This is the purpose of the identifier Z in the routine USESAVED. A new version of GROATS should really change these arguments to REAL arrays.

9. REDUNDANT ROUTINES

The package has changed considerably over the last year. In some cases these alterations have made some routines redundant. In addition a number of routines have been left in mainly as spacing devices to ensure that the numbers associated with variables and procedures in the Block Map stay the same. The variables that are not used can be seen from the listing in Appendix 1. The following procedures are redundant:-

(1) DRAWVECTOR. This is used by the Algol version of VECTOR. It may therefore be required during testing.

(2) SOFTCHARDEFNS

(3) CHARACTERINITIALISATION

(4) SEEVECTOR

They may be replaced by other procedures as the need arises.

F R A Hopgood

3 April 1970

Atlas Computer Laboratory
Science Research Council
Chilton Didcot Berkshire

APPENDIX 1

LIST OF GLOBAL VARIABLES IN GROATS

Name	Initial Value	Meaning
INTEGER		
CAMERA	2	Number of selected camera. 1 = microfilm 2 = hardcopy, 3 = both
CHARMAX	255	Maximum number of software characters allowed. May not be used
DOTTED	0	Defines whether line drawing in selected region is broken or not. 0 = not broken, I = broken with line segments approximately I raster positions in length
FRAMECOUNT	2	Each frame of output can have the number of the frame printed at the top. The value of 'framecount' is then incremented ready for printing on next frame. If FRAMECOUNT = -1 no numbering will take place
HCFRAMES	1	The number of frames of hardcopy output generated for this job
CAMERA3526	16	Defines whether microfilm camera in use is 35mm or 16mm
DARKVAR	1	The number of times each line will be overstruck in selected region
THICKVAR	1	The thickness of lines drawn in the selected region
LAST PLOT	None	The last character plotted
SHADOWVAR	0	Defines shading with respect to curves drawn in selected region
FONTVAR	0	Font defined for selected region
MFFRAMES	1	The number of frames of microfilm output generated for this job
SAVEADDR	-1	>0 means the address to which a copy of SC4020 output should be sent. <0 then no copy taken
SAVESTART	None	First address to which copy was sent in current saving
SCBFLENGTH	170	Length of buffer for SC4020 instructions

Name	Initial Value	Meaning
SCBFPOINTER	1	Position in buffer where next SC4020 instruction is to be stored
SCRECORDS	0	Number of records generated on IBM tape
SHIFT30	2^{30}	Constant used in shifting
SHIFT18	2^{18}	Constant used in shifting
SHIFT12	2^{12}	Constant used in shifting
SHIFT10	2^{10}	Constant used in shifting
SHIFT8	2^{8}	Constant used in shifting
SHIFT6	2^{6}	Constant used in shifting
OLDFRAME	None	Not used
OLDRECORDS	None	Store current value of SCRECORDS. Used by ADVANCE REPEAT
OLDHCFRAMES	None	Store current value of HCFRAMES. Used by ADVANCE REPEAT
OLDMFFRAMES	None	Store current value of MFFRAMES. Used by ADVANCE REPEAT
OLDCAMERA	None	Store current value of CAMERA. Used by ADVANCE REPEAT
SWCH	None	
SOFTMAX	4096	Maximum length of software character definition table. Used to define setting procedure definition of character
SOFTPOINTER		Position of end of character formats for font selected
STARSTRING		Address of string of asterisks
XCHARCNTR	None	Unused
SHIFT14	2^{14}	Constant used in shifting
SELECTEDREGION	0	The number of the region selected
YCHARCNTR	None	Unused
MAXSAVEADDR	None	Maximum address into which SC4020 instructions may be copied when SAVEADDR >0.
DISPLACEMENT	0	The current displacement set for characters in selected region

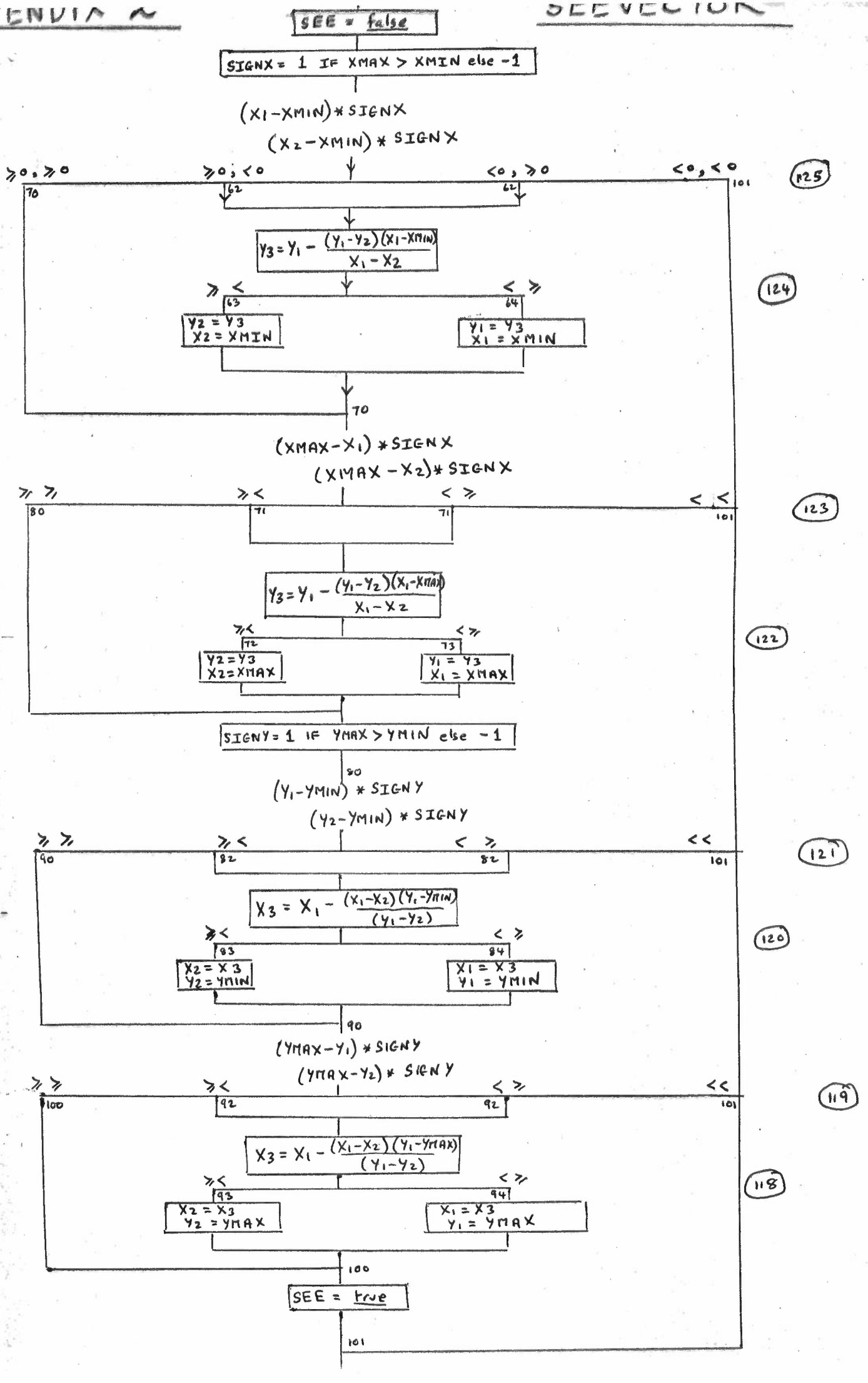
Name	Initial Value	Meaning
ACTVE	-1	The current active region
PASSVE	0	The current passive region
REAL		
XTYPE	5	The X coordinate of typing position in selected region
YTYPE	5	The Y coordinate of typing position in selected region
XMAX	1023	The upper limit of X for the selected region
XMIN	0	The lower limit of X for the selected region
YMAX	0	The upper limit of Y for the selected region
YMIN	1023	The lower limit of Y for the selected region
		(These four variables set by LIMITS (XMIN, YMIN, XMAX, YMAX);)
HEIGHTCHAR	-8	Character height in selected region
WIDTHCHAR	10	Character width in selected region
SUPSUF	0.5	Size of subscripts and superscripts
LASTX	None	The X coordinate of end of last vector drawn
LASTY	None	The Y coordinate of end of last vector drawn
ROTORIGX	0	The X coordinate for centre of expansion and rotation
ROTORIGY	0	The Y coordinate for centre of expansion and rotation
EXPANSIONFACTOR	1	Expansion defined for selected region
SINROTANGLE	0	Sin(A) where A = angle of rotation defined for selected region
COSROTANGLE	1	Cos(A) where A = angle of rotation defined for selected region
RPARAM1		Global variable used as temporary variable
RPARAM2		" "
RPARAM3		" "
RPARAM4		" "
RPARAM5		" "
RPARAM6		" "

Name	Initial Value	Meaning
RPARAM7		Global variable used as temporary variable
RPARAM8		" "
SPACECHAR	-2	Character space in selected region
FACTORCHAR	-1	Size of character defined as factor of basic font size. If set to -1 indicates character size defined in another way
XPMAX	1023	Maximum X coordinate in raster positions of selected region
XPMIN	0	Minimum X coordinate in raster positions of selected region
YPMAX	0	Maximum Y coordinate in raster positions of selected region
YPMIN	1023	Minimum Y coordinate in raster positions of selected region
SIGNX	1	Sign set to +1 if XMAX >XMIN and -1 if XMAX <XMIN
SIGNY	-1	Sign set to +1 if YMAX >YMIN and -1 if YMAX <YMIN
BOOLEAN		
WNDOW	true	true = selected region is a window, else shield
CONTAN	true	true = selected region is contained, else extended
DRAWAXES	true	true = axes drawn for horizontal and vertical lines, else vectors
INVISIBLE	true	true = zero length vectors not drawn
OLDCUTMARK	None	Not used
ABSOLUTEVEC	true	true = saved SC4020 instructions stored. in absolute form
ABSTEMP	None	Not used
BPARAM4	None	Not used
BPARAM5	None	Not used
TOIBMTAPE	true	true = records passed to IBM tape
CUTMARK	true	true = cut mark drawn for each frame

Name	Initial Value	Meaning
FIRSTTRAP	true	true = first error trap. Set to false to avoid loop on IBM tape error
TOBEPRINTED	false	true = each SC4020 record is listed on output
LPSIMON	false	true = lineprinter simulation mode
HARDWARE	true	true = characters defined as hardware size
HARDWARE SPACE	true	true = space between characters is standard so that packed up type instructions on SC4020 can be used
FACTORDEFINED	false	true = character size defined in factor mode
BOOLEAN ARRAY		
SWNDOW	true	Equivalent meaning to variable without initial letter S. Array contains values for all regions
SCONTAN	true	
SDRAWAXES	true	
SINVISIBLE	true	
SHARDWARE	true	
SHARDWARESPACE	true	
SFACTORDEFINED	false	
INTEGER ARRAY		
CHARACTER		Contains pointers to software character definitions for fonts 0 to 3. In addition contains hardware character equivalent if any and definition of setting procedure if this has replaced character definition.
SCBUFFER		Contains SC4020 instructions to be passed to IBM tape
SOFTCHARDEFN		Not used
SDISPLACEMENT	0	Equivalent meaning to variable without initial letter S. Array contains values for all regions
SSHADOWVAR	0	
SFONTVAR	0	
SDOTTED	0	
SDARKVAR	1	
STHICKVAR	1	

Name	Initial Value	Meaning
FONTHEIGHT		Basic height in raster positions of fonts
FONTWIDTH		Basic width in raster positions of fonts
FONTCENTREX		The X coordinate of centre of characters in raster positions
FONTCENTREY		The Y coordinate of centre of characters in raster positions
REAL ARRAY		
SXTYPE	5	Equivalent meaning to variable without initial letter S. Array contains values for all regions
SYTYPE	5	
SXMAX	1023	
SXMIN	0	
SYMAX	0	
SYMIN	1023	
SHEIGHTCHAR	-8	
SWIDTHCHAR	10	
SSPACECHAR	-2	
SFACTORCHAR	-1	
SXPMAX	1023	
SXPMIN	0	
SYPMAX	0	
SYPMIN	1023	
SROTORIGX	0	
SROTORIGY	0	
SEXPANSIONFACTOR	1	
SSINROTANGLE	0	
SCOSROTANGLE	1	
SSIGNX	1	
SSIGNY	-1	

This list of global variables is in the order that they appear in the GROATS declarations. Their positions therefore coincide with the positions in the trace map.


SEE = false
SIGNX = 1 IF XMAX > XMIN else -1
(X1-XMIN)*SIGNX
(X2-XMIN)*SIGNX
>=0, >=0
>=0, <0
<0, >=0
<0, <0
125
70
62
62
101
Y3 = Y1 - (Y1-Y2)(X1-XMIN) / X1-X2
124
>= <
< >=
63
64
Y2 = Y3
X2 = XMIN
Y1 = Y3
X1 = XMIN
70
(XMAX-X1)*SIGNX
(XMAX-X2)*SIGNX
>= >=
>= <
< >=
< <
80
71
71
101
123
Y3 = Y1 - (Y1-Y2)(X1-XMAX) / X1-X2
>= <
< >=
72
73
122
Y2 = Y3
X2 = XMAX
Y1 = Y3
X1 = XMAX
SIGNY = 1 IF YMAX > YMIN else -1
80
(Y1-YMIN)*SIGNY
(Y2-YMIN)*SIGNY
>= >=
>= <
< >=
<<
90
82
82
101
121
X3 = X1 - (X1-X2)(Y1-YMIN) / (Y1-Y2)
>= <
< >=
83
84
120
X2 = X3
Y2 = YMIN
X1 = X3
Y1 = YMIN
90
(YMAX-Y1)*SIGNY
(YMAX-Y2)*SIGNY
>= >=
>= <
< >=
<<
100
92
92
101
119
X3 = X1 - (X1-X2)(Y1-YMAX) / (Y1-Y2)
>= <
< >=
93
94
118
X2 = X3
Y2 = YMAX
X1 = X3
Y1 = YMAX
100
SEE = true
101