UNIVERSITY MATHEMATICAL LABORATORY
CAMBRIDGE

NOTICE TO CURRENT USERS OF GINO

As a temporary measure the Library files for GINO as described in this third edition are named CAD/GINOSAL/E and CAD/GINO/GRAPHE. CAD/GINOSAL/* and CAD/GINO/GRAPH contain versions corresponding to the second edition (June, 1969). Users will be informed when the third edition is prefered; until then it may be used on an experimental basis.

The following brief summary of the new facilities in the third edition is provided for the benefit of those familiar with the earlier edition:-

1. The versions of GINO for producing Elliott 928 display file; CALCOMP plotter output and COMPLOT on-line plotter output are new. (see routines ELLIOTTGINO, PLOTTERGINO, COMPLOTGINO).

2. Automatic scaling and positioning facilities are provided (see routines PICTURE, PICTZCLIP, OBJZDM) and some new transformation facilities (see routines AXONXYZ, PROJXYZ).

3. User-defined objects can be transformed at definition time (see section 2-3)

4. Nesting of names in objects is now permitted, but nested names in subpictures are not permitted (see section 2.10)

5. A routine for histograms is added to the graph routines (see section 3)

6. The routines for handling the link have been rationalised (see section 5)

7. Two new satellite programs for providing interactive graphics facilities without any need to write satellite code are provided, (see the interactive Handler and Terminal in section 6).

June 1970

UNIVERSITY OF CAMBRIDGE

Computer-Aided Design Group

GINO

Graphical Input/Output

THIRD EDITION

Edited by P.A. Woodsford

June 1970

The version of GINO described in this third edition of the user's manual is a result of extensions and improvements made to the version described in the second edition. These improvements are the result of the experience of over a year's usage of GINO at the University Mathematical Laboratory and at the Ministry of Technology CAD Centre, Cambridge. The extensions are mainly in the direction of accommodating a larger range of graphical devices. This edition of the manual is a revision of the second edition. One condequence of this is that 'TITAN' is used (except when obviously juxtaposed with 'ATLAS') to mean 'TITAN or 'ATLAS'. The term 'PDP' is also sometimes used to mean 'PDP or Elliott'.

Some parts of the first version of GINO, which was the work of C.A. Lang, P.J. Payne, J.C. Gray, A.P. Armit and A.R. Forrest, are still in use. The bulk of the current version is the work of P.A. Woodsford. Contributions have also been made by P. Cross and R.P. Parkins and by C. Litherland and M. Newell of the Ministry of Technology CAD Centre. The latter has also provided valuable feedback from the use of GINO at the Centre. C. A. Lang gave valuable advice about the design of GINO and A.W. Nutbourne helped with the presentation of this manual, which has been again typed most efficiently by Mrs. S. Williams.

GINO was developed by members of the Cambridge University CAD Group and is available on the TITAN computer. It is also available on the ATLAS computer at the Ministry of Technology CAD Centre, Madingley Road, Cambridge.

Computer Aided Design Group,
University Mathematical Laboratory,
Corn Exchange Street,
Cambridge CB2 3QG,
England.

CONTENTS

1. INTRODUCTION

GINO provides a set of graphical facilities for use with the TITAN and ATLAS computers. A number of graphical devices can be used including the PDP 7/9 computer with DEC340 display, the Elliott 905 computer with 928 display, the CALCOMP digital plotter and the COMPLOT on-line plotter. Programs using GINO are basically independent of the particular graphical device used. An initialisation routine is provided for each device and this is the only routine which has to be changed when changing a GINO program from using one graphical device to another.

Two or three-dimensional pictures may be generated in TITAN or ATLAS and displayed (as two-dimensional projections) on any available graphical device. Optional transformation and windowing facilities are provided for use in generating pictures. Further facilities allow for applications using interactive graphics with the PDP or Elliott as a satellite of the central computer. These facilities include the naming of picture parts and the management of communications between the two machines.

The GINO routines are designed to be usable by programmers in any of the languages available in the TITAN Mixed Language System (MLS) (1). It is anticipated that the majority of GINO users will write their programs in FORTRAN, SAL or Assembly Code. All examples of routine calls in the manual are given in the FORTRAN style, but complete sample programs in both FORTRAN and SAL are included.

The present facilities provided by GINO are:-

1. TITAN/ATLAS - Generation of Pictures

A set of easy to use routines is provided to enable pictures to be generated from lines, points, characters, arcs of circles, etc. Pictures may be sent directly to the chosen graphical device or stored on disc file, paper tape or magnetic tape.

2. TITAN/ATLAS - Generation of Graphs

Various styles of graph paper (e.g. linear, log-linear) may be specified and any number of graphs may be plotted on each "piece of paper". The graph routines are compatible with the general picture routines of (1) so pictures containing graphs and other pictorial information may be produced.

3. TITAN/ATLAS - Plotting Display Files

Pictures generated in the form of display files can be plotted on the CALCOMP plotter. This is designed to provide a general "hard copy" service for users of the display - the display files may or may not have been produced by GINO.

4. TITAN/ATLAS - Using a Satellite Computer

The PDP7 is attached to TITAN by a high speed data link, as is the PDP9 to ATLAS. Two Elliott 905 computers are also attached to ATLAS by 4800 baud lines. Any of these data channels is referred to as a "link" (2). Routines are provided to handle communications over the link; they are designed to help the FORTRAN programmer in particular.

5. PDP/ELLIOTT - Programs for Using the Display and the Link

Programs are provided for loading display file paper tapes (DFTL) and for punching out display files in a relocatable form (DFTP). The Display file Manager (DFM) may be used to run the display, particularly when display files are sent over the link. Two programs are available (the Interactive Handler and Terminal) which allow use of the PDP/Elliott as an interactive graphics satellite. These two programs may be used without writing any code for the satellite.

The current version of GINO is a revision and extension of the version of June, 1969 (20). Extensions are primarily in the direction of accommodating new graphical devices although many new facilities have been added as a result of the experience of a year's use at the Mathematical Laboratory and the Ministry of Technology CAD Centre.

An earlier version of GINO (3) was written in FORTRAN. The current version is written in the Systems Assembly Language (SAL) (4) with consequent saving in space and increase in efficiency. (A package written in SAL, equivalent to the early FORTRAN version would be less than 1/3 of the size - the new version, with very many more facilities, is less than 2/3 of the size of the old FORTRAN version). The designers of GINO have drawn on ideas used in previous graphical input/output packages (5), (6), (7), (8).

The user who is unfamiliar with the computers used by GINO will find basic hardware descriptions in Appendix 1 at the end of this manual.

The creation of pictures has traditionally been regarded as an art. Of our readers whose bent is artistic, we would crave "that willing suspension of disbelief for the moment, which constitutes poetic faith." Of the rest we would ask just that they read at least 20 pages before saying that they do not understand.

2. TITAN - GENERATION OF PICTURES

2.1 Introduction

Pictures and Picture Parts.

The routines to be described in Section 2 make up a system for generating 2D or 3D pictures for a range of graphical devices. The pictures produced appear as assemblages of lines, points and characters. However, they are generated as sequences of picture parts, and it is as such that we shall describe them.

The simplest kind of picture part is called a built-in object. Built-in objects are the basic components of pictures (e.g., lines, points, characters, arcs of circles) that are built-in to the GINO system.

Most pictures contain some kind of regularities in shape or pattern. In order to exploit such regularities, two other kinds of picture parts may be included in GINO pictures. One of these is the user-defined object. A user-defined object is defined by the user as a sequence of picture parts; once defined it has the same status as a built-in object. The other kind of picture part is the subpicture. It is also defined by the user, but has a much more specialised nature. The differences between objects and subpictures, as they affect the user, are tabulated in Fig. 2A. Subpictures are only available on devices that have a subroutining facility (i.e. the PDP & Elliott displays). They are used to shorten the length of a display file which contains repetitions of the same image by treating this image as a display subroutine. This facility is not available on the plotters and so when GINO is used for plotter output subpictures are treated like user-defined objects.

To summarise, GINO pictures consist of sequences of picture parts, where picture parts are built-in objects, user-defined objects and subpictures.

Definitions and Picture Construction.

The function of each picture building routine is to add a picture part. Precisely what this means depends on the context of the routine call. The picture part can be added to the definition of a user-defined object, to the definition of a subpicture or to the picture being constructed. Definitions are explicitly opened and closed by the user; if no definition is open, the picture construction process takes place. The three processes are illustrated in Figs. 2B, 2C and 2D and discussed fully in Sections 2.2, 2.3 and 2.4.

Coordinate Systems.

Pictures must be specified in terms of space coordinates. Space coordinates must be cartesian; they may be in 2 or 3 dimensions (they

will be denoted by upper case letters X,Y,Z). Space coordinates are abstract. They are chosen by the user to suit his picture and they are only restricted by the fact that they are held as TITAN integers (i.e., - 1048575 <X< 1048575, etc.)

Pictures are produced in terms of picture coordinates. (These will be denoted by lower case letters x, y, z.) Picture coordinates are either screen coordinates or plotter coordinates. In any case, their limits are the physical limits imposed by the output device (e.g., for the screen $0 < x < 1023$; $0 < y < 1023$). z is usually ignored, but there is an option with pictures for the display to vary the brightness of the picture with z (this is termed depth modulation).

In general a transformation is made from one coordinate system to the other. When this happens we shall speak of the total picture, expressed in space coordinates, being transformed into a view of the total picture, expressed in picture coordinates. Typical basic transformations are rotation, magnification, and projection. General transformations are built up as combinations of basic transformations.

In addition to this transformation process (described in detail in Section 2.6), the view may also be windowed (i.e., restricted to a visible 2D or 3D region specified by the user). Details of this are in Section 2.7.

Picture Part Naming.

A facility is provided for giving any picture part a name so that it can be identified by the light pen. This facility, which is only useful to those who wish to use the light pen for interactive work, is described in Section 2.8.

Multiple Pictures.

In general a GINO program can produce several "pictures". For the PDP and Elliott displays each picture is a picture segment, which is the smallest amount of picture that can be individually displayed or replaced. Any number of picture segments can be displayed simultaneously and each one can be switched on and off. On plotters each new picture corresponds to moving the plotter on to fresh paper.

This highlights another distinction between user-defined objects and subpictures. An object definition is available for any number of pictures whereas a subpicture is local to a particular picture segment and will have to be redefined for each new picture segment.

General.

The remaining sections deal with system initialisation, output facilities and routine specifications. Since most GINO programs are likely to be written in FORTRAN, sample programs and routine specifications are given in terms of FORTRAN. However, the routines may be used from SAL or assembly code and relevant details (and a sample SAL program) are also given.

COMPARISON OF USER-DEFINED OBJECTS AND SUBPICTURES

USER-DEFINED OBJECTS	SUBPICTURES
1. Must be used if transformed views are required.	1. Frozen in shape when defined and cannot subsequently be transformed.
2. Full windowing facilities available.	2. Not clipped when windowing.
3. Definition takes more buffer space than a subpicture definition.	3. Definition takes less buffer space than an object definition. Also the display file produced is shortened.
4. May be used in any number of picture segments.	4. Local to a particular picture segment.
5. Available on all devices.	5. Only effective on PDP & Elliott displays.

2.2 Built-in Objects

Built-in objects are provided by single GINO routines and include such basic picture parts as points, lines, character strings and arcs of circles. They are used both in the definition of user-defined objects and subpictures, and as picture parts when constructing pictures.

Points and lines can be defined in two or three dimensions. (2D pictures can be considered as being set up in the plane z = 0.) A point is defined by its space coordinates. A line may either be defined as a vector increment or by its end point.

GINO pictures consist of ordered sequences of picture parts. At any point in the sequence there is a "current position" in the picture. In physical terms this corresponds to the position of the electron beam on the display screen or the pen on the plotter paper. In constructing a picture it is often necessary to change the current position before adding the next picture part. Points and lines can be added invisibly for this purpose. (The corresponding routine names are derived by prefixing the letter I to the visible routine name.)

Examples of Built-in Object Calls.

CALL IPOINT(I,J)	Current position set to X = I, Y = J.
CALL LINE(10,-20)	Vector increment X = 10, Y = -20 is drawn from current position.

CALL CHARS('PLAN VIEW')	Adds character string at current position.
CALL LINEP3(100,-200,100)	Line from current position to X = 100, Y = -200, Z = 100 is drawn.

Using the Light Pen.

Built-in object calls may be named so that they can be identified by the light pen. This facility is only useful to those interested in using the display interactively.

Each routine has an optional last argument for specifying the name associated with the built-in object call. Thus,

```
CALL LINE3(100,0,-100,81)

CALL LINE(20,60,82)
```

names the specified lines as 81 and 82, respectively. A pen hit anywhere on these lines will be identified with the appropriate name. (Further details of picture part naming are given in Section 2.8.)

Further Details.

The following routines are relevant to this section:

Name	Purpose
BCDCHARS	To add characters held in binary coded decimal form.
CHARS	To add a character string in FORTRAN
CHARINT	To display an integer in character form.
CHARFPT CHARREAL	To display a floating point number in character form.
CHARTYPE	To select the size of characters.
CIRCLE	To add an arc of a circle
CIRCLE3	3D equivalent of CIRCLE
CONTROL	To control light pen sensitivity, hardware scaling and intensity.
CREATEDISPLAY	To provide formatted output facilities in FORTRAN

LINE,ILINE	To add a 2D line, specified by vector increments, visibly or invisibly.
LINEP,ILINEP	To add a 2D line, specified by its end point, visibly or invisibly.
LINE3,ILINE3	3D equivalents of LINE,ILINE.
LINEP3,ILINEP	3D equivalents of LINEP,ILINEP.
POINT,IPOINT	To add a 2D point, visibly or invisibly.
POINT3,IPOINT3	3D equivalents of POINT,IPOINT.
.SALCHARS	To add a character string in SAL.

Details of these routines are to be found in Section 2.13.

2.3 User-defined Objects

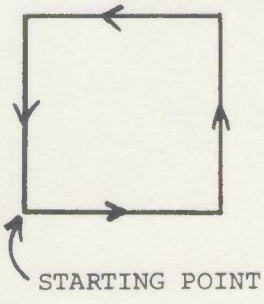
The purpose of user-defined objects is to allow the user to define for himself picture parts more complicated than those provided as built-in objects. Each user-defined object, once defined, can be called as many times as it is needed in the construction of a picture. A user-defined object is given an identifier when it is defined and this identifier is used to refer to the user-defined object when it is called The identifier is a TITAN integer chosen by the user.

The Definition Process.

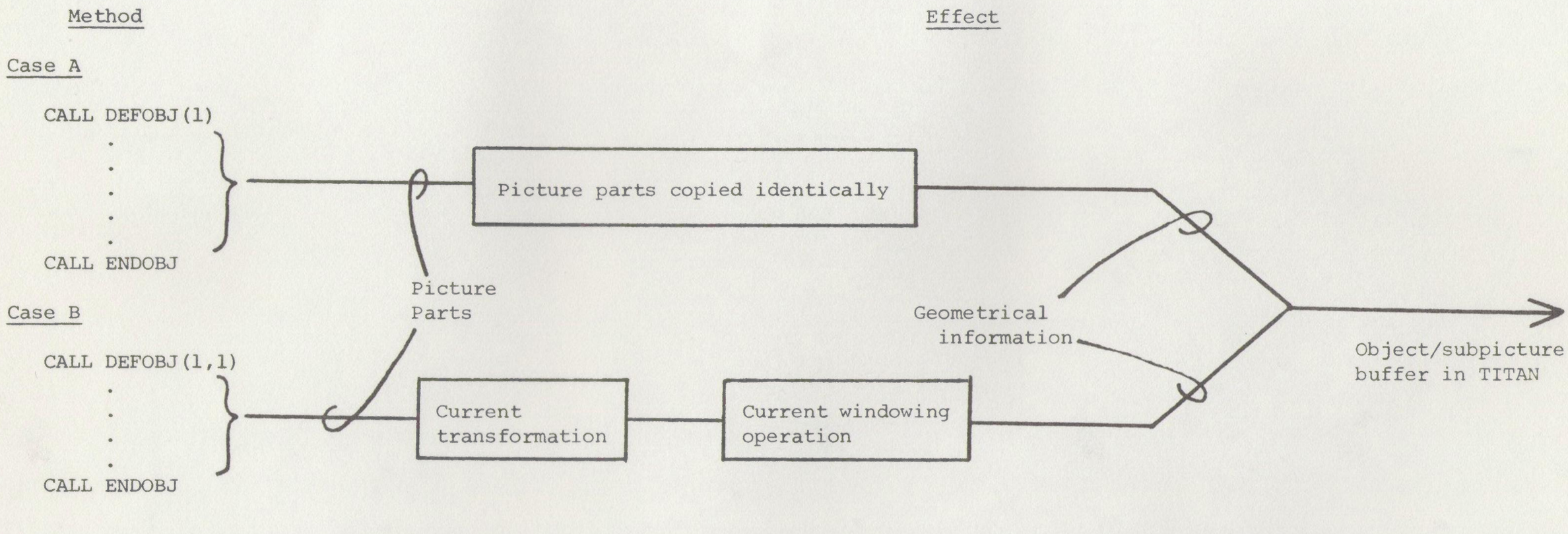
The definition process is illustrated in Fig. 2B. It is started by a call to routine DEFOBJ and closed by a call to routine ENDOBJ. Picture parts added between these two calls are added to the definition of the object and not to the picture being constructed. The object is given its identifier by the argument of DEFOBJ.

A concrete example will help the beginner. Suppose we need as a picture part a square of side 50 space units. This is not provided as a built-in object, so we must define it as a user-defined object. We decide to give it the identifier 10 and proceed as follows. The sketch illustrates the geometry.

```
CALL DEFOBJ(10)
CALL LINE(50,0)
CALL LINE(0,50)
CALL LINE(-50,0)
CALL LINE(0,-50)
CALL ENDOBJ
```



DEFINING A USER-DEFINED OBJECT



Notes:
1. An object/subpicture buffer must be created before any user-defined objects are constructed.
2. In Case A, object definitions are independent of the transformation & windowing environment current at definition time. The two cases are distinguished by the absence/presence of a second argument of DEFOBJ.
3. Subpictures used in object definitions retain their peculiar status and are never transformed or windowed.
4. Object definitions are global and may be saved on disc file.

FIG. 2B

We can now add the square to the picture how and when we need it. In fact we can use it in several different pictures or even file it away and use it another day.

There is no restriction on the picture parts which can be used in an object definition. They may be built-in objects, other user-defined objects or subpictures provided, of course, that the picture part is defined when it is called (since subpicture definitions are local to a picture segment, the case of subpictures within an object definition, whilst legal, is likely to be more trouble than it is worth).

There is only one restriction on the definition process. Definitions may not be nested. A new object definition cannot be started before ending a previous one. Neither can we start a subpicture definition inside an object definition and vice versa.

User-defined Objects in Pictures.

Having defined object 10 we can now use it in the construction of a picture by calling routine OBJECT. Thus, to obtain a square at X = 500, Y = 500 space units:

```
CALL IPOINT(500,500)
CALL OBJECT(10)
```

is all that is required.

The particular merit of user-defined objects is that they are transformed and windowed each time they are called in the picture construction process (see Figs. 2A and 2D). The definition process is independent of any transformation or windowing bounds that have been set up, and is in terms of geometrical information only.

So our humble square object is not so humble. By calling it with suitable transformations set up we can get any square (by MAGNIFY and ROTATE) and even parallelograms (by SHEAR). Further, if we position it near a window boundary it will be appropriately clipped:

The reader familiar with the term will see that user-defined objects are succinctly described as picture macros, i.e., every time an object is added to the picture, code is inserted into the display file to generate the desired picture part.

The Use of Transformations and Windowing in Object Definitions.

We have so far dealt with case A in Fig. 2B in which the object definition is an identical copy of its component picture parts. There is an alternative facility in which the component picture parts of an object definition may be subjected to transformation and windowing (case B). The reader unfamiliar with these terms should just note this option until he has read Sections 2.6 and 2.7.

If DEFOBJ is given a second argument (the value of which is immaterial) then the component picture parts of the object definition are passed through the current transformation and windowing routines and the results are entered in the object/subpicture buffer as the object definition. In other words the object definition process is exactly the same as the picture construction process shown in Fig. 2D.

This facility allows for considerable flexibility. It allows the use of transformations in building up an object definition (Example 3 in Section 2.6 is a case of this). It may also be used to reverse the built-in order of transformation and then windowing by using windowing with transformations switched off at definition time and then transformation (and perhaps windowing) at call time.

When using this facility the user must take care to explicitly set the transformation and windowing environment he requires prior to defining the object.

Light Pen Naming.

User-defined object calls can be named for the light pen in the same way as built-in object calls. The name is an optional second argument,so if we add a square to the picture by

```
CALL OBJECT(10,I)
```

a pen hit anywhere on this particular square will be associated with the name given by the value of I.

Names may also be given at definition time to the component picture parts of a user-defined object. This leads to the nesting of names. Details are given in Section 2.8.

Further Details.

Facilities are available for deleting a user-defined object (thus freeing the space used), saving an object definition on backing store and recovering an object definition from backing store. Note that if an object identifier is reused the previous definition is automatically deleted.

The following routines are relevant to user-defined objects:

Name	Purpose
BUFFERS	To set up a buffer for object definitions.
DEFOBJ	To open an object definition.
DELETEOBJ	To delete a user-defined object.
ENDOBJ	To close an object definition.
GETOBJ	To recover an object definition from backing store.
OBJECT	To call a user-defined object.
SAVEOBJ	To save an object definition on backing store.

Details are to be found in Section 2.13.

2.4 Subpictures

The purpose of subpictures is to allow users of the PDP and Elliott displays to define picture parts like user-defined objects, but of a more specialised nature. The differences between user-defined objects and subpictures have been tabulated already in Fig. 2A. The limitations and advantages of subpictures are such that they will only be useful in highly schematic contexts such as circuit layouts.

These differences hinge on the fact that only a single copy of each subpicture exists in the display file, repeated subpicture calls causing the same display file code to be executed. On the other hand, each object call has its own display file.

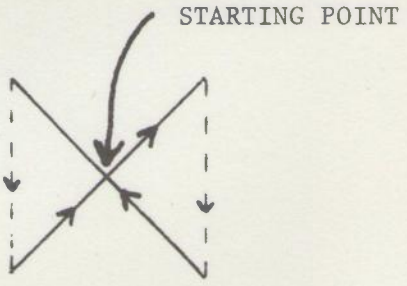
The mechanics of using subpictures is the same as for using objects. Each subpicture must be given an identifier and must be defined before it can be called.

The Definition Process.

The definition process is illustrated in Fig. 2C. To the programmer it mirrors the process of defining an object, the corresponding routines being DEFSUB and ENDSUB. Between a call to DEFSUB and a call to ENDSUB, picture parts are added to the subpicture definition and not to the picture being constructed. The restriction on nesting of subpicture definitions is precisely the same as for object definitions and so a subpicture definition cannot be started while another is unfinished.

Again we take an example. Suppose we wish to use a subpicture for a special symbol (a cross) to be used to mark data points on a graph. We have decided to give this subpicture the identifier 7 and so we proceed as follows. In the sketch illustrating the geometry, full lines are visible and broken lines are invisible.

```
CALL DEFSUB(7)

CALL LINE(5,5)

CALL ILINE(0,-10)

CALL LINE(-10,10)

CALL ILINE(0,-10)

CALL LINE(5,5)

CALL ENDSUB
```



Using Subpictures.

Having defined subpicture 7 we can now add it to the picture where and when we need it by calling routine SUBPIC. If we are using our cross subpicture to mark data points on a graph made up of straight line segments we might proceed as follows (the arrays IX, IY hold coordinates for the N data points):

```
      DO  10  J = 1,N

      CALL LINEP(IX(J),IY(J))

10    CALL SUBPIC(7)
```

Note that since LINEP draws a line from the current position to (IX(J), IY(J)), the current position must be correctly fixed before entering the DO loop.

DEFINING A SUBPICTURE

Start - CALL DEFSUB
Finish - CALL ENDSUB
Delete - by CALL BUFFERS which deletes all Subpictures

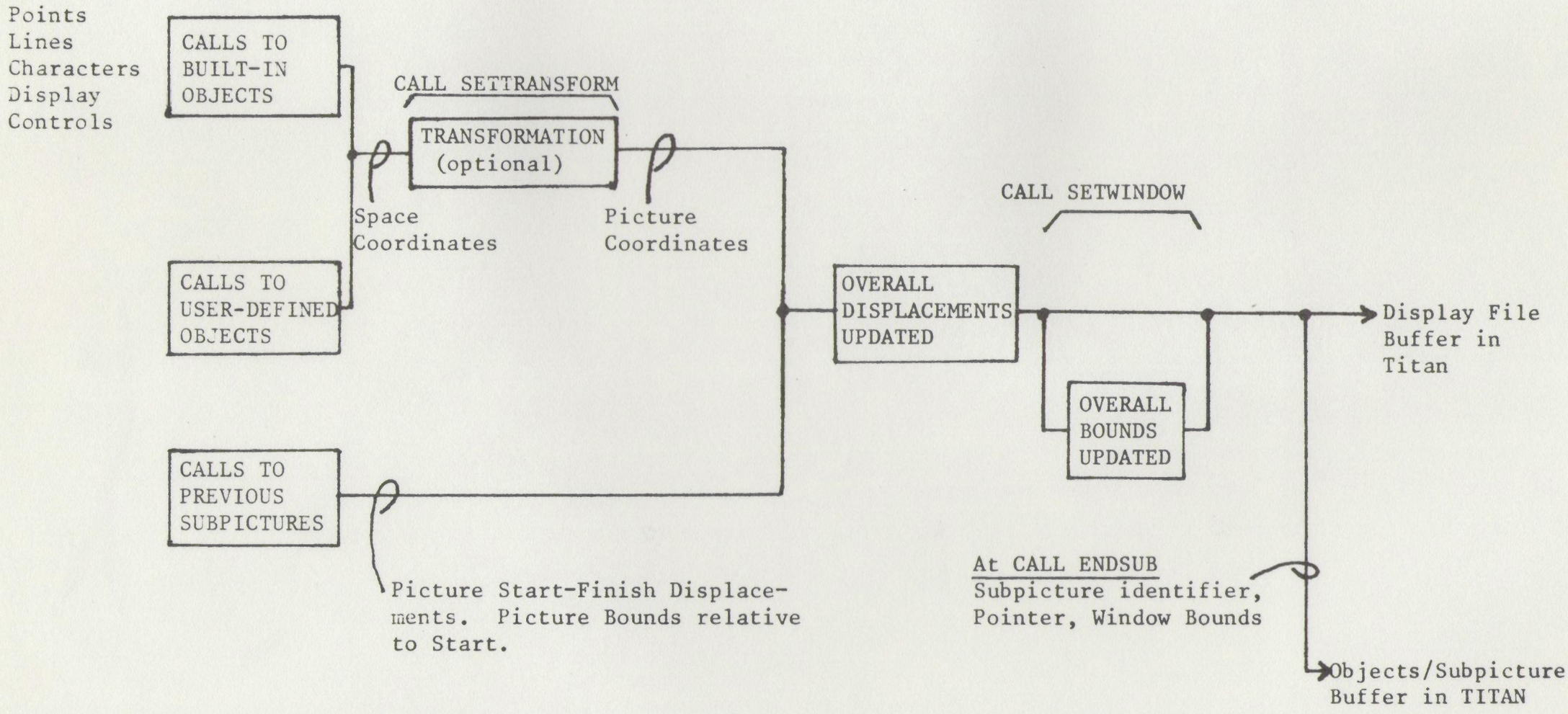


FIG. 2C

Notes: 1. The Object/Subpicture Buffer must be created before defining a Subpicture.

2. Caution: Don't use POINTS or Hardware Scale changes within the definition of a Subpicture without careful reference to the windowing procedures.

Subpictures are not as powerful as user-defined objects. They are not transformed each time they are called (see Fig. 2D) and so are frozen in shape when they are defined. Furthermore, they are not clipped when windowing. Either the whole subpicture is displayed or, if any part of it is outside the window, then it is omitted entirely. The advantages of using subpictures are that the display file is shortened and that less TITAN core space is needed. (Subpicture definitions use much less space than object definitions.) Subpictures are chiefly used for special symbols like crosses on a graph or electrical circuit symbols.

Subpictures are analogous to ordinary program subroutines without arguments.

Like user-defined objects, subpictures may be made up of any defined picture parts. It is usual, however, only to use incremental picture parts like lines and character strings in subpicture definitions. This is because a subpicture containing an absolute point will always appear in the same position, no matter what was the current position when the subpicture was called. This effect is generally unwelcome although it can be useful, eg when making character strings stand out brighter than the rest of the picture by displaying them several times over.

Light Pen Naming.

Subpicture calls, like all other picture parts, can be given a name for purposes of identification by the light pen. An optional second argument of SUBPIC is used to specify this name. A special PDP program (the Display File Manager) must be used in running display files containing named subpicture calls (this is dealt with in Section 2.8).

Further Facilities.

There is no facility for deleting an individual subpicture definition. A call to routine BUFFERS deletes all current subpicture definitions.

The following routines are relevant to subpictures:

Name	Purpose
BUFFERS	To set up a buffer for subpicture definitions.
DEFSUB	To open a subpicture definition.
ENDSUB	To close a subpicture definition.
SUBPIC	To call a subpicture.

Details are to be found in Section 2.13.

2.5 Initialisation

Initialisation of the GINO system is in two stages. First is the basic initialisation of the system to produce output for a particular graphical device. This will normally only be done once. Second is the initialisation of each new picture. When using a display, a new picture involves starting a new display file. When using a plotter it involves moving to fresh paper. Since a typical GINO program produces several pictures, this initialisation may be repeated several times.

Basic Initialisation

There is one basic initialisation routine for each available graphical device. One of these must be called before any other GINO routine. This causes GINO to be loaded with the code generating routines for the chosen device and no other. Only one graphical device can be used by a given program and a program is not allowed to contain more than one of the basis initialisation routines. However, to change from one device to another only the initialisation routine has to be changed.

The current set of basis initialisation routines is

Name	Graphical Device	Availability	DF Buffer needed
PDPGINO	DEC 340 display with PDP7 or PDP9 computer	TITAN & ATLAS	yes
ELLIOTTGINO	Elliott 928 display with 905 computer	ATLAS	yes
PLOTTERGINO	Calcomp digital plotter (2 sizes)	TITAN & ATLAS	no
COMPLOTGINO	Complot on-line plotter	ATLAS	no

New Picture Initialisation

The user program must supply buffer space to GINO. Each buffer is specified by 3 arguments - the start address, the length and an overflow error label. For the PDP and Elliott displays, the picture is built in a display file buffer, which must be supplied. Output for other devices (e.g. PLOTTERGINO, COMPLOTGINO) is sent to an output stream and so a display file buffer is not required and if one is declared it is ignored.

The second buffer which may be required is the object/subpicture buffer which must be provided if either user-defined objects or subpictures are used.

The routine used to set up the required buffer space is called BUFFERS and it must be called before any picture part routine is called. With PDPGINO and ELLIOTTGINO it may be called with 3 arguments to set up the display file buffer or with 6 arguments to set up both buffers. With PLOTTERGINO and COMPLOTGINO a call with 3 arguments sets up the object/subpicture buffer and a call with no arguments is used if user-defined objects and subpictures are not required. A call with 6 arguments is not rejected but the space declared for the display file buffer is wasted.

In addition to establishing buffer space, a call to BUFFERS also starts a new picture. Once buffer space has been established, BUFFERS may be called with no arguments to start a new picture, reusing the same buffer space. User-defined object definitions will still be available since they are global but all subpicture definitions will be lost since they are local to a particular picture. Object definitions are only lost if the basic initialisation call (e.g., ELLIOTTGINO, PLOTTERGINO) is repeated or if the location of the object/subpicture buffer is changed.

Use of COMMON.

The user does not need to declare any COMMON variables for the GINO system. Internally the system uses named COMMON blocks to hold the transformation matrix and the current position. The user may declare the necessary COMMON blocks if he wishes to have access to this information (the details are in the next section).

FORTRAN users can economise on the space taken by their programs by using arrays in blank COMMON for their buffers. This reuses space used by the loader which is otherwise wasted, e.g.,

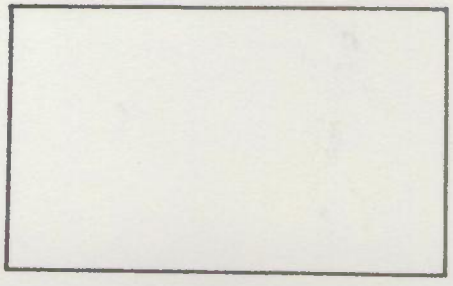
```
COMMON BUFF(400)
CALL PDPGINO
ASSIGN 99 TO L
CALL BUFFERS(BUFF(1),200,L,BUFF(201),200,L)
```

The first three arguments of BUFFERS specify the display file buffer (giving start address, length and overflow lable). The second three specify the object/subpicture buffer in similar fashion. Details of how much buffer space the user should allow, etc., are to be found in the routine specifications in Section 2.13.

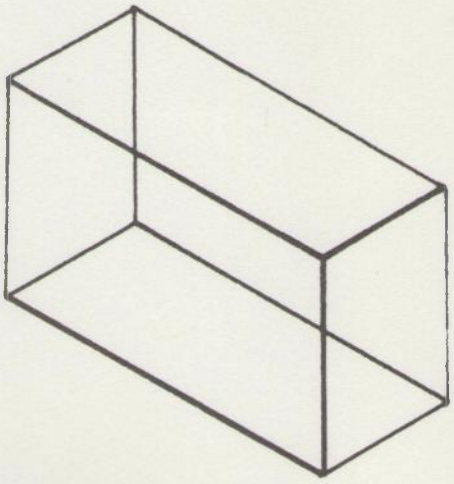
2.6 Transformations

The ability to transform picture parts during the picture construction process is essential for 3D work and very useful for 2D work. Transformations include scaling, translation, rotation and point projection in any combination. They are an optional feature in GINO.

In 3D work the basic reason for using transformations is to allow us to produce different views (projections) of total pictures. For instance having defined a simple rectangular box we might wish to view it as an isometric projection:



viewed as



Routines are provided for this and for any other projection. These routines all work by projecting on to a picture plane passing through the origin of space coordinates (they have to make some assumption about the total picture being viewed - the assumption made, that the centre of interest is at the origin, seems to be the most reasonable).

The view produced will always have to be related to the coordinate system of the display or plotter being used. All the devices currently available with GINO have their origin of coordinates at the bottom left hand corner and only permit positive coordinate values. So a second transformation is necessary to position the view on the screen or paper. It may also be necessary to scale the view to fit the available area.

We have described the use of the transformation facility for two distinct purposes - to project the total picture so as to produce the required view and then to position (and perhaps scale) the view to the appropriate position on the screen or paper. There are many other uses. For instance we may regard the screen as a window on to a large total picture and use the windowing facility (described in Section 2.7) to remove all parts of the picture outside the window. Then we would use transformations to position the appropriate part of the total picture under the visible window. Another use is in connection with the facility to transform a user-defined object while it is being defined (see Section 2.3). This enables us to use transformations in building up an object. This can be very useful for example in an object which has axial symmetry.

How Transformations Work

When GINO is set in transform mode (which can be switched on and off as required) the space coordinates in the total picture are multiplied by the current transformation matrix to produce the corresponding picture coordinates. This is shown in the diagram of the picture construction process, FIG 2D.

Mathematically we have

$$\underset{\sim}{r} = M\underset{\sim}{R} \qquad (*)$$

where $\underset{\sim}{r}$ = (x,y,z) is a position vector in picture coordinates

$\underset{\sim}{R}$ = (X,Y,Z) is the corresponding position vector in space coordinates

M = current transformation matrix (TR matrix)

(In fact, homogeneous coordinates and (4×4) transformation matrices are used, so as to include projective transformations. The theory of this is given in (9).)

Note that M in equation (*) above is the matrix current when the picture part is called in the picture construction process. The required transformation must therefore be set up before constructing the picture. The x - y plane is taken as the picture plane and z is ignored unless depth modulation (as described in the next section) is used.

The current TR matrix is built up by calls to basic transformation routines so that it represents the cumulative effect of the sequence of transformations. Thus in the simplest case, that of producing an isometric or perspective projection, two routines will be used to set up the TR matrix. The first (ISOMETRIC or FROMXYZ) sets up the projection. The second (OSHIFT) positions the resulting view. Conceptually it is easiest to think of these as separate operations - in fact they are achieved by a single matrix multiplication of the form (*) above.

In general the TR matrix will be built up from some sequence of translations, rotations, magnifications and point projections. The effect achieved is as if each picture part were subjected to this sequence of transformations. Of course the order of the sequence is often crucial, particularly in 3D work. For instance a translation followed by a rotation will in general produce a different result than a rotation followed by a translation.

Transformations do not affect the position of the space coordinate axes, which remain fixed. They only affect position,orientation and scale relative to these axes. Suppose we set up a TR matrix as follows:

```
CALL SETTRANSFORM          - initialise to unit matrix.
CALL OSHIFT(-500.,0.)      - translate through DX = -500, DY = 0
CALL ROTATE(2,-120.)       - rotate -120° about Y axis.
```

CONSTRUCTION OF A PICTURE

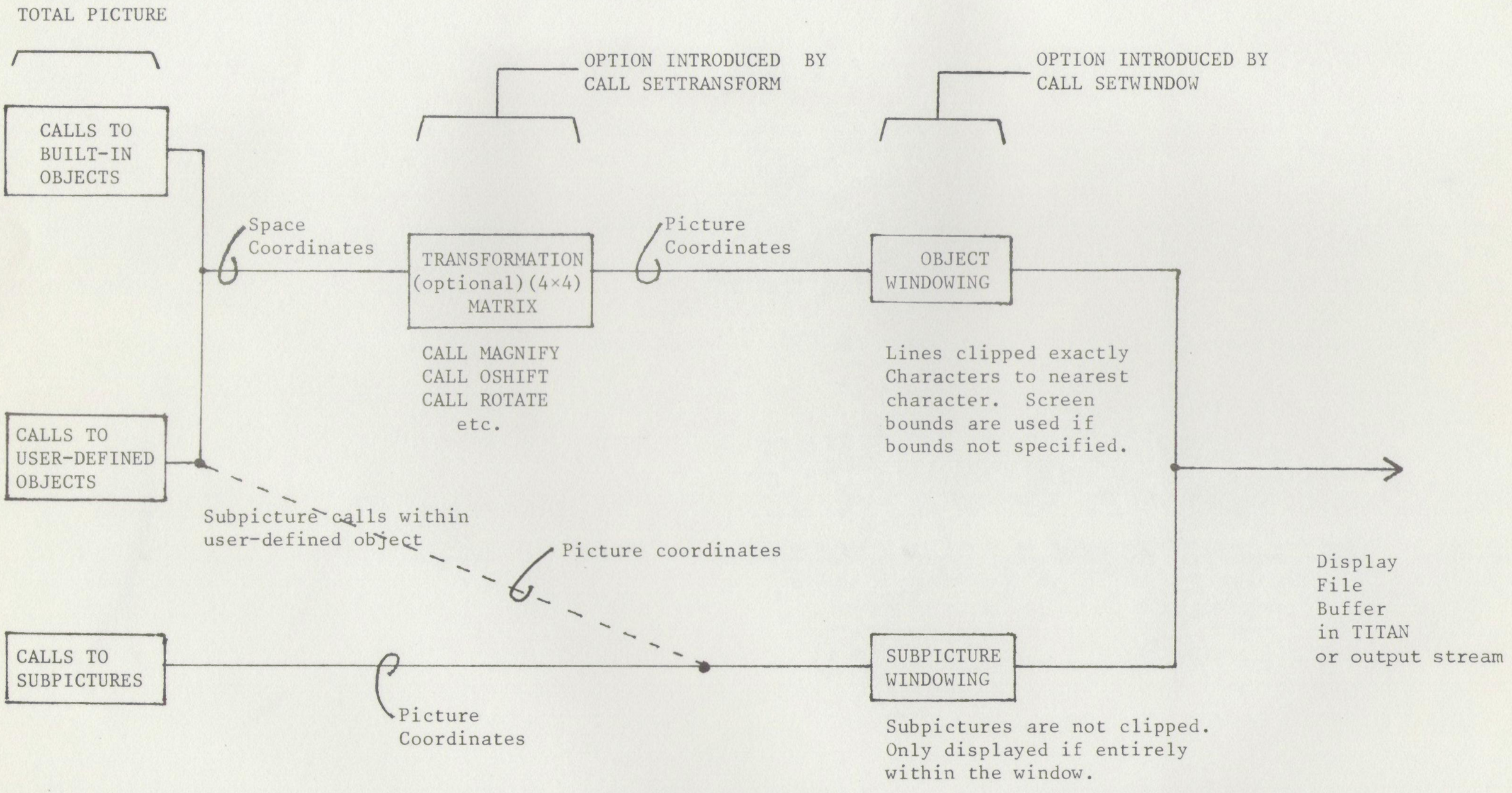
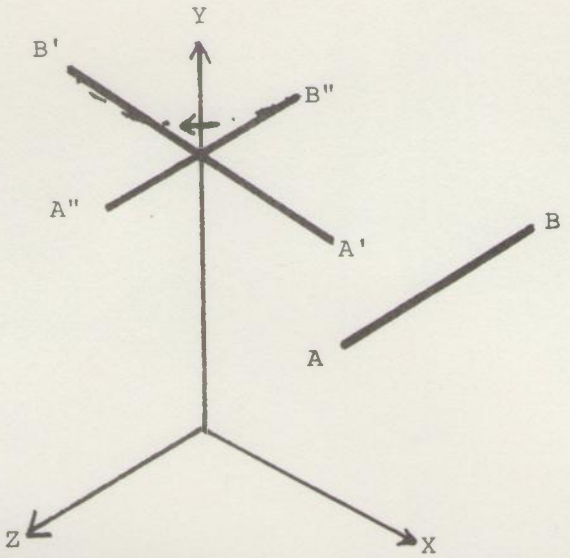


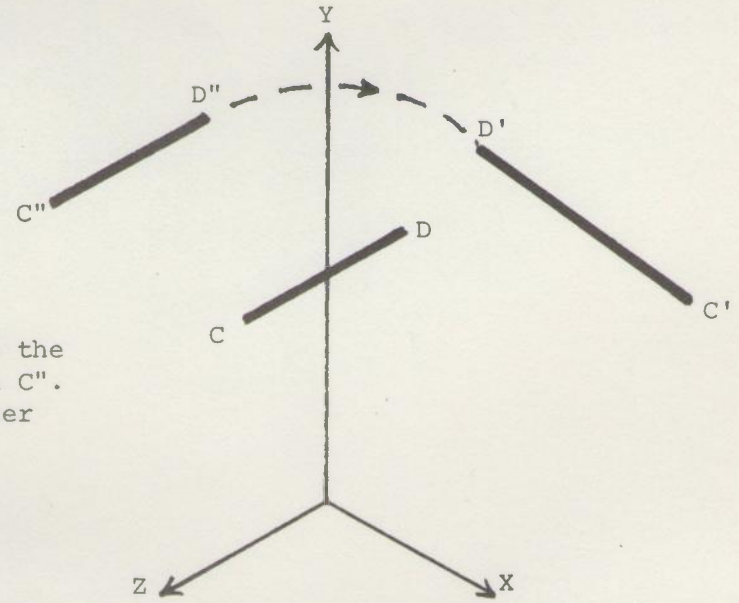
FIG. 2D

Notes:
1. If no transformations set, space coordinates must be numerically equal to picture coordinates.
2. The case of subpicture calls with user-defined objects, represented by the dotted line, is unlikely to occur often.

If we then ask for a line AB joining (400,400) to (600,600) we shall get A'B' (A" B" represents position of line if shifted only).



If we ask for CD joining (-100,400) to (100,600) we will get C'D'. Note that the rotation is still about the fixed Y axis.



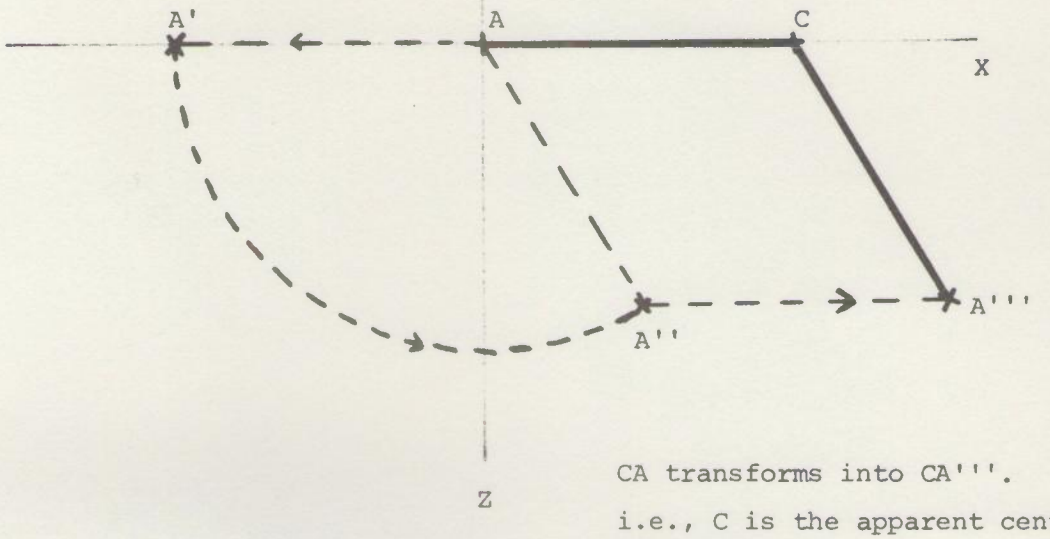
N.B. D" is closer to the axis of rotation than C". Hence D' is also closer than C'.

If we had followed the rotation by another translation through DX = 500, DY = 0, we would have the effect of a rotation not about the Y axis but about a parallel axis through X = 500.

```
CALL SETTRANS
CALL OSHIFT(-500.,0.)     - shift the point (X=500,Y=0) to Y axis
CALL ROTATE(2,-120.)      - rotate about Y axis
CALL OSHIFT(500.,0.)      - undo previous shift
```

A is (0,0,0) before transformation
C is (500,0,0) before transformation

X - Z plane



CA transforms into CA'''.

i.e., C is the apparent centre of rotation.

This is the standard technique for effectively changing the coordinate system.

How to Use Transformations

We emphasise again that in order to use transformations GINO must be set into transform mode by call routine SETTRANSFORM and the appropriate TR matrix must be built up by calls to transformation routines before the picture is constructed. The transformation applied to a given picture part is the one current when that picture part is called.

Some care is necessary in the choice of space coordinates, since rotations, scalings etc. all work relative to the fixed space coordinate axes. This is particularly important in 3D work where it is best to arrange that the centre of interest of the total picture is at the origin. Note that this means that the total picture must at least start with an absolute point (e.g. IPOINT3).

If we are using transformations to produce a particular view then the transformation matrix is built up from a call to the appropriate projection routine followed by scaling and positioning routines. If the total picture does not centre on the origin then a translation before the projection may be used to position it for the projection routine. (This is the same technique as used above for rotations.)

Routines are provided to set up several standard axonometric projections. These are projections in which the projection point is at infinity. The main virtues of axonometrics are the relative ease with which they can be produced in the drawing office and the absence of any

distortion. Using GINO it is just as easy to produce full perspective views in which the projection point is a local point. These will be characterised by having vanishing points and will generally produce a more pleasing visual effect (see FIGS 2E, 2F). FIG 2G shows a simple rectangular parallelipiped undere eight different projections.

Having chosen the required projection there remains the problem of relating the view to the coordinate system of the display or plotter. The simplest and most efficient way of doing this is to follow the projection with a shift to the centre of the screen e.g.,

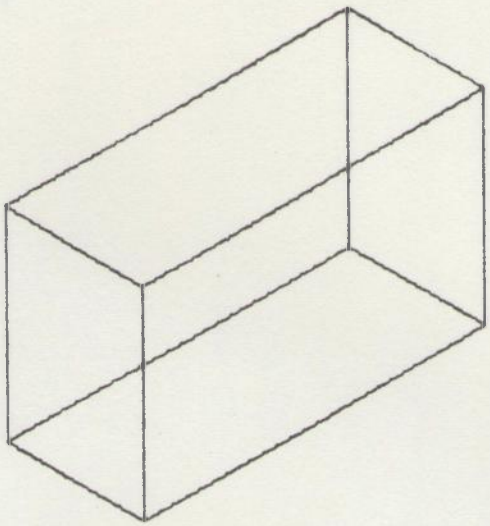
```
                CALL SETTRANSFORM
C  INITIALISE TO UNIT MATRIX
                CALL FROMXYZ(1000.,200.,1500.)
C  FULL PERSPECTIVE VIEW FROM X=1000, Y=200, Z=1500
                CALL OSHIFT(511.,511.)
C  POSITION TO CENTRE OF SCREEN
```

A problem arises if the transformed view does not fit in the available area. Two solutions are possible. One is to use the windowing facility (described in the next section) to remove everything falling outside the available area. This takes very little extra computing but may not be satisfactory if the user wants to see the whole view. The other approach copes with this. A routine (named PICTURE) is available which automatically scales and positions the transformed view to fit exactly into a specified area. It requires that the total picture be defined as a single user-defined object and does involve rather more computing than the straightforward method. Suppose the total picture consists of user-defined object 1 and an axonometric projection is required with line of sight joining X=100, Y=50, Z=80 to the origin. We are plotting and we require that the plotted view should fill the paper:-

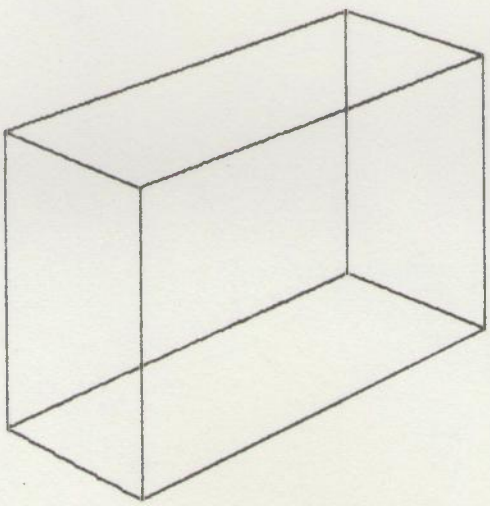
```
                CALL SETTRANSFORM
                CALL AXONXYZ(100.,50.,80.)
C  WE NOW HAVE THE REQUIRED PROJECTION.
                CALL PICTURE(1,5,1055,5,1055)
C  OBJECT 1, UNDER THE REQUIRED PROJECTION, IS MAPPED ON TO
C  5 < x < 1055, 5 < y < 1055 ON THE PLOTTER PAPER.
```

Which approach to adopt depends on circumstances. Automatic scaling and positioning is suitable for non-interactive work - when working interactively scaling and positioning is best controlled by the user.

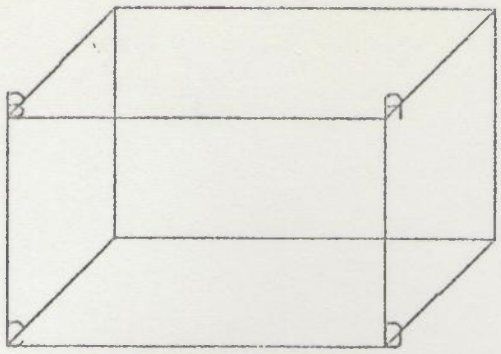
The easiest way to use transformations in building up a total picture is to use the facility to pass the component parts of a user-defined object through the current transformation (and windowing routine) while that object is being defined (see Section 2.3). Care is needed here due to the fact that the transformations have a cumulative effect, e.g. a rotation is superimposed on the existing transformation - if a rotation on its own is required, the TR matrix must be set to the unit matrix (by CALL SETTRANSFORM) before ROTATE is called. When building up an object in this way it is often convenient to save the current



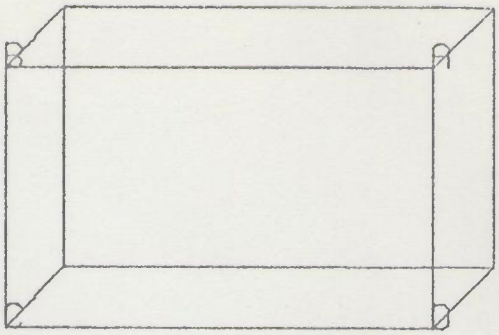
ISOMETRIC PROJECTION. FIG. 2E



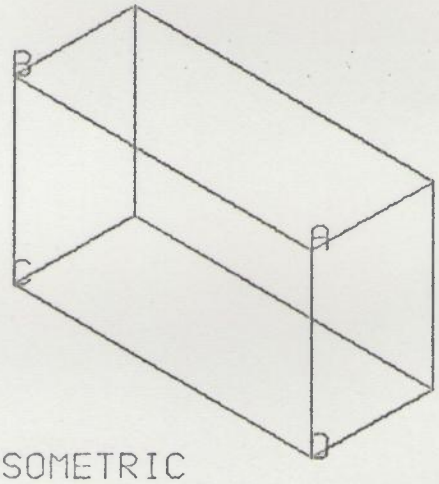
FULL PERSPECTIVE VIEW. FIG. 2F



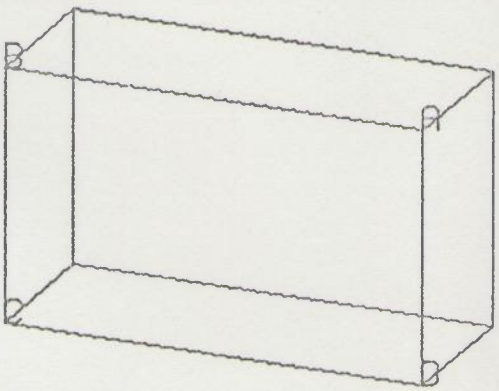
CABINET



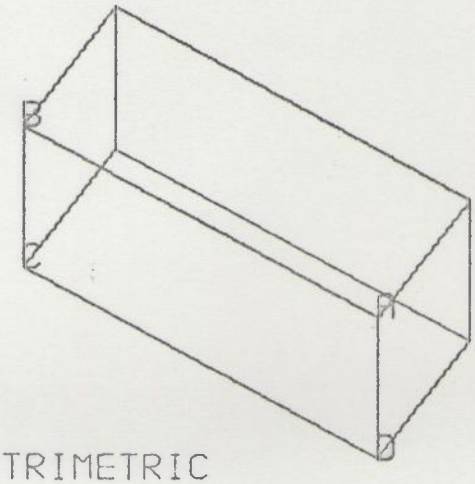
CAVALIER



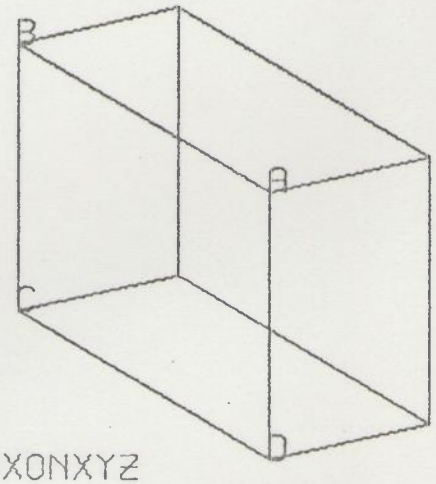
ISOMETRIC



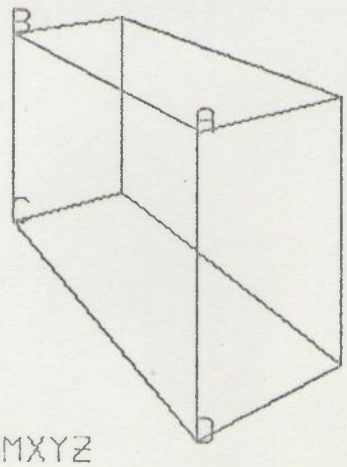
DIMETRIC



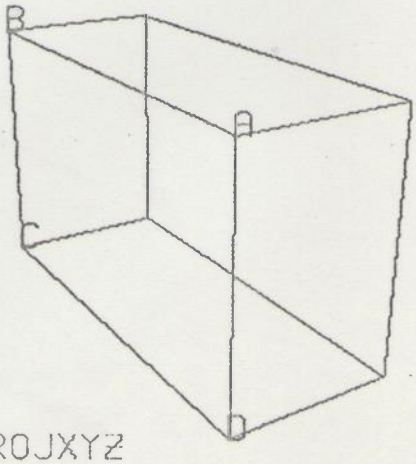
TRIMETRIC



AXONXYZ



FROMXYZ



PROJXYZ

transformation and later restore it, rather than build it up again. Routines SAVETRANSFORM and SETTRANSFORM can be used to do this. It is also useful to suspend transform mode temporarily so that picture parts are not transformed at all. Routines UNSETTRANS and RESETTRANS provide this facility.

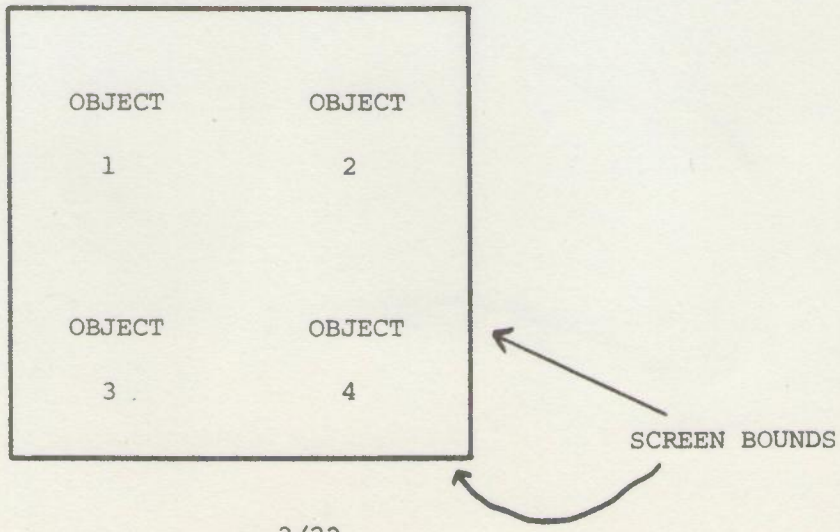
Some GINO programs will not need transformations at all. If a program does not contain calls to any transformation routines then none of the transformation machinery is loaded.

Example 1

This example shows the use of transformations in 2D to produce composite pictures. Suppose we have four 2D user-defined objects, identified as 1, 2, 3 and 4. Each would fill the screen if displayed directly. We construct a composite picture in which each object occupies one quarter of the screen area.

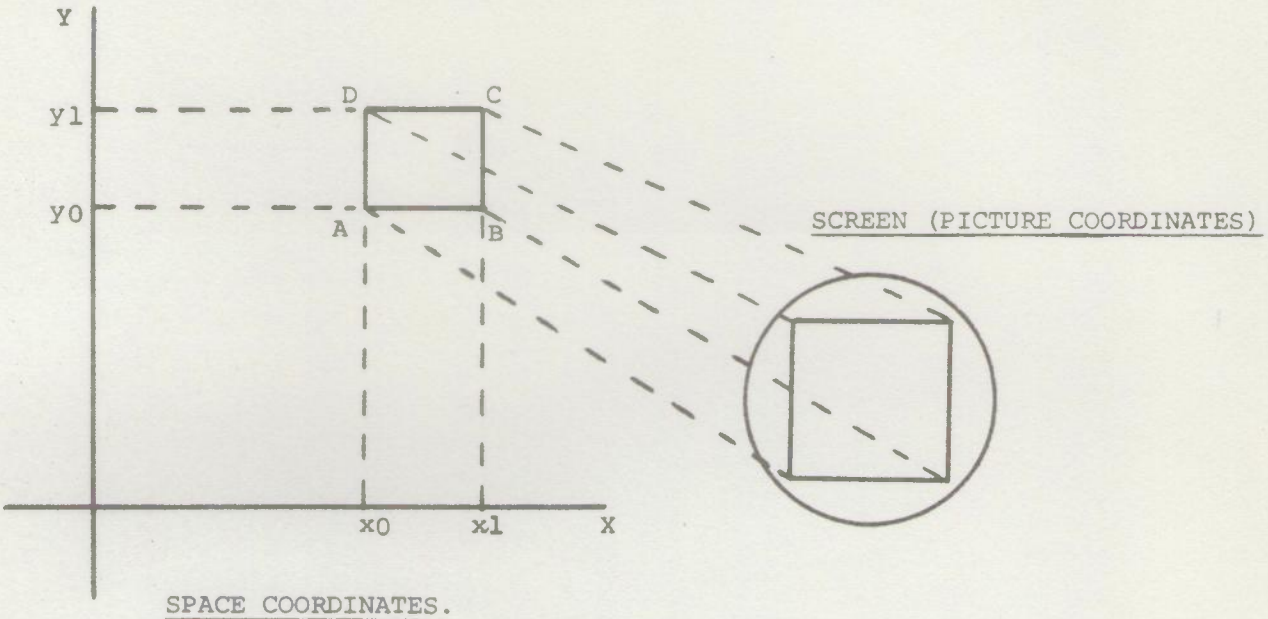
```
                CALL SETTRANSFORM
C  TRANSFORM MODE ON.  TR MATRIX = UNIT MATRIX
                CALL MAGNIFY (0.5,0.5)
C  REDUCE TO HALF SIZE
                CALL OSHIFT (0.,511.)
C  ORIGIN NOW APPEARS AT x = 0, y = 511
                CALL OBJECT(1)
                CALL OSHIFT (511.,0.)
C  ORIGIN NOW APPEARS AT x = 511, y = 511
                CALL OBJECT(2)
                CALL OSHIFT (-511.,-511.)
C  ORIGIN NOW APPEARS AT x = 0, y = 0
                CALL OBJECT(3)
                CALL OSHIFT (511.,0.)
C  ORIGIN NOW APPEARS AT x = 511, y = 0)
                CALL OBJECT(4)
                CALL DFOUTR
C  OUTPUT IN RELOCATABLE FORM FOR THE DISPLAY FILE MANAGER
```

The make-up of the resulting picture is shown in the sketch below:



Example 2

This example shows how to use transformations to map a given rectangular area on to the whole screen. Then by using the windowing facility to remove all parts of the picture that would fall outside the screen area we can examine any part of the total picture in detail.



SPACE COORDINATES.

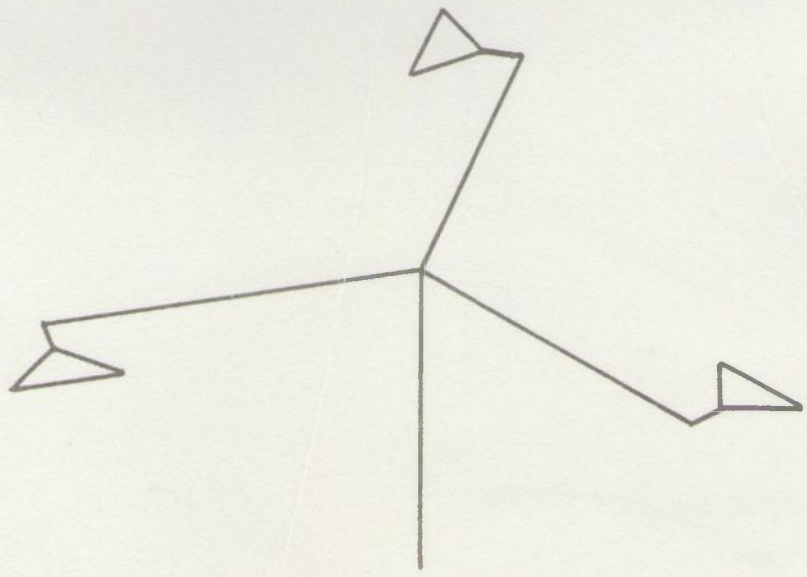
Suppose the required visible area ABCD is bounded by $x0 \leq x \leq x1$; $y0 \leq y \leq y1$ (space coordinates) and that these bounds are held in Fortran real variables x0,x1,y0,y1. Then to achieve the desired effect the following program must be executed before the picture is constructed:

```
                CALL SETTRANSFORM
C  PARTS OF THE TRANSFORMED VIEW OUTSIDE 0≤x≤1023, 0≤y≤1023 - THE SCREEN
C  SIZE - ARE TO BE REMOVED BY THE CLIPPING ROUTINES
                CALL SETWINDOW(0,1023,0,1023)
C  SHIFT SO THAT POINT A WOULD BE MOVED TO ORIGIN
                CALL OSHIFT(-x0,-y0)
C  MAGNIFY SO THAT ABCD WILL FILL SCREEN
                FX = 1023./(x1-x0)
                FY = 1023./(y1-y0)
                CALL MAGNIFY (FX,FY)
C  NOW PROCEED TO CALL PICTURE PARTS TO CONSTRUCT PICTURE.
```

Example 3

We wish to construct and view a rudimentary lawn sprinkler. The arm of the sprinkler is defined as a user-defined object. The sprinkler is defined as a second user-defined object. The component parts of this second object are transformed as the object is defined, so that the arm can be rotated into its different positions. Finally we produce an isometric projection and position it as required.

```
C  DEFINE THE SPRINKLER ARM AS OBJECT 1.  THE COMPONENT PARTS
C  OF THIS OBJECT ARE NOT TRANSFORMED. y-AXIS IS TO BE AXIS OF SPRINKLER
                CALL DEFOBJ(1)
                CALL IPOINT3(0,100,0)
                CALL LINE3(200,0,0)
                CALL LINE3(0,0,-20)
                CALL LINE3(-30,0,-30)
                CALL LINE3(60,0,0)
                CALL LINE3(-30,0,30)
                CALL ENDOBJ.
C  DEFINE WHOLE SPRINKLER AS OBJECT 2. COMPONENT PARTS ARE TO BE
C  TRANSFORMED - HENCE SECOND ARG OF DEFOBJ
                CALL DEFOBJ(2,1)
                DO 1 I=1,3
                CALL SETTRANS
                D=I*120
                CALL ROTATE(2,D)
1               CALL OBJECT(1)
C  WE NOW HAVE 3 ARMS EVENLY SPACED. THE CENTRE SHAFT DOES
C  NOT NEED TRANSFORMATION
                CALL UNSETTRANS
                CALL IPOINT3(0,100,0)
                CALL LINEP3(0,-100,0)
                CALL ENDOBJ
C  WE NOW SET UP AN ISOMETRIC PROJECTION. NOTE THAT OBJECT 2
C  IS CORRECTLY CENTRED ABOUT THE ORIGIN OF SPACE COORDINATES
C  WE HAVE FIRST TO RESET TR MATRIX TO UNIT MATRIX
                CALL SETTRANS
                CALL ISOMETRIC
C  THEN A SHIFT TO POSITION THE VIEW
                CALL OSHIFT(511.,511.)
C  FINALLY WE CALL OBJECT 2 PRODUCING THE OUTPUT SHOWN BELOW
                CALL OBJECT(2)
```



Further Details.

The reader will have noticed that most of the arguments of the transformation modifying routines are reals. This is because the TR matrix is necessarily real. The only integer arguments are those specifying axes, where 1 indicates the x-axis etc.

A user who intends using his own matrices will need to know the format of the TR matrix. This is given in the specification of SETTRANSFORM in Section 2.13. The salient point is that matrix multiplication is taken as pre-multiplication.

As mentioned in Section 2.5, transformation information is held in FORTRAN named COMMON blocks, so that it is accessible to the user. The following information is available:

(a) The TR matrix is held in a 16 element real array in a block named TRBOD.

(b) The current position in terms of space coordinates is held in three integers in a block called SPACEPOS.

(c) The current position in terms of picture coordinates is held in three integers in a block called PICTPOS.

Thus, if we declare

```
COMMON/TRBOD/T(4,4)/SPACEPOS/IX,IY,IZ/PICTPOS/JX,JY,JZ.
```

Then T contains the TR matrix, (IX,IY,IZ) is the current position in space coordinates and (JX,JY,JZ) in picture coordinates.

The following routines are relevant to transformations:

AXONXYZ	To set up an axonometric projection along the line joining (X,Y,Z) to the origin.
CABINET	To set up a cabinet projection.
CAVALIER	To set up a cavalier projection.
DIMETRIC	To set up a dimetric projection.
FROMXYZ	To set up a perspective view, with (X,Y,Z) as viewpoint and horizontal line of sight towards origin.
ISOMETRIC	To set up an isometric projection.
MAGNIFY	To superimpose a magnification.
MODTRANS	To multiply TR matrix by a user-supplied matrix.

Name	Purpose
OFIX	To fix point (0,0,0) of space coordinates to be specified point.
OSHIFT	To translate through a specified vector.
PICTURE	To call an object, mapping the resulting view (i.e. the view under the current transformation) onto a specified picture area.
PROJECT	To project from a point on a coordinate axis onto x-y plane as picture plane.
PROJXYZ	To set up a perspective view with (X,Y,Z) as viewpoint and line of sight through origin.
ROTATE	To superimpose a rotation about a coordinate axis.
SAVETRANSFORM	To save current TR matrix in a user-supplied array.
SELECTVIEW	To permute coordinate axis, thus selecting new picture plane.
SETTRANSFORM	To set transform mode and initialise TR matrix.
SHEAR	To superimpose a shear transformation.
TRIMETRIC	To set up a trimetric projection.
UNSETTRANS RESETTRANS	To unset/reset transform mode.

Details of these routines are to be found in the second half of Section 2.13.

2.7 Windowing and Depth Modulation

The range of picture coordinates is dictated by the size of the display or plotter used and is much smaller than the range of space coordinates. An attempt to draw outside the available area is termed an edge violation. The behaviour of each graphical device when an edge violation occurs is described in section 2.11. In general edge violations should be avoided and this is one of the purposes of the windowing facilities in GINO. This use was referred to in the last section (p. 2/19).

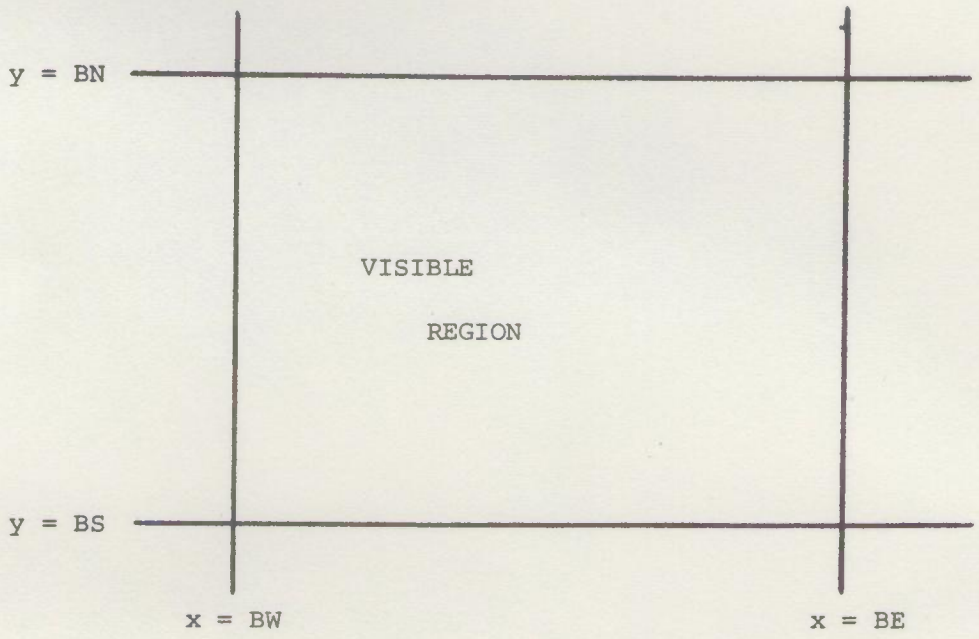
The windowing facilities can also be used to examine a particular region of the total picture. This region may be defined in 2D or in 3D. Only parts of the total picture falling in the chosen visible region will appear. One technique for doing this was illustrated on page 2/23.

Windowing Operations

Two types of windowing operations are provided in GINO - 2D windowing and 3D windowing. A program may select either or none of these. In the normal picture construction process (illustrated in Fig. 2D) the windowing operation takes place after transformation, i.e. on picture coordinates. This is the most useful order as it enables windowing to be used to avoid edge violations. When circumstances arise in which another order of operation is required, the facility to transform and window the component parts of an object definition, as already described in section 2.3, may be used. In this case the distinction between space and picture coordinates becomes blurred.

2D Windowing.

The user can specify a rectangular window in the picture plane. The picture is then clipped so that only parts within the window appear in the picture.



Picture parts added as objects (built-in or user-defined) are clipped exactly (the only exception being that character strings are clipped to the nearest character). Picture parts added as subpictures are only displayed if entirely within the window.

The user sets up the window bounds by a call to routine SETWINDOW. The presence of a call to this routine causes the system routines for 2D windowing to be loaded. (See Fig. 2D.) In the normal case the window bounds are in terms of picture coordinates. So if we just wish to avoid edge violations on the display we can set the window by

```
CALL SETWINDOW(0,1023,0,1023)
```

The user can, however, set the window bounds to any value he requires.

3D Windowing.

When 3D windowing is used, clipping is to a rectangular parallelipiped or box defined by

$$BW \leq x \leq BE;\ BS \leq y \leq BN;\ BACK \leq z \leq FRONT$$

In the usual case (when x and y are in the picture plane), z is normal to the picture plane and towards the viewer. The bounds are set and the appropriate windowing routines loaded by a call to routine SET3DWINDOW. Clipping to the 3D window is exact, so the effect of a depth cursor can be achieved by making the 3D window be a narrow band at the relevant z value. 3D windowing may be used to remove "clutter" from the foreground or background of a picture and it is also used in connection with depth modulation (see below).

3D windowing as provided in GINO should not be confused with clipping to a pyramid of view, as required when perspective transformations are used. This is essentially a 2D windowing operation and is provided by the 2D windowing option in GINO. (See Section 6 of (13)).

Depth Modulation

In all 2D representations of 3D total pictures, the third dimension presents a problem. The most useful visualisation aid provided in GINO is depth modulation. This is available on the PDP and Elliott displays. The brightness of the picture is modulated over the intensity levels of the display, with maximum intensity at the front and minimum at the back of the visible z-region. Six intensity levels are used on the PDP and five (two of them simulated) on the Elliott. Depth modulation is automatically switched on by a call to SET3DWINDOW. The user can then switch it on and off as required.

Example

The following example shows how the transformation and windowing routines together can be used to produce quite complicated effects. User-defined object 99 is an aeroplane occupying a region of space within the bounds

X -4000 (nose) to +4000 (tail)

Y -1000 (bottom) to +1000 (top) of fuselage

Z -4000 (starboard wing tip) to +4000 (port wing tip)

We wish to obtain two pictures:

a. A composite picture, the top-half showing a side elevation of the front-half of the aeroplane and the bottom-half a side elevation of its rear-half. both views to have depth modulation.

b. A view of the rear-quarter of the aeroplane. It is to be rotated about a vertical line through its centre, so that the tail swings toward the viewer. Anything then appearing behind the centre point is to be omitted.

We proceed as follows. In the comments we locate the nose (N) and tail (T).

```
          CALL SETTRANSFORM
          CALL MAGNIFY (0.25,0.25,0.25)
C  REDUCE TO QUARTER SIZE.  N AT (-1000,0,0).  T AT (+1000,0,0)
          CALL OSHIFT (1023.,767.)
C  N AT (23,767,0).  T AT (2023,767,0).
          CALL SET3DWINDOW (0,1023,512,1023,-1000,1000)
C  UPPER HALF OF SCREEN.  INTRODUCES DEPTH MODULATION ACROSS WING SPAN
          CALL OBJECT (99)
C  TOP HALF OF FIRST PICTURE.  N DISPLAYED.  T CLIPPED.
          CALL SETTRANSFORM
C  TR MATRIX RESET TO UNITY.
          CALL MAGNIFY (0.25,0.25,0.25)
          CALL OSHIFT (0.,255.)
C  N AT (-1000,255,0).  T AT (1000,255,0)
          CALL SET3DWINDOW (0,1023,0,511,-1000,1000)
          CALL OBJECT(99)
C  BOTTOM HALF OF FIRST PICTURE.  N CLIPPED.  T DISPLAYED.
          CALL DFPUNB
C  FIRST PICTURE OUTPUT ON PAPER TAPE
          .
          .
          CALL SETTRANSFORM
          CALL MAGNIFY (0.5,0.5,0.5)
C  N AT (-2000,0).  T AT (2000,0,0)
          CALL OSHIFT (-1500.,0.)
C  N AT (-3500,0,0).  T AT (500,0,0).  REAR QUARTER NOW
C  CENTRES ON ORIGIN
          CALL ROTATE (2,-30.)
C  ROTATES ABOUT Y AXIS 30° TOWARDS VIEWER
          CALL OSHIFT (511.,511.)
C  SHIFTS ROTATED VIEW TO CENTRE OF SCREEN
          CALL SET3DWINDOW (0,1023,0,1023,0,500)
C  CLIPS PART OF TAIL BEHIND AXIS OF ROTATION AND
C  POSSIBLY PART OF WING
          CALL OBJECT (99)
          CALL DFPUNB
C  OUTPUT SECOND PICTURE TO PAPER TAPE
```

Further Facilities

A problem often arises in setting the values of the z-bounds in SET3DWINDOW. If the bounds are set too far apart, the picture produced will only use part of the available range of intensities. If they are too close part of the picture that was required may be removed by the clipping. Two routines are provided to help with this. PICTZCLIP is an extension of the automatic scaling and positioning routine, PICTURE.

As well as specifying the x-y region the picture is to occupy, the visible z-range can be specified on a normalised scale in which the back of the whole object is represented by 0 (zero) and the front by 1. Thus to view the front half only values of 0.5 and 1.0 would be used.

PICTZCLIP invokes the whole machinery of 3D-windowing, which may well be redundant. In the case where depth modulation without clipping is required the routine OBJZDM should be used. This displays any view of a user-defined object, with depth modulation across the full range of z, and no clipping. Space and time are saved since SET3DWINDOW is not used.

Miscellaneous Details.

2D-windowing adds about 600 words to the length of a TITAN program. 3D-windowing and depth modulation adds about 950 words and lengthens the display file. The cost in terms of time depends on the number of boundary intersections. Even if there are no intersections, there is a slight penalty so windowing should not be used if it is not required (e.g., with the graph routines of Section 3).

There are some anomalies in the windowing of subpictures. Subpictures are not clipped. They are either displayed as a whole, or omitted. This process only works properly for subpictures containing only incremental picture parts. Subpictures containing absolute points or hardware scale changes are always displayed and may give rise to boundary violations.

Programs must not include calls to both SETWINDOW and SET3DWINDOW. 2D-windowing is included in 3D-windowing and may be obtained by setting large z-bounds (e.g., -1000000 < z < 1000000) and turning depth modulation off. If a program contains SETWINDOW or SET3DWINDOW, bounds appropriate to the size of the screen or plotter are assumed until the bounds are explicitly set.

The following routines are relevant to this section:

Name	Purpose
DEPTHON	To restore depth modulation.
DEPTHOFF	To suspend depth modulation.
OBJZDM	To provide automatic depth modulation across the full range of z.
PICTZCLIP	To provide automatic scaling, positioning, clipping and depth modulation.
SETWINDOW	To set up a 2D-window.
SET3DWINDOW	To set up a 3D-window and initiate depth modulation.

Full details are in the routine specifications in Section 2.13.

2.8 Picture Part Naming & Use of the Light-Pen

Every picture part used in the picture construction process may be given a name. The purpose of this name is to enable the picture part to be identified by the light pen. This facility is essential if the light pen is to be used as a tool in interactive graphics. It is of no interest to the user who does not intend to use the light pen in this way.

The name of a picture part is specified by giving an optional last argument in the relevant routine call, as described in Sections 2.2, 2.3 and 2.4. The use of the term "name" may cause difficulty to those who think of names as things like FRED or X or GINO. Picture part "names" are, in fact, numbers which the user may assign in order to identify picture parts. This number may be specified either by a constant, or by using a program variable, e.g.,

```
          CALL LINE (100,200,3)
C  The name of the line is 3
          CALL LINE (200,100,K)
C  The name of the line is the value of K
```

Ways of Using Picture Part Names.

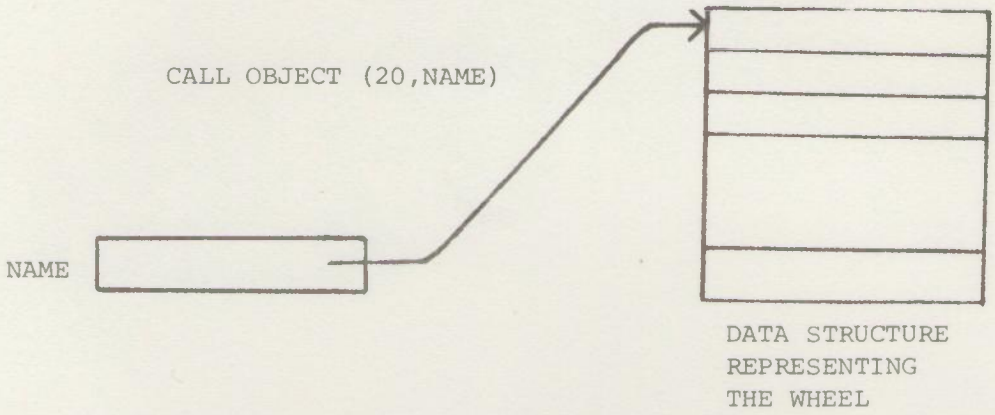
The name is used effectively to identify what a picture part represents, rather than the picture part itself. For example, when pointing at a circle which is part of a picture of a car the name may be used to identify it as "the nearside front wheel", rather than just a circle.

The name may be used in various ways. The simplest is just as a code number. For example, suppose we wish to display a sequence of squares (already defined as user-defined object 6) across the screen. The squares are to be distinguished by having names 1 to 10.

```
          DO  1  I = 0,9
          CALL IPOINT (100*I,500)
  1       CALL OBJECT (6,I+1)
```

A second way that names may be used is as pointers to a data structure. When parts of the car are pointed at with the light pen, the program may need to access information about the part pointed at. For instance, information about the wheel (tyre characteristics, materials, weight, etc.) might be held in some kind of data structure. If we make the name of the picture part representing the wheel a pointer to the data structure representing the wheel, then, when the wheel is pointed at by the light pen, we can get very fast access to the wheel information without any searching.

Suppose user-defined object 20 is a wheel. Then we might say



N.B. The value of variable NAME points to the data structure representing the wheel.

The Nesting of Picture Part Names.

Picture parts may also be given names when they are used in the construction of a user-defined object, so that the individual parts of a user-defined object can be distinguished by name.

The object may also be named when it is called. This leads to the nesting of picture part names, so that a given picture part may be distinguished as say picture part I within picture part J within picture part K. The maximum allowed depth of nesting is 16 although 4 is nearly always enough and is the maximum assumed in the standard version of the Display File Manager and the Handler (described in Section 6.3).

Subpictures are treated differently. They are regarded as atomic entities and their component picture parts cannot be distinguished by name. Names given to picture parts used in the definition of a sub-picture are ignored and a warning message given.

Light Pen Sensitivity.

One of the parameters of the display is light pen sensitivity. When the pen is turned on circuits are enabled so that the display is interrupted whenever the light pen can "see" light. The pen has a manually operated shutter and so can be used as a pointing device on the screen. When an interrupt occurs owing to the pen seeing light from a particular picture part we say that a pen hit has occurred on that picture part. When the pen is turned off, pen hits cannot occur. Pen sensitivity is controlled by the display file so it is possible to turn the pen on for some picture parts and off for others. The user can do this explicitly using routine CONTROL. Without any action on the part of the user, the GINO system turns the pen on for named picture parts and off for unnamed picture parts.

Satellite Programs for Handling Picture Part Names.

The Display File Manager (DFM) is the basic satellite program for managing display files. One of the functions of DFM is to deal with pen hits on named picture parts. When a pen hit occurs DFM passes control to the user's satellite program with the relevant name (or stack of names) in a standard position. This may result in some computation in the satellite or in the transmission of the name(s) to TITAN for computation and possible picture modification there. The facilities for transmitting information over the link are described in Section 5 and more details of DFM are in Section 6.3.

The Interactive Handler is a satellite program designed to make the facilities of an interactive graphics terminal easily accessible from FORTRAN programs running in ATLAS (or TITAN). The Handler may be used without writing any satellite code at all. It provides the following facilities for handling named picture parts and many other facilities described in Section 6.3.

(i) Automatic Bright-Up. When a named picture part is seen by the light pen the picture part is brightened so that it is easily distinguishable. This is termed a pen-see. The picture part can be reselected if a mistake is made.

(ii) Recording of Names for Pen Hits. A pen-see is confirmed as a pen-hit by pressing the space bar of the button keyboard. The stack of names and the display file segment number for the pen hit is added to a list kept in the satellite.

(iii) Transmission of List of Names to ATLAS. The current list is transmitted to ATLAS whenever a pen-hit on a picture part in segment 1 (a "light-button")is recorded or a button (not the space bar) on the keyboard is depressed. In addition to picture part names associated with pen-hits, the list contains other information - e.g., tracking cross coordinates, button numbers, etc. (see specification of the Handler and Section 6.3).

After transmission the list can be converted into a FORTRAN array and processed.

Versions of DFM and the Handler exist for both the PDP 7/9 and the Elliott 905 computers.

Further Details.

The display file segment number is always recorded with each pen hit since any modification to the picture part chosen will involve the replacement of this segment. The usual strategy in highly interactive work is to use a large number of display file segments so that the amount of display file to be replaced will be small.

The name of a picture part should not be confused with the identifier of a user-defined object or subpicture.

Picture part names are only available to the programmer. If a user wishes some name to appear on the picture beside the picture part he must program this explicitly. The Interactive Handler provides a facility akin to this. A piece of display file may be associated with a call to a user-defined object and this associated display file will only be displayed when the object is selected by the light pen. The routine for adding an associated display file is called ASSOCIATE.

Names are TITAN integers, excluding 0 (zero) which is regarded as the null name. Unless nested names are used (in which case only unique combinations should be employed) unique names should be given to each picture part.

2.9 Methods of Picture Output

The routines so far described are used to generate pictures for a variety of graphical output devices. We now consider methods of actually outputting and displaying pictures. The output devices fall into two classes, depending on whether or not they are associated with a satellite computer.

THE PDP AND ELLIOTT DISPLAYS

These two displays are associated with satellite computers. PDPGINO or ELLIOTTGINO are the appropriate initialisation routines. We need to distinguish two cases depending on whether the display file is loaded into the satellite from paper tape or sent directly over the link.

Paper Tape.

Display file paper tapes are punched out in binary form by routine DFPUNB or in character form by routine DFPUNC. Relocation (i.e., setting up the display file to be loaded at a particular satellite address) is then performed in the satellite by loading the tape with the Display File Tape Loader (DFTL). The specification of DFTL is in Section 6.2. Binary tapes are approximately one third the length of character tapes and should be used unless the user wishes to print his display file as well as display it.

The Link.

The methods of sending display files to the satellite over the link divide into two categories depending on whether the user is logged in in normal mode or expensive mode on TITAN. In normal mode the display file has to be output in binary to a disc file or pending stream and then sent over the link by use of the PDP command (this is one of the commands available on the Cambridge Multiaccess System (10)). In expensive mode link transmissions can be initiated at any time and the display file can be sent to the satellite as soon as it is constructed. In either case there are two options. The display file can be relocated in TITAN and sent to the appropriate satellite address (optionally with a small program to make it self-running). Alternatively, the display file can be transmitted in relocatable form. The Display File Manager (DFM) is used to organise relocation of such transmissions. DFM treats each link transmission as a

picture segment and provides facilities for displaying several picture segments, switching picture segments on and off, etc. (Details are in Section 6.3.)

In normal mode the first effect is achieved by using routine DFOUTB, which outputs the display file in binary, together with a small program to run the display. The second effect is obtained by using routine DFOUTR. In expensive mode both effects can be obtained by using routine SENDPDP. Note that this is the method that must be used for real time manipulation of pictures (details of this are in Section 5).

It should be noted that the PDP and Elliott links are handled in the same manner by the ATLAS supervisor. They are only differentiated at the time when the link is created (by the SIGNAL command when in expensive mode, see specification (10)).

Preserving Display Files.

Since DFOUTB, DFOUTR and DFOUTC (for relocated character output) send their output to a stream set up by the user, display files in character or binary form can be sent to disc file, paper tape or magnetic tape. They can then be displayed or plotted as and when required.

Plotting Display File.

Pictures produced as display files can be plotted on the CALCOMP plotter by outputting them to a disc file and reading them back into core and plotting them using routine DFPLOT described in Section 4. Display files which have been sent to the PDP or the Elliott to run under DFM can also be read back for plotting (see Section 6.3).

It is possible also to generate display files and plotter output in the same program by using routines PLOTPDP or PLOTELL to translate the display file produced into plotter output. To do this PDPGINO or ELLIOTTGINO will be used as initialisation routine, and two output routines will be called, one for the display file, and PLOTPDP or PLOTELL for the plotter output.

DIRECT PLOTTER OUTPUT

Output for the CALCOMP and COMPLOT plotters may be generated directly, PLOTTERGINO and COMPLOTGINO being the appropriate initialisation routines (see Section 2.5). The basic routine for outputting a picture to the plotter is PLOTEND although all the display file output routines are equivalent to PLOTEND if plotter output is being generated.

Output Options Peculiar to Plotters.

The size of the plotter paper is only limited in one direction - across the drum. The standard length along the paper per picture is 3000 units. This can be extended by cutting routine PLOTLENGTH. The standard picture orientation is with x across the page. Output with x along the page can be generated by calling routine XALONG before generating the picture. The standard orientation can be restored by calling routine XACROSS.

To aid identification each plotter picture produced by GINO is normally preceded by a stream header giving user identifier, etc. If this is a nuisance it can be supressed by calling routine NOSTRHEAD (STRHEAD restores the plotting of stream headers).

The details of these routines are included in the specification of PLOTTERGINO in Section 2.13.

Obtaining Plotter Output.

Output is sent to the CALCOMP plotter automatically when PLOTEND is called. Should control fail to reach PLOTEND (e.g. because the plot causes an edge violation) the on-line user can choose to have the output so far plotted (by typing 'CLOSE') or lost (by typing 'DELETE'). Off-line the output will be plotted as far as the edge violation. The procedure for producing output on the COMPLOT on-line plotter can be demonstrated at the Ministry of Technology CAD Centre.

Routine Details.

The following routines are relevant to this section:

Name	Purpose
DFOUTB	To output a relocated, self-running binary display file.
DFOUTC	To output a relocated display file in character form.
DFOUTR	To output a relocatable display file for use with DFM.
DFPLOT	To output a picture in the form of a display file on the plotter. (See Section 4.2.)
DFPUNB	To punch out a display file in binary form for DFTL.
DFPUNC	To punch out a display file in character form for DFTL.
DFREAD	To read a display file into core in TITAN.
DFTERMIN	To terminate a display file in the manner stipulated by the user.
PLOTPDP PLOTELL	To produce a PDP or Elliott display file together with corresponding plotter output in the same program.
PLOTEND	To end a plotter picture when using PLOTTERGINO or COMPLOTGINO.
PLOTLENGTH	To set the paper allocation when plotting.

Name	Purpose
NOSTRHEAD STRHEAD	To suspend or restore the plotting of stream headers.
SENDPDP SENDSAT	To send a display file over the link in expensive mode.
XALONG XACROSS	To control the orientation of output on the plotter paper.

Full details are in Section 2.13.

2.10 SAL Entries to the Routines

Special entry points for SAL and assembly code programmers are provided in many of the routines. Use of these rather than the FORTRAN entry points will make for more efficient code. The GINO routines will not disturb B20 to B64 but may change any other B-line.

Arguments.

Arguments are handed over in B89, B88, B87, etc., and the link is stored in B90. The following conventions apply about optional arguments. The optional last argument in the picture part routines, which specifies the name for light pen purposes, is normally not picked up. A special routine called .NAMING is provided so that if light pen names are required, they can be used. A call to .NAMING causes the routines to pick up the name argument. Every picture part routine then picks up the optional argument, until another call to .NAMING cancels the arrangement.

Routine BUFFERS may be called with only 3 arguments by setting B86 < 0 . All other routines that can have an optional number of arguments are only available via the MLS calling sequence. Also routines that have real arguments can only be called using the MLS calling sequence.

Routine Names.

For routine entries in which arguments are held in B-lines, the name of the entry points is obtained by prefixing '.' to the FORTRAN routine name. In SAL such names must be declared as global labels. In ETAL they will be name parameters. For routines using the MLS calling sequence, or which have no arguments, the only entry point is the FORTRAN routine name.

The only exception to this rule is the routine equivalent to CHARS. This is called .SALCHARS (it has to be a different routine owing to the different methods of packing characters in FORTRAN and SAL).

Examples.

In the following examples parallel FORTRAN and SAL routine calls are given. A full SAL program may be found in Section 2.12.

CALL LINE(10,-20)	B89 = 10; B88 = -20; ENTER .LINE (ENTER .NAMING has not been executed.)
CALL OBJECT(77,I)	B89 = 77; B88 = I; ENTER .OBJECT (ENTER .NAMING has been executed.)
CALL INITGINO	ENTER INITGINO or MLS INITGINO
CALL CHARS('FRED')	B89 = PTR ::FRED::; ENTER .SALCHARS
CALL OSHIFT(511.,511.)	MLS OSHIFT(511.,511.)

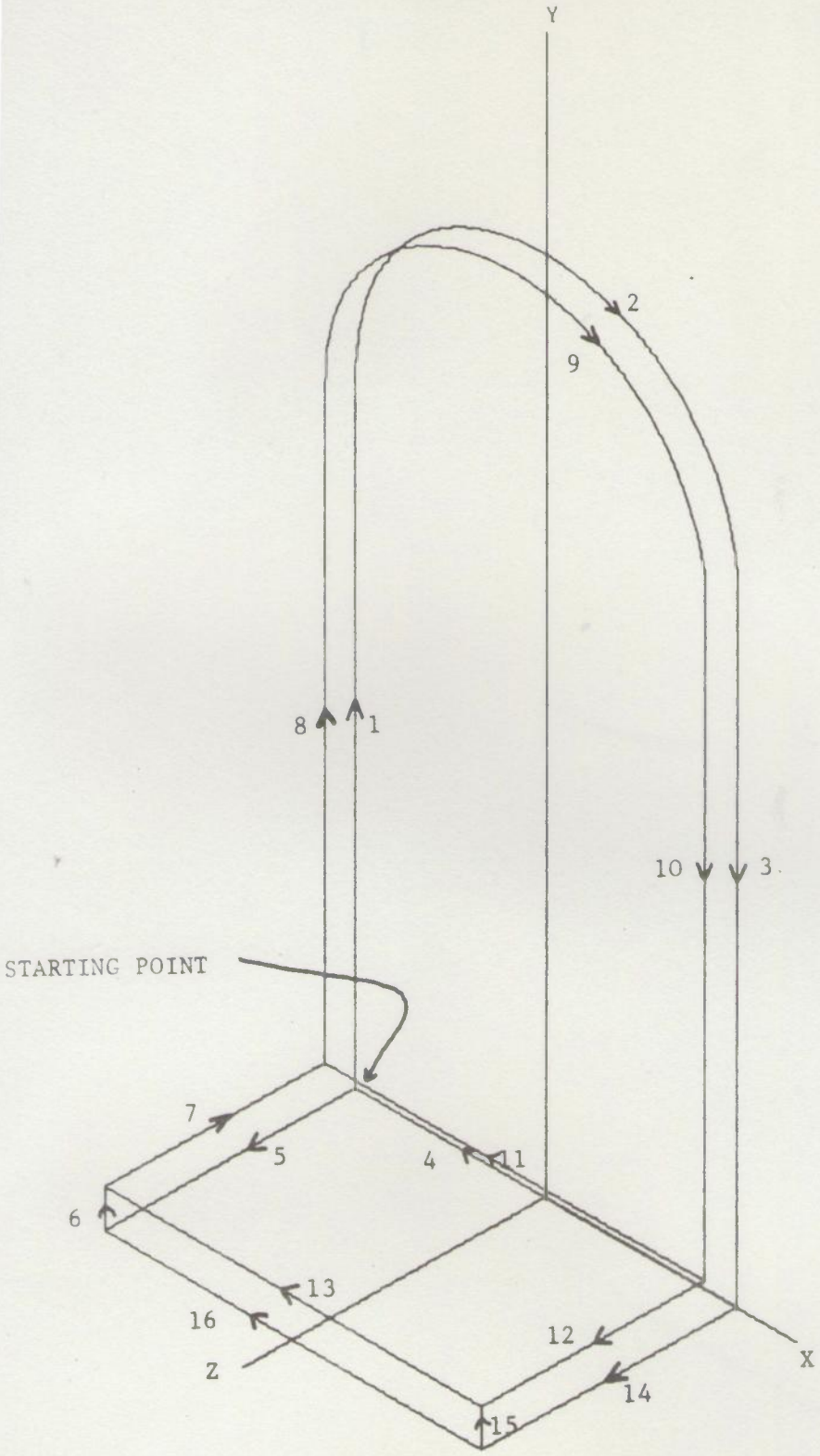
CHARACTERISTIC	PDP (DEC 340 display)	ELLIOTT (928 display)	CALCOMP PLOTTER
Raster size	1024 × 1024 (i.e. $0 \leq x \leq 1023$) Viewing area is 9 3/8" by 9 3/8"	1024 × 1024 (i.e. $0 \leq x \leq 1023$) Viewing area is 10" by 10"	1 unit = 0.01 inches Width = 1060 (small), 2920 (large) Length can be set by program (PLOTLENGTH)
Edge violation behaviour	Display stops and interrupts the PDP processor. (DFM deals with this)	Within an area of 4096 × 4096 the beam is blanked if it falls outside the viewing area.	Program monitors. Output may be deleted or plotted as far as the edge violation.
Available intensities	0-7. Six of these (2-7) are used for intensity modulation.	0-2. Two more are simulated for intensity modulation.	No variation of intensity. (Colour of ink may be changed, sparingly.)
Characters	4 sizes provided by hardware:- 6 × 7, 12 × 14, 24 × 28, 48 × 56.	2 sizes provided by hardware:- 8 × 10, 16 × 20.	4 sizes provided by software:- 6 × 7, 12 × 14, 24 × 28, 48 × 56.
GINO initialisation	CALL PDPGINO	CALL ELLIOTTGINO	CALL PLOTTERGINO
Buffer space requirement.	Display file buffer must be provided; object/subpicture only if required.		No display file buffer. Object buffer only if required.
Picture starting routine.	BUFFERS	BUFFERS	BUFFERS
Picture ending routine.	DFOUTB, DFOUTR, DFPUNB, SENDSAT		PLOTEND (DFOUTR, etc. have same effect)
Subpictures	Provided	Provided	Treated as objects, automatically.

CHARACTERISTIC	PDP (DEC 340 display)	ELLIOTT (928 display)	CALCOMP PLOTTER
Availability (June 1970)	CUML, MINTECH	CUEL, MINTECH	CUML, MINTECH

NOTES: 1/ Display files can be loaded into the PDP or the Elliott from paper tape (using DFTL) or over the link (using DFM or the Interactive Handler).

2/ Output is sent to the CALCOMP automatically. An option exists to plot x along the paper instead of across. Each plot is preceded by an identifying stream header (which may be suppressed).

3/ The COMPLOT on-line plotter is treated by GINO in a very similar manner to the CALCOMP. The initialisation routine is COMPLOTGINO. The standard width of paper is 1016 units (1 unit = 0.01 inches) and the length allowance is 3499 units. The COMPLOT is currently available only at MINTECH.



Visible picture parts are numbered in the order in which they are added.

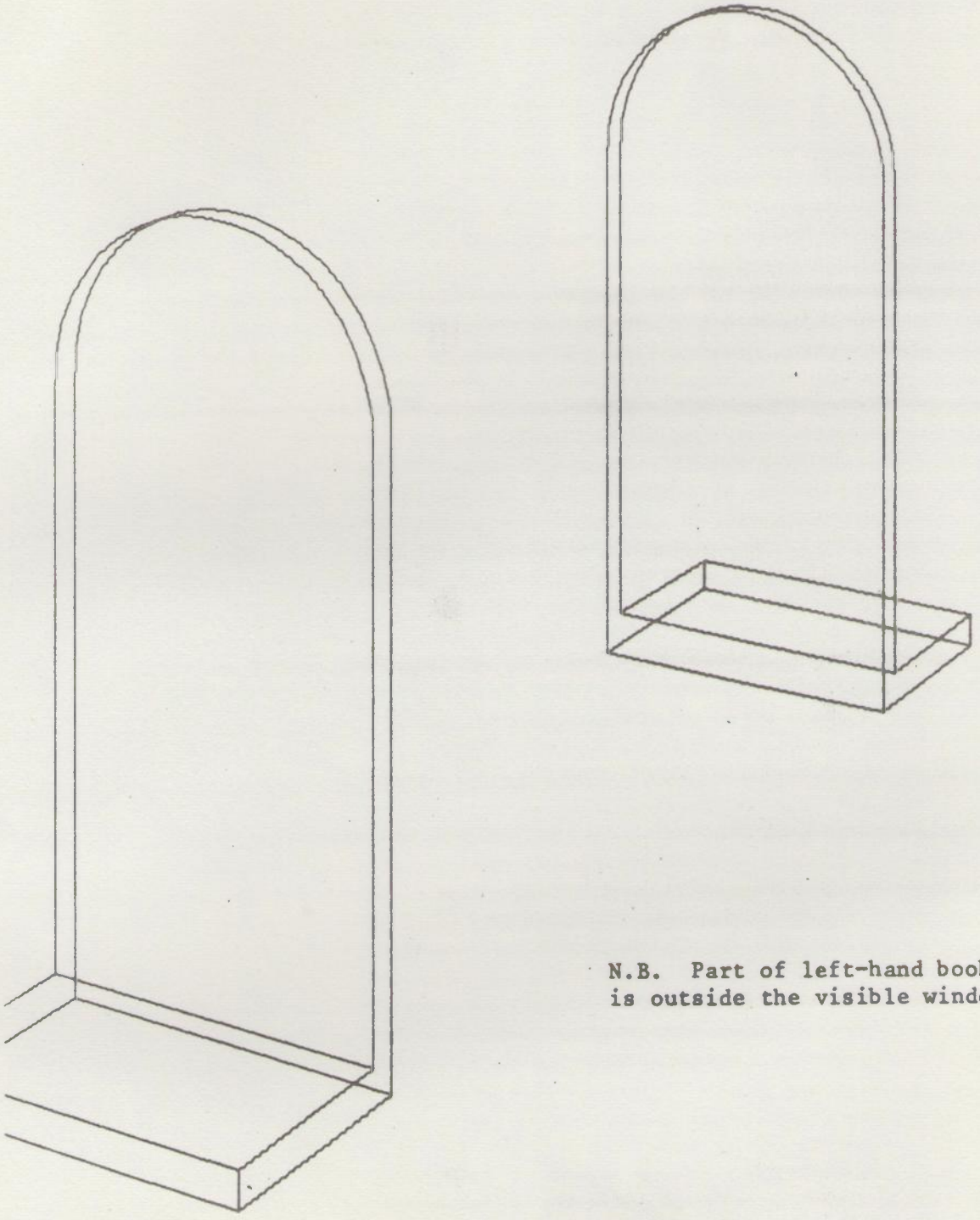
BOOKEND - ISOMETRIC PROJECTION.

FIG. 2H

2.12 SAMPLE PROGRAMS.

```
C FORTRAN PROGRAM TO PRODUCE DISPLAY FILES FOR
C PICTURES OF BOOKENDS. OUTPUT IS TO BE FILED ON
C THE DISC SO THAT PICTURES CAN BE DISPLAYED LATER.
C OUTPUT IS TO STREAM 6 WHICH SHOULD BE SET UP TO FILE.

      COMMON DBUFF(300),OBUFF(300)
C INITIALISE GINO SYSTEM. OUTPUT IS TO BE PDP DISPLAY FILE.
      CALL PDPGINO
C SET UP DISPLAY FILE BUFFER AND OBJECT/SUBPICTURE BUFFER. IF EITHER
C BUFFER OVERFLOWS CONTROL IS TO PASS TO LABEL 10 AT END OF PROGRAM.
      ASSIGN 10 TO L
      CALL BUFFERS(DBUFF,300,L,OBUFF,300,L)
C WE DEFINE A BOOKEND AS USER-DEFINED OBJECT 1. THIS IS LOCATED
C AT ORIGIN OF SPACE COORDINATES AND IS PICTURED IN FIG 2H OPPOSITE.
      CALL DEFOBJ(1)
C MOVE TO STARTING POINT IN FIG OPPOSITE. WE FOLLOW ARROWS.
      CALL IPOINT3(-150,0,0)
      CALL LINEP3(-150,500,0)
C ARGS 1,2,3 SPECIFY CIRCLE CENTRE. ARGS 4,5,6 SPECIFY END
C POINT OF ARC. IN GENERAL ANY POINT ON END RADIUS VECTOR SUFFICES.
C ARGS 7,8,9 SPECIFY DIRECTION OF INITIAL TANGENT VECTOR.
      CALL CIRCLE3(0,500,0,150,500,0,0,100,0)
      CALL LINEP3(150,0,0)
C NEXT LINE RETURNS TO STARTING POINT.
      CALL LINEP3(-150,0,0)
C WE CAN SPECIFY LINES AS INCREMENTS AS WELL AS BY ENDPOINT.
      CALL LINE3(0,0,200)
      CALL LINE3(0,30,0)
      CALL LINE3(0,0,-175)
      CALL LINE3(0,470,0)
      CALL CIRCLE3(0,500,25,150,500,25,0,100,0)
      CALL LINE3(0,-470,0)
      CALL LINE3(-300,0,0)
      CALL ILINE3(300,0,0)
      CALL LINE3(0,0,175)
      CALL LINE3(-300,0,0)
      CALL IPOINT3(150,0,0)
      CALL LINE3(0,0,200)
      CALL LINE3(0,30,0)
      CALL ILINE3(0,-30,0)
      CALL LINE3(-300,0,0)
C CLOSE DEFINITION OF BOOKEND.
      CALL ENDOBJ
C DEFINE X-Y-Z AXES AS USER-DEFINED OBJECT 2.
      CALL DEFOBJ(2)
      CALL IPOINT3(0,0,0)
      CALL LINE3(200,0,0)
      CALL CHARS(' X')
      CALL IPOINT3(0,0,0)
      CALL LINE3(0,800,0)
      CALL CHARS(' Y')
      CALL IPOINT3(0,0,0)
      CALL LINE3(0,0,240)
      CALL CHARS(' Z')
C CLOSE DEFINITION OF AXES. AS YET NOTHING HAS BEEN
C ADDED TO DISPLAY FILE BUFFER ('PICTURE').
      CALL ENDOBJ
C SET TRANSFORM MODE.
      CALL SETTRANSFORM
C SET UP ISOMETRIC PROJECTION.
      CALL ISOMETRIC
```

```
C POSITION ON THE SCREEN.
      CALL OSHIFT(511.,300.)
C FIRST PICTURE IS TO BE ISOMETRIC PROJECTION OF BOOKEND
C DRAWN WITH SPACE COORDINATE AXES.
      CALL OBJECT(1)
      CALL OBJECT(2)
C OUTPUT FIRST PICTURE ON STREAM 6 TO FILE,IN RELOCATABLE FORM.
C THIS PICTURE,PLUS ANNOTATION,IS SHOWN IN FIG 2H OPPOSITE.
      CALL DFOUTR
C NEXT PICTURE,SO CLEAR OUT DISPLAY FILE BUFFER. SINCE PDPGINO
C IS NOT CALLED AGAIN OBJECT DEFINITIONS WILL STILL BE AVAILABLE.
      CALL BUFFERS
C THE AXES ARE NO LONGER REQUIRED. WE DEFINE A NEW OBJECT 2
C CONSISTING OF TWO BOOKENDS, REDUCED TO 7/10 SIZE AND
C POSITIONED AS ON A SHELF.
C DEFOBJ IS CALLED WITH 2 ARGS AS THE COMPONENT PARTS OF OBJECT
C 2 ARE TO BE TRANSFORMED AND WINDOWED.
      CALL DEFOBJ(2,1)
C RESET TR MATRIX TO UNITY.
      CALL SETTRANSFORM
C SET WINDOW BOUNDS EXPLICITLY. AT THIS STAGE WE REQUIRE NO
C CLIPPING. SET3DWINDOW IS NECCESSARY AS IT IS USED LATER.
      CALL SET3DWINDOW(-10000,10000,-10000,10000,-10000,10000)
C ONE BOOKEND IS TO BE AS DEFINED BUT REDUCED TO 7/10 SIZE AND
C POSITIONED AT X=0,Y=0,Z=300.
      CALL MAGNIFY(0.7,0.7,0.7)
      CALL OSHIFT(0.,0.,300.)
C ADD FIRST BOOKEND TO DEFINITION OF OBJECT 2.
      CALL OBJECT(1)
C RESET TR MATRIX.
      CALL SETTRANSFORM
C SECOND BOOKEND IS TO BE SIMILARLY SCALED AND VIEWED,
C BUT AT 'OTHER END'  , AT X=0,Y=0,Z=-300.
C NEGATIVE SCALE FACTOR CAUSES REFLECTION.
      CALL MAGNIFY(0.7,0.7,-0.7)
      CALL OSHIFT(0.,0.,-300.)
C ADD SECOND BOOKEND.
      CALL OBJECT(1)
C CLOSE DEFINITION OF OBJECT 2.
      CALL ENDOBJ
C WE NOW GENERATE SECOND PICTURE - A PERSPECTIVE VIEW OF THE BOOKENDS
C WITH VIEWPOINT AT X=1200, Y=1000,Z=2000 AND INTENSITY MODULATION.
      CALL SETTRANSFORM
C SET UP PERSPECTIVE VIEW.
      CALL FROMXYZ(1200.,1000.,2000.)
C POSITION ON SCREEN.
      CALL OSHIFT(511.,300.)
C SET 3D-WINDOW BOUNDS.
      CALL SET3DWINDOW(200,800,0,1023,-450,450)
C CALL OBJECT 2 TO GENERATE PICTURE.
      CALL OBJECT(2)
C OUTPUT SECOND PICTURE - SEE FIG 2J.
      CALL DFOUTR

C THIS IS OVERFLOW EXIT.
10    STOP
      END

C DETAILS OF HOW TO DISPLAY THE PICTURES ON THE PDP ARE IN
C SECTION 6.3. FIGS 2H AND 2J WERE PRODUCED ON THE PLOTTER SIMPLY BY
C SUBSTITUTING PLOTTERGINO FOR PDPGINO AT LINE 8.
```



N.B. Part of left-hand bookend is outside the visible window.

TWO BOOKENDS - OUTPUT FROM SAMPLE FORTRAN PROGRAM

FIG. 2J

```
|SAL PROGRAM TO DISPLAY THE PRIMITIVE TWISTED CUBIC.
|NOTE THAT B-LINE ARGUMENT ENTRIES COMMENCE WITH '.'
|ROUTINES NEEDING REAL ARGUMENTS MUST USE THE MLS
|CONSTRUCTION. ENTER USES B90 AS LINK.


GLOBAL LABEL GO,INITGINO,.BUFFERS,.CONTROL,.IPOINT,.LINEP,→
.SALCHARS,.SET3DWINDOW,.IPOINT3,.LINEP3,.ILINEP,SETTRANSFORM
INTEGER DBUF(200)
BLINE 20 T |B-LINES 20-64 ARE UNTOUCHED BY GINO.


|ENTRY POINT. INITIALISE GINO AND DF BUFFER.
GO,     ENTER INITGINO
        B89=PTR DBUF(0);B88=200;B87=DERR;B86=-1;ENTER .BUFFERS
|NOW SET SCALE 1,INTENSITY 3 FOR AXES.
        B89=0;B88=1;B87=3;ENTER .CONTROL
        B89=0;B88=0;ENTER .IPOINT
        B89=1023;B88=0;ENTER .LINEP
        B89=511;B88=0;ENTER .ILINEP
        B89=511;B88=1023;ENTER .LINEP
|SCALE UNCHANGED,INTENSITY 7 FOR TITLE.
        B89=0;B88=-1;B87=7;ENTER .CONTROL
        B89=350;B88=900;ENTER .IPOINT
        B89=PTR::THE PRIMITIVE TWISTED CUBIC.::
        ENTER .SALCHARS
|RESTORE SCALE 0
        B89=0;B88=0;B87=-1;ENTER .CONTROL
|SET UP WINDOW AND TRANSFORMATION FOR TWISTED CUBIC.
        B89=0;B88=1023;B87=0;B86=1023;B85=-600;B84=600
        ENTER .SET3DWINDOW
        MLS SETTRANSFORM        |COULD HAVE ARG. SO MUST BE MLS.
|MAGNIFY, SO THAT CUBIC FILLS SCREEN.
        MLS MAGNIFY(0.5,10.0,50.0)
        MLS OSHIFT(511.)        |SHIFT TO THE DISPLAYED AXES.

|THE CUBIC IS (T*T*T,T*T,T) FOR T IN [-10,10]
        B89=-1000;B88=100;B87=-10;ENTER .IPOINT3
        FOR T=-9 STEP 1 UNTIL 10 DO
                B87=T;B88=T*B87;B89=T*B88
                ENTER .LINEP3
        REPEAT

DERR,   MLS DFPUNB
|BINARY PAPER TAPE OUTPUT,FOR USE WITH DFTL.
|COULD HAVE ARGUMENT,SO MUST BE MLS.

        STOP

|SEE A.R.FORREST'S THESIS FOR DERIVATION OF GENERAL TWISTED
|CUBIC FROM THIS PRIMITIVE FORM.

FINISH
```

2.13 Specifications of Picture Routines

Routine specifications are given in terms of FORTRAN. The type of each routine argument is indicated by the first letter of the argument name, following the standard FORTRAN convention (I - N indicates integers, others indicate reals). Where optional arguments are allowed, several examples of the routine call are usually given or the optional arguments are indicated by '?'.

Each routine specification is on a separate sheet, so that it may be rapidly updated. The practice of giving routine lengths has been discontinued since the lengths usually depend on which device (e.g. PDP, Elliott, plotter) is being used. Those lengths which remain should be regarded as very loose upper bounds. As an approximate guide, enough GINO to do points, lines and characters and to output pictures will take about 1 1/2 K of core. User-defined objects and the full range of transformation facilities use about 1/2 K each and 3D-windowing with intensity modulation takes about 1K. A full selection of facilities will not take more than 4K (excluding buffer space).

All routines needed by other routines are loaded automatically. Each routine specification contains details of any error messages that may be provoked. In the case of serious faults the GINO system stops the user's job by forcing a fault 49. For FORTRAN jobs this causes line number information to be given, which, together with the GINO error message, should pinpoint the mistake. For SAL and assembly code jobs, the fault causes an entry to System Monitor. The user may, of course, trap fault 49 or set up his own post mortem routine.

The routines for handling transformations are grouped together and given after the general picture routines.

This section contains specifications of the following routines:

ASSOCIATE
BCDCHARS
BUFFERS
CHARINT, CHARFPT, CHARREAL
CHARS
CHARTYPE
CIRCLE
CIRCLE3
COMPLOTGINO
CONTROL
CREATEDISPLAY
DEFOBJ
DEFSUB
DELETEOBJ
DEPTHOFF, DEPTHON
DFCONTIN
DFOUTB
DFOUTC

DFOUTR
DFPUNB
DFPUNC
DFREAD
DFTERMIN
ENDOBJ
ENDSUB
GETOBJ
INITGINO, PDPGINO, ELLIOTTGINO
LINE, ILINE
LINE3, ILINE3
LINEP, ILINEP
LINEP3, ILINEP3
OBJECT
OBJZDM
PICTURE, PICTZCLIP
PLOTTERGINO
PLOTPDP, PLOTELL
POINT, IPOINT
POINT3, IPOINT3
.SALCHARS
SAVEOBJ
SENDSAT, SENDPDP
SETWINDOW
SET3DWINDOW
SUBPIC

CABINET, CAVALIER
FROMXYZ, AXONXYZ, PROJXYZ
ISOMETRIC, DIMETRIC, TRIMETRIC
MAGNIFY
MODTRANS
OFIX, OSHIFT
PROJECT
ROTATE
SELECTVIEW
SETTRANSFORM, SAVETRANSFORM
SHEAR
UNSETTRANS, RESETTRANS

Name: ASSOCIATE

Purpose: To set up a call to a user-defined object to be "associated" with a named picture part for use with the Interactive Handler program in the PDP and Elliott computers.

Example of use: CALL ASSOCIATE(IDN)

Argument: IDN - the identifier of a user-defined object.

Description: This routine only produces any effect when used with PDPGINO and ELLIOTTGINO. The display file for the object is added but will not be visible when displayed.

The routine can be used only when the last picture part added to the total picture was named.

The object specified in ASSOCIATE is associated with this object so that when a pen-see occurs on the named picture part the associated display file is made visible. This facility is only available when the Interactive Handler satellite program is used (see Section 6.3).

Associated display files thus provide a method of making additional information available when using the light pen.

Errors: 1/ GINO ERROR - ASSOCIATE USED IN WRONG CONTEXT.
The last picture part called was not named.

2/ GINO ERROR - ASSOCIATE USED IN OBJECT/SUBPICTURE DEFINITION.
This is forbidden.

BCDCHARS

Issue 2, Feb. 1969

Name: BCDCHARS

Purpose: To add a character string which is held in binary coded decimal form in a FORTRAN real or real array.

Examples of use:

```
CALL BCDCHARS(A,N)
CALL BCDCHARS(CHARRAY(10), 8, NAME)
```

Arguments:

A - a real or real array element, which contains the characters in BCD form. In the case of an array element, characters in excess of 4 are taken from the next consecutive array elements.

N - an integer - the number of characters in the string.

NAME - an optional integer argument - the call name associated with the string.

Description: As for CHARS

BCDCHARS is useful for displaying character strings read in in FORTRAN A4 format or by use of SYMREAD (see the IAL library documentation.)

e.g.

```
      DIMENSION CH(4)
        .
        .
        .
      READ(2,10) CH
10    FORMAT(4A4)
      CALL BCDCHARS(CH, 16)
```

Language: SAL

Length: 340 (uses some space as CHARS)

BUFFERS

Issue 3, April 1970

Name: BUFFERS

Purpose: (a) To establish buffer space in TITAN. For displays, a display file buffer is always needed; this is not needed for plotters. A subpicture/object buffer will be required if subpictures or objects are used.

(b) To start a new picture. For displays this means that a new display file is initialised, any existing display file being discarded. For plotters the output stream is initialised so that the plotter moves to new paper.

Example of use:
```
CALL BUFFERS(DPLACE,IDSIZE,IDERR,SNPLACE,
             ISNSIZE,ISNERR)
```

Arguments:

DPLACE - Location of start of display file buffer. In ASA this may be an array name; in T3 it may be base address + 1.

IDSIZE - Size of display file buffer.

IDERR - Error label for display file buffer overflow.

SNPLACE - Location of start of subpicture/object buffer.

ISNSIZE - Size of subpicture/object buffer.

ISNERR - Error label for subpicture/object buffer overflow.

Description: Calls with arguments are used to set up buffer space. For displays (e.g. when PDPGINO, ELLIOTTGINO are used) the first three arguments specify the display file buffer, which must be provided. Each display file word occupies half a word in the buffer. The second three arguments, which may be omitted, specify the subpicture/object buffer. Each subpicture uses four words of buffer space. The space for an object definition depends entirely on contents. Each GINO call within the definition uses approximately three halfwords.

For plotters (e.g. when PLOTTERGINO is used) a display file buffer is not required and only three arguments need to be given, to specify the subpicture/object buffer. In this case a call to BUFFERS with no arguments will be used if subpictures and objects are not required.

Every call to BUFFERS removes existing subpicture definitions. It does not remove existing object definitions unless preceded by INITGINO or its equivalent. Every call to BUFFERS also starts a

new picture, as defined above. A call without arguments can always be used to start a new picture without changing the location of the buffers. The display file buffer space (if required) will be reused, the subpicture definitions will be lost but the object definitions will still be available.

The user wishing to take specific action on buffer overflow (e.g., to output the picture so far, to output diagnostic information) must assign appropriate labels to IDERR, ISNERR using FORTRAN ASSIGN statements. If buffer overflow is not expected, the simplest course is to give IDERR, ISNERR negative values. Should overflow then occur, a machine code monitor results.

CHARINT
CHARFPT
CHARREAL

Issue 2, June 1970

Names: CHARINT, CHARFPT, CHARREAL

Purpose: To display an integer or floating point number in character form.

Examples of use:

CALL CHARINT(INTEGER, N)

CALL CHARFPT(REAL,N)

CALL CHARREAL(REAL,N,M)

Arguments: The first argument is the number to be displayed. N - field width. M - no. of places after decimal point (CHARREAL only).

Description: CHARINT outputs an integer, justified to the right, preceded by space or minus and filled out on the left with spaces.

CHARFPT outputs a floating point number in the following format. The number is standardised so that its mantissa satisfies $10^{-1} <$ mantissa < 1. The mantissa is output to a maximum of six figures, preceded by space or minus and a decimal point. It is followed by an exponent marker and a three figure exponent. The whole is right justified and filled out with spaces on the left. Thus

CALL CHARFPT(123.456,16) produces

␣␣␣␣␣␣.123456E 3

CHARREAL outputs a real number in a format like the FORTRAN 'F' format.

The routines use the basic routine CHARS.

Errors: If a negative or zero field width is demanded, a single 'E' is output. If the number is too large for the specified field, the field is packed with 'E' 's.

Name: CHARS

Purpose: To add a character string.

Examples of use: CALL CHARS('HERE BE CHARACTERS.')

CALL CHARS('NAMED STRING', NAME)

Arguments: The first argument is a compiled Hollerith string (produced by the ASA compiler).

NAME - optional integer argument - call name associated with the characters.

Description: The size of the characters generated depends on

(a) the latest call to CHARTYPE (q.v.) which sets the current character scale.

(b) the particular output device used.

On the PDP, characters are produced using the 340 character generator. The basic character size is 6 x-units by 7 y-units so character scale 1 (size 12×14) is recommended and assumed if CHARTYPE is not called. Four character sizes are available. The basic character size of the Elliott 1 (size 16×20) is recommended and assumed. Two character sizes are available.

Other output devices (e.g. the CALCOMP plotter) produce characters by software, the characters being exactly the same as those produced by the PDP character generator.

In every case the starting point for displaying the character string is the bottom left hand corner of the initial character. At the end of each character the current position is at the bottom right hand end of the string so that another string can be added without repositioning. When necessary, character strings should be positioned by using routine IPOINT.

Character strings cannot be transformed: they may be windowed (windowing is to the nearest character).

Characters tend to be dim in large display files on the PDP. One solution to this is to define the characters and their positioning point(s) as a subpicture, which is then called repeatedly.

Language: SAL

Length: Approximately 350 if hardware characters, 600 if software characters.

Warning: The directive 'INNERSET' to the ASA compiler should not be used with CHARS.

Name: CHARTYPE

Purpose: To choose the required scale for characters.

Example of use: CALL CHARTYPE(N)

Argument: N - the chosen character size. The basic character size is magnified by 2^N.

Description: For the PDP, N may be 0 - 3 giving four sizes:-

6×7 (the basic size), 12×14, 24×28, 48×56.

For the Elliott, N may be 0 or 1 giving two sizes

8×10 (the basic size), 16×20.

Other devices, which draw characters by software, have the same range of sizes as the PDP.

N = 1 is assumed until CHARTYPE is called. The scaling of characters is independent of the scaling of lines (which may be obtained by use of routine CONTROL).

CIRCLE

Issue 2, June 1969

Name: CIRCLE

Purpose: To add a two-dimensional circular arc. The arc subtends an angle α° where $0 < \alpha < 360$.

Examples of use:

```
CALL CIRCLE(IXC,IYC,IXE,IYE,ISENSE)

CALL CIRCLE(400,400,600,600,1,NAME)
```

Arguments:

IXC,IYC - the X and Y coordinates of the centre of the arc.

IXE,IYE - the X and Y coordinates of a point on the line joining the centre to the end point of the arc.

ISENSE - 0 for clockwise arcs, 1 for anti-clockwise arcs.

NAME - optional argument specifying the call name of the arc.

Description: The arc is drawn from the current position, which can of course be set by a suitable call to IPOINT. Enough short vectors are generated to produce a visibly smooth arc.

Error: If a degenerate case occurs (e.g. centre and starting point coincident)

GINO ERROR - CIRCLE NOT DEFINED.

is printed on stream 0 and the job continues.

Language: SAL and FORTRAN

Length: CIRCLE and CIRCLE3 together total 285 words.

CIRCLE3

Issue 1, June 1969

Name: CIRCLE3

Purpose: To add a three-dimensional circular arc. The arc subtends an angle α where $0 < \alpha < 360$.

Examples of use:

```
CALL CIRCLE3(IXC,IYC,IZC,IXE,IYE,IZE,IXT,IYT,IZT)

CALL CIRCLE3(400,400,400,600,600,600,100,0,0,NAME)
```

Arguments:

IXC,IYC,IZC - the X, Y and Z coordinates of the centre of the arc.

IXE,IYE,IZE - the X, Y and Z coordinates of a point on the line joining the centre to the end point of the arc.

IXT,IYT,IZT - The X, Y and Z components of the "approximate tangent vector" at the start of the arc (see below).

NAME - optional argument specifying the call name of the arc.

Description: The arc is drawn from the current position. The "approximate tangent vector" is used as follows:

(i) Where the arc is not a semicircle or a full circle, the tangent vector determines whether the major or minor arc is required. For this purpose it need only point into the correct half plane.

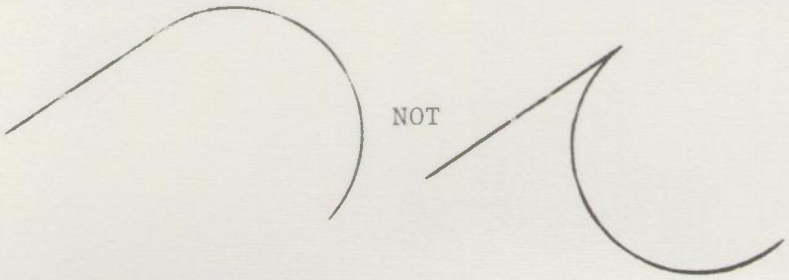
(ii) Where the arc is a semicircle or a full circle, the tangent vector determines the plane in which the arc is to be drawn. For this purpose it must point in the correct direction.

Note that the magnitudes of IXT, IYT and IZT are not used - the tangent vector is only used to establish direction.

In practice, arcs are usually drawn as part of a larger figure. Then, for instance,

```
CALL LINE3(100,200,-100)

CALL CIRCLE3(-,-,-,-,-,-,100,200,-100)
```

is used to produce continuity i.e.



Error: If a degenerate case occurs

```
GINO ERROR - CIRCLE NOT DEFINED.
```

is printed on stream 0 and the job continues. Degeneracy in the tangent vector (i.e. it passes through the centre) does not produce an error if the plane of the arc is determined. In this case the minor arc is drawn.

Language: SAL and FORTRAN

Length: CIRCLE and CIRCLE3 together total 285 words.

COMPLOTGINO

Issue 1, June 1970

Name: COMPLOTGINO

Purpose: To initialise the GINO system and to select the COMPLOT on-line plotter for graphical output.

Example of use:
```
CALL COMPLOTGINO
CALL COMPLOTGINO(NSTREAM, NSIZE)
```

Arguments:

NSTREAM - the output stream to be used. This should not exceed 200.

NSIZE - the output limit in blocks.

Both of these arguments are optional. If they are not given then stream 7 is used with an output limit of 25 blocks. This should be quite adequate for most programs.

Description: This routine is very similar to PLOTTERGINO (q.v.). It must be called before any other GINO routine. It cannot be used with any of the other basic initialisation routines (e.g. PDPGINO, PLOTTERGINO). The routine performs internal initialisation and loads the internal code generating routines for the COMPLOT on-line plotter.

It is not necessary to supply a display file buffer when routine BUFFERS is called. A plot must be terminated by calling routine PLOTEND, which moves the plotter to new paper. A new plot may then be started by another call to BUFFERS. It is not necessary to call COMPLOTGINO again unless all object definitions are to be deleted.

If the output stream specified exists when COMPLOTGINO is called, it is used. Otherwise, it is created as a pending stream. Output is <u>not</u> sent to the plotter automatically. It is the user's responsibility to secure the services of a COMPLOT plotter and to plot the output by use of the PLOT command. Thus if working with the plotter at one's side one would type

```
PLOT 07      ('0' for output)
```

after running a COMPLOTGINO program, to plot output sent to pending stream 7. Output may also be sent to a file and plotted (using 'PLOT <file title>') when convenient.

Errors: If the program moves the pen over the edge of the paper allocation a monitor message is printed and the job stops. Such edge violations can be avoided by use of the windowing option.

Extra Facilities: These are as for PLOTTERGINO except that the stream header is normally omitted (i.e. it has to be explicitly requested by use of routine STRHEAD).

One unit on the plotter is 1/100 inch. The standard width across the paper is 1016 units and the maximum available is 1079 units. The length limit is 3499 units.

Name: CONTROL

Purpose: To set display file controls for pen sensitivity, scale and intensity.

Examples of use: CALL CONTROL(IPEN,ISCALE,INTENS)

IPEN - pen - 0 = no change
2 = disable pen
3 = enable pen

ISCALE - scale - 0, 1, 2, 3. If ISCALE is negative, the scale is not changed.

INTENS - intensity - 0, 1, 2, 3, 4, 5, 6, 7. If INTENS is negative, the intensity is not changed.

Description: Note that the windowing routines do not deal with hardware scale change within subpicture definitions. If CONTROL is called within a subpicture definition so as to change the scale, the subpicture will not be windowed - it will be displayed every time it is called, at the user's risk.

These display file controls are modelled exactly on those for the PDP. With other devices equivalent effects are achieved as far as possible.

For plotters the same image is produced as on the PDP screen but pen sensitivity and intensity arguments are ignored.

For the Elliott, the effect of PDP hardware scaling is simulated by the software. At scales 2 and 3 lines will appear dotted. Only 3 intensity levels are available; INTENS = 0, 1 or 2 giving the dimmest, INTENS = 3, 4 or 5 giving the next and INTENS = 6 or 7 the brightest.

Name: CREATEDISPLAY

Purpose: To provide full formatted output facilities by setting up a notional output stream to the display for use with FORTRAN WRITE statements.

Example of use: CALL CREATEDISPLAY(N)

Argument: N - the number of notional output stream.

Description

After CALL CREATEDISPLAY(N) all output sent to stream N by FORTRAN WRITE statements is not output on stream N but added to the display file buffer. Carriage control characters are not included so only a single record can be output at a time. The beam must be repositioned, using IPOINT, for each new record.

Only one such notional stream can be set up and the routine must not be used together with the IAL routine CREATEPLOTTER. Note that only output via the FORTRAN I/O package is affected - other output to stream N will be lost (it goes to NONE!)

Errors: Illegal use of CREATEDISPLAY causes loader errors.

Name: DEFOBJ

Purpose: To open the definition of a user-defined object.

Example of use: CALL DEFOBJ(IDENTIFIER)

Argument: IDENTIFIER - the identifier by which the object is to be known to the GINO system.

Description: DEFOBJ is used to open an object definition. Until ENDOBJ is called, nothing further is added to the total picture. Instead the object definition is built up in the object buffer. Normally this definition will be a record of the object exactly as defined (i.e. not transformed, not windowed). However if a second argument is given for DEFOBJ then the object will be transformed and windowed as it is defined. In this way quite complicated effects can be achieved. In all cases objects may be transformed and windowed when called (by routine OBJECT).

Object definitions may include calls to existing user-defined objects and subpictures and they may include named calls. Object definitions may not be nested, i.e. after DEFOBJ has been called it may not be called again until ENDOBJ has been called. Also objects may not be defined within subpicture definitions and vice versa.

The value chosen for IDENTIFIER is up to the user. It may be a code number or a pointer to a piece of data structure. If there is already an object defined with the given identifier, the new definition supersedes the old (which is automatically deleted).

Errors:

1. GINO ERROR - ATTEMPT TO NEST OBJECT DEFINITION.

   The program stops.

2. GINO ERROR - OBJECT BUFFER OVERFLOW.

   Control passes to the location specified by the sixth argument of BUFFERS. The normal course is to try again with more buffer space.

3. GINO ERROR - OBJECT BUFFER REQUIRED.

   The job stops. Try again with 6 arguments for BUFFERS.

DEFSUB

Issue 2, Feb. 1969

Name: DEFSUB

Purpose: To open the definition of a subpicture.

Example of use: CALL DEFSUB(IDENTIFIER)

Argument: IDENTIFIER - the identifier by which the subpicture is to be known to the GINO system.

Description: DEFSUB is used to open a subpicture definition. The value chosen for IDENTIFIER is up to the user. Until ENDSUB is called, thus closing the subpicture definition, further display file commands are added to subpicture definition and not to the main display file. Subpicture definition is subject to the transformation current at definition time: the subpicture is then frozen and cannot be changed except for hardware scaling.

Subpicture definitions may include calls to previously defined objects and subpictures: they may not include named calls. Subpicture definitions may not be nested, ie, after DEFSUB has been called it may not be called again until after a call to ENDSUB. Also objects may not be defined within subpicture definitions and vice versa.

Errors: 1. GINO ERROR - ATTEMPT TO NEST SUBPICTURE DEFINITION. The program stops.

2. GINO ERROR - SUBPICTURE BUFFER OVERFLOW. Control passes to the location specified by the sixth argument of BUFFERS.

3. GINO ERROR - SUBPICTURE BUFFER REQUIRED. The program stops. Try again with six arguments for BUFFERS.

Language: SAL

Length: 98

DELETEOBJ

Issue 1, April 1969

Name:	DELETEOBJ
Purpose:	To delete the definition of a user-defined object.
Example of use:	CALL DELETEOBJ(IDENTIFIER)
Argument:	IDENTIFIER - the identifier of the object that is to be deleted.
Description:	The information making up the definition is deleted and the buffer is "closed up" so that the space is freed. The deleted object can no longer be called.
Error:	If the object specified by IDENTIFIER is not currently defined then GINO ERROR - OBJECT <n> NOT DELETED. is printed on stream 0 and the job continues.
Language:	SAL
Length:	53

DEPTHOFF/DEPTHON

Issue 1, April 1969

Names:	DEPTHOFF,DEPTHON
Purpose:	To suspend/restore depth modulation when 3D windowing is being used.
Examples of use:	CALL DEPTHOFF
	CALL DEPTHON
Language:	SAL
Length:	4

Name:	DFCONTIN
Purpose:	To "unterminate" the display file.
Example of use:	CALL DFCONTIN
Description:	DFCONTIN is used whenever it is required to continue adding to the display file after it has been terminated, either explicitly by a call to DFTERMIN or implicitly by a call to one of the output routines (DFPUNB, DFOUTR, etc.).
Language:	SAL
Length:	37

DFOUTB

Issue 1, April 1969
2, May 1970

Name: DFOUTB

Purpose: To output a relocated, self-running binary display file.

Examples of use: CALL DFOUTB(1000)

CALL DFOUTB(IADR,NSTREAM)

Arguments: IADR - satellite address to which display file is to be relocated.

NSTREAM - optional integer argument - specifies the stream for output.

Description: Display files output by DFOUTB are suitable for transmission over the link by use of the PDP command (10). It is assumed that they will be transmitted and run with the appropriate version of LINKBOOT in core in the satellite.

The display file is output in binary to the stream specified, or to stream 6 otherwise. If the output stream does not exist, it is created to pending and a suitable message given. The display file is preceded by a small program to run the display.

Suppose a display file has been output to file /DFB by CALL DFOUTB(2000), then it can be run as follows:

1/ Load and start LINKBOOT

2/ Ensure that the link is engaged.

3/ On the TITAN console type

PDP (/DFB) W A2000 D2000

Further details of the PDP command and of LINKBOOT are in reference (10). The above procedure is the simplest way of transmitting a single display file to the satellite - better facilities are provided by the Display File Manager and routine DFOUTR.

Message: After CALL DFOUTB(M)

DFOUTB - <N> WORDS FOR SATELLITE ADDRESS <M>

is printed on stream 0. The display file and program for running it total N satellite words.

Name:	DFOUTC
Purpose:	To output a relocated display file in character form.
Examples of use:	CALL DFOUTC(2000) CALL DFOUTC(IADR,NSTREAM)
Description:	As for DFPUNC except that the display file is first relocated to address IADR. The user must set up the output stream (stream 6, if not specified by the optional argument).
Message:	DFOUTC - < n > PDP WORDS.
Language:	SAL
Length:	65

Name:	DFOUTR
Purpose:	To output a display file in the relocatable form used by the Display File Manager (DFM).
Examples of use:	CALL DFOUTR CALL DFOUTR(NSTREAM)
Arguments:	NSTREAM - optional integer argument - specifies stream for output.
Description:	The display file is terminated with a stopcode and output in binary form, preceded by a header consisting of a code word and a bit map (see description of DFM, Section 6.3). DFM is capable of receiving a sequence of display files output in this form in a single FDP command and so DFOUTR may be used repeatedly to the same stream. This makes for economy of disc file space. Output is to the stream specified, or stream 6 otherwise. The user is responsible for setting up the output stream.
Message:	DFOUTR - < n > PDP WORDS. The display file (excluding the bit map) is n words long.
Language:	SAL
Length:	48

DFPUNB

Issue 2, June 1970

Name: DFPUNB

Purpose: To punch out a display file in reversed binary form.

Examples of use:

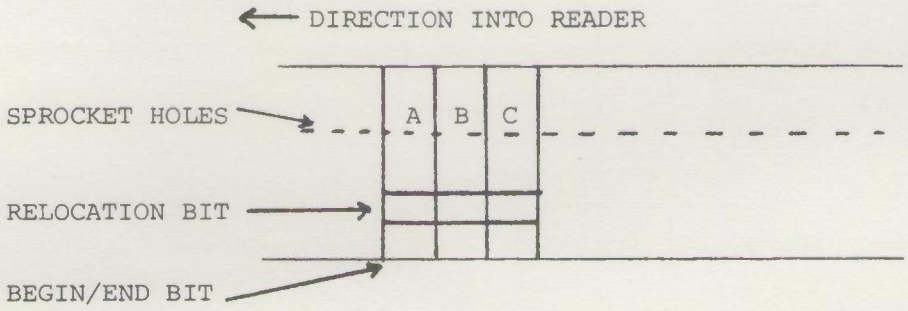
```
CALL DFPUNB
CALL DFPUNB(NSTREAM)
```

Arguments: NSTREAM - optional integer argument - specifies output stream number.

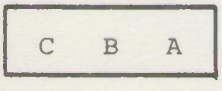
Description: Display files punched out in reversed binary form can be loaded into core in the PDP or Elliott computers using DFTL. Each word of display file occupies 3 rows of holes on 8-track paper tape. DFPUNB is the routine recommended for paper tape output.

The output stream number may be specified (by the optional argument). Stream 15 is used if the stream is not specified. The output stream is created to 8-track tape if it does not exist. The display file is terminated, if necessary, with a stop code or frame timer hold.

The format of reversed binary tape is:



A, B, C each consist of 6 bits. They form a single display file word:



LEAST SIG. END

Message: After each call to DFPUNB

DFPUNB - <n> DISPLAY WORDS

is printed on stream 0. The display file is n display words long.

Name: DFPUNC

Purpose: To punch out a display file in character form.

Examples of use:

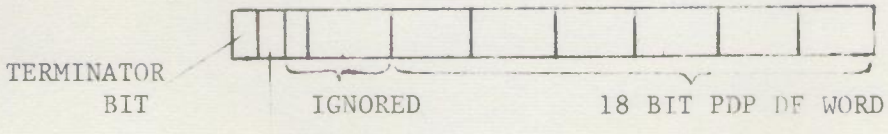
```
CALL DFPUNC

CALL DFPUNC(NSTREAM)
```

Arguments: As for DFPUNB

Description: Display files punched out in character form can be loaded into PDP core at any location using DFTL. Each word of display file occupies 8 rows of holes on the paper tape. Character tapes are thus longer than reversed binary tapes. The user is advised to use DFPUNB rather than DFPUNC unless he has a special reason for needing character output.

The output stream number may be specified (by the optional argument). Stream 15 is used if the stream is not specified. The output stream is created to 7-track tape if it does not exist. The display file is terminated, if necessary, with a DJP PA TOP.

Each TITAN half-word is output as 8 octal digits, the format being:



Message: After each call to DFPUNC

```
DFPUNC - < n > PDP WORDS
```

is printed on stream 0. The display file is n PDP words long.

Language: SAL

Length: 59

Name: DFREAD

Purpose: To read a display file into core in TITAN.

Example of use: CALL DFREAD(ADDRESS,ISTREAM)

Arguments: ADDRESS - Address in TITAN where display file is to be read to. (eg., an array element in ASA, base address + 1 in T3 FORTRAN).

ISTREAM - Number of input stream from which display file is to be read.

Description: DFREAD reads a display file stored in character form (as produced by DFPUNC, DFOUTC) or in reversed binary form (as produced by DFPUNB). The display file may have been produced in TITAN, or punched out from the PDP by DFTP (as described in Section 6.2).

DFREAD distinguishes between character and binary streams. The input stream is deleted after it has been read. To set up an input stream from a reversed binary paper tape, the following form of job description is used:

```
INPUT  D2  (PAW/BDF)
  .
  .
  .
***C
  runout
(PAW/BDF)
***F
```

The binary tape must then follow the JD on the same tape reader. It is set up as input stream 2.

Language: SAL

Length: 78

Name: DFTERMIN

Purpose: To terminate the display file in the manner stipulated by the user.

Example of use: CALL DFTERMIN(IHOW)

Argument: IHOW - specifies the terminator as follows:

IHOW = 0 - the display file is to run in isolation.

IHOW ≠ 0 - the display file is to run with others. Specifically in the PDP case, when IHOW = 0 the terminator is DJP PA top, otherwise it is PAR PA SI. In the Elliott case a frame timer hold is inserted if IHOW = 0, otherwise it is omitted.

Description: DFTERMIN terminates the display file if it is in an unterminated state. It is provided to allow the user to overide the terminators used by other GINO routines.

Name:	ENDOBJ
Purpose:	To close the definition of a user-defined object.
Example of use:	CALL ENDOBJ
Description:	ENDOBJ closes an object definition.
Error:	GINO ERROR - UNMATCHED ENDOBJ. A call to ENDOBJ without a corresponding call to DEFOBJ has occured. The job stops.
Language:	SAL
Length:	10

ENDSUB

Issue 2, Feb. 1969

Name:	ENDSUB
Purpose:	To close a subpicture definition.
Example of call:	CALL ENDSUB
Description:	ENDSUB closes a subpicture definition.
Error:	GINO ERROR - UNMATCHED ENDSUB. A call to ENDSUB without a corresponding call to DEFSUB has occurred. The job stops.
Language:	SAL
Length:	94

Name: GETOBJ

Purpose: To recover an object definition that has been saved on backing store by routine SAVEOBJ.

Example of use: CALL GETOBJ(IDENTIFIER,NISTREAM)

Arguments: IDENTIFIER - the identifier of the object to be recovered.

NISTREAM - the number of the input stream containing the object definition.

Description: The stream is scanned and the relevant object definition is added to the subpicture/object buffer. The object is thus defined and can be called (using routine OBJECT).

The stream is rewound and left open, so that GETOBJ can be called repeatedly.

Error: If the object definition specified by IDENTIFIER is not on the stream then

GINO ERROR - OBJECT < n > NOT ON STREAM

is printed on stream 0 and the job stops.

If the subpicture/object buffer overflows then control passes to the error label set up by the 6th argument of BUFFERS.

Language: SAL

Length: 75

INITGINO,PDPGINO
ELLIOTTGINO

Issue 3, May 1970

Name: INITGINO, PDPGINO, ELLIOTTGINO

Purpose: To initialise the GINO system and to select the required graphical output device.

Examples of use: CALL ELLIOTTGINO

CALL PDPGINO

Arguments: None

Description: One of these routines must be called before all other GINO routines. Only one of them may be used in a given program - failure to observe this rule will result in errors at load time.

The routine performs internal initialisation and loads the internal code generating routines appropriate to the required device. To change the graphical output device used by a GINO program, only this routine needs to be changed.

PDPGINO (or INITGINO, which is synonomous for historical reasons) causes output for the DEC 340 display associated with the PDP7 and PDP9 computers.

ELLIOTTGINO causes output for the 928 display associated with the Elliott 905 computers.

Both of these devices will require routine BUFFERS to set up a display file buffer. The first call to BUFFERS after PDPGINO or ELLIOTTGINO will clear the object buffer. This is the only reason for which it is necessary to call PDPGINO or ELLIOTTGINO more than once in a GINO program.

Issue 2, Feb. 1969

Names: LINE, ILINE

Purpose: To add a two-dimensional line, specified by vector increments.

Examples of use:

```
CALL LINE(IDX,IDY)

CALL ILINE(-100,100)

CALL LINE(IDX,IDY,NAME)
```

Arguments:

IDX - the X increment of the line

IDY - the Y increment of the line

NAME - optional integer argument - the call name associated with the line.

Description: The line is drawn from the current position, with the specified increments. LINE adds the line visibly; ILINE adds it invisibly. If transformations and windowing are used, the increments can be any integers. If not, increments must be in the range $-1023 < IDX < 1023$. This restriction is imposed by the display raster size. Note that the hardware scaling effect is superimposed on the vector increment. Thus, CALL LINE(100,-40) when the hardware scale is 2 will produce a vector on the screen with x-increment 400 and y-increment -160.

Error: If LINE or ILINE causes a vector too large for the screen to be added to the display file

```
GINO ERROR - LINE TOO LONG
```

is printed on stream 0 and the job stops. This can only happen when not windowing.

Language: SAL

Length: The combined LINE and POINT routines occupy 290 words.

Names:	LINE3, ILINE3
Purpose:	To add a three-dimensional line, specified by vector increments.
Examples of use:	CALL LINE3(IDX,IDY,IDZ) CALL ILINE3(100, 0, -100) CALL LINE3(IDX,IDY,IDZ,NAME)
Arguments:	IDX - the X increment of the line. IDY - the Y increment of the line. IDZ - the Z increment of the line. NAME - optional integer argument - the call name associated with the line.
Description:	LINE3 adds the line as a visible line; ILINE3 as an invisible line. Three-dimensional lines will only produce useful results if transformations are used.
Error:	As for LINE
Language:	SAL
Length:	The combined LINE and POINT routines occupy 290 words.

Names: LINEP, ILINEP

Purpose: To add a two-dimensional line from the current position to the specified end point.

Examples of use:

```
CALL LINEP(IX,IY)

CALL ILINEP(200,0)

CALL LINEP(IX,IY,NAME)
```

Arguments:

IX - the X coordinate of the end point of the line.

IY - the Y coordinate of the end point.

NAME - optional integer argument - the call name associated with the line.

Description: As for LINE, except that the line is drawn from the current position to the specified end point. This is independent of the hardware scaling factor.

Error: As for LINE

Language: SAL

Length: The combined LINE and POINT routines occupy 290 words.

Names:	LINEP3, ILINEP3
Purpose:	To add a three-dimensional line, joining the current position to the specified end point.
Examples of use:	CALL LINEP3(IX,IY,IZ) CALL LINEP3(100,200,-50,NAME)
Arguments:	IX - the X coordinate of the end point of the line. IY - the Y coordinate of the end point. IZ - the Z coordinate of the end point. NAME - optional integer argument - the call name associated with the line.
Description:	LINEP3 adds the line as a visible line; ILINEP3 as an invisible line. Three-dimensional lines will only produce useful results if transformations are used.
Error:	As for LINE.
Language:	SAL
Length:	The combined LINE and POINT routines occupy 290 words.

OBJECT

Issue 3, June 1970

Name: OBJECT

Purpose: To add a user-defined object.

Examples of use:

```
CALL OBJECT(IDENTIFIER)

CALL OBJECT(IDENTIFIER,NAME)
```

Arguments:

IDENTIFIER - the identifier by which the object is known to the GINO system. This identifier is given at definition time.

NAME - optional integer argument - the call name associated with this particular call of the object.

Description:

If the object is called in the picture construction process, graphical output for the object is generated. In general, this will represent the transformed, windowed view of the object.

Calls to objects may also be used in other object or subpicture definitions.

For compatibility with issue 1, the routine is also called CALOBJ.

Errors:

If IDENTIFIER is not currently defined

```
GINO ERROR - OBJECT <n> NOT FOUND
```

is printed and a fault 49 is forced, which the user may trap if he wishes.

OBJZDM

Issue 1, June 1970

Name: OBJZDM

Purpose: To call a user-defined object with automatic intensity modulation across its full z-range.

Example of use: CALL OBJZDM(IDN,NAME?)

Arguments:

IDN - the identifier of the user-defined object being called.

NAME - optional argument. The name associated with this call.

Description: This routine is designed to provide automatic intensity modulation without the time and core space penalties of 3D-windowing. Only the range of z-values is calculated before the object is called - this range is then used for intensity modulation so that the view produced uses the full range of intensities. This z-range is left set up on return from the routine so that other objects can be called using the same range.

The routine should be used without windowing as no clipping is necessary.

PICTURE
PICTZCLIP

Issue 1, June 1970

Names: PICTURE, PICTZCLIP

Purpose: To provide automatic scaling and positioning and 3D-windowing (PICTZCLIP) of any transformed view of a user-defined object.

Examples of use:
```
CALL PICTURE(IDN,IW,IE,IS,IN,FAC?)
CALL PICTZCLIP(IDN,IW,IE,IS,IN,BACK,FRONT,FAC?)
```

Arguments:

IDN - the identifier of the user-defined object being called.

IW,IE,IS,IN - the bounds, in picture coordinates, of the area on to which the transformed view is to be mapped.

FAC - optional real argument. If this argument is given, it will be set to the linear scaling factor used to scale the view to the required area, and the TR matrix will be left on return from the routine at the value used within the routine. If this argument is omitted, the TR matrix is restored on return to its value before the routine was called.

BACK,FRONT - real arguments (PICTZCLIP only). The back and front bounds of the z-clipping on a normalised scale in which 0.0 is the back of the object and 1.0 the front.

Description: The routines calculate the extent of the view of the object under the transformation current when they are called and superimposes suitable scaling and positioning transformations so as to map the view onto the defined area. The result is that the required view is produced in the specified area. The scaling is uniform in x and y so no distortion is caused.

In addition, PICTZCLIP sets the z bounds for SET3DWINDOW automatically. This routine involves the whole 3D-windowing machinery and cannot be used with 2D-windowing.

Name: PLOTTERGINO

Purpose: To initialise the GINO system and to select the CALCOMP plotter for graphical output.

Example of use:

```
CALL PLOTTERGINO
CALL PLOTTERGINO(NSTREAM,NSIZE)
```

Arguments:

NSTREAM - the output stream to be used. If NSTREAM exceeds 200 then stream (NSTREAM − 200) is used to the big plotter. If NSTREAM is less than 200, the small plotter is used.

NSIZE - the output limit in blocks.

Both of these arguments are optional. If they are not given then stream 7 is used with an output limit of 25 blocks. This should be quite adequate for most programs.

Description: This routine must be called before any other GINO routine. It cannot be used with any of the other basic initialisation routines (e.g. ELLIOTTGINO, PDPGINO). The routine performs internal initialisation and loads the internal code generating routines for the CALCOMP plotter.

It is not necessary to supply a display file buffer when routine BUFFERS is called. A plot must be terminated by calling routine PLOTEND. A new plot may then be started by another call to BUFFERS. It is not necessary to call PLOTTERGINO again unless all object definitions are to be deleted.

If the output stream specified exists to plotter when PLOTTERGINO is called, it is used. If it exists as a pending stream it is also used, so as to permit dummy runs. In all other cases an existing output stream is terminated and recreated to plotter.

Errors: If the program moves the pen over the edge of the paper allocation a monitor message is printed. The stream will not be terminated so the on-line user has the option of deleting it (e.g. by typing 'DELETE 07') so as to suppress the output or closing it, so as to cause it to be plotted up to the point where the violation occurred. Doing nothing is tantamount to failing to suppress the plot, so use of 'DELETE' is to be encouraged.

Edge violations can of course be avoided entirely by use of the windowing option (see routine SETWINDOW).

Extra facilities:

(a) Orientation. The standard orientation is x across the paper and y along. This may be altered to x along and y across by the routine XALONG:

```
CALL XALONG
```

The standard orientation may be restored

```
CALL XACROSS
```

(b) Paper length. The standard allocation per plot is 3000 units (one unit on the CALCOMP plotter = 1/100 inch). This is the allowed limit along the paper. This limit can be altered by

```
CALL PLOTLENGTH(N)
```

which sets the limit to NX 1024 units.

It is not of course possible to vary the width across the paper. This is 1060 units for the small plotter and 2920 units for the big plotter.

(c) Stream headers. Each new plot (i.e. each new call to BUFFERS) is normally annotated with a stream header, giving user identifier, stream number, job number, date and time. These headers can be suppressed by calling routine NOSTRHEAD and restored by calling routine STRHEAD.

PLOTPDP
PLOTELL

Issue 2, June 1970

Name: PLOTPDP, PLOTELL

Purpose: To produce plotter output in addition to generating display files when using PDPGINO or ELLIOTTGINO.

Example of use: CALL PLOTPDP(IFRAME,NSTREAM)

Arguments: IFRAME - integer in range 0 to 7 - specifies plotter control information as follows:

0 - no frame drawn round picture, plotter advanced.

1 - frame drawn, plotter advanced.

2 - no frame, plotter not advanced so new picture is superimposed on previous picture.

3 - frame drawn, plotter not advanced.

In each of the above cases the y axis is along the plotter paper and the x axis across it. If 4 is added to any of the above values, the same effect is obtained but with x and y interchanged.

NSTREAM - optional integer amount - specifies the number of the output stream to be used for plotter output (stream 7 is used if NSTREAM is not specified).

Description: If the output stream does not exist it is created as a plotter stream. The display file currently occupying the user's display file buffer is terminated (if necessary) and output on the CALCOMP plotter, using the routines of Section 4.2. PLOTPDP must be used with PDPGINO, PLOTELL with ELLIOTTGINO. PLOTTERGINO must not be used.

Messages: As for DFPLOT

Names:	POINT, IPOINT
Purpose:	To add a two-dimensional point.
Examples of use:	CALL IPOINT(IX,IY) CALL POINT(200,300) CALL POINT(IX,IY,NAME)
Arguments:	IX - the X coordinate of the point. IY - the Y coordinate. NAME - optional integer argument - the call name associated with the point.
Description:	POINT adds the point visibly; IPOINT adds it invisibly. If transformations and windowing are being used, the point coordinates need not be within the range $0 < IX < 1023$; $0 < IY < 1023$ of the display raster. Note, however, that the system does not allow the beam to be positioned off the screen by a call to POINT or IPOINT.
Error:	GINO ERROR - ILLEGAL POINT The point would have been outside the screen area. The job stops. This can only happen if windowing is not being used.
Language:	SAL
Size:	The combined LINE and POINT routines occupy 290 words.

Issue 2, Feb. 1969

Names:	POINT3, IPOINT3
Purpose:	To add a three-dimensional point.
Examples of use:	CALL IPOINT3(IX,IY,IZ) CALL POINT3(200,300,-100,NAME)
Arguments:	IX - the X coordinate of the point. IY - the Y coordinate of the point. IZ - the Z coordinate of the point. NAME - optional integer argument - the call name associated with the point.
Description:	POINT3 adds the point as a visible point; IPOINT3 adds it as an invisible point. Three-dimensional points will only produce useful results if transformations are used.
Language:	SAL
Length:	The combined LINE and POINT routines occupy 290 words.

Name: .SALCHARS

Purpose: To add a character string in the SAL format. (This routine is not of interest to FORTRAN programmers.)

Example of use:

```
B89 = PTR :: CINO MAY BE USED FROM SAL. :: ;
            ENTER .SALCHARS.
```

B89 should contain the address of the string.

B88 should contain the call name, if names are being used. It is ignored otherwise.

B90 should contain the return address.

Description: As for CHARS except that, if the string is terminated by a newline, carriage return and line feed characters are included in the display file.

Language: SAL

Length: 400 (shared space with CHARS)

SAVEOBJ

Issue 1, April 1969

Name:	SAVEOBJ
Purpose:	To save the definition of a user-defined object on backing store.
Example of use:	CALL SAVEOBJ(IDENTIFIER,NOSTREAM)
Arguments:	IDENTIFIER - the identifier of the object that is to be saved. NOSTREAM - the number of the output stream to be used.
Description:	The definition of the object specified is output as a single record on the output stream. More than one object definition can be saved on a single stream.
Error:	If the object specified by IDENTIFIER is not currently defined then GINO ERROR - OBJECT < n > NOT SAVED is printed on stream 0 and the job continues.
Language:	SAL
Length:	49

SENDSAT
SENDPDP

Issue 3, June 1970

Name: SENDSAT (also SENDPDP for compatibility)

Purpose: To send a display file to the satellite computer (PDP or Elliott) over the link, whilst operating in expensive mode.

Examples of use:

```
CALL SENDSAT(IPLACE,IHOW,IDATA)
CALL SENDSAT(10)
```

Arguments: IPLACE determines the interpretation placed by the satellite on the data transmitted. If IPLACE < 256, it is interpreted as a segment for DFM, IPLACE being the segment number (see Section 6.3) and the other two arguments (which may be omitted) are used as follows:-

IHOW = 0 or omitted: segment will be displayed

IHOW < 0 : segment will be invisible

IHOW > 0 : on/off status of new segment same as that of existing stament of same number; on if no existing segment.

IDATA - data word of transmission.

If IPLACE > 256 it is taken as the satellite address at which the display file is to be put. In this case:-

IHOW = 0 : the display file is started, otherwise not.

IDATA is used as jump address in the satellite after starting the display file.

Description: In addition to being logged in in expensive mode, the user must create the link (by using the SIGNAL command) and select it (using routine SELECTSAT, Section 5) before calling SENDSAT.

Errors: Direct fault 26 occurs if the link is not created or selected. Delayed fault 60 occurs if the link is disengaged, or if a transfer error occurs. These faults may be trapped by the user.

SETWINDOW

Issue 1, Feb. 1969

Name:	SETWINDOW
Purpose:	To introduce 2D-windowing, so that the picture is visible only within a specified 2D-region.
Example of use:	CALL SETWINDOW(IBW,IBE,IBS,IBN)
Arguments:	IBW,IBE,IBS,IBN - these specify the horizontal (west and east) and vertical (south and north) bounds of the window. The bounds are in terms of picture coordinates.
Description:	Parts of the picture outside the visible window are omitted from the display file. Points on the boundaries are included. Clipping is exact. Since SETWINDOW has an effect at load time, windowing occurs throughout a job that includes a call to SETWINDOW. Until a call to SETWINDOW is executed the assumed window is the whole screen (i.e. IBW = IBS = 0; IBE = IBN = 1023). Note that SETWINDOW may be called more than once in order to vary the window bounds. In addition to the extra space required for windowing, there is a time penalty although the windowing routines do seek to minimise this.
Language:	SAL
Length:	640

SET3DWINDOW

Issue 1, April 1969

Name: SET3DWINDOW

Purpose: To introduce 3D-windowing, so that the picture is visible only within a specified 3D-region. At the same time, depth modulation is introduced.

Example of use: CALL SET3DWINDOW(IBW,IBE,IBS,IBN,IBACK,IFRONT)

Arguments: IBW,IBE,IBS,IBN - these specify the horizontal and vertical bounds of the window, as for SETWINDOW.

IBACK,IFRONT - these specify the bounds on the notional z axis normal to the screen. The positive z-direction is towards the viewer. Note that all the bounds are in terms of picture coordinates.

Description: Parts of the picture outside of the visible region set up are omitted from the display file. Points on the boundaries are included. Clipping is exact.

Depth modulation is also introduced. The intensity of the picture is modulated, with maximum brightness at the front of the visible region and minimum brightness at the back. Depth modulation may be suspended/restored at any time by calling routines DEPTHOFF and DEPTHON.

Until a call to SET3DWINDOW is executed the assumed window bounds are the whole screen, with IBACK = -1000000; IFRONT = 1000000 and no depth modulation.

Language: SAL

Length: 1010

SUBPIC

Issue 3, June 1970

Name: SUBPIC

Purpose: To call a previously defined subpicture.

Examples of use:

```
CALL SUBPIC(IDENTIFIER

CALL SUBPIC(IDENTIFIER,NAME)
```

Arguments:

IDENTIFIER - the identifier, given at definition time, by which the subpicture is known to the GINO system.

NAME - optional integer argument - the call name associated with this particular call of the subpicture.

Description:

Subpictures are not transformed at call time. When windowing, subpictures which cross window boundaries are omitted. Subpictures may be called at any hardware scale. This is the only way in which they can be altered. If any other effect is required, user-defined objects must be used.

Subpicture calls may be included in object definitions and such calls may be named. They may also be included in subpicture definitions but in this case they may not be named.

Error:

If IDENTIFIER does not exist

```
GINO ERROR - SUBPICTURE <n> NOT FOUND
```

is printed and the job stops.

CABINET
CAVALIER

Issue 1, April 1969

Names: CABINET, CAVALIER

Purpose: To set up standard oblique projections.

Examples of use:

```
CALL CABINET

CALL CAVALIER
```

Description: In each case the projection is set up for a total picture centred on the origin, with the X-axis horizontal and the Y-axis vertical. The projections are specified as follows:

1/ The X and Y axes remain orthogonal and are not foreshortened.

2/ The Z axis is equally inclined to the X and Y axes.

3/ In the CABINET projection, the Z axis is not foreshortened; in the CAVALIER projection it is foreshortened by a factor of 1/2.

Language: SAL

Length: 16, 16.

FROMXYZ
AXONXYZ
PROJXYZ

Issue 2, June 1970

Names: FROMXYZ, AXONXYZ, PROJXYZ

Purpose: To set up perspective or axonometric views from the specified viewpoint, with centre of interest at the origin of space coordinates.

Example of use: CALL FROMXYZ(X,Y,Z)

Arguments: X,Y,Z ; reals - the space coordinate of the projection point.

Description: In all cases the view is obtained by projecting from the viewpoint on to a picture plane at the origin of space coordinates. The kind of views obtained are illustrated in Fig. G on p. 2/21.

AXONXYZ uses parallel projection onto a picture plane normal to the line of sight, producing an axonometric view.

PROJXYZ is similar but uses point projection, producing a perspective view.

FROMXYZ also uses point projection but uses a vertical picture plane normal to the vertical plane containing the line of sight. This has the effect of keeping vertical lines vertical - it can introduce extreme distortion in cases where Y is large.

In all cases the view produced still centres on the origin. It will generally have to be positioned by calling OSHIFT.

ISOMETRIC
DIMETRIC
TRIMETRIC

Issue 1, April 1969

Names: ISOMETRIC, DIMETRIC, TRIMETRIC

Purpose: To set up standard axonometric projections (i.e. projections with projection point at infinity).

Examples of use:

```
CALL ISOMETRIC

CALL DIMETRIC

CALL TRIMETRIC (THETA, FI)
```

Arguments: TRIMETRIC only.

THETA, FI - reals - rotation in degrees about Y and X axes, respectively.

Description: In each case the projection is set up for a total picture centred on the origin, with the X-axis horizontal and the Y-axis vertical. The specification of each projection is as follows:

ISOMETRIC - each axis is equally foreshortened.

DIMETRIC - the X and Y axes are equally foreshortened. The routine sets up the special case in which the third axis is foreshortened by a factor of a half.

TRIMETRIC - this is the general case. The only condition is that the orthogonality of the coordinate axes is preserved.

Language: SAL

Length: 15, 16, 24

MAGNIFY

Issue 1, Feb. 1969

Name: MAGNIFY

Purpose: To modify the current transformation by superimposing the specified scale changes.

Examples of use:

```
CALL MAGNIFY(2.0,2.0)

CALL MAGNIFY(U,V,W)
```

Arguments: U,V,W - one, two or three real arguments - scale factors. If only one is given it is taken as the X-scale factor; if two, as the X and Y scale factors, respectively.

Description: The arguments specify the scale factors for the X, Y and Z directions, respectively. Negative scale factors may be used to reverse the customary right-handed orientation of the axes. (e.g. in the production of stereo pairs.)

Note that scaling is relative to the origin of space coordinates current when MAGNIFY is called. Thus after

```
CALL OFIX(-500.,-500.)
CALL MAGNIFY(2.0,2.0)

.

.

CALL POINT(600,600)
```

produces a point whose picture coordinates are (200,200) whereas if the order is reversed, it produces a point at (700,700).

Language: SAL

Length: 15

MODTRANS

Issue 1, Feb. 1969

Name:	MODTRANS
Purpose:	To modify the current transformation by superimposing a user-defined transformation.
Example of use:	CALL MODTRANS(ARRAY)
Argument:	ARRAY - the name of a real (4×4) array (ASA Fortran) containing the matrix of the modifying transformation.
Description:	The current transformation matrix is multiplied by the modifying matrix. The format of elements in the transformation matrix is given in the specification of SETTRANSFORM.
Language:	SAL
Length:	37

OFIX,OSHIFT

Issue 2, June 1970

Names: OFIX, OSHIFT

Purpose: To modify the current transformation by superimposing the specified translation.

Examples of use:

```
CALL OFIX(511.,511.)
CALL OSHIFT(X,Y,Z)
```

Arguments: X,Y,Z - one, two or three real arguments - one argument is taken as X, two as X and Y, respectively.

Description: OSHIFT causes a translation through the vector specified by (X,Y,Z) OFIX causes the translation necessary to bring the point (0,0,0) in space coordinates to the point (X,Y,Z) and thus may be used to translate to an absolute position (e.g., as the last step in a sequence of translations.)

Note that the space coordinate system remains fixed even when OSHIFT or OFIX are used.

OSHIFT may be used to achieve the effect of rotation about an axis not through the origin of space coordinates as follows:-

(i) Call OSHIFT to translate the required centre to (0,0,0)

(ii) Call ROTATE to achieve the required rotations.

(iii) Call OSHIFT again to translate back to the original position.

PROJECT

Issue 1, Feb. 1969

Name:	PROJECT
Purpose:	To modify the current transformation by superimposing the specified perspective in the given coordinate direction.
Example of use:	CALL PROJECT(IDIR,RATIO)
Arguments:	IDIR - integer in range 1 to 3 - specifies the coordinate direction. RATIO - real - specifies the magnitude of the perspective.
Description:	RATIO is typically a small real number, say 10^{-3}. The larger the value of RATIO, the stronger is the perspective employed. More precisely the point at infinity on the axis specified will be transformed into a point distant 1/RATIO from the origin on that axis and this will become the vanishing point of lines that were parallel to the axis before the perspective was applied. If RATIO is negative, the vanishing point will be on the negative axis. This is useful for perspective in the z direction since with the standard orientation of axes CALL PROJECT(3,0.001) produces a vanishing point in front of the screen instead of behind it.
Language:	SAL
Length:	30

ROTATE

Issue 1, Feb. 1969

Name:	ROTATE
Purpose:	To modify the current transformation by superimposing the specified rotation.
Examples of use:	CALL ROTATE(3,30.) CALL ROTATE(IDIR,DEGREES)
Arguments:	IDIR - integer in range 1 - 3 - specifies the axis of rotation (1 for X, 2 for Y, 3 for Z). DEGREES - real - specifies the amount of rotation in degrees.
Description:	The positive sense of rotation is that of a right-handed corkscrew along the positive axis of rotation. So positive rotation about the third axis is anticlockwise rotation on the screen.
Language:	SAL
Length:	43

SELECTVIEW

Issue 1, Feb. 1969

Name: SELECTVIEW

Purpose: To modify the current transformation by permuting the coordinate axes.

Example of use: CALL SELECTVIEW(NHORIZ,NVERT)

Arguments: NHORIZ - integer in range 1 - 3 - specifies axis that is to be the horizontal axis.

NVERT - specifies axis that is to be the vertical axis.

Description: The effect of SELECTVIEW is to permute the rows of the transformation matrix. The remaining axis (i.e. the one not specified by NHORIZ,NVERT) becomes the third axis (i.e. in PDP terms, it is normal to the screen and 'out of' the screen).

It is prudent to make SELECTVIEW the last transformation modifying routine used. Since it changes the transformation matrix rather violently the user is liable to lose track of things after it has been called. Suppose we wish to obtain plan and elevation views of a picture made up of a single user-defined object 10. Having set up the transformation to give us the orientation and view we required, we could procede as follows: (A is a real 16-element array.)

```
                CALL SAVETRANSFORM(A)
C  Preserve the current transformation.
                CALL OBJECT(10)
                .
                .
                .
C  Display x - z view (front elevation)
                CALL SETTRANSFORM(A)
                CALL SELECTVIEW(1,3)
                CALL OBJECT(10)
                .
                .
                .
C  Display y - z view (side elevation)
                CALL SETTRANSFORM(A)
                CALL SELECTVIEW(2,3)
                CALL OBJECT(10)
```

Language: SAL

Length: 27

SETTRANSFORM
SAVETRANSFORM

Issue 1, Feb. 1969

Names: SETTRANSFORM,SAVETRANSFORM

Purpose: SETTRANSFORM is used to set transform mode and to initialise the transformation matrix. SAVETRANSFORM is used to save the transformation matrix by copying it into a user supplied array.

Examples of use:

```
CALL SETTRANSFORM

CALL SETTRANSFORM(TRMATRIX)

CALL SAVETRANSFORM(A)
```

Argument: The argument is the name of a real 16-element array (ASA FORTRAN). If SETTRANSFORM is called without an argument, the transformation matrix is set to the unit matrix, otherwise it is set to the user supplied array. SAVETRANSFORM copies the current matrix into the user supplied array.

Description: SETTRANSFORM must be called before any other transformation routines. (Failure to do so will usually produce a loader error and will certainly produce nonsensical results.)

If the transformation matrix is A(4,4), the significance of the elements is:

A(1,1),A(2,2),A(3,3) represent magnification factors.

A(4,4) represents the overall scaling factor and should be 1.0.

A(1,4),A(2,4),A(3,4) represent origin change terms.

A(4,1),A(4,2),A(4,3) represent perspective terms.

Other off-diagonal elements represent shear terms.

A user wishing to set up his own transformation must use this format. Note that a transformation saved by

```
CALL SAVETRANSFORM(T)
```

may be restored by

```
CALL SETTRANSFORM(T)
```

SAVETRANSFORM does not unset transform mode. This is achieved by a call to routine UNSETTRANS.

Language: SAL

Length: 110

Name:	SHEAR
Purpose:	To modify the current transformation by superimposing the specified shear.
Example of use:	CALL SHEAR(IDIR,IDEP,A)
Arguments:	IDIR - integer in range 1 - 3 - specifies direction of shear (1 = X, etc.) IDEP - integer in range 1 - 3 - specifies dependent direction of shear. A - real - shearing factor.
Description:	This routine is included for the sake of completeness.
Language:	SAL
Length:	21

UNSETTRANS
RESETTRANS

Issue 1, Feb. 1969

Names:	UNSETTRANS, RESETTRANS
Purpose:	To suspend or restore transform mode.
Examples of use:	CALL UNSETTRANS CALL RESETTRANS
Description:	After a call to UNSETTRANS, points and lines are not transformed, i.e. CALL POINT(600,0) puts a point at (600,0) in picture coordinates. The current transformation matrix is unaffected. Transform mode may be restored by a call to RESETTRANS. This facility is useful for placing titles, etc. where required.
Language:	SAL
Length:	6

3. TITAN - GENERATION OF GRAPHS

3.1 Introduction.

The first two routines of this section may be used to produce graphs. One routine generates the graph raster and the other generates the graph. More than one graph may be plotted on any one raster and more than one raster may be used at a time. Four different symbols are available for points and five intensities of lines can be selected to distinguish graphs on the same raster. Scaling is done by an internal routine and can be linear or logarithmic on either axis. Graphs are plotted from arrays of their true values to suit the raster on which they appear. Information about the raster is transferred from the raster routine to the graph plotting routine by a 12-element array argument. Scaling numbers appear alongside the axis when sufficient space is available. The graphs can be plotted as individual points or as straight lines joining consecutive points. Points off the raster are not plotted, but lines to such points are taken to the edge of the raster in the correct direction.

The third routine will produce histograms. The histogram is defined to the routine by an array containing the heights of the columns in the correct sequence together with information about scales and the size and position of the frame. Columns outside the frame are not plotted but the end columns are taken to the edges of the frame. Provision is made for having scaling numbers alongside the axis or not and for omitting the internal vertical lines of the histogram. A control for brightness is provided (only applicable for display output) to make it possible to have two distinguishable histograms in the same frame: this is done by two calls to the routine with those parameters which affect the scaling unaltered.

These routines use the picture part routines of Section 2, so that graphs may be incorporated in some more complex picture if desired. GINO must be initialised (Section 2.5) using the routine appropriate to the display or plotter before calling the graph routines. Note that an object/subpicture buffer must be provided so BUFFERS should be called with 6 arguments (when using PDPGINO or ELLIOTTGINO) or 3 arguments (when using PLOTTERGINO or COMPLOTGINO). Also the LIBRARY command must arrange for the graph routine library file to be scanned before the main library file (see Section 7). The graphs are output just like any other GINO picture (see Section 2.9).

3.2 Specification of Routines.

This section contains specifications of the following routines:

PLOTGRAPH

PLOTPAPER

PLOTHIST

PLOTGRAPH

Issue 2, April 1969

Name: PLOTGRAPH

Purpose: To build display file commands to generate a point or line graph from FORTRAN arrays of X and Y coordinates.

Example of use: CALL PLOTGRAPH(N,X,Y,M,P,IG?)

Arguments:

N Integer number of elements in each input coordinate array.

X) N - element real arrays containing the
Y) x and y coordinates of the points in correct sequence. Actual values should be used, the routine scales them to suit the specified raster.

M Integer code number to select type of graph.

$0 \leq M \leq 4$ Straight lines plotted between consecutive points or to raster edge. Intensity of line controlled by (7 - M).

$-4 \leq M \leq -1$ Gives graph of points only as one of following symbols:

M = -1 Cross, M = -2 Triangle,

M = -3 Square, M = -4 Diamond.

P 12-element information real array previously created by the subroutine 'PLOTPAPER'. This defines the raster on which the graph will appear, also whether linear or log.

IG Optional integer argument - call name associated with the graph. This argument should be omitted unless light pen techniques are being used.

Description: Points outside the bounds of the raster are not displayed but may be used to control the direction of line segments which will be displayed as far as the edge of the raster.

In the case of linear scaling, coordinates with very large values which might cause overflow in the arithmetic unit are set to a value which can be handled by the program. The resulting graph will go to the edge of the raster in a position indistinguishable from the correct one.

For logarithmic graphs, negative or zero coordinates are given values off the raster in order to avoid a monitor. The resulting graph will contain erroneous excursions to the raster edge which will usually appear as obvious errors. These features relieve the user of the obligation to remove such invalid points from his input arrays.

Language: FORTRAN

Length: 1150

PLOTPAPER

Issue 3, June 1970

Name: PLOTPAPER

Purpose: To generate a suitably scaled raster.

Example of use:
```
CALL PLOTPAPER(WI,HI,XO,YO,LX,XMIN,XMAX,LY,
                    YMIN,YMAX,P,IP?)
```

Arguments: LX and LY are integers, P is a 12-element REAL array and must be so dimensioned in the calling program; the rest are real variables. (Note that real arguments require decimal points when called numerically.)

WI, HI Width and height of the raster in screen or plotter units.

XO, YO Coordinates of the bottom-left hand corner of the raster in display or plotter units. This is used to position the raster.

The following size checks are made by PLOTPAPER. If output is for the PDP or Elliott displays then the values must satisfy

$0 \leq WI \leq 1023$; $0 \leq HI \leq 1023$

$XO \geq 0$; $YO \geq 0$; $0 \leq XO + WI \leq 1023$; $0 \leq YO + HI \leq 1023$.

For plotter output the only check is $XO \geq 0$; $YO \geq 0$, so as to permit long graphs. (In this case it is the user's responsibility to avoid edge violations.)

If scaling numbers are required alongside the axes, a margin of 56 units at the bottom and left side and 7 units at the top and right side must be allowed. This changes the constraints to:

$XO \geq 56$ $YO \geq 56$ $XO+WI \leq 1016$ $YO+HI \leq 1016$

This means that the largest raster obtainable on the screen with scaling numbers is 960 × 960.

LX Integer code number for type of scaling on horizontal axis.

LX = 1 gives linear scaling.

LX = 0 gives logarithmic scaling (up to 10 decades).

XMIN Value to be used by scaling algorithm in deciding the left-hand extreme value to be shown on the raster. On return from the subroutine XMIN is set to this extreme value as computed by the scaling algorithm; in some cases this will involve no change.

XMAX Similar to XMIN but for right-hand extreme. Note that XMAX may be less than XMIN if graph is to be plotted this way round.

LY) Similar to LX, XMIN and XMAX but for
YMIN) vertical axis.
YMAX)

P Output array (12-element) contains information about the raster for transfer to PLOTGRAPH. If a raster too large for the screen is requested, P(1) is set to zero as an error indication.

IP Optional integer argument - call name associated with the raster. This argument should be omitted unless light pen techniques are being used.

Description:

The linear scaling algorithm is designed to choose scales that will place the given MIN and MAX as near to the edges of the raster as possible, i.e. to make the graph fill the paper. In some cases this results in scales somewhat less convenient than a human plotter might have chosen; in particular, the algorithm has no inbuilt prejudices about the number 10 and has only a slight bias towards multiples of 5. However, if the user loads MIN and MAX with simple whole numbers the scaling algorithm will usually pass them through unscathed. The range 0 to 11 is the smallest that is not accepted; it is changed to 0 to 12.

The logarithmic scaling algorithm only provides complete decades in sufficient quantity (up to 10) to accommodate the given values of MIN and MAX. Exact multiples of 10 will be accepted by the algorithm if the user requires to force the scales. For one or two decades, lines appear at positions representing all the whole numbers 1 to 9; for 3 or 4 decades lines appear for all even numbers; for 5 or 6 decades lines appear at the 4 and 8 positions only; for 7 to 10 decades the decade lines only appear.

Note, however, that the scaling algorithm cannot provide for the logarithms of negative or zero values and will refuse to construct the raster file.

The raster display is constructed such that the grid lines appear dotted and fainter than the subsequently plotted graph, (i.e. they are built in display scale 2). In linear graphs, a line corresponding to a zero value (if present) appears brighter than the other lines. Similarly in logarithmic rasters, the lines bounding each decade are brighter.

When sufficient space has been allowed, scaling numbers appear along each axis. For linear graphs these numbers are one or two digit signed integers and represent those digits of the actual values that are varying. For instance, an actual range of 9950 to 10150 is shown as:

95 0 5 10 15

The true extreme value nearest the bottom left corner is additionally displayed in exponential form to 3 significant figures, e.g., .995E 4 in example above. For logarithmic scaling, only the exponents of the integral powers of ten are displayed along the axes; the extreme value again appears in exponential form.

Language: FORTRAN

Length: 1950

PLOTHIST

Issue 1, June 1970

Name: PLOTHIST

Purpose: To generate a histogram from a FORTRAN array of column heights.

Example of use:
```
CALL PLOTHIST(WI,HI,XO,YO,XMIN,XMAX,YMIN,YMAX,L,
              Xl,XN,Y,N,MSCALE,MLINES,INTENS,IH?)
```

Arguments: The arguments WI,HI,XO,YO,XMIN,XMAX,YMIN,YMAX are concerned with the scaling and the size and position of the frame. They have the same meanings as in the routine PLOTPAPER.

L — Integer code number for type of scaling on vertical axis, i.e. heights of columns.
L = 1 gives linear scaling
L = 0 gives logarithmic scaling

Xl) XN) — Are the x-coordinates of the centres of the first and last columns. Actual values are used, the routine scales them to suit the frame.

Y — An N-element array containing the heights of the columns in correct sequence. Actual values are used, the routine scales them to suit the frame.

N — Integer number of columns in input array. Any columns not within the frame will not be plotted.

MSCALE — Integer control number

MSCALE = 1 Scales, end values and ticks provided along edges of frame if space is sufficient.

MSCALE = 0 Scales end values and ticks are omitted; frame only plotted.

MSCALE = -1 Scales and ticks provided but no end values.

MLINES — Integer control number.

MLINES = 1 Internal vertical lines of histogram inserted.

MLINES = 0 Internal lines omitted, outer boundary only plotted.

MINTENS — Integer control number for brightness. Recommended value for a typical single histogram, 5. When plotting two histograms in same frame, use 6 and 4.

IH — Optional integer argument call name associated with the histogram. This argument should be omitted unless light pen techniques are being used.

Description: Scaling is done by the same routine as is used by PLOTPAPER which should be consulted for details.

Columns which would go above or below the frame are drawn across the top or bottom of the frame with no other indication of error.

Language: FORTRAN

3.3 SAMPLE PROGRAM.

```
C EXAMPLE OF USE OF GRAPH ROUTINES.
C LEAST SQUARES FIT TO STRAIGHT LINE LAW,Y=A*X+B.
C DATA IS TO BE READ OFF STREAM 0. CALCLINE IS EXTERNAL ROUTINE.
      COMMON SPACE(20),X(100),Y(100),XX(4),YY(4),P(12)
C INITIALISE GINO. CALCOMP PLOTTER OUTPUT IS REQUIRED.
      CALL PLOTTERGINO
C SET UP OBJECT BUFFER. NO OVERFLOW IS ANTICIPATED SO USE
C -VE THIRD ARGUMENT.
      CALL BUFFERS(SPACE,20,-1)
C N.B. IF PDPGINO OR ELLIOTTGINO ARE USED, BUFFERS MUST HAVE
C 6 ARGS. AS A DISPLAY FILE BUFFER IS NEEDED AS WELL.
C READ DATA, THE X AND Y VALUES OF POINTS.
      CALL SELECTIN(0)
      N=READREAL(0)
      DO 1 I=1,N
      X(I)=READREAL(0)
1     Y(I)=READREAL(0)
C CALL SUBROUTINE TO COMPUTE MAX AND MIN VALUES AND THE
C CONSTS A AND B.
      CALL CALCLINE(N,X,Y,A,B,XMIN,XMAX,YMIN,YMAX)
C PLOT RASTER, LINEAR/LINEAR, MAXIMUM SIZE.
      CALL PLOTPAPER(960.,960.,56.,56.,1,XMIN,XMAX,1,YMIN,YMAX,P)
C TEST FOR ERROR.
      IF (P(1)) 5,5,2
C PLOT POINTS AS GIVEN AS SMALL SQUARES.
2     CALL PLOTGRAPH(N,X,Y,-3,P)
C COMPUTE 4 X-VALUES STRADLING EDGES OF RASTER FOR
C STRAIGHT LINE.
      Z=(XMAX-XMIN)/50.
      XX(1)=XMIN-Z
      XX(2)=XMIN+Z
      XX(3)=XMAX-Z
      XX(4)=XMAX+Z
C COMPUTE CORRESPONDING Y-VALUES.
      DO 3 I=1,4
3     YY(I)=A*XX(I)+B
C PLOT STRAIGHT LINE.
C THIS WILL BE CLIPPED TO RASTER EDGES BY THE ROUTINE.
      CALL PLOTGRAPH(4,XX,YY,0,P)
C END THIS PLOT. IT WILL GO TO THE PLOTTER AUTOMATICALLY.
5     CALL PLOTEND
C N.B. PLOTEND WOULD BE REPLACED BY DFPUNB, DFOUTR, OR SENDSAT
C WHEN USING PDPGINO OR ELLIOTTGINO.
      END
```

4. TITAN - PLOTTING DISPLAY FILES

4.1 Introduction

Pictures in the form of PDP display files may be drawn on the CALCOMP plotter attached to TITAN. The display files may be produced by GINO, or punched from the PDP (see the description of DFTP in Section 5.2), or produced in any other way. Corresponding routines for Elliott display files will be produced shortly.

The basic routine used for plotting a display file is DFPLOT (DFPLOTNS can also be used for the same purpose, and has a simplified argument format). At the time of plotting the picture contained in the display file may be clipped to a rectangular region, scaled and than positioned on the plotter paper. The user can also control the plotter output stream so as to plot a number of display files as one composite picture, or separately, or in any other grouping.

Routine PLOTLIMITS is used to set up limits on X and Y in display coordinates. Parts of the display file outside these limits are omitted; clipping being exact. The picture inside these clipping limits may be scaled (routine PLOTSCALES is used to specify X and Y scaling factors) and positioned on the plotter paper (routine PLOTSHIFT being used to specify transverse and longitudinal shifts). The scaled and positioned picture is clipped (if necessary) so as to fit the plotter paper. Routine PLOTSTREAM may be used to control the plotter stream.

All these routines are optional and the parameters they set have default settings such that if none of them are called, DFPLOT "copies" the picture from the screen to the plotter (there is a slight change in size owing to the differing basic increments on the two devices).

Display files produced by the other GINO routines may be plotted directly, without an intermediate output stage to paper tape or file, by using routines PLOTPDP or PLOTELL (see Section 2.13).

4.2 Specifications of the Plotter Routines

This section contains specifications of the following routines:

DFPLOT
DFPLOTNS
PLOTLIMITS
PLOTSCALES
PLOTSHIFT
PLOTSTREAM

Routines DFREAD, PLOTPDP and PLOTELL of Section 2.13 are also relevant to this section.

DFPLOT

Issue 2, June 1969

Name: DFPLOT

Purpose: To send a display file to the plotter.

Examples of use:

CALL DFPLOT(IENTRY,ISTART,ITLOC,ISCALE,IFRAME)

CALL DFPLOT(IENTRY,ISTART,ITLOC,ISCALE)

CALL DFPLOT(0,0,DFARRAY,1,1)

Arguments:

IENTRY - the PDP address at which display execution starts.

ISTART - the PDP address which is the lower limit of the whole display file; in most cases it will be the same as IENTRY.

ITLOC - the address in TITAN of the array containing the display file (the first halfword of this array corresponds to ISTART in the PDP.)

ISCALE - a PDP screen to plotter reduction factor, which may be 1,2,4 or 8. This is applied before any scaling and clipping under the conditions set up by PLOTSCALE,PLOTLIMITS (q.v.)

IFRAME - optional integer argument in range 0 to 7 giving control information as follows:

Let IFRAME = $N_1 + N_2 + N_3$

N_1 = 0 or 1. If N_1 = 1, then draw a frame around the picture; otherwise omit. This is the "frame" option. The frame corresponds to the current values of XLIM1, XLIM2, YLIM1, YLIM2 (see PLOTLIMITS). N_2 = 0 or 2. If N_2 = 2 then the picture is superimposed on the previous picture; otherwise the plotter stream is broken so that the plotter advances to fresh paper. This is the "superimpose" option. N_3 = 0 or 4. If N_3 = 4 the Y axis is drawn parallel to the axis of the plotter drum (i.e. "across" the paper); otherwise the X axis is drawn across the paper. This is the "exchange" option.

Description:

DFPLOT will draw on the plotter any display file stored in consecutive halfwords with each right justified. The display file is interpreted as it would be by the PDP hardware - illegal conditions which would stop the 340 display cause error messages to be printed. By previous calls to PLOTLIMITS, PLOTSCALES, PLOTSHIFT and PLOTSTREAM the user may obtain arbitrary scaling and clipping of the picture (subject to further clipping if required to ensure the picture will fit the plotter), and may specify an output stream. If there has been no call to PLOTSTREAM, DFPLOT uses the lowest numbered stream existing to plotter. Should there be no such stream, stream 7 will be terminated if necessary and recreated to the plotter (using CREATEPLOTTER), with an output limit of 20 blocks.

A display file on paper tape or disc file in one of the standard GINO formats (detailed in the specifications of DFPUNB, DFPUNC in Section 2.13) may be read into core in TITAN by use of routine DFREAD (see Section 2.13).

DFPLOT insists that the display file be terminated either by a stop code (in parameter mode) or by a display jump to the entry address.

DFPLOT is used by routine PLOTOUT (see Section 2.13) which may be used to plot a display file produced by GINO directly from the display file buffer.

Message:

For each display file successfully sent to the plotter the message

DISPLAY FILE n PDP WORDS AT m PLOTTED.

- where m is the value of ISTART - is sent to stream 0.

Errors:

The following error messages may occur on stream 0.

DFPLOT ERROR - DISPLAY ENTRY BEFORE DISPLAY FILE START.

DFPLOT ERROR - OP CODE ZERO IN SUBROUTINE MODE.

DFPLOT ERROR - SLAVE MODE.

Language:

SAL

DFPLOTNS

Issue 1, June 1969

Name:	DFPLOTNS
Purpose:	To send a display file to the plotter.
Examples of use:	CALL DFPLOTNS(IENTRY,ISTART,ITLOC,IFRAME) CALL DFPLOTNS(IENTRY,ISTART,ITLOC)
Arguments:	As for DFPLOT except that ISCALE is omitted. (ISCALE is included in DFPLOT for compatibility with issue 1.)
Description:	As for DFPLOT
Language:	SAL

PLOTLIMITS

Issue 1, June 1969

Name: PLOTLIMITS

Purpose: To set up clipping limits for plotting with DFPLOT.

Examples of use:

```
CALL PLOTLIMITS(IXLIM1,IXLIM2,IYLIM1,IYLIM2)

CALL PLOTLIMITS(200,300,0,700)
```

Arguments: The arguments are 4 integers as follows:

IXLIM1,IXLIM2 - the limits (in display units) to be imposed on X when plotting.

IYLIM1,IYLIM2 - similar limits for Y.

Description: The lower of each pair of values is chosen to be the lower limit so that, for instance, it is not necessary that IXLIM1 < IXLIM2.

Additional clipping may occur if the picture clipped as specified by PLOTLIMITS and scaled as specified by PLOTSCALES will not fit on the plotter. The initial limit settings are:

IXLIM1 = 0, IXLIM2 = 1023, IYLIM1 = 0, IYLIM2 = 1023

Thus until a call is made to PLOTLIMITS the whole of the visible display screen will be plotted (subject to plotter overflow).

Language: SAL

PLOTSCALES

Issue 1, June 1969

Name:	PLOTSCALES
Purpose:	To set up arbitrary scaling in X and Y for plotting with DFPLOT.
Examples of use:	CALL PLOTSCALE(XSCALE,YSCALE) CALL PLOTSCALE(1.,6.5)
Arguments:	2 reals as follows: XSCALE - conversion factor from X on the display (in display units) to X on the plotter (in plotter units). YSCALE - similarly for Y.
Description:	Scaling by arbitrary factors may be applied independently in X and in Y. With incremental pictures, only the end points of a line are scaled (the line increments then being recomputed). Lines that are supposed to meet will, therefore, meet in the plotted picture. The initial settings are XSCALE = YSCALE = 1.0
Language:	SAL

6. PDP/ELLIOTT - PROGRAMS FOR USING THE DISPLAY AND THE LINK

6.1 Introduction

Several programs are available for both the PDP and Elliott computers. The first group (DFTP and DFTL) may be used without the link to TITAN/ATLAS. The second group (DFM, the Interactive Handler and Terminal) use the PDP/ELLIOTT as a satellite. Full specifications of DFM, the Interactive Handler and Terminal are published separately as they are to a greater or lesser extent dependent on which machine is used as satellite. The purpose of Section 6.3 is to provide a summary of the facilities available and an indication of how to use a basic subset of them.

The Display File Tape Producer (DFTP) extracts a display file from the PDP and punches it on paper tape in relocatable form. DFTP which is itself relocatable, may be loaded anywhere in core and needs only the first address of the user's display file. An Elliott version will be produced when required.

The Display File Tape Loader (DFTL) loads into the PDP relocatable display file tapes produced by DFTP or GINO, at any desired location. DFTL can also start the display from any address. A version of DFTL for the Elliott is available.

The Display File Manager (DFM) is a program used to manage the display, particularly when the PDP/Elliott is being used as a satellite. It can be used in a standard preassembled form (as described in Section 6.3) or as a module that can be incorporated in user's satellite programs. Versions of DFM are available for both computers.

The Interactive Handler and Terminal are two "prepackaged" programs designed to make interactive graphics facilities easily available, without any neccessity to write code for the satellite computer. The Handler provides interaction using the light pen, the button keyboard, the satellite teletype and the joystick (PDP only). Terminal provides text editing facilities which can be used for the preparation, modification and monitoring of program data. Versions of the Handler are available for both computers. A version of Terminal for the PDP is available and an Elliott version is being prepared.

6.2 Specification of DFTL and DFTP

Name: Display File Tape Loader

Length: 375 octal

Destroys: C(AC), C(L), C(registers Ø and 1)

Relocatable display tapes produced from TITAN (by GINO routines) or from the PDP (by DFTP) may be read into the PDP using DFTL. The display file so produced can be placed from any address in the machine (above DFTL); this loader also allows the display to be started from any address. Also, any number of display files can be loaded so as to be displayed together.

Method of use: DFTL is loaded by setting 22 on the address switches and pressing 'readin'. At this time the C(AC) is the first address above DFTL.

A display tape is loaded as follows:

1. Set address switches to 22.
2. Set accumulator switches to the first address for the display file.
3. Press 'start'.

These steps may be repeated for other display tapes. At the end of each successful loading the C(AC) is the last address of the display file, or zero otherwise.

To start the display running:

1. Set address switches to 23.
2. Set accumulator switches to the address from which the display is to start.
3. Press 'start'.

Any interrupt causes the display to be restarted from the address set on the accumulator switches when the display was last started by the 'start' key.

To load display files to be displayed together:

1. Load the first tape in the usual manner.

2. For successive tapes do (a) or (b):

   (a) If the display has not been started since the last tape was loaded, place the next tape in the reader and press 'continue'.

   (b) If the display has been started, place the next tape in the reader, set the address switches to 24 and press 'start'. (Method (b) may be used instead of (a)).

Further details: Display tapes are produced in one of three codes - Titan flexowriter, ASCII and a 'reversed binary' - and DFTL automatically recognises and decodes these.

In flexowriter and ASCII codes, the data on the display tape is a succession of numbers, each comprising 8 octal digits. If the first octal digit is 2 then the number to be interpreted will be relocated against the first address of the display file. When the first octal digit is found to be 4, the display file built is taken to be complete. The second octal digit is ignored. The third to eigth octal digits are interpreted as an 18-bit number which is placed in the store (possibly after relocation).

Successive sets of 8 octal digits are similarly treated. The loader ignores all the codes for Ø to 7 and for the asterisk, which halts reading after discarding the next three rows of tape holes.

In reversed binary code each word is represented by three rows of tape holes, with the LEAST significant six bits read FIRST. Channel 7 of the third row is punched whenever relocation is needed and channel 8 is punched at the start and end of the tape.

Display files can legally end 'PAR PA SI' or 'DJP PA first address'. DFTL alters the latter type of ending to the former so that display files of either type can be added and displayed together.

Elliott 905 Version

This loads reversed binary paper tapes (from DFPUNB) only. DFTL is loaded using the "initial instructions".

1/ To load and run a display file. With DFTL in core start with 400200_8 set on the handkeys. Place the display file paper tape in the reader, set the address to which it is to be loaded (mod 17777_8) on the handkeys and set handkey switch 18 to zero. The display file is read in (to module 1) and displayed. The next free location is display in the accumulator.

2/ To start a display file already in core (i.e. previously loaded by DFTL) start at 400201_8, set the start address on the handkeys and set switch 18 to zero.

DFTP

Issue 2, April 1969

Name: Display File Tape Producer

Length: At least 1210 octal

Destroys: C(AC), C(MQ), C(L)

Method of use: For each display subroutine or 'DJP' instruction in the user's display file DFTP uses the next three registers above its highest address so far.

DFTP loads itself anywhere in store by the following procedure:

1. set both the address and accumulator switches to the first address available for DFTP.

2. press 'read-in'

3. when reading halts, press 'start' to complete the loading.

To produce a display tape:

1. set the accumulator switches to the first address of the display file.

2. set AC0 up to produce a display file ending 'DJP PA first address', or down to end 'PAR PA SI'.

3. set AC1 up if this is the first of a number of display files to be joined together (and subsequently displayed together). If AC1 is up the state of AC0 is not considered.

4. set AC2 up for ASCII code, or down for reversed binary code for the display tape.

5. Press 'start'. If AC1 is down, DFTP will complete the display tape and the following steps do not apply. When AC1 is up, DFTP halts when all the first display file has been punched. The next word to be punched is expected in 'PA' mode. When display files are joined this is supplied by the first word of the next display file. To add the next display file if it is already in store:

   1. leave AC1 up

   2. set accumulator switches to the first address of the next display file.

   3. press 'continue'.

To add the next display file if it is not in store:

1. note the address on the program counter when DFTP halts after punching the last display file,

2. set up the next display file in the machine,

3. set accumulator switches to the display file starting address,

4. set AC1 up,

5. set the address switches to the address noted in (1),

6. press 'start'

After the last display file has been punched, complete the display tape:

1. set AC1 down,

2. set ACØ up for a 'self-running' or down for an 'interrupt running' display, as mentioned before,

3. press 'continue'.

At the end of each intermediate stage the AC and MQ lights are alternately on and off (i.e. bits 0, 2, 4 etc. on, and 1, 3, 5 etc. off). At the completion of the display tape the AC and MQ lights are all on and the bell on the teletype is sounded. If the program halts under different conditions then DFTP has been unable to follow the display file.

Notes: It is intended that DFTP will operate with any likely (or not so likely) method of programming the display. DFTP follows the user's display file doing any 'DDS' instructions met. For this reason the display should not be running when using DFTP.

A display file is considered terminated by a 'DJP PA < first address > ' or by an instruction stopping the display.

Only the display instructions used will be punched. Subroutines used are only punched once. However, if display files using common subroutines are joined by DFTP, subroutines used in each display file will be punched once for each display file. If this presents a problem the user might deposit 'PAR SB' (16ØØØØ in octal) for the 'PAR PA SI' and follow this by 'DJP PA < next first address > ' (4XXXXX in octal where XXXXX is the first address of the next display file). DFTP will then produce a display tape holding the subroutines once only.

At the end of the display tape the code for four asterisks is punched if the display tape is in ASCII code. Details of the tape formats are given under the Display File Tape Loader.

6.3 DFM, The Interactive Handler and Terminal

Description of DFM

The Display File Manager (DFM) offers a convenient way of displaying pictures produced by the GINO routines. Pictures are handled as display file segments, each of which can be switched on or off, or replaced or preserved on backing store. Segments are identified by their segment number. Any number of display file segments can be used (up to a limit preset by the user) and the segments may be generated in the satellite or in the central computer, using GINO. In the later case, core allocation in the satellite is performed automatically by DFM within a region of core set aside by the user for display files. DFM also deals with all display interrupts and with named picture parts.

Those wishing to use the full range of facilities of DFM should consult the detailed specifications of the PDP version (14) and the Elliott version (15). Use of the standard preassembled versions, which provide a subset of facilities suitable for picture viewing but not for fully interactive work, is described below.

Use of DFM for Picture Viewing

(i) Setting up DFM in the PDP.

(a) Log in in normal mode on the console allocated to the PDP

(b) Load the LINKBOOT paper tape by setting the PDP address switches to 17600_8 and pressing READIN. Engage the link.

(c) Transmit and start the DFM program by typing on the multiaccess console

```
PDP(CAD/DFM) W AO DO22     (022 is '0' for 'Octal')
```

(ii) Setting up DFM in the Elliott. This procedure may be modified in the future to allow normal mode access.

(a) Log in in expensive mode.

(b) Set some octal number (N) on the handkeys of the Elliott to be used and load the LINKBOOT paper tape using the initial instructions.

(c) Create the link by typing on the multiaccess computer

```
SIGNAL LINK SATELLITE <N>  (N is the octal number set
                            on the handkeys)
```

(d) Transmit and start the DFM program by typing on the multiaccess console

```
PDP(CAD/DFM/905) W AO DO22   (O22 is 'O' for 'Octal')
```

(iii) Displaying pictures (both computers)

Display file segments produced by DFOUTR may be transmitted using PDP commands with address ≤ 256. The address is taken as the segment number. Address zero is taken as meaning "allocate the lowest unused

segment number to this segment." Several segments may be sent in one PDP command. In this case all but the first are treated as having zero address. In expensive mode segments may also be transmitted using SENDSAT.

Thus if file (USER/DF) contains a display file segment

```
PDP(USER/DF) W A6
```

transmits it as segment 6, overwriting any existing segment 6.

```
PDP(USER/DF) W A0
```

would transmit it as a new segment, the number being allocated by DFM.

(iv) Manipulating display file segments (both computers).

The standard version allows up to 16 segments. Numbers 1 - 9 can be switched (on if off and off if on) by typing the corresponding digit on the satellite teletype. Segments can also be switched using the data word (+N switches segment N on, -N switches it off, N=256 switches all segments). This is most likely to be useful in expensive mode (using WRITEPDP to transmit zero words of data with the required data word) but it can also be used with the PDP command, e.g.

```
PDP  W  NO  DO777771
```

may be used to switch off segment 7 (O777771 = -7 in octal).

(v) Reading display file segments back to TITAN/ATLAS.

Any segment originally sent to the satellite over the link can be read back, e.g. to read segment 7 to file (USER/SEG7)

```
PDP(01 USER/SEG7)  R  A7  P1
```

is used. One circumstance in which this facility is useful is when a plotter copy of the display file segment is required (see Section 4).

DFM Error Conditions and Restart Procedures

The following error conditions may arise when using either the standard versions of DFM described above or any program (e.g. the Interactive Handler or Terminal) that uses DFM. The error is indicated by a message on the screen. Error messages are:-

(i) LINK FAULT. A partial restart is possible on the PDP. A complete restart may be necessary on the Elliott.

(ii) INSUFFICIENT SPACE FOR DF. The zone set aside for display file segments from the link is full.

(iii) DF RECEIVED IS IN WRONG FORMAT. Either it is not in display file segment format at all or the format is wrong - usually caused by omitting a call to BUFFERS.

(iv) SEGMENT NUMBER TOO LARGE. The attempted segment is ignored.

(v) PDP COMMAND WITH WRONG P. In an attempt to read a display file segment the argument given with P was not 1.

(vi) SEGMENT CANNOT BE READ. An attempt has been made to read a display file segment up the link when the segment is not in the correct format (i.e. it did not originate from the link.)

DFM may also halt if an attempt is made to switch a segment whose number is too large.

In all cases the satellite program must be restarted. Two kinds of restart are available - a complete restart and a partial restart in which existing segments are preserved. In the standard versions for picture viewing described above 22_8 in the complete restart address and 23_8 in the partial restart address. In some circumstances a partial restart may not be possible, so that a complete restart cannot be avoided.

Description of the Interactive Handler

The Interactive Handler is designed to make the facilities of the PDP/Elliott, as an interactive graphics terminal, easily available from FORTRAN programs running in the central computer. The user of the Handler need write no satellite code at all.

Facilities provided include graphical input using the light pen (tracking cross, sketching, character input), means of selecting named picture parts and reporting the names to the central computer (as already described on page 2/32), the provision of light buttons and the handling of the button keyboard and the joystick (PDP only) as a means of interacting with the program. Versions are provided for both computers and a full user's guide is available (16).

Description of TERMINAL (PDP only)

The version of TERMINAL provided for the PDP is designed to enable the PDP to be used as a graphical on-line terminal for TITAN/ATLAS. It can be used without writing any PDP code.

It consists of a standard DFM plus a scope text editor program. The facilities of DFM have already been described. The text editor allows the display of text and provides deletion and insertion facilities using the lightpen or the teletype. When using the editor, part of the text buffer is displayed on the screen. The part of the text buffer chosen for display can be varied at will. More sophisticated editing facilities are also provided. The full specification of TERMINAL is to be found in Ref. (17).

TERMINAL may be used in normal mode or in expensive mode to prepare, modify and monitor program data. For a discussion of this method of interaction see Ref. (18).

7. HOW TO USE THE GINO ROUTINES

The routines are kept in precompiled form on disc files as detailed below. Programs will usually be run under the COMMAND program although they can also be run under MLS. The GINO routines are obtained by scanning the appropriate library files using the '.LIBRARY' command.

The main GINO library file on both TITAN and ATLAS is called CAD/GINOSAL/*. This contains all routines except the graph routines of Section 3, which are held in a dise called CAD/GINO/GRAPH and the routines for plotting display files, described in Section 4, which are held in file CAD/GINO/PLOTTER.

For the benefit of those new to the COMMAND program and the on-line system we give a few basic details of how to run a program which uses GINO. Those needing more information should consult "The TITAN ASA FORTRAN System Manual"(1) and "The Cambridge Multiple-Access System - User's Reference Manual" (11) and "Command Specifications" (10).

A typical FORTRAN job is run in three stages: compilation, loading of library routines and execution. Suppose the file (USER/GINO/PROG) contains an ASA program which uses GINO routines from Section 2 only. Then the following commands are required:

```
.ASA USER/GINO/PROG
.LIBRARY CAD/GINOSAL/*
.ENTER
```

This may fail for large programs, when the standard space allocations are not enough (see p.11/1 to 11/6 of Volume I of Ref. (1) for details). The allocations may be increased by using various options with the '.ENTER' command. One common source of trouble is insufficient space for forward references. This can be avoided by adding 'FR 1024' after the '.ENTER' command. The program may also require more than the standard allocation of core space. More core may be requested by, say, 'STORE 12K' with the '.ENTER' command. Thus for a large program we might need

```
.ENTER   FR1024   STORE 16K
```

An environment declaration may also be used with the '.ENTER' command to set up input or output streams for the program. This must be enclosed in round brackets and must follow the '.ENTER', preceding any of the options. It is made up of directives such as 'O n <file title>' used to set up output stream n to a file.

If the graph routines are used the appropriate library file must be scanned before the main library file, i.e. the command must be

```
.LIBRARY  CAD/GINO/GRAPH   CAD/GINOSAL/*
```

An incorrect '.LIBRARY' command causes a loader printout with unset parameters (e.g., LINE --- unset).

The most appropriate way to run a job on-line is to use the COMMAND command, which takes a list of commands and obeys them. The commands may be taken from a file or from the on-line input stream. This may be set up in the environment declaration of the COMMAND command. It is introduced by the letter 'H' followed by a single terminating character. All that follows until the terminating character is found on a line of its own is taken as the on-line input stream, i.e. the list of commands to be obeyed. Thus to compile and run a GINO program held on file USER/GINO/PROG and sending display file output via stream 6 to file USER/GINO/DF, we type

```
COMMAND(H*
.ASA  USER/GINO/PROG
.LIBRARY  CAD/GINOSAL/*
.ENTER(06  USER/GINO/DF)
*
)
```

The final ')' closes the environment of the COMMAND command and must be followed by a carriage return. The command must, of course, only be typed when in command status (i.e. when the system is 'READY').

The system responds immediately by echoing the first command (i.e. '.ASA USER/GINO/PROG') and then, when it has finished the job it echoes the remaining commands and types any messages produced by the program. In this instance this will be

```
DFOUTR  -  n DISPLAY WORDS
```

followed by

```
READY
```

The user is then in a position to display his picture on the PDP or Elliott, following the instructions of Section 6.3.

If PLOTTERGINO is being used there is no need to set up output stream 6 with the .ENTER command. The message will be

```
PLOTTERGINO  -  END OF PLOT
```

and the plotted output will duly appear in the output area.

8. REFERENCES

(1) Titan ASA FORTRAN System Manual (2 Volumes). UML, Cambridge. April 1969.
Section IV of Volume 1 contains the current specification of MLS.

(2) The ATLAS-PDP Link (3rd Edition) by C.A. Lang and P. Cross. UML, Cambridge. June 1969.

(3) GINO Graphical Input/Output (1st Edition) by C.A. Lang, P.J. Payne, J.C. Gray, A.P. Armit and A.R. Forrest. University of Cambridge CAD Group. April 1968.

(4) SAL User's Manual (1st Edition) by H. Brown. UML, Cambridge. June 1968.

For a description of SAL see
SAL: Systems Assembly Language by C.A. Lang. SJCC 1969.

(5) New B-Core System for Programming the ESL Display Console by C.A. Lang. ESL Memo 9442-M-122. M.I.T., Cambridge, Mass. April 1965.

(6) Display Interface System, User's Manual by A. Mozley. TM67/1, UML, Cambridge, 1967.

(7) Fortran Package for Generating a PDP-7 Display File by J.W. Brackett, A.C. Kilgour and J.V. Oldfield. CAD Project, University of Edinburgh. November 1967.

(8) The Adage Graphics Terminal by T.G. Hagen, R.J. Nixon and L.G. Schaeffer. FJCC 1968.

(9) Coordinates, Transformations and Visualisation Techniques by A.R. Forrest. University of Cambridge CAD Group Doc. No. 23 (June 1969).

(10) Cambridge Multiple-Access System, Command Specifications. Ed. D.F. Hartley. UML, Cambridge. July 1969.

(11) Cambridge Multiple-Access System, User's Reference Manual. Ed. D.F. Hartley. UML, Cambridge, November 1968.

(12) 340 Display Programming Manual by Sanford C. Adler. DECUS No. 7-13, Maynard, Mass.

(13) GINO - Design and Implementation Features by P.A. Woodsford. University of Cambridge CAD Group Doc. No. 27 (October 1969).

(14) Specification of Display File Manager (PDP version) by P.A. Woodsford,
University of Cambridge CAD Group.
Doc. 43 (to appear).

(15) Specification of Display File Manager (Elliott version) by
C. Litherland.
Ministry of Technology CAD Centre,
Cambridge. (to appear).

(16) The Interactive Handler (PDP and Elliott versions) by M. Newell.
Ministry of Technology CAD Centre,
Cambridge (to appear).

(17) Specification of Terminal - Mark 2 by R.P. Parkins.
University of Cambridge CAD Group.
Doc. 29 - Revised. (Nov. 1969)

(18) How to Use Interactive Computer Graphics in Engineering Analysis
Programs by R.J. Pankhurst.
University of Cambridge CAD Group.
Doc. 32 (Nov. 1969)

(19) 928 Graphical Display by J.A. Monro.
Elliott Brothers (London) Limited
(Nov. 1967)

(20) GINO Graphical Input/Output (Second Edition)
University of Cambridge CAD Group
(June 1969)

9. ROUTINE INDEX

This index indicates where to find the routine specifications. The comment '(with X)' means that the routine is grouped with routine 'X'. Thus to find the specification of DIMETRIC look in Section 2.13a for ISOMETRIC.

Name	Section
SQUASH (with EXPAND)	5.2
STRHEAD (with PLOTTERGINO)	2.13
SUBPIC	2.13
TRIMETRIC (with ISOMETRIC)	2.13a
UNSETTRANS	2.13a
WRITEPDP	5.2
XACROSS (with PLOTTERGINO)	2.13
XALONG (with PLOTTERGINO)	2.13

10. GLOSSARY

This section is presented as an attempt to summarise some of the vocabulary of the manual. It also acts as an index, together with Section 9. Reference is made to the principle sections of the manual bearing on each topic.

		Section
ASA FORTRAN	American Standards Fortran (Fortran IV).	
ATTENTION	An interrupt to TITAN, issued by the PDP when it wishes to initiate a link transfer, or to receive some kind of attention from TITAN.	5.1
ATTENTION DISPLAY	A special display item displayed only when an attention is being processed, as a visible sign that an attention request is outstanding.	6.3
AXONOMETRIC PROJECTION	A projection in which the projection point is at infinity so that parallel lines remain parallel.	2.6, 2.13a
BACKING STORE	Auxilliary storage. In the TITAN operating system this may be magnetic disc or magnetic tape.	2.3
BUFFER	An area of core set aside for a special purpose. In the GINO system there are 2 buffers, one for display file and one for subpicture/object data.	2.3, 2.4, 2.5
BUILT-IN OBJECTS	Picture parts that are built-in to the GINO system, i.e. points, lines characters and circular arcs.	2.1, 2.3
CLIPPING	The restriction of a picture to a specified region by omitting parts of the picture that are outside the region. Not to be confused with BLANKING, in which parts outside the visible region are not omitted, but rendered invisible. BLANKING is not used in GINO.	2.7, 4.1

		Section
CURRENT POSITION	Pictures are generated as a sequence of picture parts so at any time there is a current position.	2.1, 2.2
DELAYED TRAP STATUS	The state into which the user's TITAN program is put when the PDP issues an attention or a link transfer error occurs. All these events occur as delayed fault 60. Further delayed faults (including fault 60's) are held up while the current one is processed.	5.1
DEPTH MODULATION	A visualisation aid whereby the intensity of the display is made proportional to the z coordinate so that a 3D effect is suggested.	2.7
DEPTH CURSOR	A visible region (or window) that is a narrow band in the z-direction. By varying the depth of the cursor various sections of 3D pictures may be obtained.	2.7
DISPLAY FILE	A list of display commands. To produce an image on the screen a display file must be executed repeatedly.	2.1, 2.9, 4.1, 6.2, 6.3, Appendix 1
DFTL	The Display File Tape Loader	6.2
DFTP	The Display File Tape Producer	6.2
DISPLAY FILE MANAGER	Any PDP program which deals with the running of the display. GINO includes a version (DFM).	2.9, 6.3
DISPLAY FILE SEGMENT	The unit in which DFM handles display files. A display file, of any length, terminated by a stop code.	2.9, 6.3
DISPLAY INTERRUPTS	Edge violations, light pen hits and stop codes cause the display to stop and interrupt the (PDP) central processor, which can then identify and process the interrupt.	2.7, 6.3
EDGE VIOLATION	An attempt to position the display beam or the plotter pen outside the allowed area.	2.7,4.1,6.3

		Section
ETAL	A version of TITAN assembly code available under MLS.	
EXECUTIVE	The PDP program which deals with the link.	5.1, 6.3
GRAPH	In this manual a graph is the straightforward graph of schooldays - a plot of Y against X.	3.1
HARDWARE SCALING	Scaling performed by the display hardware when executing display file.	2.13/CONTROL Appendix 1
HOMOGENEOUS COORDINATES	The technique of using (n+1) coordinates for points of an n-dimensional space.	2.6
IDENTIFIER	The integer used by the GINO system to identify a user-defined object or subpicture. It is assigned by the user when he starts the definition of the user-defined object or subpicture.	2.3, 2.4
LIGHT PEN	A photoelectric pointer used with the display. When the pen is enabled, a pen hit interrupt occurs when the pen is pointed at a visible display item.	2.8, 6.3 Appendix 1
LINK	The high speed data link connecting the multiaccess TITAN to the PDP7 as satellite.	2.9, 5.1, 6.3
MLS	The Mixed Language System, under which programs using FORTRAN (ASA & T3), ETAL and SAL are run in TITAN.	1, 7
MODES OF ACCESS	There are several modes of on-line access to TITAN. In normal mode only certain system commands (e.g. EDIT) can be obeyed interactively. In expensive mode any interactive command can be called and the link can be used directly.	2.9, 5.1

		Section
NAME	The integer used to identify light pen hits on a picture part. Sometimes denoted "call name" to emphasise the fact that each call to a picture part should be give a unique name. Names for picture parts are optional.	2.2, 2.3, 2.4, 2.8, 5.1, 6.3
OBJECT	This term is used to include built-in objects and user-defined objects but not subpictures.	2.2, 2.3, 2.4
PAPER TAPE FORMATS	Display files may be punched out on paper tape in character form or in reversed binary form.	2.13/DFPUNB 2.13/DFPUNC 6.2
PDP COMMAND	The multiaccess system command used to transfer data over the link when in normal mode.	2.13/DFOUTB 5.1, 6.3
PERSPECTIVE	The term "full perspective view" is used in this manual to describe projections with a local projection point. Such views are characterised by the existance of vanishing points.	2.6
PICTURE COORDINATES	The coordinate system of the transformed view. This is identical with the screen or plotter coordinate system.	2.1, 2.6
PICTURE PARTS	Built-in objects, user-defined objects and subpictures.	2.1
PROJECTION POINT	The common point of the pencil of lines used in a projection. It may be regarded as the viewpoint. The intersection of the pencil of lines with the projection plane gives the projection on to the plane from the projection point.	2.6
RELOCATABLE	A relocatable program can be loaded anywhere in core. Relocation is the process of filling in a relocatable program ready for loading.	2.9, 6.2, 6.3

		Section
SAL	The Systems Assembly Language.	
SPACE COORDINATES	The coordinate system in terms of which the total picture is defined, i.e. the untransformed coordinate system.	2.1, 2.6
STOPCODE	A display file command causing the display to stop and interrupt the central processor.	2.9, 6.2, 6.3
STREAM	An ordered set of input or output information in the TITAN operating system.	2.9
SUBPICTURE	A user-defined picture subroutine. Calls to a subpicture share the same display file.	2.1, 2.4
TOTAL PICTURE	A collection of picture parts, defined in terms of space coordinates, i.e. the "untransformed picture".	2.1, 2.6
TRACKING CROSS	A displayed item that can be positioned on the screen using the lightpen. It's coordinates are available to the PDP central processor.	6.3
TRANSFORMATION	Mathematically, a change of coordinate base effected by a matrix multiplication. Geometrically this includes scaling, rotation, shifting and point projection.	2.1, 2.6
TRANSFORM MODE	When GINO is in transform mode, picture parts are transformed so that the display file reflects a transformed view of the total picture.	2.6
T3 FORTRAN	A FORTRAN variant peculiar to Cambridge.	
USER-DEFINED OBJECT	A picture part, defined by the user, which causes new display file to be generated each time it is called.	2.1, 2.3
VIEW	The picture presented on the display or plotter. If transform mode is set this view is a transformation of the total picture.	2.6

		Section
WINDOWING	Restriction of the visible picture to a window - a defined region in picture coordinates. 2D or 3D windows may be set up in GINO.	2.7

APPENDIX 1 - HARDWARE DETAILS

Details of the hardware on which the GINO system is implemented will be of interest to readers unfamiliar with the machines available at Cambridge.

TITAN/ATLAS

ATLAS is the ATLAS II computer, of which TITAN is the prototype. Both machines have 128K of 48 bit words and operate under a multi-access system with background batch processing. ATLAS has hardware for paging.

PDP7/PDP9

These two computers are practically identical. They have 8K of 18 bit word store. The two PDP's used by GINO are equipped with DEC340 displays. A detailed specification of this display is to be found in Ref. (12).

Elliott 905

This computer has 16K of 18 bit word store. It is equipped with a 928 display, the specification of which is to be found in Ref. (19).

Data Links

The speed of the data links to the PDP's can be adjusted by the hardware. The PDP7-TITAN link operates at approximately 20000 baud; the PDP9-ATLAS link at a rather higher speed.

The data links to the Elliotts operate at 4800 baud and use the Elliott 916 Modem Controller.

APPENDIX 2 - DESIGN AND IMPLEMENTATION DETAILS

These are described in CAD Group Document 27 (13), which covers all but the most recent features of GINO. An article is in preparation which gives an up-to-date account.