CLM-P182

United Kingdom Atomic Energy Authority

RESEARCH GROUP

Preprint

THE STRUCTURE AND IMPLEMENTATION OF GHOST

F. M. LARKIN

Culham Laboratory
Abingdon Berkshire

1968

UNCLASSIFIED
(Approved for Publication)

CLM - P 182

THE STRUCTURE AND IMPLEMENTATION OF GHOST

by

F.M. LARKIN

(To be submitted for publication in Computer Bulletin)

A B S T R A C T

This paper describes the objectives and current status of GHOST, the Culham computer graphical output system. The system is discussed from several different viewpoints - the user, the system programmer and the machine operator. The direction of possible useful development is also indicated.

U.K.A.E.A., Research Group,
Culham Laboratory,
Abingdon,
Berks.

August, 1968 (SER)

C O N T E N T S

1. INITIAL OBJECTIVES

The past ten or so years have seen the development of a number of the so-called 'high-level' computing languages. Most of these languages, notably those one would classify as being of FORTRAN type, are provided with moderately good facilities for the input and output of alphanumeric information, but their designers have so far largely ignored the problem of graphics. Of course, during the early years of automatic computation, cheap, reliable plotters were hard to find, but with the benefit of hindsight the graphic shortcomings of the main high-level computing languages are now glaringly clear. With increasing use of on-line graphics terminals and high speed electronic plotters, the need for convenient graphic facilities within the normal computing languages becomes even more apparent.

The Culham system, GHOST, was designed in order to provide users with convenient means for effecting graphical output. It was planned, on a fairly modest scale, as an experimental facility which could be adapted to users' requirements and modified in the light of operating experience.

Of necessity, the original version of GHOST is FORTRAN oriented. However, the set of basic graphic commands may be regarded as a rudimentary, self contained language in its own right, which could be implemented to fill the graphical output gap in any of the main computing languages. Heap and Nott (1968) have in fact produced an Algol version, for users at the National Physical Laboratory.

In order to avoid circumlocution the acronym GHOUL has been coined, by an obvious choice of letters from the phrase 'graphical output language', to denote the hardware independent set of users' graphic

commands, while the term GHOST is reserved for the complete Culham implementation of GHOUL in the form of a 'graphical output system'.

GHOST was designed from the user's viewpoint; i.e. the outline structure of GHOUL was specified initially, bearing in mind the hardware available; the necessary software support and operating system then followed fairly naturally.

Before going on to a more detailed description, it will be useful to mention the main design criteria for GHOUL. The least that may be expected of a language is that it should be capable of handling, succinctly and unambiguously, all the concepts for which it is designed. Ideally, it should also be capable of indefinite extension in order to accommodate any further concepts which may be introduced at a later date - a forlorn hope! In the case of GHOUL the former criterion may be reformulated as follows:-

(i) The standard graphical output commands should be readily interpretable in terms of simple manual or mental operations which a human would naturally employ when writing, plotting a graph or drawing a picture.

(ii) All of these natural, human operations should be catered for by basic commands in the language.

It would be presumptuous to suggest that any language could be perfect, so the best one can do towards satisfying (ii) is to rely on experience of users' requirements in order to define the basic elements of the language, and then to allow for modifications to be made. It is virtually impossible to be sure of allowing for indefinite extendability, but one can go some way towards this desirable goal by adhering to the following rule:

(iii) The only operands associated with any graphical output operation (command) should be those immediately required for its definition. Furthermore, the operand lists should be as short as possible.

The implication of (iii) is that compound operations, such as 'advance the plotting frame and/or draw axes' should be avoided, as far as the basic language is concerned. This is desirable because, for example, a particular user might wish to invent a special purpose command 'draw axes and annotate them' without incurring a frame advance as well.

We can summarise the above criteria by saying that the basic graphical output commands should be as natural and powerful as possible, consistent with the requirement that they be 'atomic', in the sense that a user should never feel the need for a partial operation. Of course, out of these atomic commands, which constitute the basic language, a user will be free to construct whatever compound commands suit his purpose. Indeed, to carry the physical analogy still further, certain frequently used 'molecular' commands can profitably be included as standard facilities.

A further, practical requirement is that:

(iv) The basic operations should be capable of implementation on widely available plotting equipment; in practice, this means 'incremental pen plotters' having no built-in, hardware characters.

The particular set of commands initially chosen for GHOUL may be found in appendix A of the "Users' Guide", (Larkin, 1967). It must be emphasised that the names of these commands, limited by the FORTRAN

language, are comparatively unimportant. The important features are the conceptual operations and operands. For a detailed description reference should be made to the "Users' Guide". GHOUL includes, in one form or another, the graphical output operations proposed by Knight (1965), but with rather more insistence upon short and 'natural' argument lists.

In addition to the basic GHOST operations, standard library routines for common operations such as stereo projection, histogram plotting, etc., have, of course, also been developed.

No claim is made that GHOUL is a 'best' graphical output language in any sense, merely that it is a serviceable one. In practice it appears to be easy enough to use, and so far it has not proved difficult to modify and extend without conflicting with existing programs which refer to it.

2. THE USER'S VIEWPOINT

From the point of view of the user, GHOST appears as a set of FORTRAN subroutines which may be called upon in the same fashion as any other library routines; that is, they are included in the operating version of his program simply by being mentioned, and need not be loaded separately. The user may, for example, give the commands

```
CALL POINT(X1, Y1)
CALL JOIN(X2, Y2)
```

and, provided that appropriate mappings have been established, this will result in a straight line drawn upon each one of the designated output media. It is important to note that the interpretation of the arguments X1, X2, Y1 and Y2, and subsequent output action, will depend upon previously executed commands. It is this feature of interdepend-

ence of the commands which justifies the use of the term 'system' for what would otherwise simply be a collection of subroutines.

GHOST commands may be classified either as administrative commands, which supply information to the system (for example, to select a character set), and executive commands (such as POINT and JOIN, above) which can be thought of as directly affecting the graphical output hardware. It is the information supplied by the administrative commands which determines the particular interpretation placed upon the information supplied by subsequent executive commands. The manner of use of these two types of command will become clear from the examples below.

Alternatively, the GHOST commands may be classified according to function, and we now sketch the main features of these different classes:-

(i) Control of graphical output hardware.

The user-image of the GHOST hardware is of a number of separate, parallel output channels, each serving a different plotter. Graphical output from a program may be directed simultaneously to any, or all, of the available channels, which may conceptually be individually switched on or off at any point in the program. This idea of logical switching is extended to cover other facilities, such as broken lines, italic characters, etc. Facilities for frame advance on individual, and all, output channels, as well as for storage and retrieval of the graphical information, are also provided.

(ii) Mapping from model space to plotter space.

When drawing a picture, a human being, consciously or otherwise, employs two distinguishable frames of reference. One such frame of

reference is in plotter space (P-space), i.e. the real, physical plane of the plotting medium, in which titles, annotation, etc., are arranged in convenient positions, with relative distances usually measured in inches or centimetres. The human operator establishes a correspondence between this P-space and the model space (M-space) in which his picture originates. He mentally identifies points in P-space, which he can see, with corresponding points in M-space, which he usually has to imagine. This mapping between M-space and P-space must be defined before plotting of any M-space quantities can take place, and appropriate commands are provided for the purpose. The same commands also effect mappings between pairs of P-spaces.

In the current version of GHOST it is assumed that the user will wish to map rectangular regions in M-space into rectangles in any selected number of the available P-spaces. In all cases, the sides of these rectangles are assumed to be parallel with rectangular, Cartesian, co-ordinate axes fixed in the corresponding spaces; however, their shapes, sizes and off-sets from their respective origins are under program control.

An important distinction is made between the rectangular mapping regions, described above, and limiting rectangles. As well as defining a mapping region in a particular space, the user may specify a limiting rectangle within which all his plotting will be confined. Whereas the mapping regions are necessary in order to establish a metrical correspondence between M-space and the various P-spaces, the limiting rectangles have been introduced partly in order to simplify the location of textual output. A limiting rectangle is analogous to the smallest rectangle which just contains all the text on a page of

a typical book (excluding page number and footnotes). It also serves, incidentally, to delineate the part of a P-space outside of which no plotting may occur.

Since a human operator is constantly shifting his mental frame of reference from M-space to P-space and back, while plotting a graph, special commands are necessary in order to simulate the process. This shift of attention is referred to below as selection of the corresponding space, a process which should not be confused with the operation of switching on an output channel. In GHOST the two operations are performed by the same routine, but this is not necessary and, in some respects, perhaps even rather undesirable.

Once any particular space is selected (regardless of whether or not it is still switched on) all subsequent dimensioned arguments will be interpreted as being measured in the units (inches, centimetres, model units, etc.) which currently apply in the selected space. A P-space cannot be deselected by switching it off - only by explicitly selecting a different space.

Note that any information plotted when a particular P-space (or M-space) is selected will also appear, suitably scaled, on all currently switched-on output channels, since correspondences between their mapping regions will have been established either explicitly, or by default. It should be clear that, whereas any number of the available output channels (each with its associated P-space, but not M-space, of course) may be switched on simultaneously, only one P-space (or M-space) may be selected at any given instant. Furthermore, it is even possible to switch off a selected output channel without deselecting it.

Clearly, all class (ii) commands are of the administrative type.

(iii) Straight lines and curves

The basic straight-line drawing facility has already been illustrated above, so we mention here only the curve drawing commands.

Four basic commands are provided for the purpose of drawing smooth curves which interpolate a finite number of given points. These cover the following four most frequently occuring situations:-

(i) Non-periodic, single-valued graphs;

(ii) Periodic, single-valued graphs;

(iii) Open-ended, possibly multi-valued, curves;

(iv) Closed, multi-valued curves.

In all cases, the shapes of the interpolated graphs or curves will be invariant under a uniform translation of the co-ordinate axes. A rotation of the co-ordinate axes will result in a change of shape in cases (i) and (ii), but not in cases (iii) and (iv). Thus, for example, the instruction

```
CALL GRAPHN (XV, YV, M, N)
```

will result in the drawing of a smooth graph through the points ((XV(I), YV(I)), I = M,N). The abscissae are assumed to satisfy the conditions

$$XV(M) < XV(M+1) < \ldots < XV(N-1) < XV(N).$$

In this case the curve Y(X) will be single-valued and its shape will be invariant under a translation, but not a rotation, of the co-ordinate axes.

As another example, the command

```
CALL GRAPHF (FUNXN)
```

where FUNXN is the name of an EXTERNAL function subprogram, will result in the graph of FUNXN(X) being plotted over that part of the

range of its argument X which lies within the mapping region of the currently selected space.

(iv) Characters and plotting symbols.

Paradoxically, one of the most important requirements of a graphical output language is the capability of handling non-graphical information. For this reason, liberal facilities are provided in GHOUL for the purpose of annotating graphs with alphabetic and numerical characters, as well as symbols indicating plotting positions.

The user may call upon several character sets, each consisting of 64 characters whose size, position and orientation are under program control.

A single character may be located on a page, either by giving the cartesian co-ordinates of its center, or by specifying the number of lines (measured downwards from the upper edge of the limiting rectangle) and the number of spaces (measured to the right from the left-hand edge of the limiting rectangle) on a (space, line) co-ordinate grid. Characters may be output individually, or in strings. Facilities for suffixing, superfixing and italicising characters are also included.

For the purpose of character output two modes, plotting mode and typing mode, are distinguished. All plotting mode commands must supply the (x,y) co-ordinates of the position at which a character is to be plotted.

Whenever a character is output a current typing position is established, immediately to the right of the position of the last character. If a typing mode command then follows, the character next referred to is output at the current typing position. Of course,

no positional information is supplied with a typing mode command.

Provision is made for the plotting (or typing) of numbers in integer, fixed point and floating point formats.

(v) Miscellaneous Operations

In addition to the above classified operations, it has been found useful to include in GHOUL the miscellaneous group listed in Appendix A(v) of the "Users' Guide". These include facilities for causing lines and curves to be drawn in broken form, rotating and expanding the picture (but not the mapping region) and altering the line plotting density. Also a number of compound operations are included, such as adding scales and graticules to pictures, and drawing borders and contours. A lengthy description of these facilities would be out of place in this paper, but details may be found in the Users' Guide.

Fig.1 shows a view of the Benson-Lehner 120 printer/plotter, Fig.2(a) illustrating field lines and contours of field strength in a section through a toroidal magnetic field, is an example of B-L 120 output produced using standard GHOST facilities; Fig.2(b) illustrates a non-technical application.

3. THE SYSTEM PROGRAMMER'S VIEWPOINT

Briefly, the system programming task for GHOST was to implement the commands outlined in the previous section, subject to constraints imposed by the available hardware and by operational requirements.

Apart from the overall limitations imposed by the system configuration, the principle hardware constraints were as follows:-

(a) A design feature of the KDF9 computer limits the amount of immediately addressable program to a maximum of 8000 consecutive words. Thus, any graphical output software loaded

in-core with a user's program should occupy a space small compared with 8,000 words.

(b) Standard KDF9 magnetic tapes were not compatible with the Benson-Lehner 120 printer/plotter. However, a single, compatible tape drive was available.

The principle operational requirements were as follows:-

(c) The system had to cope with a throughput of several hundred frames of output per day, apart from occasional scenes of ciné film.

(d) Because of its experimental nature, the software would be subject to frequent modification.

(e) Users required a fast turnround of results. To be comparable with printed output, permanent graphs had to be available within about two hours of program execution. For on-line working, of course, output on the CRT display had to be immediate.

As often happens in this kind of situation, the governing constraints were not entirely compatible, and a compromise was necessary. After some exploratory work it was decided to provide two parallel sets of software. The 'standard' software would occupy a relatively small fraction of the user's core store and allow use of all the available output channels, but would necessitate batching of the graphical output from a sequence of users' jobs, and hence would involve delay in turnround. The 'CRT' software would occupy more of the user's core store, even after deletion of some of the standard GHOUL facilities; it would allow use only of the CRT display, but output would appear during job execution. As much as possible of the software was written

in FORTRAN in order to simplify debugging, development and subsequent modification.

Fig.3 gives a schematic illustration of the situation. The user may elect to use either the 'standard' or the 'CRT' software. Under the control of the standard software, information defining the required operations and their arguments is stored on a standard KDF 9 ¾" magnetic tape. In this way, graphical output from a sequence of users is accumulated onto a single storage tape. At the discretion of the machine operators a 'processor' job is initiated, which reads and interprets the stored information, and writes appropriate B-L 120 control instructions onto the compatible ½" magnetic tape and/or produces display files which are directed via the PDP8 to the on-line CRT.

With the streamlined, CRT GHOST software all production of CRT display files is done as part of the user's job. This permits limited interactive use of graphical output in conjunction with the Culham on-line computing system, COTAN (CAM Group, 1967). It is not possible to produce permanent pictures directly with the CRT software but, of course, an on-line user can quite conveniently re-run his job with the choice of standard software, whenever he feels his pictures are worth preserving; alternatively, he can photograph the CRT screen using a hand camera.

4. THE MACHINE OPERATOR'S VIEWPOINT

As far as the machine operators are concerned, jobs in the input stream produce a sequence of display files which may go either direct to the on-line CRT or to the ¾" storage tape. The operators have little control over the CRT output, but, since the B-L 120 output must

be batched, they need to be able to exercise flexible control over the scheduling, both of the standard GHOST processor and of the B-L 120 itself. Furthermore, users, machine operators and magnetic tapes being what they are, it sometimes becomes necessary for the operators to reprocess graphical output storage tapes, partially or completely!

Operator control has conveniently been achieved by providing the standard GHOST processor with limited interactive facilities. For example, when the processor types

'PLOT TP ACTION'

on the operators' console, an operator may reply

'FIRST'

indicating 'rewind storage tape and process output originated by first job', or

'NEXT'

indicating 'process output originated by next job on tape', or give a variety of other responses. The operator can also interrupt execution of the processor and, if required, can divert its course of action. Furthermore, regardless of whether or not users have requested CRT output via the standard GHOST software, the operator can choose to follow the progress of processor execution on the screen of the CRT display. This enables him, for example, to check on certain types of user error thus avoiding unnecessary waste of computer time.

During execution of users' jobs, information about the number of graphs produced, and output channels used, is printed on the operators' console. This can then be checked against similar information printed during the subsequent processor run, and is a useful aid to the operators in scheduling processing, and use of the B-L 120.

5. DIRECTION OF FUTURE DEVELOPMENT

GHOST has given valuable service for two years now, but modern developments suggest possible improvements in several directions. The original system was implemented on a computer of IBM 7090 vintage, at a time before any on-line facilities were available. However, recent experience, in conjunction with the Culham COTAN system, tends to support widely held opinions on the usefulness of a properly integrated on-line, multi-access graphics facility, preferably also with real-time computational ability. This would necessitate computers at the upper ends of manufacturers' latest ranges, probably in association with smaller, satellite machines, but, given such hardware, a reasonably comprehensive graphics system could be implemented now.

The simplified functional diagram of Fig.4 illustrates quite a powerful, but nevertheless technically feasible, graphics system. The central concept of GHOST (the distinction between M-space and P-space) is extended and integrated with other established software techniques into a coherent whole.

Basic control over the initiation and interruption of system operations is to be exercised by the user via the console keyboard and/or the light-pen. For example, the user may fetch a program from the Archives and execute it to obtain data in the M-space store defining, say, a road bridge. He may then fetch standard processing software to project this essentially 3-dimensional object onto a plane, and record the 2-dimensional image in the P-space data store. The image is displayed on the CRT, possibly modified (e.g. by deleting hidden lines) with the light-pen, or by annotating from the console, and a permanent microfilm copy made of the current stage on the design. If required, the M-space image may then be modified, for example by

rotating with a tracker ball, and the projection process repeated. At any stage, the current image, either in the M-space or P-space form, may be recorded in the Archives.

In a real-time computing application, for example in air traffic control, the M-space image might be constructed and updated by means of external sensors. Appropriate processing software would provide a P-space image which would be monitored on the CRT display by the controller, who might initiate automatic optimal sequences of emergency action (e.g. clear flight-paths and runways, call fire-fighting equipment, etc.) if the occasion arose.

In view of the wide variety of possible users' programs and types of processing software, applications of such a system seem limited principally by the imagination. To say that it is well suited to providing detailed, flexible, executive control over complex situations almost obscures the point by generalisation; however, the examples of engineering design, industrial process control, typesetting, computational physics, business, transport and military strategy strongly suggest themselves as fields for fruitful application. The research, development and commercial possibilities can hardly be over-emphasised.

6. ACKNOWLEDGEMENTS

A number of people, both permanent staff and visitors to the Laboratory, have assisted in the development of the current version of GHOST. The author gratefully acknowledges the help of all those contributors, especially of Mr. A. Sykes, Mrs. G.M. Leslie and Mrs. V.J. Fox. Thanks are also due to M.A. Coates and Mrs. V.J. Fox for permission to use the examples shown in Figs.2(a) and 2(b).

7. REFERENCES

1. Computing and Applied Mathematics Group; 'A User's Guide to COTAN'. CLM-M75 (In preparation).

2. HEAP, B.R. and NOTT, C.W.; 'Users Guide to the NPL Graphical Output System'. National Physical Laboratory, Ma. 63, March 1968.

3. KNIGHT, G.B.; 'Standard Graphic Output Subroutines as Proposed by a Committee of SHARE'. Proc. 4th Meeting of UAIDE, New York, Oct., 1965.

4. LARKIN, F.M.; 'A User's Guide to the Culham Graphical Output System'. CLM-R84, 1967.

Fig. 1 (CLM-P 182)
View of the Benson-Lehner 120 printer plotter

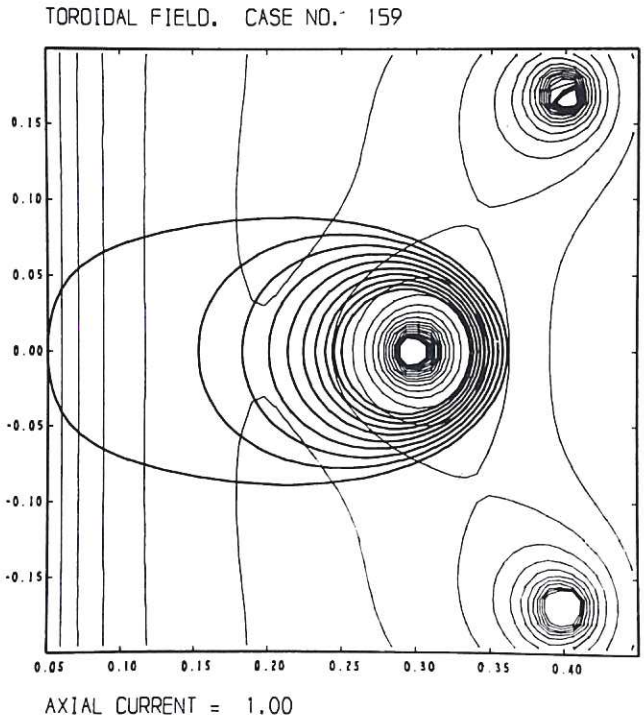

Fig. 2(a) Technical example of the use of GHOST (CLM-P 182)

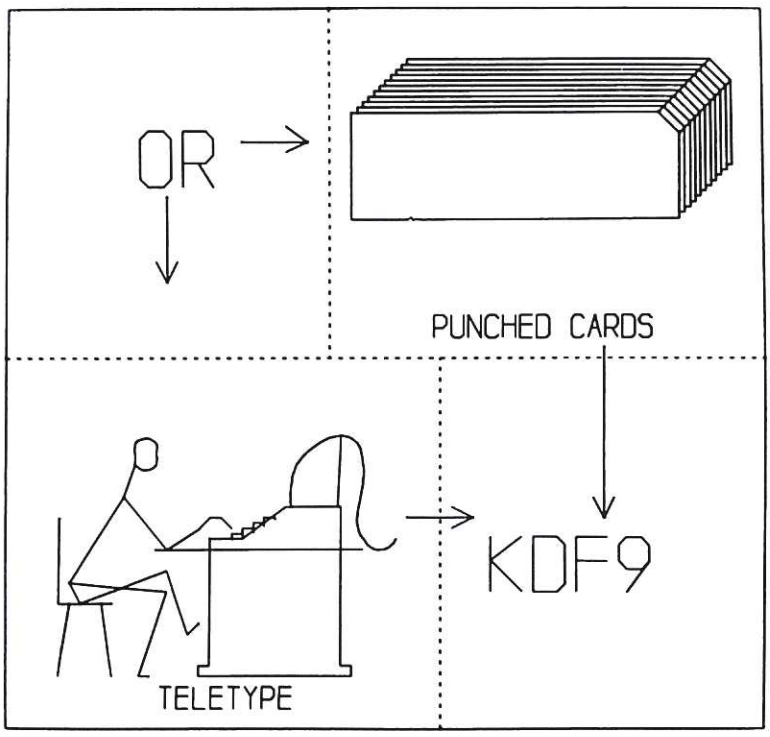

Fig. 2(b) Non-technical example of the use of GHOST (CLM-P 182)

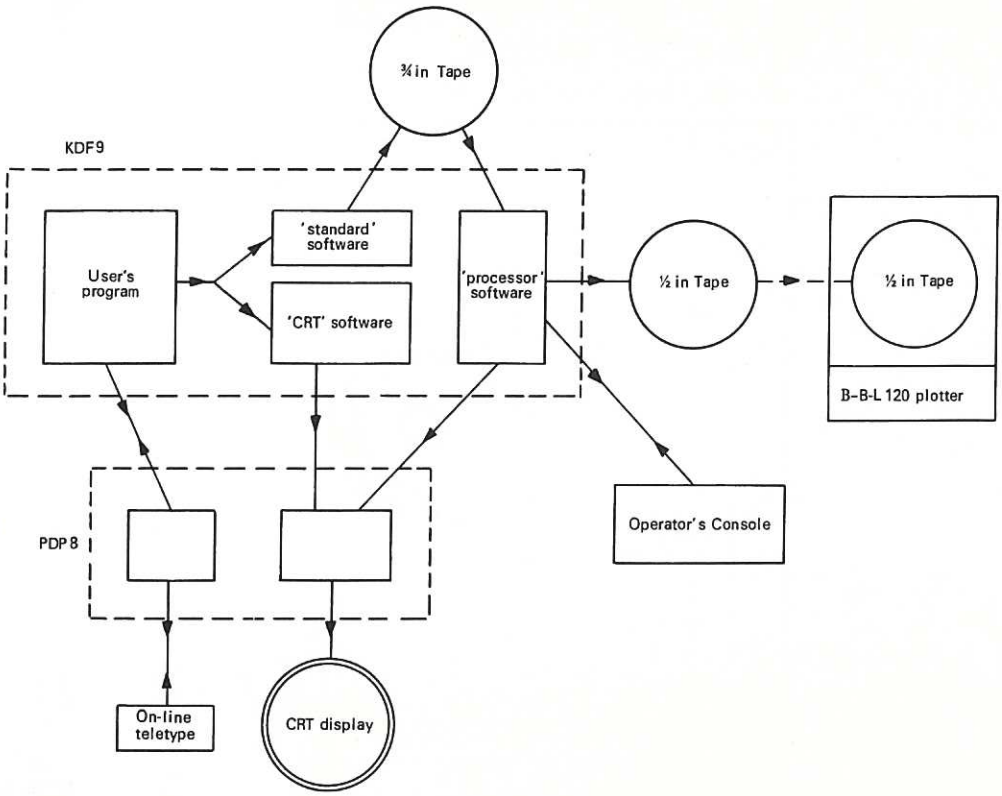

Fig. 3 (CLM-P 182)
Illustration of the different GHOST operational modes (April, 1968)

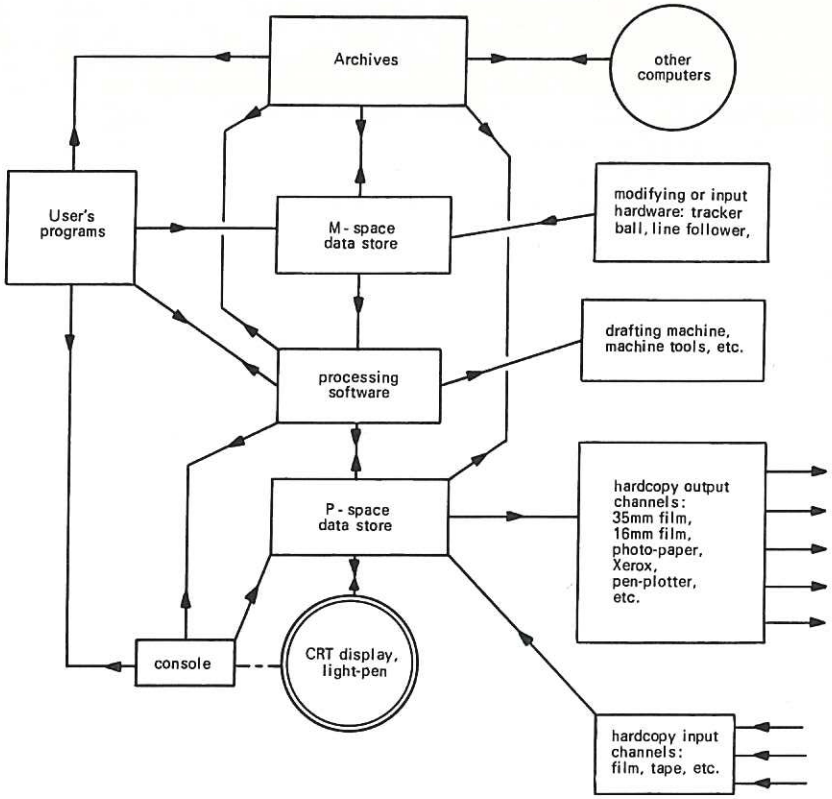

Fig. 4 (CLM-P 182)
Functional diagram of possible graphics-system development