IBM Research

DIALOGUE/360: A SIMPLE APPROACH TO INTERACTIVE NUMERICAL EXPERIMENTATION FOR PHYSICAL SCIENTISTS

Paul M. Grant

July 28, 1970

RJ 734

Yorktown Heights, New York

San Jose, California

Zurich, Switzerland

DIALOGUE/360: A SIMPLE APPROACH TO INTERACTIVE NUMERICAL EXPERIMENTATION FOR PHYSICAL SCIENTISTS

Paul M. Grant
IBM Research Laboratory
San Jose, California

ABSTRACT: Those aspects of interactive computation attractive to workers in the physical sciences are identified and discussed. A PL/I-based graphics terminal package designed to operate on an IBM 2250 Display Unit and attack problems in "numerical experimentation" is presented. The package permits the writer of the basic problem program to communicate with his job merely by calling one of three subroutines. These subroutines allow him a simple interface for entering data, outputing answers, and plotting graphical results. Also included are functions for annotation insertions and performing elementary arithmetic calculations. A typical numerical problem, that of calculating Faraday rotation in a thin absorbing film, is used as a vehicle to demonstrate the system.

RJ 734 (#13873)
July 28, 1970
Output devices

INTRODUCTION

This paper deals with a simple, easy to learn, interactive graphics package of wide generality designed primarily for use by physical scientists. Probably the most extensive computer activity engaged in by scientists of all disciplines entails what we shall call numerical experimentation. By that term we mean interacting on a time scale comfortable to humans, and usually by trial and error, with mathematical models and situations considered intractable prior to the advent of digital computers. This definition would encompass most experimental data analyses and many theoretical modeling problems, but would rule out, for the time being at least, applications comparable to _ab initio_ quantum mechanical calculations. Later in the article we will treat an example of a representative numerical experiment. First of all, however, let us try to identify and define what we feel to be those aspects possessed by numerical experiments _in toto_.

(1) Physical insight is usually inhibited by the complexity of the pertinent equations (often highly non-linear).

(2) Thus one wishes to interact freely with the problem by changing at will interesting parameters, the effect of whose variation on the result is not immediately obvious, in the hope of revealing new and unexpected effects.

(3) Although the mathematics involved may be of labyrinthic construction, the numerical techniques applied frequently make small time demands on present-day central processing units.

(4) Most scientists will desire that both intermediate and final results be presented in graphical, rather than tabular, form.

(5) Because many numerical experiments require much trial and error, retention in hard copy form is necessary only for important results with ready disposability of failures and intermediate calculations maintained as the usual alternative.

In the section to follow, we apply these reflections to the design of an interactive system suitable for physical scientists with a modicum of programming experience. Most of the observations we make have been made before; unfortunately, it seems, at least to the author, that the approaches usually offered are either too specific to one area of science to be universally applied, or are so flexible as to require a level of software sophistication beyond that possessed or desired by the typical physicist or chemist.

The succeeding sections treat the physical features of the display unit involved and the formatting of its output screen, a description of the user-invoked subroutines provided to access the display, an implementation in a typical computer center, and finally an example numerical experiment. Readers not particularly interested in the software structure underlying the accessing subroutines may ignore the discussion of Figs. 5-9 in the appropriate sections and skip the implementation section as well. On the other hand, these figures should be sufficient to satisfy those curious as to the actual framework of the package.

NECESSARY ATTRIBUTES OF A TERMINAL SYSTEM FOR NUMERICAL EXPERIMENTATION

In order to provide the freedom of access to computer resources vital to numerical experimentation, it has proven expedient to develop on-line terminals and time-sharing operating systems of various types. Whether the terminal shares a CPU with many others or occupies a partition in a multi-programming environment is unimportant for our considerations. We will be concerned with the physical features of the terminal and the programming interface the user must learn. Toward this end, let us list some of the principal qualities deemed desirable in a numerical experimentation oriented terminal system.

(1) High-speed graphics capability would seem an absolute necessity. Unfortunately many terminal devices are without this feature; those limited to CRT

alphanumeric displays often lack resolution while those of typewriter design lack both speed and resolution. Mechanical plotters, on the other hand, possess resolution but are slow.

(2) Usually only small numbers of messages and results need be presented at a given time by an interactive terminal system inasmuch as only small amounts can be absorbed mentally by the operator. Thus sacrifice of output volume for high speed display of only important factors would be allowable.

(3) On-line keyboard input of data and commands is an obvious requirement.

(4) The system should grant amnesty to the user for syntax and arithmetical errors by permitting him to restart his problem from his terminal without having his entire job aborted.

(5) Hard-copy backup should be available to support graphical and alphanumeric terminal output. Thus the user would be able to review his whole terminal session off-line if desired.

(6) A "scratchpad" facility for taking notes and performing simple arithmetic calculations (analogous to slide rule operations) during the course of the numerical experiment should be provided.

(7) Creation and editing of source code at the terminal, as well as problem program execution, although not absolutely necessary, would be highly welcome.

(8) Use of a universal high-level language as the terminal programming medium is essential. The language should be as mathematically directed as possible yet allow some manipulation of character variables for logical purposes.

Having now put forth what we believe to be vital criteria for a numerical experimentation terminal system, let us consider an actual embodiment. The heart of our system will prove to be three simple subroutines callable by a user-written main program to perform the functions of writing to, reading from, and plotting on a high-speed CRT graphics display unit.

THE IBM 2250 GRAPHICS DISPLAY TERMINAL*

The most complete description of the physical and logical characteristics of the IBM 2250 Display Unit is to be found in the appropriate IBM Systems Manuals. For the sake of clarity throughout the rest of the paper it will be necessary to describe a few of its features here. Readers interested in recent papers on interactive graphics involving this unit are referred to Ref. 1. Also a review of other presently available graphic terminals and programming support packages may be found among the proceedings of a recent symposium held at Brunel University, UK.[2]

The 2250 Display Unit comprises a CRT display, a light pen, an alphanumeric keyboard, and a set of attention keys. The CRT display consists of an array of 1024 by 1024 addressable points over a 144 in.^{2} area. The farthest distance between two adjacent points is thus 0.017 in., sufficiently near the limits of visual acuity to permit high-resolution plotting. In addition, character generation facilities provide for displaying 52 lines of alphanumeric text containing 74 characters/line, or, in other words, 3848 addressable character positions. Vectoring between points is possible and image regeneration is effected via a 16 Kbyte buffer store.

The alphanumeric keyboard shown in Fig. 1, encompasses the standard EBCDIC character set. When the keyboard is active, a dash (cursor) appears under the screen position where the next character to be typed will be displayed. Errors in typing are corrected by moving the cursor along the line via the BACKSPACE or ADVANCE key and re-typing the proper characters. Depressing the CONTINUE key and any other character key allows that character to be continuously repeated as long as the CONTINUE key is held down. Depressing the JUMP key moves the cursor to some pre-designated area on the screen. The function of the ALT key will be discussed later on.

*We point out that the IBM 2250 is typical of several terminals available on the market today, any of which would serve to treat the problems under discussion.

The attention, or program function, keys, of which there are 32, are used to initiate I/O interupts that identify the particular key depressed so that an appropriate pre-programmed reaction can take place.

SOME FEATURES OF PL/I and GSP

Choosing a programming language suitable for numerical experimentation from among the several extant tongues involves various trade-offs. Ideally one would like the language to be mathematically oriented yet permit easy manipulation of non-arithmetic quantities such as character strings and I/O operations. Also promise of long life, widespread usage, inter-system compatibility, and existence of an efficient compiler would be an extremely valuable trait. It has been the author's experience that data handling through I/O devices poses a far more formidable coding problem than the formulation of the pertinent mathematical algorithm, and any attributes a language may have to alleviate this situation weigh heavily in its favor for use in numerical experimentation. For this reason, and the others noted above, we selected PL/I as the basis for our system. Its algebraic syntax and statement construction is so similar to Fortran (the most widely used scientific language at present) that those of its users wishing to convert have won half the battle upon finding the location of the semicolon symbol on their keyboards.[3] In addition, if one is careful, linkage can be achieved on certain machines between PL/I calling programs and Fortran subroutines.[4] Furthermore, PL/I, while offering formatted I/O resembling Fortran, extends and eases information transfer greatly through its LIST- and DATA-directed features. Also PL/I is much better adapted to the management of logical and character string variables.

The standard IBM-supplied interface to the 2250 is GSP (Graphic Subroutine Package).[5] This package allows a wide variety of operations to be performed with the 2250 and, in fact, is in itself much more than we need for our considerations here.

However, it is through a combination of GSP subroutines that we have created the simple interface which we now wish to discuss and through which the presence of GSP becomes invisible to the user.

LAYOUT OF THE DISPLAY SCREEN AND DESCRIPTION OF ACCESSING SUBROUTINES

Employing GSP, we partition the area of the 2250 CRT screen into three regions as shown in Fig. 2. Each region may be accessed by the user-written problem procedure through use of the sub-procedure indicated. The largest region is reserved for presentation of standard format graphs 500 by 500 points in size. The next largest region displays 10 lines of output 74 characters in length while the smallest region contains one line of input data which is to be entered from the keyboard.

Figure 3 illustrates the overall software organization of the interactive system. The user-written program runs as an external procedure under control of the main procedure monitor program GTASK. Up to five problem procedures can be selected by depressing the appropriate attention key. Each procedure is compiled separately and is then used by the OS/360 LINKAGE EDITOR to replace any of five dummy procedures in GTASK. Within the problem procedure itself, however, the presence of GTASK is transparent to the writer. The only part of the monitor system the problem procedure author need recognize consists of the sub-procedures GREAD, GWRITE, and GPLOTS referred to in Fig. 2 and whose functions will be discussed shortly.

The principal operations of GTASK are summarized by the flow chart of Fig. 4. Apart from initialization services and task selection, GTASK provides two PL/I on-units to monitor the problem procedure. One creates an ON CONDITION to furnish an expedient means for the operator to terminate a particular problem procedure at any time he is in sub-procedure GREAD, while the other is designed to trap all syntactic and arithmetic errors occurring in the problem procedure thus allowing the user to restart his task without being aborted by the Operating System and having to resubmit his job. The particular problem procedure to be executed is selected by depressing one of the enabled (illuminated) attention keys. When this is done an I/O interrupt is received by GSP subroutine RQATN signifying which attention key was activated and a GO TO is computed to call the chosen problem procedure from TASK01 through TASK05 (these are dummy procedures which the Linkage Editor has replaced with live problem procedures). The terminal session is ended by pressing attention key 31.

Let us now consider in some detail each of the three external sub-procedures which our problem procedure author uses to reference the pertinent display areas of Fig. 2. Figure 5 describes sub-procedure or subroutine GWRITE. GWRITE has a single parameter or argument which must be a character string no longer than 74 symbols. The 2250 area serviced by GWRITE has room for 10 such lines and a call to GWRITE inserts the present value of the argument into the bottom-most position shifting the previous occupant along with those above it up one line. The topmost message is, of course, removed from the screen entirely; however, each call to GWRITE also results in its argument being written onto the system output device so that hard copy of all transactions may be eventually obtained. In practice, it has been found that retention of the most recent 10 lines of output on the screen is sufficient for carrying out numerical experimentation.

To perform the function of obtaining data for the problem procedure, the user must invoke sub-procedure GREAD whose flow-chart is contained in Fig. 6. The lowest line of the 2250 screen (see Fig. 2) is reserved for this purpose. Immediately upon being called, GREAD (1) places a question mark in the leftmost character position, (2) places a cursor in the next position so that entries may begin from the alphanumeric keyboard, and (3) sounds an audible alarm alerting the operator that an entry is to be made. In addition, GREAD enables attention key 0 and the alphanumeric keyboard END key. If the user finds his numerical experiment proceeding poorly, he may at this time bail out by pressing attention key 0 thus returning him to GTASK monitor and yielding the opportunity to restart the same or a different problem procedure. Most problem procedures will be found to return to GREAD very often during their operation, hence providing a return option to GTASK from this sub-procedure appears quite logical. More typically, however, the user will commence entering data via the alphanumeric keyboard, each character appearing on the bottom line of the screen as its corresponding key is struck. Up to a total of 73 characters may be entered this way. When the desired message or data has been typed, the operator simultaneously depresses the ALT key and the "5" key (see Fig. 1) thus activating the END function. GREAD then returns control to the calling problem procedure passing in its argument or parameter the character string comprising the contents of the typed-in entry minus the question mark. Experience has shown that one line at a time allows enough room for the usual amount of data input required by the problem procedure.

Although GREAD and GWRITE bestow the ability to transfer character string data to and from the 2250 terminal, it would be extremely advantageous if data conversion features analogous to FORTRAN FORMAT control or PL/I LIST-DATA-EDIT STREAM I/O could somehow be included. It turns out that the user can easily accomplish this for himself within the problem procedure by employing the STRING I/O function in PL/I. That is, PL/I contains an option whereby one can "write" or "read" ("put" or "get") to or from a character string variable just as if that variable were an actual external physical I/O device. Thus, for example, it is possible to use GREAD to obtain numerical data entered from the alphanumeric keyboard in character string form and then "read" this string by LIST-, DATA-, or EDIT-directed STRING I/O as the case may be. Table I gives an example. Conversely, a similar operation is feasible using GWRITE for output, namely, one performs a PUT LIST, DATA, or EDIT to a 74-character string variable which is subsequently passed as the argument to GWRITE and is displayed on the 2250 screen with the desired format conversion. STRING I/O thus greatly enhances the interactive versatility of GREAD and GWRITE.

The third sub-procedure called by a problem procedure, viz., GPLOTS, creates a standard format graph in the display region indicated in Fig. 2. The logic of its argument or parameter list is explained in Table II and its flow diagram given in Fig. 7. Not shown in Fig. 7 are those provisions for off-line hard copy plotting. The sign and value of parameter MARK govern the plotting symbol used, whether the plot appears on the 2250 or is put in direct access storage for later CALCOMP plotting, and whether the plot is to be overlaid onto a previous plot. When the overlay option is selected, the x-axis configuration remains unchanged but the y-axis is replaced with the new one on the screen (for the CALCOMP option, the new y-axis is drawn slightly to the left of the old one). If XAXIS(1) and/or YAXIS(1) are zero on entry to GPLOTS, appropriate scale factors are chosen to allow the data to fill the plotting area; no rounding algorithms are used. The features of GPLOTS will become clearer to the reader in a subsequent example section.

THE "SCRATCHPAD" OPTIONS

It is standard practice for an experimental scientist to maintain a laboratory notebook wherein are recorded important observations and calculations regarding the outcome of his efforts. To a large extent, a similar facility is provided the numerical experimentalist by GWRITE which lists all calls to it on the system output printer. Nevertheless, there arise occasions when one would wish to annotate meaningful or unusual results as they occur and indeed perhaps perform some simple calculations before continuing the experiment. This need is fulfilled by sub-procedure GNOTES which may be activated from either GTASK or GREAD through enabled attention keys 28 and 29 (see Figs. 3, 4, and 6). Thus the user may employ this feature while between problem procedures (waiting in GTASK) or during a problem procedure itself (while waiting for a reply to a CALL GREAD). As seen from Fig. 8, two paths are available on entry to GNOTES; one allows the placing onto the 2250 and also the hard-copy output of whatever comments one desires to insert, whereas the other permits simple numerical computations to be performed. Return to the invoking procedure occurs on the depression of an attention key (29) in the case of the former, and the typing-in of an end message for the latter. Keyboard entries are made in both cases as for GREAD.

Now consider the path labelled "DCC" in Fig. 8. Our objective is to provide the user with the ability to perform "slide-rule" type calculations while running his principal problem procedure. The format we have chosen follows closely that of CALL/360:DCC (for Direct Computation Capability). It is based on operations among four registers; one of them, A, being the register on which the specified arithmetic manipulations take place, and the others, B, C, and D, providing storage space for intermediate results. The operations available are summarized in Table III, and their usage may be seen in the example to be considered in a following section.

AN IMPLEMENTATION

Computer operating systems often interpose great barriers between man and machine. The highest difficulty appears most often not to be the learning of some high level computer language or the use of an interactive system such as described herein, but rather simply the problems encountered in getting one's work to run under a given operating system in a given computer installation. In the present section we will discuss the implementation of our package in the computer center of the IBM San Jose Research Laboratory. However, we wish to reiterate the thoughts of the second section and stress that the basic philosophy and principles of our numerical experimentation system are essentially independent of operating system and physical hardware embodiment.

Our computer center presently consists of an IBM System/360 M91 with 2 megabyte high-speed storage operating under OS/360/MVT/V18 (our package would operate just as well on an M50, however). More than half the storage is allotted to long compute-bound jobs. There are three jobstream regions of 256 kilobytes each serviced by high-priority initiators, the highest priority of which also handles graphics terminal work. Graphics jobs are allowed one minute limits on CPU time which in real time expands to around two hours for the usual numerical experiment performed interactively with such a powerful computer. Work requiring more than one minute or more than 256 kilobytes can be run on a pre-scheduled basis. The interleaving of interactive graphics with compute-bound applications in a multi-programming environment proves quite efficient. The interactive system is in a "wait" state for long periods (anticipating a reply to a CALL GREAD, for example) during which compute-bound programs run unhindered. Yet when short bursts of CPU time for computation are required by the numerical experiment, the high priority of the graphics initiator permits immediate interruption of the compute-bound work for such purposes.

Under our present system organization all graphics jobs are entered from the jobstream and held until activated from the graphics terminal. Jobs using our numerical experimentation package are thus submitted as shown in Fig. 9. Cards 20 through 90 compile the problem procedure. Once compiled, the Linkage Editor builds an executable load module about the problem procedure employing the libraries designated on cards 180, 190, and 200. The library named on card 200 contains GTASK, GNOTES, GWRITE, GREAD, and GPLOTS. Card 230 instructs the Linkage Editor to replace dummy procedure TASK01 with problem procedure YOURTSK. Cards 290 through 310 assign I/O devices required by the resulting module. Card 290 defines a semi-temporary data set for the reception of graphs from GPLOTS which are to be subsequently dumped onto a CALCOMP mechanical plotter. The only changes a user need make from job-to-job is in the problem procedure deck itself signified by cards 70-90 and the Linkage Editor control card 230. If it is desired to assign further problem procedures to the other attention keys available, this is best done by separately compiling and storing them as members of a partitioned data set with subsequent link-editing similar to that specified on cards 180-260. When the size or number of the problem procedures causes the load module to become greater than 256 kilobytes, one may construct an appropriate OVERLAY tree with proper Linkage Editor control cards. The time taken to flip overlaid procedures between core and direct-access memory is well within human reaction time.

SUGGESTED ORGANIZATION OF THE PROBLEM PROCEDURE WITH AN EXAMPLE

The author's experience coding numerical experiments to be run under the PL/I-based system discussed herein has led to the conviction that an optimum design exists for problem procedures. This design is outlined in Fig. 10. As shown, the first task facing the user is frequently the entry of several, or perhaps many, numerical parameters pertinent to his experiment. It is desirable that this be accomplished in the most expeditious way possible. This end is achieved through the use of GREAD, STRING I/O, and, most importantly, PL/I LIST-directed input which supports the highest level free-form numerical syntax permitted by the language. Many parameters can be most likely defaulted, examples being the initial values of XAXIS, YAXIS, and MARK arguments for sub-procedure GPLOTS and quite possibly even the number of points. The required calculations are then performed with these initial parameters. Upon completion, an appropriate PUT statement, STRING I/O, and GWRITE are employed to output a small number of important results on the 2250 keeping in mind the human factors problem pursuant to rapid interactive analysis of any large amount of data. The most informative output will of course be the graph, and the writer of the problem procedure should take care that such will be the case. After reviewing current results, the numerical experimentor may well wish to change selectively one or more particular elements of the variable set, leaving the rest invariant, and loop back to redo the calculation. GREAD, STRING I/O, and PL/I DATA-directed GET statements permit him to perform this function with adroitness and celerity. That is, by explicitly naming and setting each variable to be changed equal to its new value (exactly analogous to FORTRAN NAMELIST with the exception that no list header is required and the ubiquitous semicolon replaces &END), the user may change any variable at random as long as it was defined or DECLARED prior to the GET DATA statement. Upon reception of the new data, the experiment is repeated with the cycle continuing as long as the experimentor wishes. Often, once a satisfying result has been achieved, the GET DATA block provides a convenient technique for beautifying the plot scales and providing hard copy on the next loop through the problem. The problem procedure may be exited at a CALL GREAD via attention key 0.

Expositions of interactive graphics systems in the absence of a live demonstration prove to be tedious and difficult. Lacking the opportunity for said demonstration, we will instead attempt to instruct the reader in the use of the system through a tutorial example. The example we choose comprises the calculation of Faraday rotation and circular dichroism as a function of incident photon energy in a thin absorbing film. Readers conversant with the subject area will recognize the following equations

$$T = \left[\frac{1 + R_+^2}{1 + R_-^2}\right]\left[\frac{1 - R_-^2 \exp(-i2En_-a/hc)}{1 - R_+^2 \exp(-i2En_+a/hc)}\right] \exp(-iE(n_- - n_+)a/hc) \quad , \qquad (1)$$

where

$$R_\pm = \frac{1 - n_\pm}{1 + n_\pm} \; , \qquad (2)$$

$$n_\pm = [1 + 4\pi(\chi_{11} \pm i\chi_{12})]^{1/2} \; , \qquad (3)$$

$$\chi_{ij} = \sum_{g} \sum_{e>g} (P_g - P_e) \frac{\langle g|x_i|e\rangle\langle e|x_j|g\rangle}{E_{ge} + E - i\Gamma_{ge}} + \frac{\langle g|x_j|e\rangle\langle e|x_i|g\rangle}{E_{ge} - E + i\Gamma_{ge}} \ , \tag{4}$$

with

$$P_{g,e} = e^{-E_{g,e}/kT} \Big/ \left(1 + \underset{e>1}{} e^{-E_e/kT}\right) , \tag{5}$$

$$E_{ge} \equiv E_e - E_g \ , \tag{6}$$

and

$$\theta = \mathcal{I}\ (\log T) \ , \tag{7}$$

$$\alpha_{CD} = \mathcal{R}\ (\log T) \ , \tag{8}$$

as describing this system in terms of quite fundamental parameters. Equation (1) relates the relative transmission of the left (-) and right (+) circularly polarized components of the linearly polarized incident light for film thickness a. The next two equations define the complex optical response functions $R_{\pm}$, the fresnel reflectivity coefficient, and $n_{\pm}$, the index of refraction, which are employed in Eq. (1). These are followed by the complex susceptibility tensor, χ_{ij}, as derived from first-order time dependent perturbation theory. Here in Eq. (4) are contained the fundamental quantities defining the material comprising the film. The subscripts g and e enumerate, respectively, the possible initial and final electronic states of the system whose energy values are given by E_g and E_e and whose difference E_{ge} is defined by Eq. (6). Γ_{ge} is then the linewidth of the optical transition between any two states and E is the energy of the incident photon. In the numerator of Eq. (4) we have the matrix elements $\langle g|x_i|e\rangle$ of the cartesian components of the dipole operator connecting the magnetically decoupled system of energy levels. P_g and P_e denote occupancy of the initial and final states, respectively, with the population being Boltzmann-distributed with temperature as shown in Eq. (5). Finally the sought for Faraday rotation and circular dichroism are displayed in Eqs. (7) and (8).

A perusal of the above equations should leave the reader with an appreciation of their mathematical complexity and also of the vicissitudes to be encountered in attempts to trace through to the final result the effect of variations in the fundamental parameters without numerical experimentation. This morbid algebraic construction is, of course, common to all problems susceptible to this technique. For our tutorial example, we will apply Eqs. (1)-(8) to optical transitions between a ^2S ground state and a ^2P excited state in the presence of spin-orbit coupling and a magnetic field. The pertinent multiplet structure is shown in Fig. 11 with the transitions concerned indicated thereon. Figures 12-17 contain photographs of the sequence of events leading through the problem. Detailed descriptions of the operations underlying each figure will be found in the associated caption. The experiment consists of the following sequence of events:

(1) The initial calculation of Faraday rotation between 0.0 and 5.0 eV for a transition at 2.0 eV with a linewidth of 0.4 eV, spin-orbit splitting of 0.1 eV, and oscillator strength of 0.01 eV in a magnetic field of 10^5 gauss is performed for a sample 5μ thick at a temperature of 4.2°K (Figs. 12 and 13).

(2) The spin-orbit splitting is then changed to 0.8 eV and the new calculation overlaid against the old (Figs. 13 and 14).

(3) The display of the circular dichroic phase shift is then requested (Fig. 15).

(4) A return to the previous value of spin-orbit coupling is made but with a change in temperature to 300°K. A notation is inserted using GNOTES (NOTES)

commenting on what is to be expected from this modification (Fig. 16), and a new graph produced.

(5) Finally a calculation is effected employing GNOTES (DCC) to determine the ground state splitting in eV units (Fig. 17).

Thus we have illustrated with an actual example the philosophy of problem procedure organization outlined at the beginning of this section. The interplay of the various parameters of the problem is almost endless - it should be clear by now that unless some way of rapidly testing their interdependence was found, such as provided by the interactive graphics package described here, one would be very reluctant to tackle these tasks at all.

CONCLUSION

In creating programming packages, one attempts to emerge with a result that balances simplicity with power. To achieve maximum flexibility one would have to continually resort to machine language code. On the other hand, a package would petrify if for the sake of extreme simplicity it is directed toward the solution of only one class of problems. We chose the path of extending a well-known language through provision of a subroutine library, one commonly taken in many other areas of computer application. By identifying what we felt to be the basic qualities desired in a terminal system by the casual scientific user, namely, ease of data entry, simple text output format, and graphing of computational results, we were able to limit this library to only three subroutines.

ACKNOWLEDGMENTS

The author would like to express his gratitude to the staff of the IBM San Jose Research Laboratory Computer Facility, and particularly to E. Ryan, K. G. Field, and E. I. Mahoney for their assistance in implementing the package discussed in this paper. He is appreciative to R. Kay for a critical reading of the manuscript.

REFERENCES

1. "Interactive Graphics in Data Processing," IBM Systems Journal 7 (1968).
2. Computer Graphics: Techniques and Applications, ed. by R. D. Parslow, R. W. Prowse, and R. E. Green, Plenum Press, 1969.
3. "A Guide to PL/I for Fortran Users," IBM Form C20-1637.
4. E. J. Kinsinger, "Procedures Allowing PL/I to Call Fortran and Fortran to Call PL/I," SHARE Document C-5225, 1969.
5. A. D. Rully, IBM Systems Journal 7, 248 (1968); "IBM System/360 Operating System Graphic Subroutine Package (GSP) for FORTRAN IV, COBOL, and PL/I," IBM Form C27-6932.

TABLE I. Use of GREAD and STRING I/O to Enter Numerical Data From 2250 Keyboard. An Example of the INPUT Character String Read Below Might be "6.28, 2.99E + 10, 137".

```
          .
          .
          .
DECLARE INPUT CHARACTER (73);
CALL GREAD (INPUT);
GET STRING (INPUT) LIST (TWOPI, C, IRFSC);
          .
          .
          .
```

TABLE II. Parameter List for Sub-Procedure GPLOTS.

Parameter	Description
X	Independent variable vector
Y	Dependent variable vector.
N	Number of points in X and Y.
XAXIS	Configuration of x-axis. XAXIS(1) = Scale factor in user's units per division. The box for the graph is divided on its sides into ten divisions. The scale factor is in terms of these divisions. If XAXIS(1) = 0 then automatic scaling is used and XAXIS(1) returns with the computed scale. XAXIS(2) = beginning value for x-axis in user's units.
YAXIS	Analogous to XAXIS
MARK	Symbol and overlay identifier. =1, points plotted on 2250 =2, lines drawn between points on 2250 =3, points plotted on CALCOMP =4, lines drawn between points in CALCOMP <0, new plot is overlaid on previous plot with new y-axis drawn
NAMEX	36 character title for x-axis
NAMEY	Analogous to NAMEX

TABLE III. List of Direct Calculation Capability Operations Available in Sub-Procedure GNOTES.

Arithmetic Operation	Description
+ (optional)	Add operand to value in register A
-	Subtract operand from A
*	Multiply A by operand
/	Divide A by operand
**	Exponentiate A by operand

Function Operation on A	Description
SIN	Sine
COS	Cosine
TAN	Tangent
COT	Cotangent
SEC	Secant
CSC	Cosecant
ASN	Inverse Sine
ACS	Inverse Cosine
ATN	Inverse Tangent
RAD	Degrees to Radians
DEG	Radians to Degrees
HSN	Hyperbolic Sine
HCS	Hyperbolic Cosine
HTN	Hyperbolic Tangent
EXP	Exponential Function
ATN	Inverse Hyperbolic Tangent
LGT	Briggs Logarithm
LOG	Naperian Logarithm
ALG	Briggs Antilog
ERF	Error Function
FAC	Factorial
PYT	$(A^2 + B^2)^{1/2}$
SQR	Square Root
EV	Ergs to Electron-volts
WN	Ergs to Wavenumbers
REV	Electron-volts to Ergs
RWN	Wavenumbers to Ergs
CS	Change Sign
R	Reciprocal
Z	Clear A, B, C, and D
ZA	Clear A only
ZB	Clear B only
ZC	Clear C only
ZD	Clear D only
END	End Message

TABLE III. List of Direct Calculation Capability Operations Available in Sub-Procedure GNOTES (Cont.)

Assignment Operation	Description	Examples

A = (or B=, C=, D=)	Replace Contents of Specified Register With Value on Right of = sign.	A = -6.2
		C = A

Built-in Constants	Description	Value and Units
&PI	π	3.141593
&E	e	2.718282
&C	Velocity of Light (c)	$2.997925 \cdot 10^{10}$ cm/sec
&Q	Electronic Charge	$-1.60210 \cdot 10^{-20}$ emu
&M	Electronic Mass	$9.1091 \cdot 10^{-28}$ g
&A	Avogadro Number	$6.02252 \cdot 10^{+23}$
&H	Plank Constant (h)	$6.6256 \cdot 10^{-27}$ erg·s
&HB	Plank Constant/2π	$1.05450 \cdot 10^{-27}$ erg·s
&AO	Bohr Radius	$5.29167 \cdot 10^{-9}$ cm
&MU	Bohr Magneton	$-9.2732 \cdot 10^{-21}$ emu
&K	Boltzmann Constant	$1.38054 \cdot 10^{-16}$ erg/°K
&HC	hc	$1.98631 \cdot 10^{-16}$ erg·cm

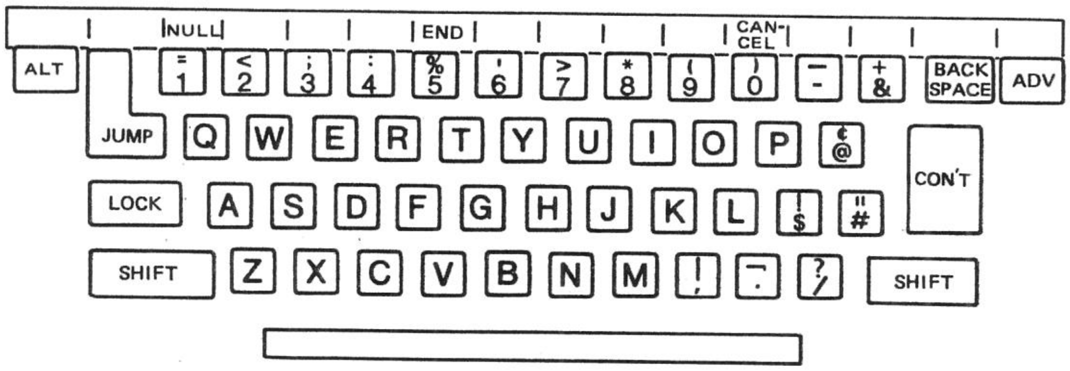


Figure 1 Alphanumeric keyboard of the IBM 2250 Display Unit. The "END" or "CANCEL" functions are activated by depressing the ALT key simultaneously with either the 5-% or 0-) key.

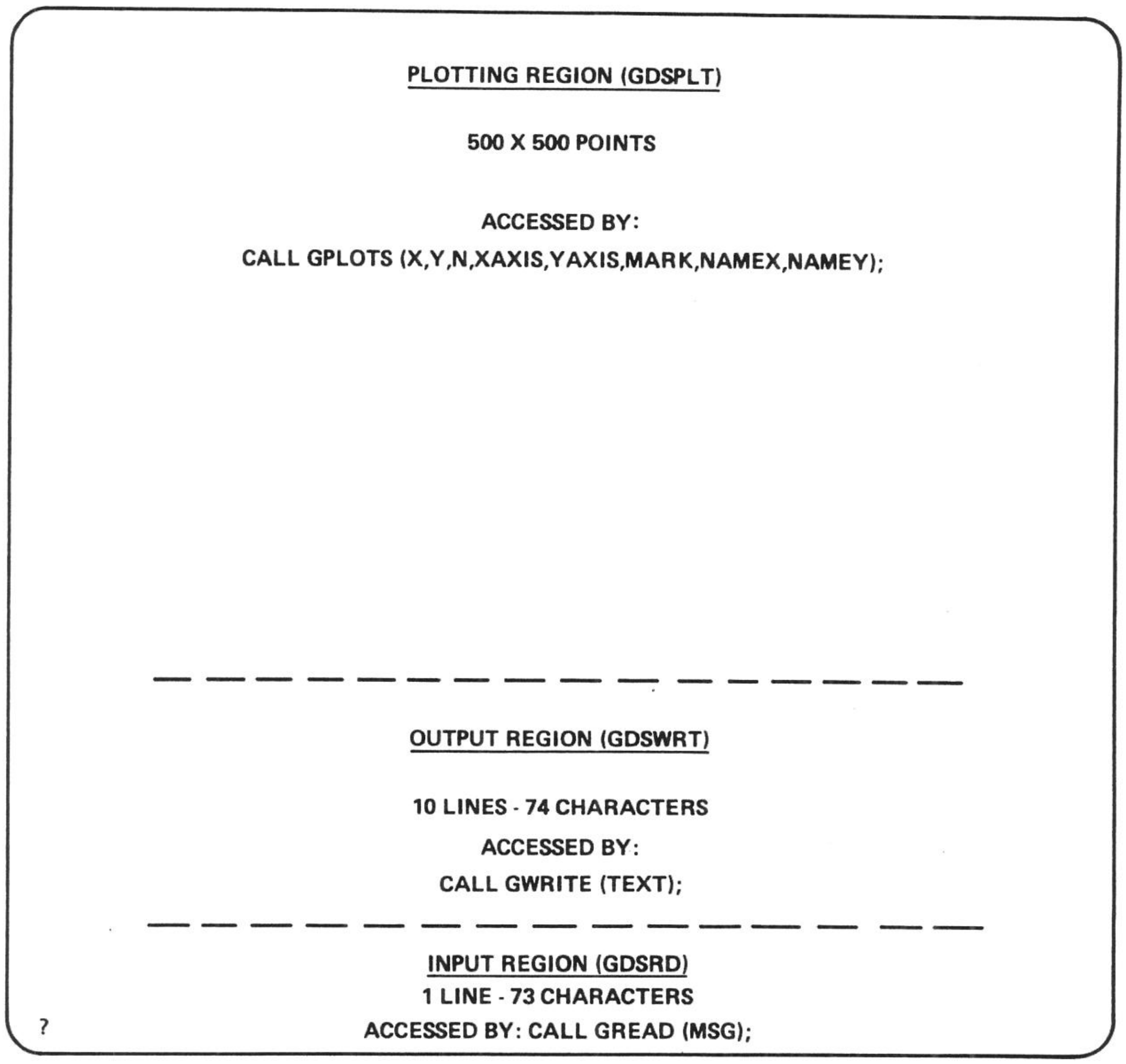


Figure 2 Layout of the 2250 CRT Screen. Each region is defined within GSP by the graphic data set enclosed in parentheses.

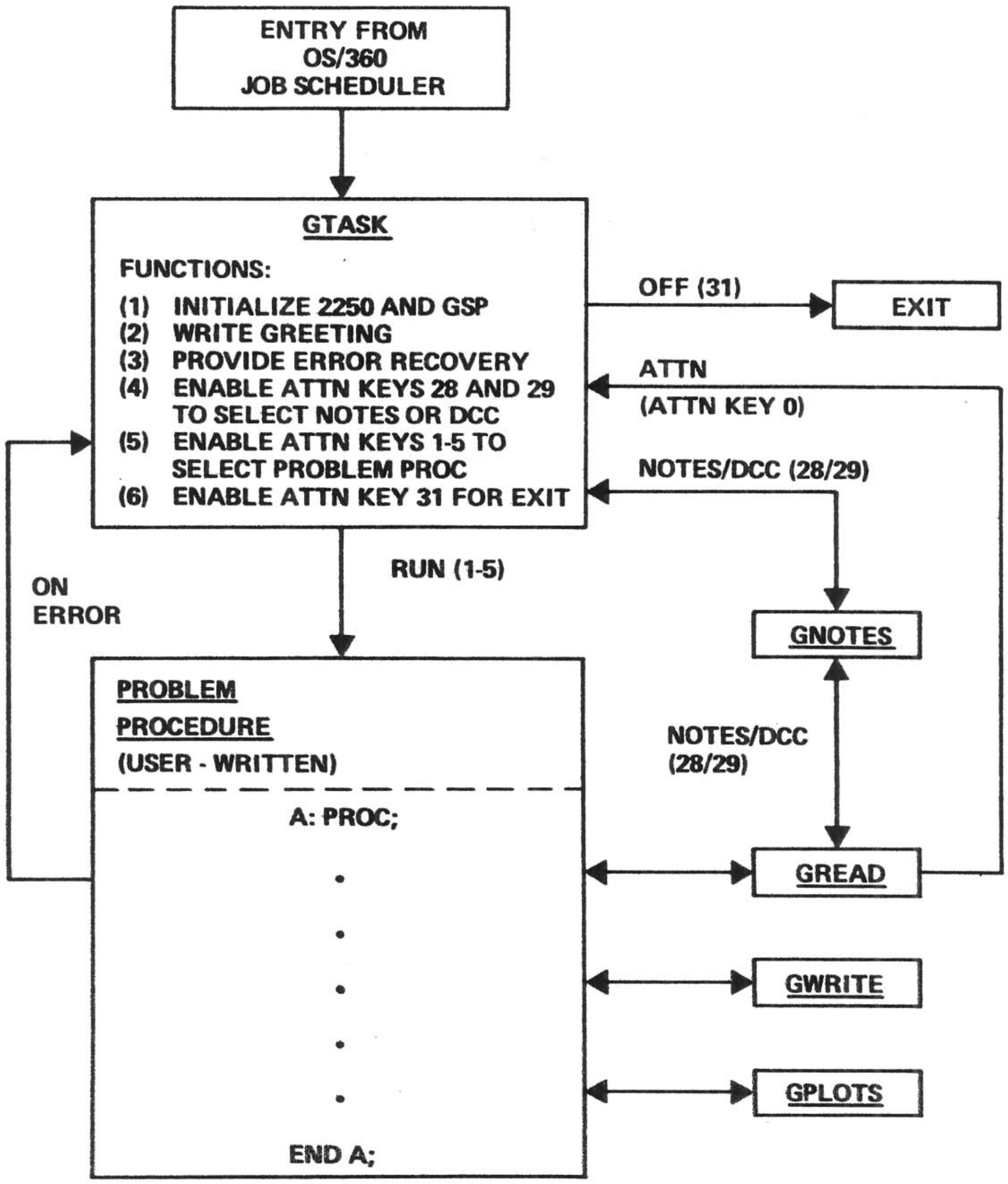

ENTRY FROM OS/360 JOB SCHEDULER
GTASK
FUNCTIONS:
(1) INITIALIZE 2250 AND GSP
(2) WRITE GREETING
(3) PROVIDE ERROR RECOVERY
(4) ENABLE ATTN KEYS 28 AND 29 TO SELECT NOTES OR DCC
(5) ENABLE ATTN KEYS 1-5 TO SELECT PROBLEM PROC
(6) ENABLE ATTN KEY 31 FOR EXIT
OFF (31)
EXIT
ATTN
(ATTN KEY 0)
NOTES/DCC (28/29)
RUN (1-5)
ON ERROR
GNOTES
PROBLEM PROCEDURE
(USER - WRITTEN)
A: PROC;
END A;
NOTES/DCC (28/29)
GREAD
GWRITE
GPLOTS

Figure 3 Flowchart of the overall organization of the interactive system. The user need only concern himself with the coding of the problem procedure and the proper use of GREAD, GWRITE, and GPLOTS. The flow paths from GTASK are pictorial only and are not meant to point to a particular function invoked on returning or exiting.

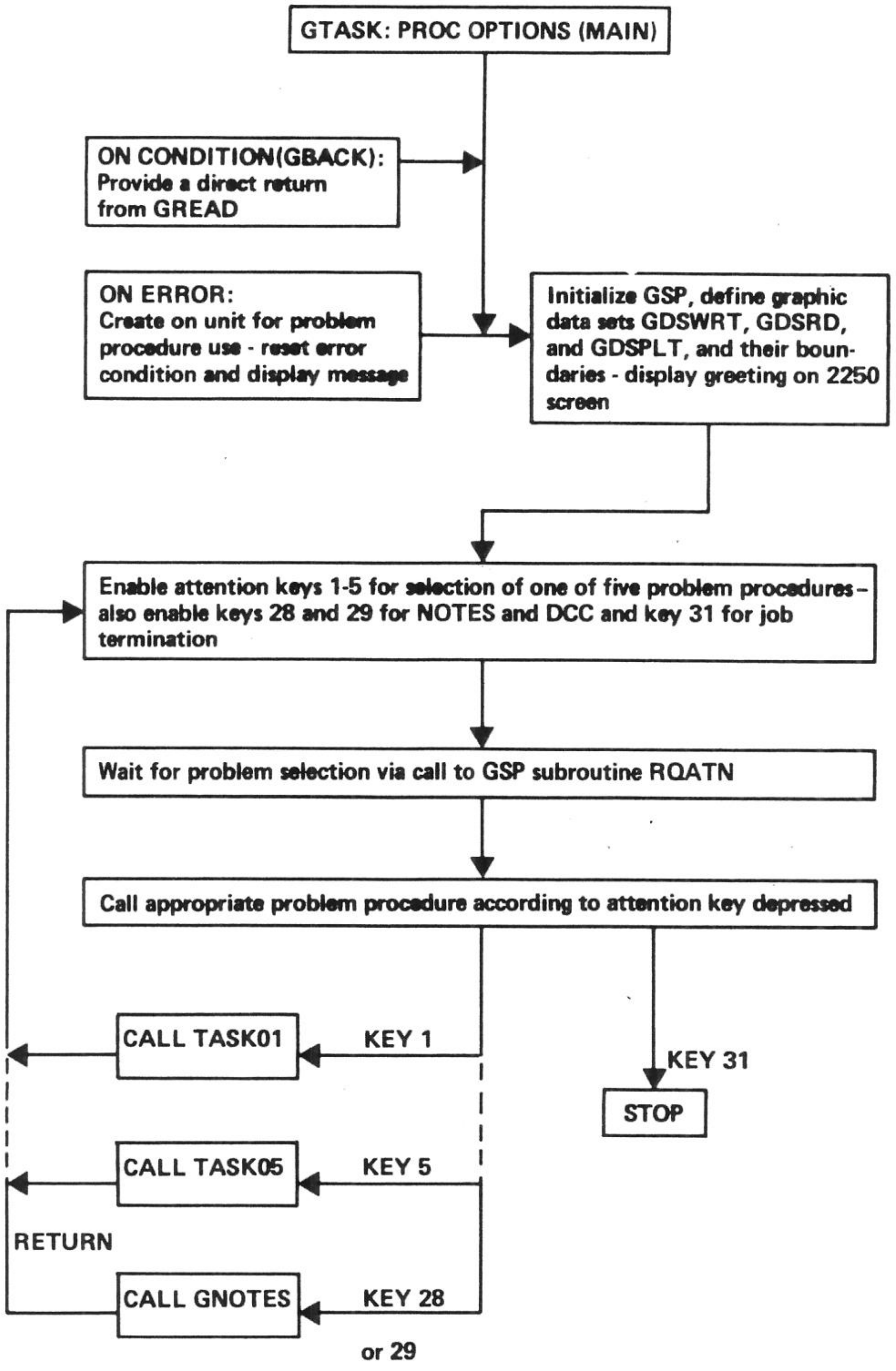

GTASK: PROC OPTIONS (MAIN)
ON CONDITION(GBACK): Provide a direct return from GREAD
ON ERROR: Create on unit for problem procedure use - reset error condition and display message
Initialize GSP, define graphic data sets GDSWRT, GDSRD, and GDSPLT, and their boundaries - display greeting on 2250 screen
Enable attention keys 1-5 for selection of one of five problem procedures - also enable keys 28 and 29 for NOTES and DCC and key 31 for job termination
Wait for problem selection via call to GSP subroutine RQATN
Call appropriate problem procedure according to attention key depressed
KEY 1
CALL TASK01
KEY 5
CALL TASK05
RETURN
KEY 28 or 29
CALL GNOTES
KEY 31
STOP

Figure 4 Flowchart of main procedure GTASK.

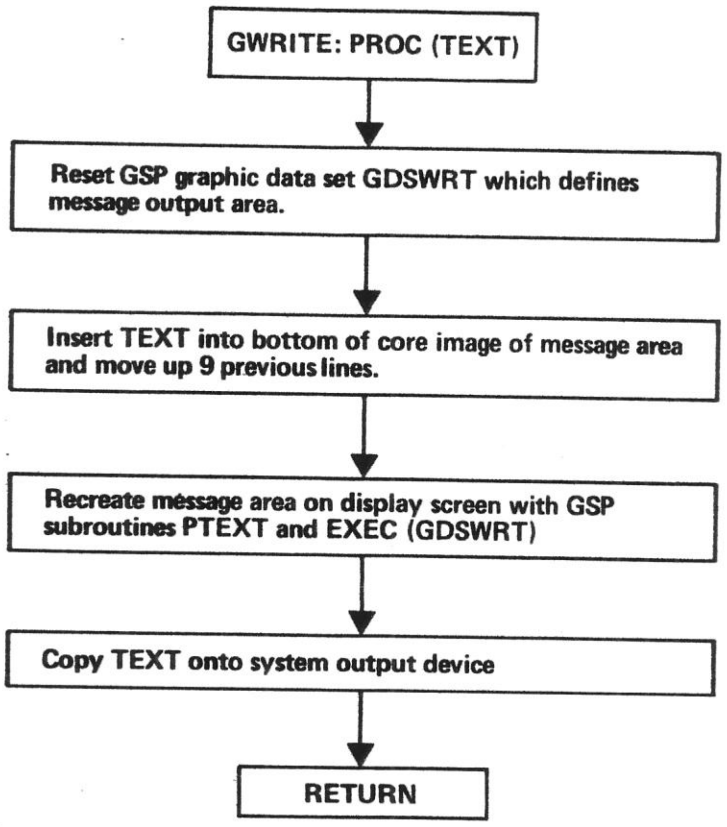

GWRITE: PROC (TEXT)
Reset GSP graphic data set GDSWRT which defines message output area.
Insert TEXT into bottom of core image of message area and move up 9 previous lines.
Recreate message area on display screen with GSP subroutines PTEXT and EXEC (GDSWRT)
Copy TEXT onto system output device
RETURN

Figure 5 Flowchart of sub-procedure GWRITE.

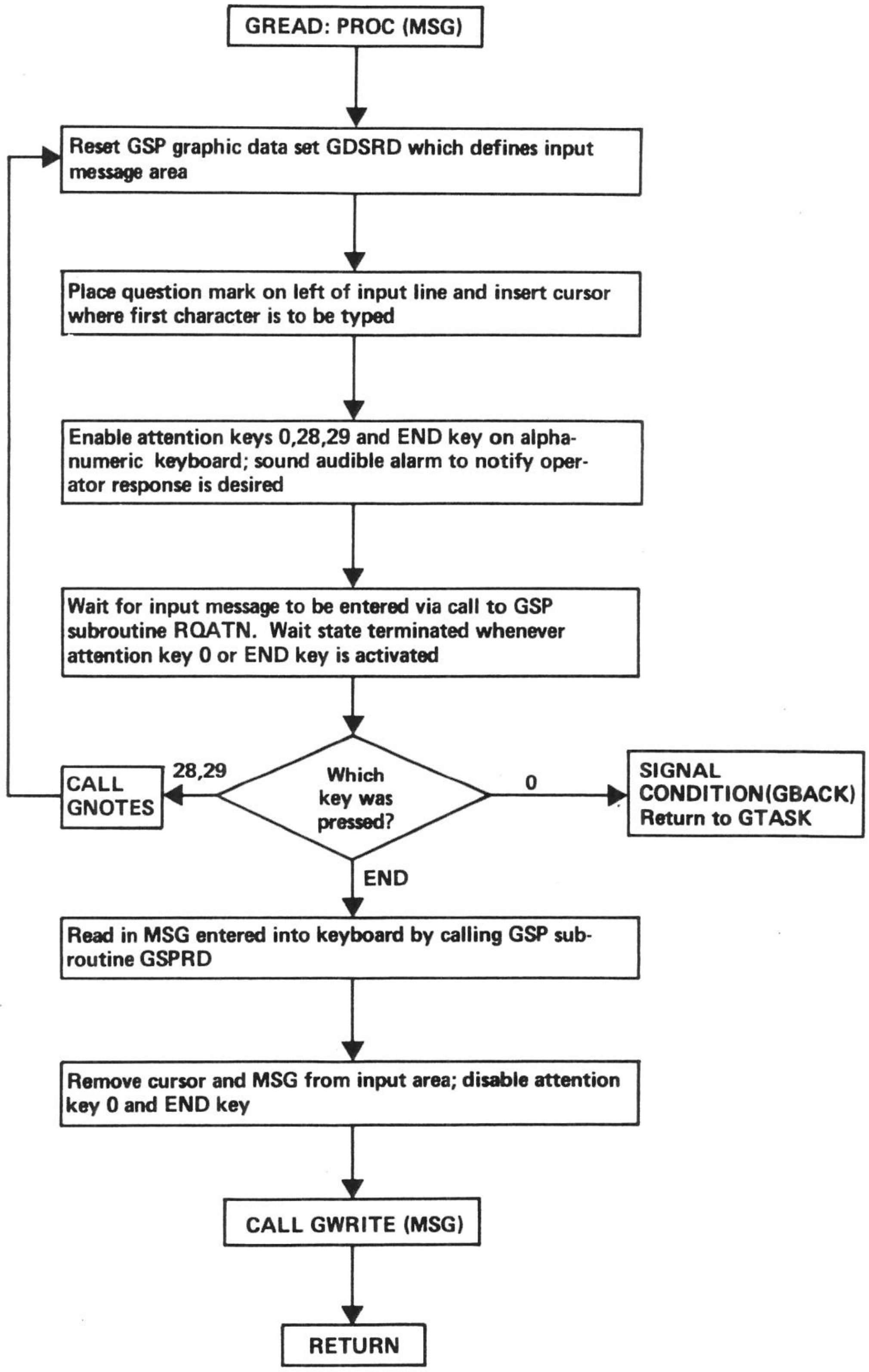

GREAD: PROC (MSG)
Reset GSP graphic data set GDSRD which defines input message area
Place question mark on left of input line and insert cursor where first character is to be typed
Enable attention keys 0,28,29 and END key on alphanumeric keyboard; sound audible alarm to notify operator response is desired
Wait for input message to be entered via call to GSP subroutine RQATN. Wait state terminated whenever attention key 0 or END key is activated
Which key was pressed?
28,29
CALL GNOTES
0
SIGNAL CONDITION(GBACK) Return to GTASK
END
Read in MSG entered into keyboard by calling GSP subroutine GSPRD
Remove cursor and MSG from input area; disable attention key 0 and END key
CALL GWRITE (MSG)
RETURN

Figure 6 Flowchart of sub-procedure GREAD.

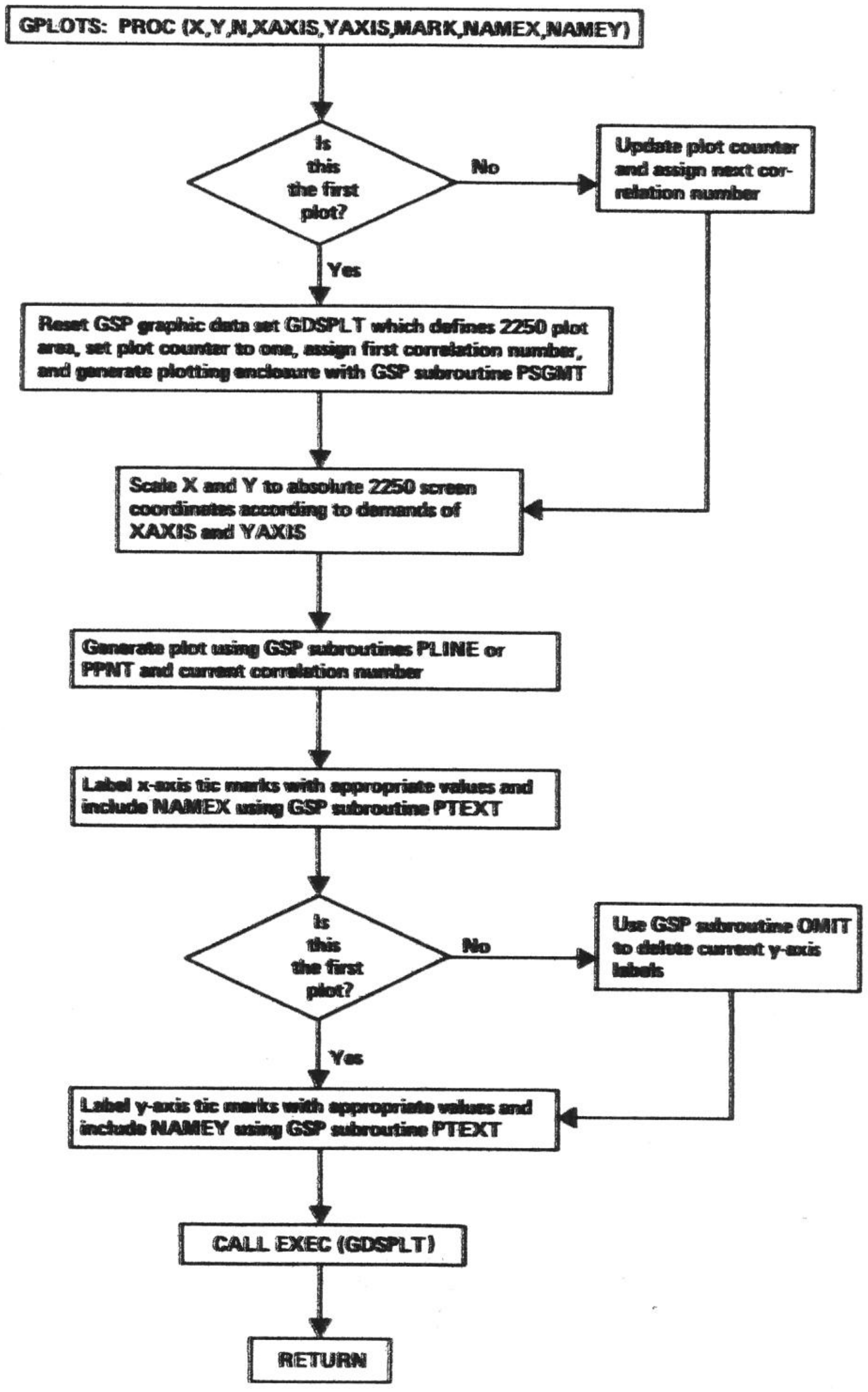

GPLOTS: PROC (X,Y,N,XAXIS,YAXIS,MARK,NAMEX,NAMEY)
Is this the first plot?
No
Update plot counter and assign next correlation number
Yes
Reset GSP graphic data set GDSPLT which defines 2250 plot area, set plot counter to one, assign first correlation number, and generate plotting enclosure with GSP subroutine PSGMT
Scale X and Y to absolute 2250 screen coordinates according to demands of XAXIS and YAXIS
Generate plot using GSP subroutines PLINE or PPNT and current correlation number
Label x-axis tic marks with appropriate values and include NAMEX using GSP subroutine PTEXT
Is this the first plot?
No
Use GSP subroutine OMIT to delete current y-axis labels
Yes
Label y-axis tic marks with appropriate values and include NAMEY using GSP subroutine PTEXT
CALL EXEC (GDSPLT)
RETURN

Figure 7 Flowchart of sub-procedure GPLOTS. Not shown are those sections used to produce off-line Calcomp hard-copy graphs.

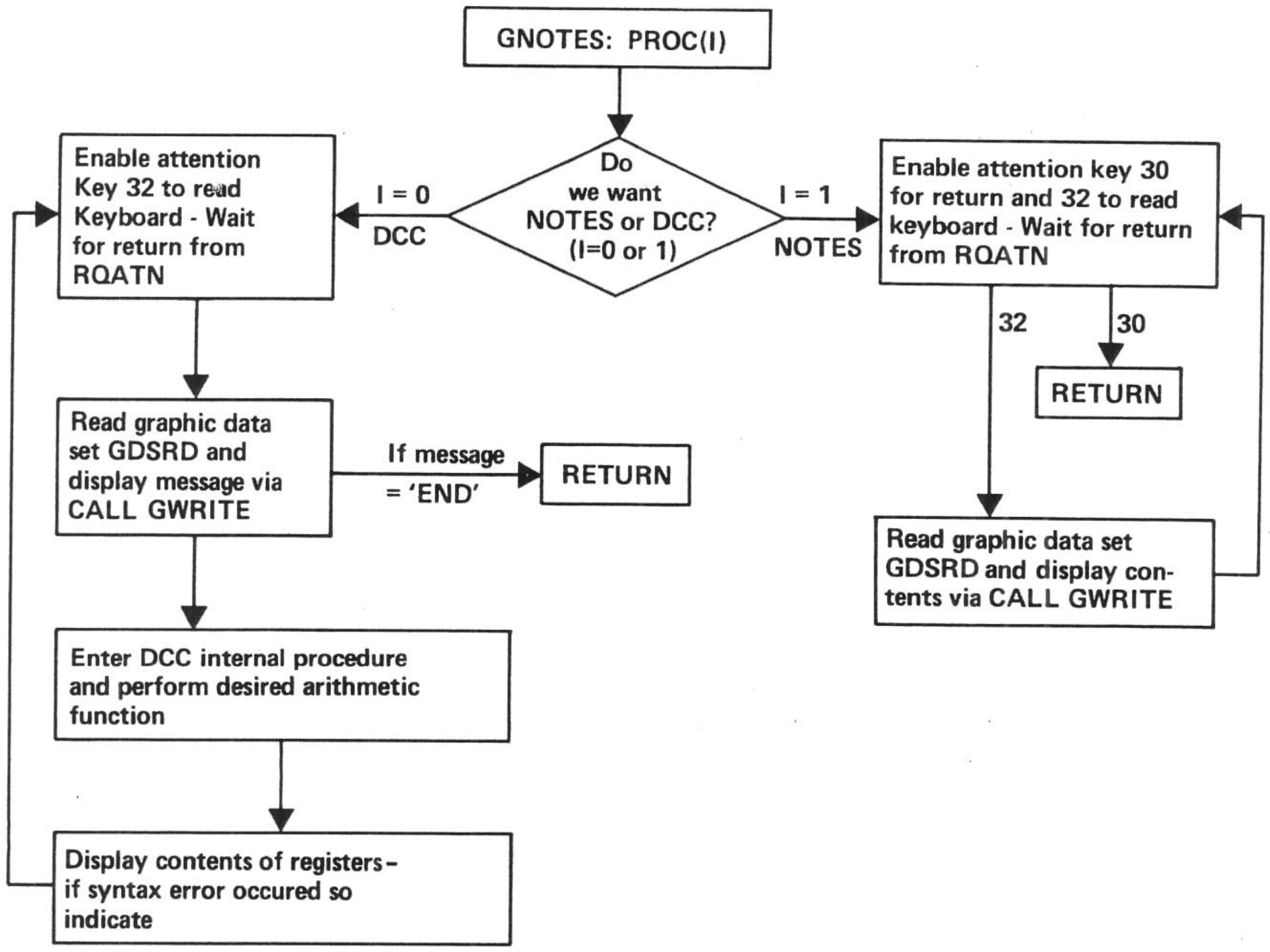

GNOTES: PROC(I)
Do we want NOTES or DCC? (I=0 or 1)
I = 0
DCC
I = 1
NOTES
Enable attention Key 32 to read Keyboard - Wait for return from RQATN
Read graphic data set GDSRD and display message via CALL GWRITE
If message = 'END'
RETURN
Enter DCC internal procedure and perform desired arithmetic function
Display contents of registers - if syntax error occured so indicate
Enable attention key 30 for return and 32 to read keyboard - Wait for return from RQATN
32
30
RETURN
Read graphic data set GDSRD and display contents via CALL GWRITE

Figure 8 Flowchart of sub-procedure GNOTES. This sub-procedure is accessible by the problem procedure only when it calls GREAD. Attention key 28 (I = 0) and attention key 29 (I = 1) choose DCC and NOTES, respectively.

```
//PMGSCOPE  JOB  K05-4167,'GRANT, PM'
//*             **********     C O M P I L E     **********
//* PERFORM IN JOBSTEP PROBPROC THE STANDARD IBM PL/I CATALOGED
//* PROCEDURE FOR COMPILE, LINK-EDIT, AND EXECUTE
//PROBPROC  EXEC PL1LFCLG                  FIRST EXECUTE THE PL/I COMPILER
//PL1L.SYSIN  DD  *                        CARDS TO BE COMPILED FOLLOW
      YOURTSK: PROC;           /* FIRST CARD OF PROBLEM PROCEDURE */
           /*  PL/I CARDS FOR YOUR PROBLEM PROCEDURE  */
      END YOURTSK;             /* LAST CARD OF PROBLEM PROCEDURE */
/*
//*             **********     L I N K - E D I T     **********
//* NOW USE THE LINKAGE-EDITOR TO OBTAIN THE REQUISITE SUBPROGRAMS
//* CALLED BY YOUR PROBLEM PROCEDURE AND REPLACE DUMMY SUBPROGRAM
    //* TASK01 IN MAIN PROCEDURE GTASK WITH YOURTSK, THUS CREATING AN
//* EXECUTABLE MODULE
//* THE FOLLOWING CARDS DENOTE THE LIBRARIES THAT MAY BE ACCESSED BY
//* YOUR PROBLEM PROCEDURE
//LKED.SYSLIB  DD
//             DD  DSN=SYS1.FORTLIB,DISP=SHR
//             DD  DSN=K05.GRANT,DISP=SHR
//* THE FOLLOWING CARDS CREATE THE EXECUTABLE LOAD MODULE
//LKED.SYSIN  DD  *            INPUT CARDS TO THE LINKAGE EDITOR FOLLOW
      REPLACE TASK01(YOURTSK),TASK01A(YOURTSKA)
      INCLUDE SYSLIB(GTASK)
      ENTRY IHENTRY
/*
//*             **********     E X E C U T E     **********
//* FINALLY ASSIGN APPROPRIATE I/O DEVICES AND RUN THE JOB
//GO.PLOTTER  DD  DSN=K05.PMGPLOT,DISP=SHR   DEFINES CALCOMP STORAGE
//GO.FT10F001  DD  UNIT=GRAPH              ASSIGNS A 2250 FOR THIS JOB
//GO.SYSIN  DD  *              ANY INPUT CARDS NEEDED BY YOURTSK FOLLOW
/*
```

Figure 9 Sample Job Control Language deck necessary for implementing an interactive problem procedure in a typical computer center jobstream environment.

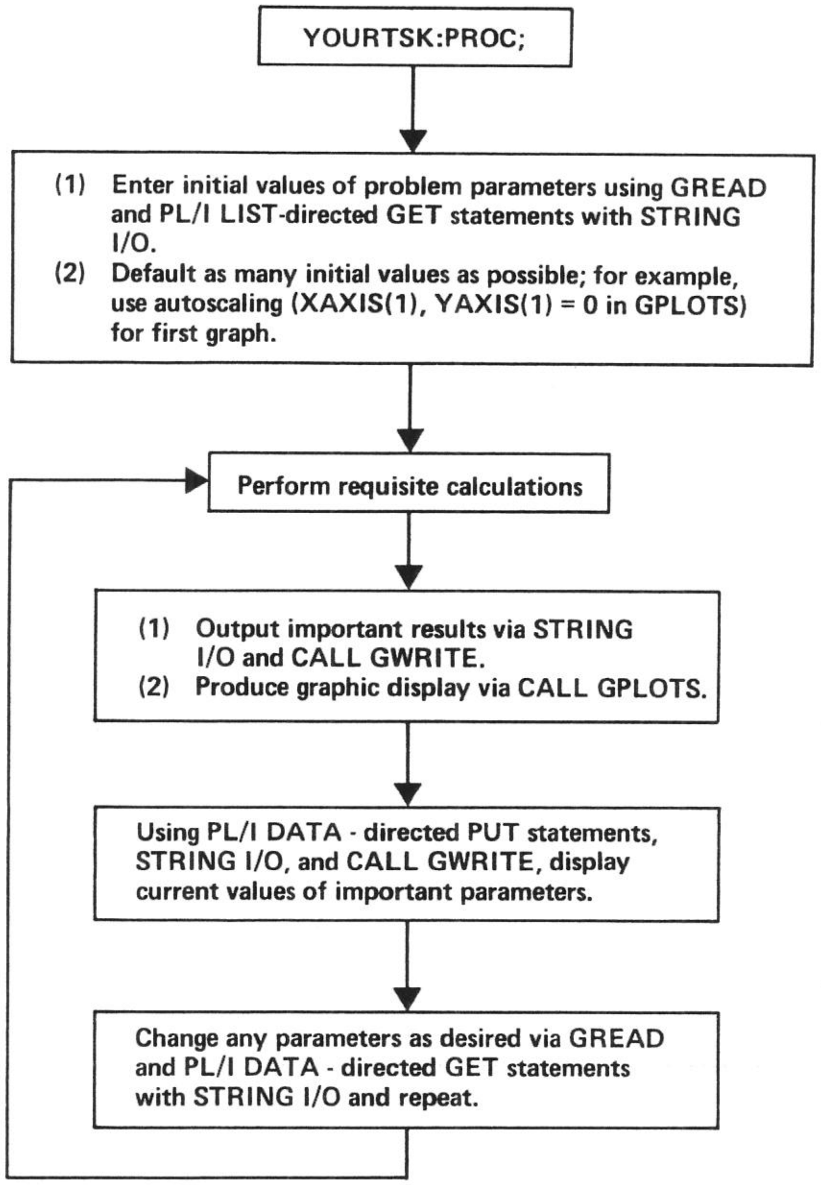

YOURTSK:PROC;
(1) Enter initial values of problem parameters using GREAD and PL/I LIST-directed GET statements with STRING I/O.
(2) Default as many initial values as possible; for example, use autoscaling (XAXIS(1), YAXIS(1) = 0 in GPLOTS) for first graph.
Perform requisite calculations
(1) Output important results via STRING I/O and CALL GWRITE.
(2) Produce graphic display via CALL GPLOTS.
Using PL/I DATA - directed PUT statements, STRING I/O, and CALL GWRITE, display current values of important parameters.
Change any parameters as desired via GREAD and PL/I DATA - directed GET statements with STRING I/O and repeat.

Figure 10 Suggested organization of a typical problem procedure.

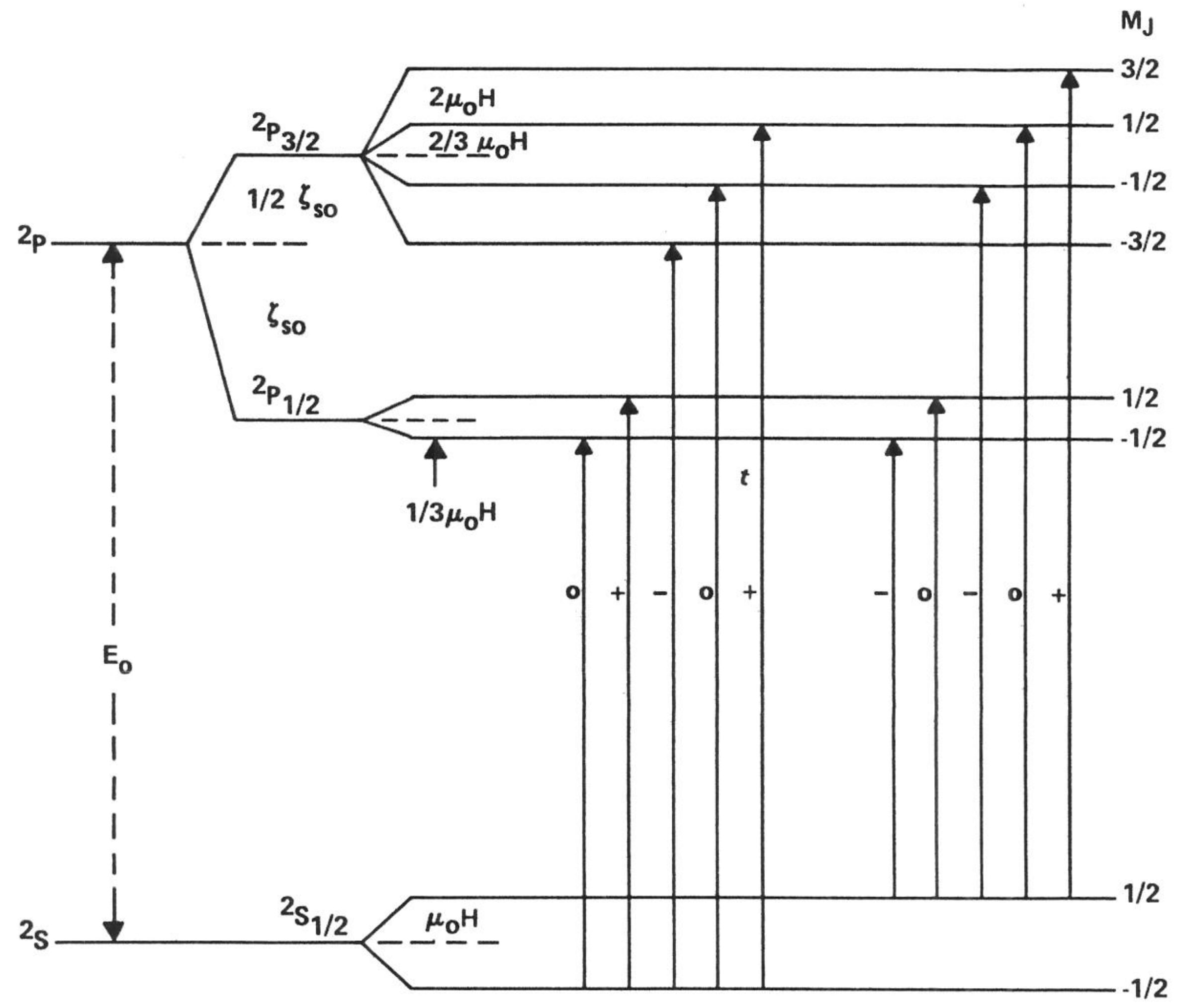
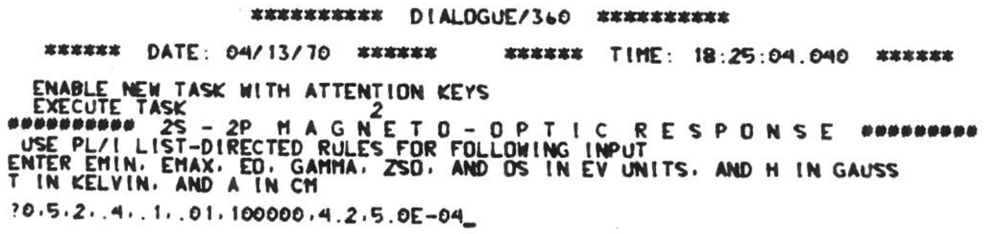
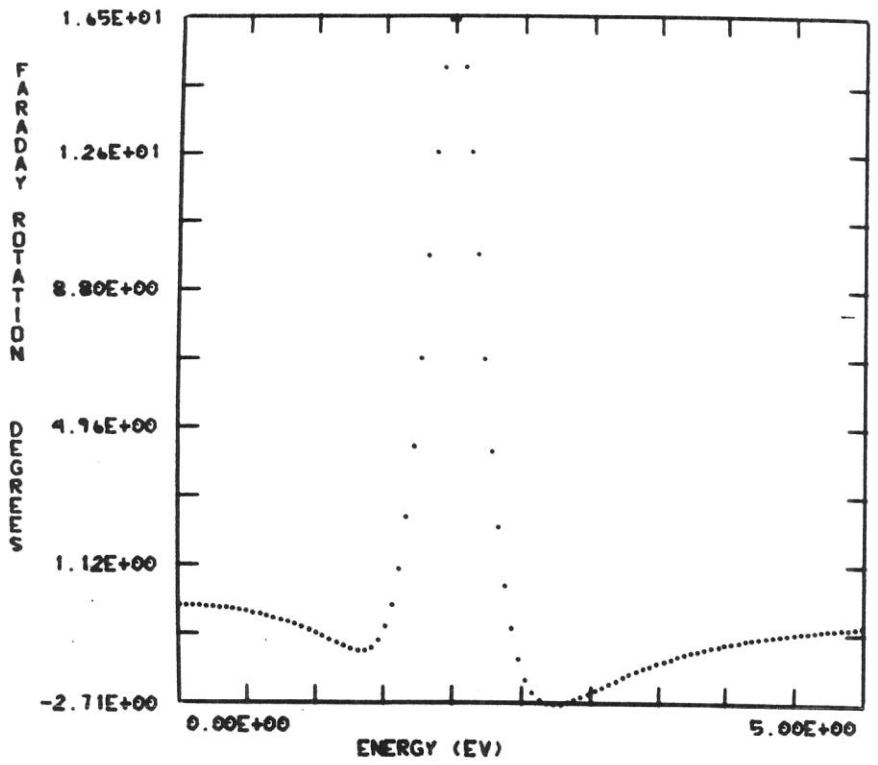


Figure 11 Term scheme for a $^2S \rightarrow {}^2P$ transition with spin-orbit splitting in the presence of a magnetic field. ζ_{so} is the spin-orbit coupling parameter, H the field strength, and μ_o the Bohr magneton. Shown are allowed transitions for left (-) and right (+) circularly polarized radiation and unpolarized radiation (0) from both components of the magnetically split ^2S ground state. The transition strengths are related to each other to within a constant factor by the appropriate 6-j and 3-j coefficients.



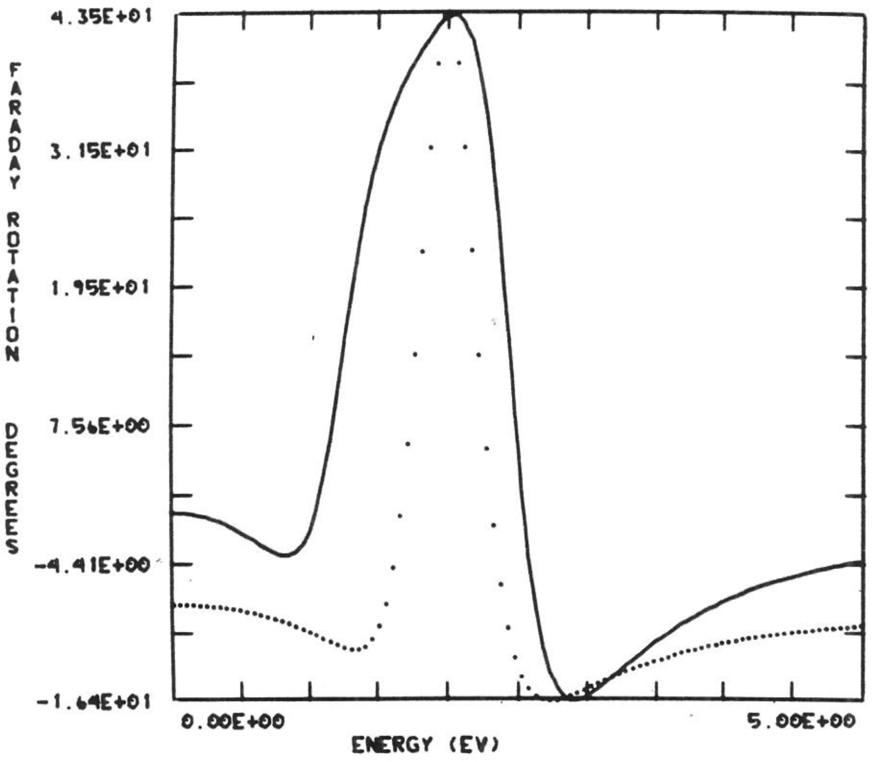
```
                 ***********  DIALOGUE/360  ***********

   ******  DATE: 04/13/70  ******     ******  TIME: 18:25:04.040  ******

  ENABLE NEW TASK WITH ATTENTION KEYS
  EXECUTE TASK          2
**********  2S - 2P  M A G N E T O - O P T I C  R E S P O N S E  *********
 USE PL/I LIST-DIRECTED RULES FOR FOLLOWING INPUT
ENTER EMIN, EMAX, ED, GAMMA, ZSO, AND OS IN EV UNITS, AND H IN GAUSS
T IN KELVIN, AND A IN CM
?0,5,2,.4,.1,.01,100000,4.2,5.0E-04_
```

Figure 12   Photograph of sign-on greeting, problem procedure selection, and entry of initial data to same. Text following "EXECUTE TASK 2" is generated by the problem procedure via calls to GWRITE. The last line is the operator response to a call to GREAD. At the moment we are waiting in GREAD for the operator to strike the "END" function on the 2250 keyboard.



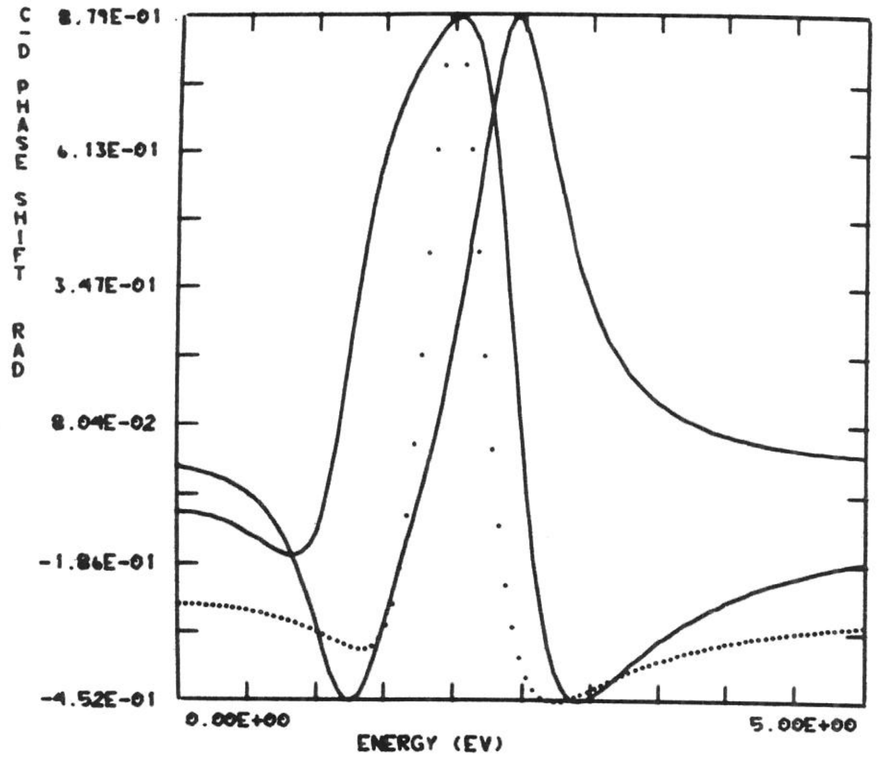
```
 USE PL/I LIST-DIRECTED RULES FOR FOLLOWING INPUT
ENTER EMIN, EMAX, ED, GAMMA, ZSO, AND OS IN EV UNITS, AND H IN GAUSS
T IN KELVIN, AND A IN CM
?0,5,2,.4,.1,.01,100000,4.2,5.0E-04
DO YOU WISH CIRCULAR DICHROIC PHASE SHIFT ALSO?  REPLY YES OR NO
?NO
DO YOU WISH TO REVIEW STATUS OF IMPORTANT VARIABLES?   REPLY YES OR NO
?NO
MAKE ANY CHANGES USING PL/I DATA-DIRECTED RULES
DENOTE END OF CHANGES BY TYPING:  LAST='END';
?ZSO=.8,YAXIS(1)=0,MARK=-2,LAST='END';_
```

Figure 13 The "END" function has been executed and the data entered in Fig. 12 has been read. The Faraday rotation for the model of Fig. 11 has been computed over the energy limits 0-5 eV and is displayed with a call to GPLOTS. The problem procedure then offers two options which are rejected. Finally, using GREAD with a subsequent GET DATA statement and STRING I/O (see Fig. 10), we change only the spin-orbit coupling constant ZSO, two parameters to GPLOTS, and are ready to repeat the calculation.



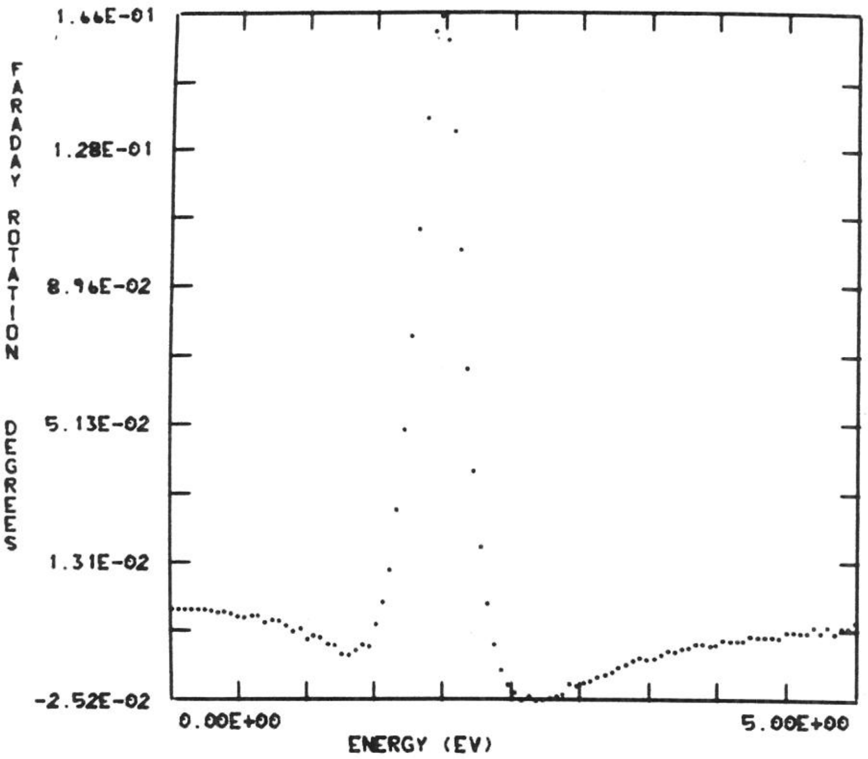
```
T IN KELVIN, AND A IN CM
?0,5,2,.4,.1,.01,100000,4,2,5.0E-04
DO YOU WISH CIRCULAR DICHROIC PHASE SHIFT ALSO?  REPLY YES OR NO
?NO
DO YOU WISH TO REVIEW STATUS OF IMPORTANT VARIABLES?  REPLY YES OR NO
?NO
MAKE ANY CHANGES USING PL/I DATA-DIRECTED RULES
DENOTE END OF CHANGES BY TYPING:  LAST='END';
?ZSO=.8,YAXIS(1)=0,MARK=-2,LAST='END';
DO YOU WISH CIRCULAR DICHROIC PHASE SHIFT ALSO?  REPLY YES OR NO
?_
```

Figure 14 The calculation has been repeated with the new value of ZSO and overlaid as a line plot (see Table II) on the old data. We now await a reply to a query we declined on the first pass.



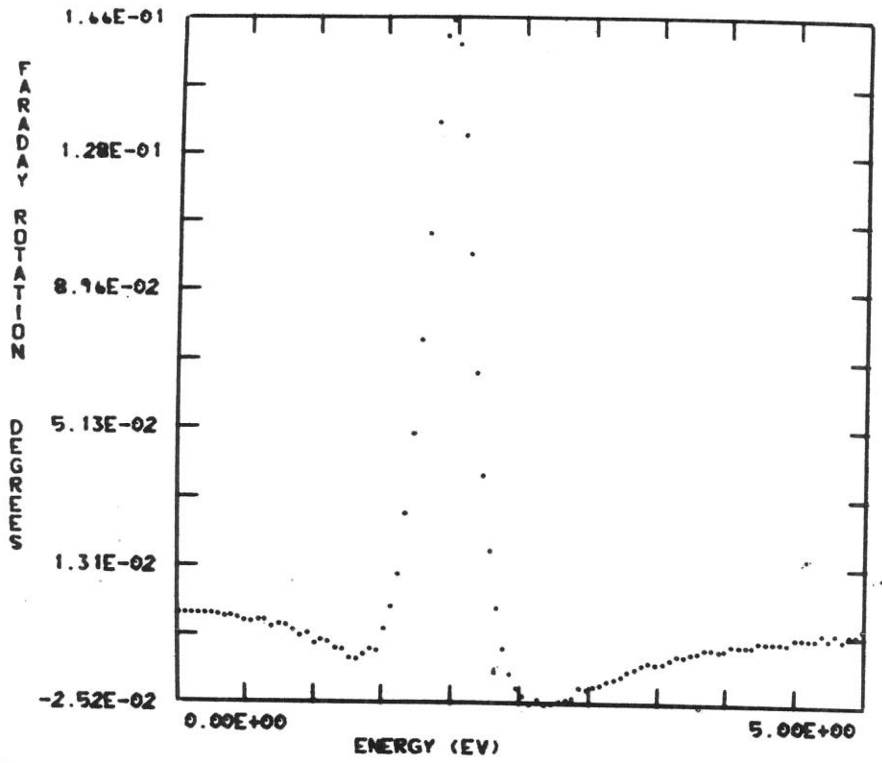
```
?YES
EMIN= 0.00000E+00 EMAX= 5.00000E+00;
EO= 2.00000E+00 GAMMA= 3.99999E-01;
ZSO= 7.99999E-01 DS= 9.99999E-03;
H= 1.00000E+05 T= 4.19999E+00 A= 4.99999E-04;
XAXIS(1)= 4.99999E-01 XAXIS(2)= 0.00000E+00;
YAXIS(1)= 1.33053E-01 YAXIS(2)=-4.51802E-01;
MARK=         -2;
MAKE ANY CHANGES USING PL/I DATA-DIRECTED RULES
DENOTE END OF CHANGES BY TYPING:  LAST='END';
?_
```

Figure 15 We reply in the affirmative to both the request for circular dichroic phase shift display and for a review of the present value of important variables. The phase shift is plotted in overlay fashion and the pertinent variables appear through use of a PUT DATA statement, STRING I/O and GREAD in the problem procedure.



```
MAKE ANY CHANGES USING PL/I DATA-DIRECTED RULES
DENOTE END OF CHANGES BY TYPING:  LAST='END';
****************************  N O T E S  *****************************
LET US NOW RETURN TO THE ORIGINAL PROBLEM WITH ZSO=.1 BUT WITH THE
TEMPERATURE RAISED TO 300 KELVIN.
THE EFFECT SHOULD BE TO DECREASE THE FARADAY ROTATION DUE TO
POPULATING THE SPIN=1/2 COMPONENT OF THE GROUND STATE.
**************************  E N D  N O T E S  *************************
?ZSO=.1,T=300,YAXIS(1)=0,MARK=1,LAST='END';
DO YOU WISH CIRCULAR DICHROIC PHASE SHIFT ALSO?  REPLY YES OR NO
?_
```

Figure 16 Before replying to the request for more data, we enter an annotation concerning what we are about to do by selecting the NOTES option of GNOTES with attention key 29. Once the desired comment has been made, we exit via attention key 30 (see Fig. 8) and answer the original query with new values of spin-orbit coupling and temperature. The new result is then displayed.



```
******************************  N O T E S  ******************************
COMPUTE GROUND STATE SPLITTING FOR H=100000 GAUSS
***************************  E N D  N O T E S  **************************
*******************************  D C C  *********************************
100000
A= 1.00000E+05 B= 0.00000E+00 C= 0.00000E+00 D= 0.00000E+00:
*&MU
A=-9.27319E-16 B= 0.00000E+00 C= 0.00000E+00 D= 0.00000E+00:
EV
A=-5.78815E-04 B= 0.00000E+00 C= 0.00000E+00 D= 0.00000E+00:
*2_
```

Figure 17 We now begin a simple calculation of ground state splitting before continuing the problem procedure. An appropriate comment is entered prior to invoking DCC with attention key 28. Note that after each entry and computation the present contents of all registers are displayed. Note also that no question mark preceeds entries to NOTES or DCC.